Wrocław University of Science and Technology

Safety and Trustworthiness of Deep Learning in Computer Vision – With Application of Out-of-Distribution Detection Techniques

By

Kamil Szyc

Doctoral Thesis

Supervisor:
Henryk Maciejewski
PhD, DSc, Assoc. Prof.

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Faculty of Information and Communication Technology
Department of Computer Engineering

Wrocław, V 2022

Kamil Szyc, *Safety and Trustworthiness of Deep Learning in Computer Vision – With Application of Out-of-Distribution Detection Techniques*, V 2022

SUPERVISOR:
Henryk Maciejewski, PhD, DSc, Assoc. Prof.

LOCATION:
Wrocław

DATE:
V 2022

I have dedicated this thesis to my wonderful, beloved wife Dorota and children Jakub and Zofia. Thank you for all your love and support. I cherish every moment with you.

I would like to express my gratitude to my supervisor Henryk Maciejewski PhD, DSc, Assoc. Prof., for his guidance and insightful comments in completing this thesis.

I would like to thank my parents, Maciej and Monika, for inspiring me and igniting my passion for knowledge.

I would also like to send my appreciation to my brother Michał, Tomasz Walkowiak, Ph.D., family and friends. Without their valuable opinions, ideas, and numerous discussions, the research would not have been accomplished.

Kamil Szyc
V 2022, Wrocław

Abstract

Although the newest computer vision models achieved impressive accuracy, further challenges in the problem of image classification still exist. Due to the broad applications of these algorithms in real-life, improving security and trustworthiness seems to become the most important task nowadays for researchers.

There are many challenges and threats to deep learning approaches connected with the above. One of them is susceptibility to natural and adversarial attacks. The network can be easily fooled with special kinds of images. The goal of adversarial attacks lies in preparing new images by adding additional unique noise that forces the network to point out some class with high certainty despite the image not presenting that class. Images that do not belong to any known classes - however, the network returns high certainty results for them, can be considered as natural attacks. Another challenge is the robustness of the network. The robustness is the ability of the network to classify similar (but not within the same distributions) images to the training examples – for instance, working with extra distortions like with rotated images, with different weather conditions (e.g., fog, rains), or obscured by some object. The robustness is also the ability to return similar outputs for similar inputs, increasing the stability of the response. It connects with the next challenge - interpretability. Modern models work like "black boxes", so it is impossible to interpret the factors influencing the outputs. However, we - people - would like to understand why the network responds in that way to get to know the principles.

All of the above can be connected to one of the most important threats, in our opinion – the close-set nature of classification. The artificial network usually is forced to choose one winning class, while there is no "I do not know" or "unknown" class. Often the classifier returns a high certainty answer to a random label (from a human perspective). When we will "open" the model by adding the ability to detect unknown data distribution, we increase the safety of the ML model.

The main thesis of this dissertation is that the safety and trustworthiness of AI models can be assured only when the models can distinguish between known and unknown data - therefore, we showed and thoroughly researched the Out-of-Distribution detection techniques. Still, there are no clear conclusions and recommendations in the literature on which methods are most successful for a given recognition problem. Furthermore, our research suggests that the best method depends on pairs of known and unknown samples. However, the OoD samples are unknown by their nature. In this thesis, we looked at OoD from a practical perspective. The safety and trustworthiness models must work as open-set classification – we focused on how to force classic close-set models to be OoD aware. Our contributions are as follows.

We analyzed the selected popular OoD methods. We showed there is no best OoD approach, and it should depend on the tested ID and OoD pair. We showed limitations and analyzed the assumptions of the Extreme Value Machine (EVM) algorithm and instability of parametric models based on MultiVariate Normal (MVN) distribution, i.e., using the Mahalanobis distance, in high dimensional data. We showed that the nonparametric, density-based LOF approach performs better than based on MVN or logits in many cases. We see a lack of comparison between LOF and other benchmark methods in the literature.

We showed the efficiency of the OoD method based on large-scale (e.g., the high image resolutions or the high number of known classes) dataset benchmarks. Many OoD methods

are evaluated only on a low-scale dataset, which is insufficient in real-life problems. We conclude that the OoD methods based on logits work poorly, and Mahalanobis is slightly worse compared to other methods on a large-scale. The LOF seems to be the most stable method.

We noticed the significant influence of the feature extraction strategy on improving the efficiency of OoD detection. The standard method uses only Global Average Pooling (GAP). However, various approaches can focus on different components (e.g., on edges, patterns, or whole objects), so for different pairs of ID and OoD, different feature extractors can be useful in separating data. We recommend it as a new hyperparameter. We showed that reducing the size of feature vectors leads to severe efficiency deterioration for many methods. The LOF-based methods seem to be the only method we recommend using together with dimensional reduction.

We showed that OoD detection can filter many adversarial examples. Moreover, we recommend choosing a different feature extraction strategy than GAP (notice that the GAP is the base for the classifier) - it may improve the efficiency of detecting attack samples.

We showed the importance of proper data augmentation techniques in OoD detection problems and robustness.

We researched the problems that occur with unknown examples of detection. We analyzed the problem with instability and experimented with repeatability of the OoD detection methods – and concluded that results in literature should be taken with caution. The slightly different model state can change the OoD method's efficiency drastically. Moreover, we showed the mismatch between the image and feature space - i.e., similar images (in our human understanding) can generate distant features, and images from different classes can be close to each other in feature space. We conclude that the common usage of near and far OoD examples definitions is inaccurate.

KEYWORDS

Image Classification – Computer Vision – Out-of-Distribution Detection – Open-Set Classification – Deep Learning – Robustness – Security and Trustworthiness of AI – Features Extraction Methods – Adversarial Attacks – Convolutional Neural Networks

Streszczenie (Abstract - Polish Version)

Najnowsze modele wizyjne osiągają imponującą dokładność, jednak wciąż istnieją kolejne wyzwania w problemie klasyfikacji obrazów. Szerokie zastosowanie takich modeli w praktycznych projektach sprawia, że obecnie poprawa bezpieczeństwa i wiarygodności sztucznej inteligencji wydaje się najważniejszym zadaniem dla badaczy.

Z powyższym zagadnieniem wiąże się również wiele innych wyzwań i zagrożeń. Jednym z nich jest podatność na ataki naturalne i typu adversarial (z ang. adversarial attacks). Sieć można łatwo oszukać za pomocą specjalnych obrazów. Celem ataków typu adversarial jest przygotowanie nowych obrazów poprzez dodanie dodatkowego unikalnego szumu, który zmusza sieć do wskazania wybranej klasy z dużą pewnością, mimo że obraz nie przedstawia tej klasy. Obrazy, które nie należą do żadnych znanych klas, a mimo to sieć jest pewna swojej błędnej odpowiedzi nazywamy atakami naturalnymi. Kolejnym wyzwaniem jest krzepkość (z ang. robustness) sieci. Krzepkość to zdolność sieci do klasyfikowania obrazów podobnych do przykładów treningowych - na przykład obrazów, które zostały w pewien sposób zniekształcone. Przykładem takich zniekształceń mogą być obrócone zdjęcia, zdjęcia zrobione w różnych warunkach pogodowych (np. we mgle lub w deszczu) lub gdy główny obiekt jest zasłonięty. Krzepkość to także zdolność do zwracania podobnych wyników dla podobnych danych wejściowych, określana także jako stabilność odpowiedzi. Łączy się to z kolejnym wyzwaniem - interpretowalnością. Współczesne modele działają jak ”czarne skrzynki”, więc nie da się zrozumieć czynników wpływających na wyniki. My - ludzie - chcielibyśmy jednak zrozumieć, dlaczego sieć reaguje w ten sposób, aby poznać zasady jej działania.

Wszystko to można powiązać z jednym z najważniejszych, moim zdaniem, zagrożeń – klasyfikacją w zbiorze zamkniętym. Sztuczna sieć jest zazwyczaj zmuszona do wyboru jednej zwycięskiej klasy, podczas gdy nie ma klasy ”nie wiem” lub ”nieznane”. Często klasyfikator zwraca odpowiedź o wysokim stopniu pewności do losowej (z ludzkiego punktu widzenia) etykiety. Gdy ”otworzymy” model, dodając możliwość wykrywania nieznanego rozkładu danych, zwiększymy bezpieczeństwo modelu ML.

Główną tezą tej rozprawy jest to, że bezpieczeństwo i wiarygodność modeli sztucznej inteligencji można zapewnić tylko wtedy, gdy modele te będą umieć rozróżnić dane znane od nieznanych - dlatego też pokazałem i dokładnie zbadałem techniki wykrywania danych nieznanych (z ang. Out-of-Distribution, OoD detection). Wciąż jednak w literaturze nie ma jednoznacznych wniosków i zaleceń dotyczących tego, które metody są najskuteczniejsze. Co więcej, przeprowadzone w tej pracy badania sugerują, że najlepsza metoda zależy od pary danych znanych i nieznanych. Jednakże dane nieznane są z natury nieokreślone, co sprawia, że problem jest trudny. W tej pracy skupiono się na praktycznym podejściu do klasyfikacji w zbiorze otwartym. Bezpieczne i wiarygodne modele muszą działać jako klasyfikatory otwarte - skupiłem się na tym, jak zmusić klasyczne modele, aby były świadome danych nieznanych. Poniżej opisany jest najważniejszy wkład tej pracy.

Dokładnie przeanalizowałem wybrane metody wykrywania danych nieznanych. Wykazałem, że nie ma najlepszej metody, a jej wybór powinien zależeć od badanej pary znanych i nieznanych danych. Znalazłem ograniczenia i zbadałem założenia metody EVM oraz niestabilność modeli parametrycznych opartych na wielowymiarowym rozkładzie normalnym (z ang. MultiVariate Normal distribution, MVN) tj. wykorzystujących odległość Mahalanobisa. Nieparametryczna metoda LOF oparta na gęstości w wielu przypadkach

sprawdza się lepiej, niż metoda oparta na MVN lub bazująca na odpowiedzi z sieci (z ang. logits). W literaturze niestety brakuje porównań metod typu LOF z innymi.

Sprawdzono skuteczność metod wykrywania danych nieznanych w dużej skali (np. obrazy z dużą rozdzielczością czy bazy danych z dużą liczbą klas znanych). Wiele metod jest ocenianych jedynie na zbiorach danych w małej skali, co jest niewystarczające w przypadku rzeczywistych problemów. Pokazałem, że metody oparte na odpowiedzi z sieci działają słabo, a metoda Mahalanobisa nie działa tak dobrze. LOF wydaje się być najbardziej stabilną metodą.

Zauważyłem znaczący wpływ metody ekstrakcji cech na poprawę skuteczności wykrywania danych nieznanych. Klasycznie wykorzystuje się jedynie metodę GAP (z ang. Global Average Pooling). Jednak inne podejścia mogą skupiać się na innych elementach obrazu (np. na krawędziach, wzorach lub całych obiektach), więc dla różnych par danych znanych i nieznanych, różne ekstraktory cech mogą być przydatne w separacji danych znanych i nieznanych. Zaproponowałem dobór ekstrakcji cech jako nowy hiperparametr. Ponadto sprawdziłem wpływ redukcji rozmiaru wektorów cech. Metody oparte na LOF wydają się być jedynymi metodami, przy których taka redukcja ma sens.

Pokazałem, że przy użyciu metod służących do wykrywania danych nieznanych można odfiltrować wiele przykładów ataków typu adversarial. Zalecam wybór innej strategii ekstrakcji cech niż GAP (cechy GAP są wykorzystane przez klasyfikator) - może to poprawić skuteczność wykrywania.

Wykazałem, że dobór odpowiedniej techniki rozszerzania danych (z ang. data augmentation) wpływa znacząco na efektywność metod wykrywa danych nieznanych i krzepkość sieci.

Zbadałem również problemy występujące przy wykrywaniu danych nieznanych. Przede wszystkim zbadałem problem niestabilności omawianych metod - doszedłem do wniosku, że wyniki podane w literaturze należy traktować z ostrożnością. Nieco inny stan modelu może drastycznie zmienić skuteczność metody. Ponadto wykazałem możliwą rozbieżność między przestrzenią obrazów a przestrzenią cech - tj. podobne obrazy (w naszym ludzkim rozumieniu) mogą generować odległe cechy, a całkowicie różne obrazy mogą być bliskie sobie w przestrzeni cech. Powszechnie stosowane definicje przykładów danych nieznanych bliskich (z ang. near OoD) i dalekich (z ang. far OoD) są nieprecyzyjne.

Słowa Kluczowe

Klasyfikacja obrazów – Widzenie komputerowe – Wykrywanie danych nieznanych – Klasyfikacja w gupie otwartej – Głębokie uczenie – Bezpieczeństwo i wiarygodność sztucznej inteligencji – Metody ekstrakcji cech – Ataki typu adversarial – Konwolucyjne sieci neuronowe

Contents

Acronyms

AI	Artificial Intelligence
AUC	Area Under Curve
AUROC	Area Under Receiver Operating Characteristic curve
AUPR	Area Under the Precision and Recall curve
CNN	Convolutional Neural Network
DNN	Deep Neural Network
DTACC	Detection Accuracy
EVM	Extreme Value Machine
EVT	Extreme Value Theory
FFNN	Feed Forward Neural Network
GAN	Generative Adversarial Network
GAP	Global Average Pooling
GMP	Global Maximum Pooling
GPU	Graphics Processor Unit
ID	In-Distribution
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
LSTM	Long Short-Term Memory
MLP	MultiLayer Perceptron
ML	Machine Learning
MVN	MultiVariate Normal distribution
OoD	Out-of-Distribution
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SOTA	State Of The Art
TNR	True Negative Rate
TPR	True Positive Rate

Part I

Introduction and Background

1 Introduction

1.1 Motivation

Computer vision is nowadays one of the most important areas of computer science. Although a few years ago, processing images and videos on a large scale was a real challenge, now - thanks to the rapid development of artificial intelligence (AI) - we can achieve impressive results. There are numerous applications of this technology: in self-driving cars, in automated convenience stores (without cashiers), in-camera systems used in modern cities, or security analysis in factories and construction sites - e.g., detecting if everyone is wearing on a helmet or a vest. Nowadays, we can come across such solutions in our daily life. Whether we like it or not, computer vision algorithms can make decisions for us and contribute to the safety of our lives.

We have achieved such a level of technological advancement thanks to the development of hardware (i.e., modern graphics cards) and new algorithms (i.e., the branch of machine learning called "Deep Learning"). It allowed us to create models for solving most image computer vision (and other AI) problems - often achieving better performance than humans.

Following the history of the development of deep learning image classification algorithms, we can see rapid progress. The results achieved by the algorithms are usually presented as the accuracy on popular benchmarks - like the ImageNet. The goal is to classify each image never seen before to one of the available classes - there are 1000 classes in the ImageNet. The AlexNet[123] performed 63.3% accuracy, the ResNet[87] 78.6%, the Dual-Path Networks[39] 81.4%, the EfficientNet-B7[244] 84.4%, and one of the best current models the FixEfficientNet [250] achieved 88.5%. The 90%[173] threshold has already been reached, thanks to additional approaches. The vision algorithms are usually based on Convolutional Neural Networks, which are the focus of this dissertation. However, new concepts (like the Transformers[54] or the MLP-Mixer[248]) have become more popular nowadays. We want to focus on the image classification, which is a basis for other computer vision problems.

Although the newest computer vision models achieved impressive accuracy – however, further challenges in the problem of image classification still exist. Due to the broad applications of these algorithms, improving security and trustworthiness has become the most important task for researchers.

Safety and trustworthiness are very vast terms[101]. Safety is the invariance of a classifier's outcome to perturbations within a small neighborhood of an original input. Trustworthiness ensures that the model works as expected and safely – moreover, the user should be able to understand the returned value.

There are many challenges and threats to deep learning approaches in practice. One of them is susceptibility to natural and adversarial attacks. The network can be easily fooled with special kinds of images. The goal of adversarial attacks lies in preparing new images by adding additional unique noise that forces the network to point out some class with high certainty despite the image not presenting that class. Images that do not belong to any known classes, however, the network returns high certainty results for them, can be considered as natural attacks. Another challenge is the robustness of the

network. The robustness is the ability of the network to classify similar (but not within the same distributions) images to the training examples – for instance, working with extra distortions like with rotated images, with different weather conditions (e.g., fog, rains), or obscured by some object. The robustness is also the ability to return similar outputs for similar inputs, increasing the stability of the response. It connects with the next challenge - interpretability. Modern models work like "black boxes", so it is impossible to interpret the factors influencing the outputs. However, we - people - would like to understand why the network responds in that way to get to know the principles.

All of the above can be connected to one of the most important threats, in our opinion – the close-set nature of classification. The artificial network usually is forced to choose one winning class, while there is no "I do not know" or "unknown" class. There can be serious problems when the network sees the object class for the first time. Often the classifier returns a high certainty answer to a random label (from a human perspective). When we will "open" the model by adding the ability to detect unknown data distribution, we increase the safety of the ML model.

We believe that solving the above problems lies in understanding the representations of features. Numerous experiments show that features generated by deep neural networks are not robust enough. During the learning phase of the networks, the global loss is minimized. However, usually, in the loss functions, researchers do not consider anything except correct classification. Due to the above, we do not know how the networks treat unknown or adversarial examples and how the final features are represented. The goal is to improve the ability to detect samples from unknown distribution networks without retraining the models.

The close-set nature of classification poses severe threats to the security and trustworthiness. The standard CNN models are vulnerable to such problems. The solution for the above could be allowing networks to use the "unknown" class. This new label can also be used for outputs for adversarial attacks. The methodology for this is called open set classification or the Out-of-Distribution detection technique. The problem is not new, but nowadays - when the accuracy of networks is high enough - it is a real challenge. The popular papers in this field usually suggest "one best solution", but as we showed below, proposed solutions work well only for specific cases. The problem of OoD detection is not so trivial, and the final results depend on many factors.

The typical OoD approach is as follows. There are two data sets: train and test. However, the testing set can contain both known (same classes as in the training set) and unknown (Out-of-Distribution) examples of images. The goal is to create a classifier that can deduce which images from test sets are known – based only on the training set and the classic (with close-set nature) image classification model.

Since there are no clear guidelines in literature for the optimal way to detect OoD examples, we analyzed the effectiveness of OoD in the context of the network architecture, various benchmark data sets (treated as both: known and unknown data), methods of choosing features from the network, post-processing of these features, or OoD algorithms themselves.

Our study aimed to understand how currently used Out-of-Distribution methods work, find gaps in popular approaches, and propose suitable recommendations. Finally and above all, we aimed to improve network security and trustworthiness.

1.2 The Research Problems Formulation

The classic - close-set - nature of CNNs and the fact that they are vulnerable to adversarial examples can be a real threat in real-life situations. The main motivation to undertake this research comes from an in-depth analysis of literature and noticing that detection of out-of-distribution examples can help with above issues. The area of this research is connected with many other machine-learning fields as architectures of artificial neural networks, global and local features extraction methods from the networks, adversarial attacks, understanding how the networks make decision boundaries, and the Out-of-Distribution detection techniques. The goal was to improve network security and trustworthiness by proposing new methods, improving existing ones, and researching new paths.

As the literature explains and our experiments confirmed, there is no one OoD strategy, which works well for all cases. We believe the reason for that lies in how features generated by the networks are represented. Working with high-dimensional data (usually more than 1000 dimensional) and limited examples (usually up to a few thousand per class) is a real challenge.

We researched the Out-of-Distribution detection problem in computer vision. We showed the weakness of currently used methods and pointed the direction of further improvements. First, we focused on the influence of the effectiveness of OoD and the decisions which have to be undertake - we need to choose which OoD method, which CNN architecture, which features extraction method, or which metrics to evaluate. We tested a variety of configurations, also working on large-scale data. Moreover, we analyzed the assumptions of the chosen OoD detection methods. Knowing the limits of each method is important in the context of trustworthiness. Next, we focused on feature representation showing that the proper features obtained strategy might follow to improve OoD results. We showed the mismatch between semantic and feature space - similar images (in our human understanding) can generate distant features, and images from different classes can be close to each other in feature space. We tested OoD detection methods as one of the defense approaches against adversarial attacks.

The following specific research problems are addressed:

1. **We analyzed the effect of key parameters for the OoD detection procedure, which have a stronger impact on the effectiveness.**

   We discussed, among others, the evaluation metrics, dataset benchmarks for known and unknown data, CNNs model architectures, feature extraction strategy, and differences in OoD methods.

2. **We analyzed the effectiveness of the OoD detection methods based on operating principles.**

   OoD methods can be divided into different operation modes. We checked if methods - based on multivariate Gaussian distribution, logits, probability of inclusion, or non-parametric density-based ones - have some advantages over others.

3. **We showed there is no best OoD approach.**

   We examined the effect of different distributions of unknown data, such as noise (randomly generated images) or images similar to the In-Distribution. We postulated (by being in line with the current literature) that there is no best OoD approach for all problems. We confirmed that the best suitable OoD method depends on the pair of the known and unknown sets of images. Therefore, the goal should be to choose the proper OoD strategy based on the analysis of the features extracted from DNNs.

4. **We analyzed the impact of the network (architecture and its state) as the data representation generator.**

   Various models allow building decision boundaries in different ways. It suggests that different close-set models can work differently with OoD methods.

5. **We analyzed the influence of models with different accuracy in terms of the efficiency of OoD detection methods.**

   A closed classifier is used for feature generation, so the intuition suggests that the model with higher accuracy should increase the OoD detection efficiency.

6. **We analyzed the OoD detection methods using large-scale data.**

   By large-scale data, we mean, i.e., a large number of training samples, a high number of known classes, and high image resolutions. Algorithms applied in real-life problems require working on these data types – however, they are not commonly tested in the literature. Different OoD methods can work differently in large-scale problems.

7. **We analyzed the assumptions and limitations of the popular OoD methods.**

   It is necessary to verify if the input data for OoD methods fulfilled their assumptions. Moreover, the features from the CNNs are high-dimensional (+1000D), which is a challenge. We analyzed the assumptions and limitations of two popular OoD methods: based on the Mahalanobis distance and Extreme Value Theory.

8. **We checked the influence of the feature extraction strategy on OoD detection.**

   To the best of our knowledge, other papers focusing on OoD detection problems use only Global Average Pooling as default feature extraction technique. We proposed adding a new hyperparameter to choose the proper extraction strategy. It can increase OoD efficiency due to various approaches that can focus on different components (e.g., on edges, patterns, or whole objects), so for different pairs of ID and OoD, different feature extractors can be useful in separating data.

9. **We analyzed the effect of the feature reduction on the efficiency of OoD detection.**

   The features obtained from the CNNs are high-dimensional (+1000D), which can be a problem for many OoD detection methods. We reduced the dimensions of the features to check the influence of this operation on the efficiency of the OoD detection.

10. **We analyzed the influence of image augmentation on the efficiency of OoD detection.**

    The training images are the basis for learning (by the networks) on how to build feature representation to maximize the ability to distinguish the known classes. Using the proper augmentation strategy should improve the model robustness and OoD detection, what we have verified.

11. **We analyzed the stability and reproducibility of OoD methods.**

    Most research presents results based on a few benchmarks using a few models. We observed a lack of stability and reproducibility in the literature. We checked the stability of the OoD models based on the same model architecture with similar close-set accuracy, but slightly different hyperparameters. Moreover, we analyzed and showed the behavior of many OoD approaches during the training phase.

12. **We analyzed the mismatch between images and feature representations.**

    The DNNs can generate features from images contrary to our human intuition - i.e., similar images (in our human understanding) can generate distanced features, and different images can be close to each other in feature space. We showed mismatch for many images in image and feature space, which can be the source of incorrect behavior of OoD methods. We show that this mismatch occurs regardless of models or approaches to feature extraction. This observation guided us to make special subsets that are hard or easy in the context of OoD detection.

13. **We analyzed OoD methods in terms of defenses against adversarial attacks.**

    We suggest that adversarial attacks can be distinguished from known image distribution by using OoD methods. Moreover, we noticed that classifiers are based on the Global Average Pooling. It leads us to the idea of changing the feature strategy to improve this defense approach.

Our main contributions can be formulated as the following points:

1. We showed limitations and analyzed the assumptions of the Extreme Value Machine (EVM) algorithm.

2. We showed instability of parametric models based on MultiVariate Normal (MVN) distribution, i.e., using the Mahalanobis distance in high dimensional data. This characteristic of this method is the limitation of this popular approach.

3. We showed that the nonparametric, density-based LOF approach performs better than based on MVN or logits in many cases. We observe a lack of comparison between LOF and other benchmark methods in the literature.

4. We showed the efficiency of the OoD method based on large-scale dataset benchmarks. We conclude that the OoD methods based on logits work poorly, and Mahalanobis is slightly worse compared to other methods on a large scale. The LOF seems to be the most stable method.

5. We showed that the correctly chosen feature extraction strategy can improve the efficiency of OoD detection. The standard method uses only Global Average Pooling (GAP). However, we recommend it as a hyperparameter, while various approaches can focus on different components of images.

6. We showed that reducing the size of feature vectors leads to severe efficiency deterioration for many methods. The LOF-based methods seem to be the only method we recommend using together with dimensional reduction.

7. We showed that OoD detection can filter many adversarial examples. Moreover, we recommend choosing a different feature extraction strategy than GAP - it may improve the efficiency.

8. We analyzed the problem with instability and experiments repeatability of the OoD detection methods – and conclude results in literature should be taken with caution. The slightly different model state can change the OoD method's efficiency drastically.

9. We showed the mismatch between the image and feature space - i.e., networks can generate features from images contrary to our human intuition. We conclude that the common usage of near and far OoD examples definitions is inaccurate.

10. We showed there is no best OoD approach, and it should depend on the tested ID and OoD pair.

11. We showed the importance of proper data augmentation techniques in OoD detection problems and robustness.

    The main thesis of this dissertation is that the safety and trustworthiness of AI models can be assured only when the models can distinguish between known and unknown data - therefore, we showed and thoroughly researched the Out-of-Distribution detection techniques.

1.3 Document organization

This work is organized as follows. Next, chapter 2 is a review of basic terms and techniques used in deep learning. The end of this chapter is focused on describing in detail the main methods used in our research. We focused - among others - on CNNs architectures, feature extraction methods, out-of-distribution techniques, adversarial attacks, and challenge problems occurring in current deep learning algorithms. The theory introduced in these sections should be sufficient to understand all issues raised in this dissertation.

We described in detail our research in chapter 3. We presented achieved results. We argue that there is no universal method to solve the Out-of-Distribution detection problem and propose recommendations based on our experiments.

Chapter 4 is the summary of our work. We present here our final thoughts and future works.

2

Background

This chapter is divided into three parts.

The first part, presented in section 2.1, is focused on a general review of deep learning types of networks, such as classic NN, CNN, GAN, or RNN. Next, we presented standard techniques necessary to successfully design and train modern models dealing with issues such as optimization, regularization techniques, and others.

The second part is focused on CNNs for computer vision - see section 2.2. We described popular benchmark datasets, layers used in CNNs, and popular model architectures. We also illustrated specific computer vision issues such as object detection, architectures for mobile (and edge) devices, contrastive learning, or image retrieval. We also described popular feature extraction methods used later in our research.

The last part was the starting point of our research. We showed the challenges, which in our opinion, are one of the most important in machine learning nowadays. These are Out-of-Distribution detection, defenses against adversarial attacks, and interpretability and trustworthiness of AI. All of our research is strongly related to the above challenges.

2.1 Review of Deep Learning

Technically, deep learning is just using artificial neural networks with more than one hidden layer. Nowadays, networks contain millions of neurons within even thousands of layers, and each layer is specifically designed for a particular problem. The key idea of this type of machine learning algorithms is extracting high-level features from raw data - in contrast to classical approaches, where engineers prepare features by hand. Deep learning allows working with complex data such as images, music, video, or texts wherein during the training phase, the network "learns" how "dog", "cat", "poem", "rock song", or "swimming action" looks. Based on thousands or millions of examples, these networks can extract key features from input data to "understand" them.

There are many problems where these networks can be used. The basic is a classification problem. However, many other problems also exist (also dedicated to the specific input data type), like image retrieval, image generation, object detection, image segmentation, text generation, text translation, or market stocks prediction. Moreover, for each type of data, different types of layers can be used - for instance, convolutional layers for images, transformers for text, or LSTMs for time-series data.

Thanks to the rapid development of technology, and hence more available input data and better graphic(or tensor) cards, the networks can be bigger and trained on more data in less time. The deep networks can achieve the best results in nearly all benchmarks datasets for all kinds of data. That is the reason for their popularity.

2.1.1 *Types of Deep Architectures*

2.1.1.1 *Classic Neural Network*

The history of artificial neuron networks is long. Already in 1943 [155] McCulloch and Pitts proposed a method of how neural networks could work and modeled a simple one

using electrical circuits. Later in 1958, Rosenblatt introduced an idea of a perceptron [193] - a basic unit similar to the nowadays artificial neuron. However, this simple construct was successfully used in some applications, e.g., ADALINE – researchers can not overcome many challenges like the XOR problem. The first breakthrough was developed backpropagation algorithm[200] (used until today) then started joining artificial neurons in networks creating feedforward neural networks (*Feed Forward Neural Network* (FFNN)) / multilayer perceptron (*MultiLayer Perceptron* (MLP))) with two or more layers. This type of network can solve complex problems, and it is used nowadays too. There were introduced many other variants of neural networks like Hopfield network [94], Boltzmann machine [2], or self-organizing map [121] – however, we will not focus on these.

Although FFNNs work quite well, they were not commonly used because of their limitations. They need much input data to successfully learn patterns and computer resources to speed up the learning process. Because of that, the other algorithms were more often used, like SVM or decision trees. This changed in the 2010s when graphics processor units (*Graphics Processor Units* (GPUs)) were well developed, and they were available for large datasets. The era of deep learning has been started [129]. Technically, a deep neural network is any network with more than one hidden layer, but in practice, quite new network architectures for different types of input data have occurred. Nowadays, we use Convolutional Neural Networks (mostly for images data), Recurrent Neural Networks (mostly for text and time-distributed data), Generative Adversarial Networks (mostly for generating new data), or Reinforcement learning (mostly in robotics). The standard FFNN is still used, for example as a classification part in Convolutional Neural Network (*Convolutional Neural Network* (CNN)) or in Transformers [255].

$$y_k = \varphi\left(\sum_{j=0}^{m} w_{kj} x_j\right) \tag{2.1}$$

The artificial neuron used in FFNN is a simple block. Each input x is multiplied by the corresponding weight w and summed up. The result is passed through a non-linear function known as an activation function φ (see more in section 2.1.2.3.1). Without the activation function, the whole network could be transformed into a single neuron. The FFNN is based on layers that are based on these artificial neurons. The output for neuron k can be presented as in equation 2.1.

2.1.1.2 *Convolutional Neural Network*

The main work described in this document is based on Convolutional Neural Networks, so these are described in much more detail in the separate section 2.2.

2.1.1.3 *Generative Adversarial Network*

Generative Adversarial Network [72] was designed in 2014 by Ian Goodfellow and his colleagues. They were a real breakthrough in *Machine Learning* (ML), allowing generate completely new images (fake ones - never seen before) based on a set of real examples. Although the generated images were low resolutions initially, it is now possible to create images that are difficult to distinguish from real ones.

The classic approach is a game between two models: a generator and a discriminator. The generator is responsible for generating new images using an initial vector (usually Gaussian noise) as input. The discriminator should guess if the input of this model is a real

or fake image. Two models learn simultaneously, competing with each other. Both models can improve themselves thanks to the loss calculated from the discriminator's output. Over time, the generator can create better and better models, and the discriminator is more strict in judgment. Usually, both models are CNNs where the generator uses transposed convolution [148]

One fo the first significant improvement for the standard *Generative Adversarial Network* (GAN) was DCGAN (deep convolutional)[178], which use CNNs for both submodels. This paper's authors proposed a number of improvements - the mains are: using Leaky ReLU in the discriminator, replacing pooling operations with convolutional stride, eliminating fully-connected layers, and using batch normalizations.

The important paper was Wasserstein GAN[6] where was proposed a new method to calculate the loss using Wasserstein distance. This method makes gradients smoother, so in consecrations, it can improve stability when training the model.

The BiGAN[51] (or AliGAN[57]) paper kept - in the generator part - the standard mapping from latent representation to the image but also added a new submodel - the encoder. The encoder makes inverse mapping - from data to the latent vector. The motivation is to create a network that can learn rich representations as in unsupervised learning applications. The discriminator should guess its output based not only on an image but also on the latent vector. For real images, this vector is generated by the encoder.

The next improvement was focused on increasing fake image resolutions. The authors of ProGAN[114] showed how to generate big resolution images by splitting generating into phases. First, the 4x4 pixels images are generated. Next - based on these images - the bigger ones are produced with a resolution of 8x8. Next, the whole process is repeated - up to achieve a 1024 x 1024 pixels resolution. Each phase adds new convolutional layers to the generator to increase resolution and to the discriminator to decrease them. Increasing image resolutions allow for adding new essential features step by step - for instance, for generating a new face image, the shape or the color of hair are produced in the first order, and features like eye color or freckles in the final phases. Based on this idea, the authors also proposed StyleGAN [113] showing how to control specific attributes on images like smiling or hair color using adaptive instance normalization (AdaIN). Next, solutions like BigGan [23], and later BigBiGan [52] focused on improving particular parts of the image using Self-Attention Module (convolutional layers with 1x1 filters and softmax operations), allowing the removal of unwanted artifacts. Consequently, make it possible to generate even more realistic images.

The GANs can also be used for transforming one image into another. There are a few influential papers in this area. The CycleGAN [291], which allows the changing style of images keeping content using two sub-GAN models - first transforming the image to the new one with the unique style and next by repeating this operation making cycle. The Pix2Pix [105] allows mapping input image to output image, for instance, edges to full images, or synthesizing photos from label maps. To the discriminator is provided both source and the target image (fake or real), and the goal is to determine whether the target is a believable transformation of the source image. Another model is SPADE[169] allows control style and content using semantic image synthesis. The authors used a spatially-adaptive normalization to perform generating realistic photo images.

2.1.1.4 *Recurrent Neural Network*

The classic artificial neural networks can not work well with data changing over time like texts, videos, market stock charts, or music. To be able to analyze these types of data,

the networks have to have memory blocks. The memory is used to remember the past to interpret the currently analyzed part of data with a proper context. The way to add this memory block is recursion. The network's output is reused as input. The network itself decides which essential features referred to in the past are worth keeping, and that data is called the "hidden state". The networks which use this technique are called Recurrent Neural Networks.

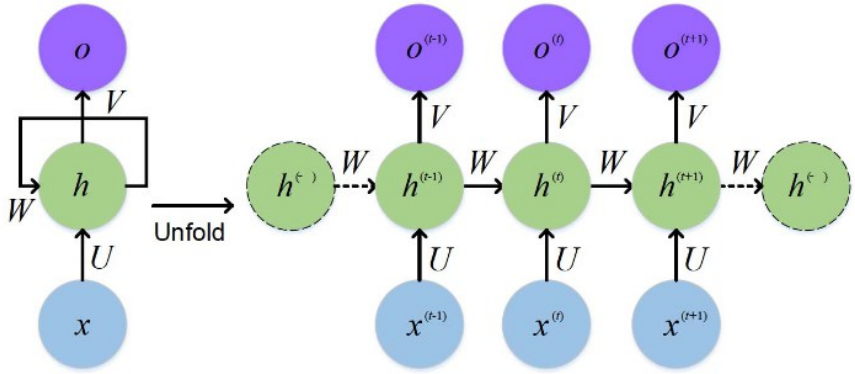


Figure 2.1: RNN - image from [60]

2.1.1.4.1CLASSIC RNN The RNN is well known for detecting [200]. The model is looped using three types of layers: the input layer (x), the hidden layer (h), and the output layer (o). Because the model works with data changing over time, each input is denoted with time (t). The hidden state and the output can be calculated as shown in equation 2.2. The W, U, and V are trainable weights, and b is the bias. The σ is an activation function – usually, the sigmoid is used. The RNN can unfold as shown on figure 2.1. The unfolded version of the RNN easily explains how to train that kind of network - exactly the same as the classic one - layer by layer using the backpropagation technique.

$$\begin{aligned} a^{(t)} &= Wh^{(t-1)} + Ux^{(t)} + b_1 \\ h^{(t)} &= \sigma(a^{(t)}) \\ o^{(t)} &= Vh^{(t)} + b_2 \end{aligned} \tag{2.2}$$

We do not have to use all outputs. The RNN problems can be divided as follows: (1) one-to-one (vanilla neural network) - e.g., images classification, (2) one-to-many - e.g., generate captions for images, (3) many-to-one - e.g., Texts classification, or (4) many-to-many - e.g., texts translation. Depending on the above dividing, we can use a different number of output layers (usually all or only last).

The RNN can have a problem with a long memory (keeping in a hidden state distant in time information). The reason for that is the vanishing gradient problem. Because of that, the RNN often makes decisions based mostly on a few last pieces of data.

2.1.1.4.2LSTM The Long Short-Term Memory[93] is a type of RNN. The authors tried to solve the vanishing gradient problem of classic RNN by adding an additional hidden state denoted as c for long memory. The idea is to use the additional gates (see figure 2.2) to control what could be forgotten (forget gate f), what should be remembered from the current input (input gate i), and what should be returned as output (output gate o). The

equation 2.3 explains how to calculate each gate. The W and U are trainable weights, and b is the bias, the $*$ is element-wise multiplication.

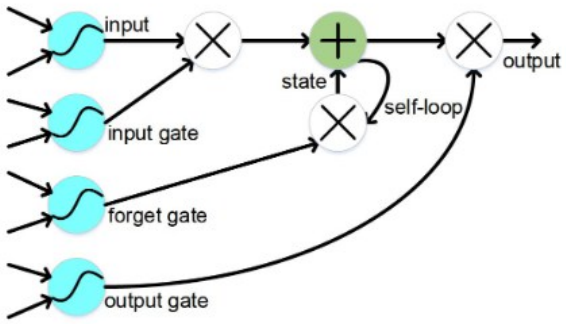


Figure 2.2: LSTM - image from [60]

$$
\begin{aligned}
i^{(t)} &= \sigma(W_i h^{(t-1)} + U_i x^{(t)} + b_i) \\
f^{(t)} &= \sigma(W_f h^{(t-1)} + U_f x^{(t)} + b_f) \\
o^{(t)} &= \sigma(W_o h^{(t-1)} + U_o x^{(t)} + b_o) \\
c^{\tilde{(t)}} &= tanh(W_c h^{(t-1)} + U_c x^{(t)} + b_c) \\
c^{(t)} &= f^{(t)} * c^{(t-1)} + i^{(t)} * c^{\tilde{(t)}} \\
h^{(t)} &= o^{(t)} * tanh(c^{(t)})
\end{aligned}
\tag{2.3}
$$

2.1.1.5 *Transformer*

A Transformer is a breakthrough model primary for natural language processing introduced in a paper called "Attention is All You Need" [255]. The Transformer is based on an Attention mechanism (primarily introduced in [12]) that allows the network to remember long memory, usually in a better way than RCN, including LSTM and GRU. The Transformer does not process input sequences but all inputs simultaneously. It does not need a labeled dataset because it learns the input by absorbing the correlations appearing in the analyzed data. A good survey of the attention and the Transformer is presented in [33].

The Transform model is based on encoder-decoder architecture and self-attention. The Encoder maps input data to the representation vectors. These vectors with outputs from the previous step are passed to decoders, which generate a new output. The Transformer can use many encoder-decoders, where each can learn new attention representation. It uses a Multi-Headed Attention based on self-attention, where Query and Key are split for heads. See the figure 2.3 to see details.

The Transform is the base for numerous other models. The most popular solutions are BERT[49] (also based on ELMo[172]), GPT[179], GPT-2[180], GPT-3brown2020language, ROberta[146], ALBERT[127], BART[134], XLNet[277] and many others. Each of them uses a different Encoder, Decoder, or only one of them with different objectives and modifications in architecture. The detailed comparison can be found here[143].

The Transformer is also used in computer vision recently[115]. We can split two directions pure attention models like key ViT[54], SASA[183], DeiT[251], Swin Transformer [147] and Convolution + Attention Models like CvT[268], BotNet[228], CoAtNet[47]. Although ViT

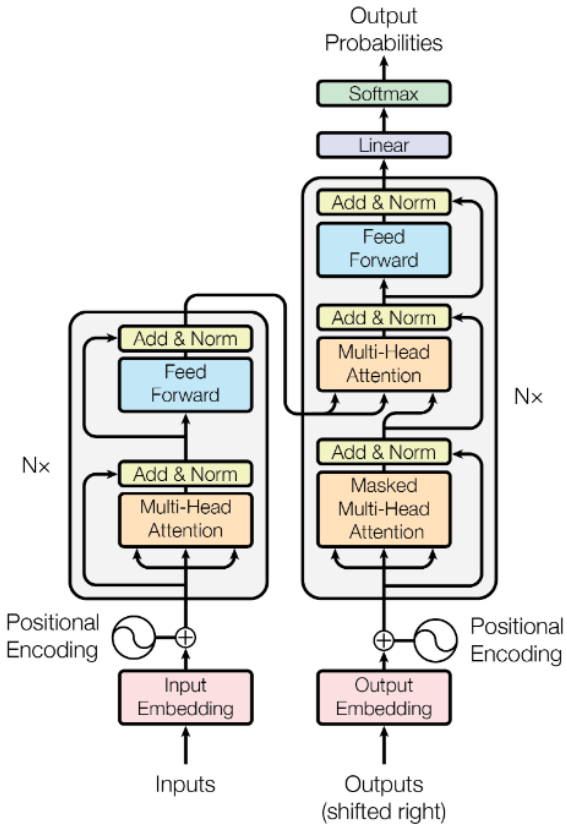


Figure 2.3: The Transformer - image from [255]

has shown impressive results with huge JFT 300M training images, its performance still is behind CNNs in the low data number regime.

2.1.1.6 *Reinforcement learning*

Reinforcement learning is one of the more vast areas in machine learning. Due to the ability to learn nearly any task based on trial and error techniques, machines, robots, and bots are allowed to solve human-level tasks. These techniques were used in such popular (even from mainstream media) solutions as overcoming humans in games like GO[221], or StarCraft [257] or developing algorithms for self-driving cars[77].

The idea of reinforcement learning is as follows. An agent (in which we want to learn new things) can perform actions in the environment. The environment rewards each action and informs the agent about new observations. Thanks to the trial and error technique, the agent becomes better and better without using any additional data. The agent can be based on Value Function and Policy. The policy describes the agent's decision-making process - for each state, it refers to the action that the agent should perform in that state. A value function represents a value for an agent to be in a certain state or, in other words, it describes the expected reward from a given state. The RL methods can be based on optimization of the policy (the policy-based RL), the Value Function (the value-based RL), or both (the actor-critic RL). The key papers in this area are as follows.

One is Deep Q-Learning[159][85][264], which tries to use deep learning to model the state-action-value function Q(S,A) (mimic Q-table) based on the Bellman equation. Another is Policy Gradient. In contrast to the Q Learning approach, where the goal is not to learn the state-value or the state-action-value function but directly learn the parametrized

policy – the Policy Gradients approach seeks to optimize the policy space directly. The key papers for Policy Gradient are [160][212][211][213][82]. Deep Deterministic Policy Gradient[220][64] is an algorithm that learns both Q function and policy. It uses off-policy data and the Bellman equation to learn a Q function and then uses a Q function to learn policy. Distributional Reinforcement Learning [15][46] is where the main idea is to work directly with the full distribution of the random return received by the reinforcement learning agent rather than with its expectation. In contrast to Model-Free RL[162][126] is approach, where optimal behavior is obtained by learning a model of the environment.

There are many fields in RL, which are key to well understanding the present models, such as: Exploration[95][27], Transfer and Multitask RL[201][107], Hierarchy[256], or Memory[207].

2.1.1.7 *Other Architectures*

Deep learning is a vast area. There are numerous methods for specific tasks, and in this document, only selected ones are described. Nevertheless, some techniques are noteworthy to point out.

Autoencoder[122][14] is a fundamental approach used in many applications like dimensionality reduction, information retrieval, image processing, or machine translation. They are a type of unsupervised learning network. The idea is to encode the input to a latent space vector, which can be decoded to the same input again. Next, it is possible to operate on this latent space vector.

Back the well-known concept was presented in the paper called MLP-Mixer[248], where authors based only on MLP layers created a model that achieved results comparable to the state-of-the-art methods. First, the input image is split into image patches (presented in ViT[54]). Next, each of them is passed by MLP into features, next the set of MixerLayer is used, where each layer uses layer normalization, transposing, skip-connections, and main parts - the two MLP, finally the output is passed to the classifier.

Worthy to note are Capsule neural networks[202], which focus on improving CNNs by adding special structures called "capsules". These capsules allow networks to be more stable with perturbations. For example, they can "understand" the "Picasso problem" - when the image contains all the right parts but with an incorrect spatial relationship. The classic CNNs ignore this relationship.

A graph neural network is a type of network designed for graph data structures[208] – the complexity of graph data challenges existing machine learning algorithms. The primary methods in this area are DeepWalk[171] and GraphSAGE[83]. The good survey is provided by [270].

2.1.2 *Popular Techniques Used in Deep Learning*

2.1.2.1 *Optimization Techniques*

The networks can learn thanks to tuning their weights. The typical learning process - called supervised learning - can be split into a few parts. First, the network has to process inputs. These inputs are transformed into features and classified into predicted labels. Beyond the training examples, the correct labels are also available. They are used to calculate loss between predicted and proper labels that are made by the network. Next, these errors can be back-propagated to each artificial neuron in the network, so as a result, there is known how each weight should be corrected. However, it is not so simple because adjusting the

network, for one example, can distort others. To deal with this problem the optimization techniques[196][226] are used. An optimizer is responsible for modifying weights in the network to finally find a global minimum loss (minimal average errors for all examples) to provide the most accurate results possible.

2.1.2.1.1GRADIENT DESCENT The Gradient Descent is the most basic optimization algorithm. It modifies weights(θ) by computing the gradient of the cost function ($J(\theta)$) on the whole training set. The cost function is from the value of the loss function, which is back-propagated to all layers one by one. The algorithm is presented in the equation 2.4.

$$\theta_{new} = \theta_{old} - \alpha * \bigtriangledown J(\theta_{old}) \tag{2.4}$$

The α is the learning rate. It controls how quickly the model adapts to the problem - how much the weights should be changed to minimize the cost function. If this parameter is too high, the network can never reach global minima due to unexpected behavior. If the parameter is set to low, then the network can not reach the minima in a reasonable time.

The main disadvantage of this approach is the fact that it cannot reach the global minimum (may trap in the local minimum), and working with a huge dataset is, in practice, impossible because it works to the whole dataset at once.

2.1.2.1.2STOCHASTIC GRADIENT DESCENT One of the most popular techniques of optimization weights is the Stochastic Gradient Descent. The idea is simple: modify the weights based on single input (not all at once like in the Gradient Descent). The variant of this method is the Stochastic Mini-Batch Gradient Descent, where inputs are joined in batches, and the weights are modified based on loss from each batch. The inputs for batches are randomly picked, making the whole process more irregular – for each batch, the network has to fit onto a little other data. The SGD allows modification weights more frequently with high variance and fluctuations in loss functions. In consequence, it can find a new minimal and reach the global one.

2.1.2.1.3MODIFICATIONS OF SGD It is possible to improve algorithms based on the Gradient Descent. One of the popular techniques is adding a momentum term. See equation 2.5. The parameter γ is usually set to 0.9. The momentum helps reduce variance and soften convergence. It accelerates convergence in the right direction and reduces fluctuations.

$$\begin{aligned} V_{new} &= \gamma v_{old} + \alpha * \bigtriangledown J(\theta_{old}) \\ \theta_{new} &= \theta_{old} - V_{new} \end{aligned} \tag{2.5}$$

The extension of the above idea is using Nesterov Momentum (see equation 2.6). It is similar to traditional momentum, except that the update is performed using the predicted update's partial derivative rather than the derived variable's present value. The traditional momentum can be a good method, but if γ is too high, the algorithm may not meet local minima and may continue to grow – the Nesterov help with that.

$$\begin{aligned} V_{new} &= \gamma v_{old} + \alpha * \bigtriangledown J(\theta_{old} - \gamma v_{old}) \\ \theta_{new} &= \theta_{old} - V_{new} \end{aligned} \tag{2.6}$$

There is possible to change the learning rate during training progress - this technique is called Learning Rate Schedules. The idea is to reduce the α according to a predefined schedule. If we are close to finding the global minimum, there is a good idea to decrease the size of the "steps" to keep reducing total loss even more. There are varieties of possibilities. We can do that considering: [137] time-based decay, step decay, exponential decay, or cosine decay.

2.1.2.1.4ADAPTIVE GRADIENT DESCENT ALGORITHMS There are a series of optimizers where the learning rate adapts to the data. The Adagrad [56] performs larger updates for more sparse parameters and smaller updates for less sparse parameters. However, it can be a problem with the monotonic learning rate – working with deep neural networks can lead to being too aggressive and stopping too early. The Adadelta[282] and RMSpropcite[125] aim to reduce the aggressive monotonically decreasing learning rate. The Adam[118] is an extension of the RMSProp optimizer by adding the momentum.

The Adam is usually faster than SGD. However, many researchers willingly use SGD with momentum, arguing that for a longer training time, it can converge better. A good comparison with practical use in deep learning is here [290].

2.1.2.2 *Regularization Techniques*

Regularization techniques are all methods that are designed to make the network more generalized. The network can start overfitting during the learning process, allowing it to "remember by heart" the inputs and outputs. It is a bad situation. Learning aims to generalize the training input data to create the overall model, which can work well on new - never seen before - examples. Numerous techniques are used to do that, for example, data augmentation, where we artificially add new input data, L1 or L2 regularization, where we do not allow the network to set a high weight in the network, or dropout, where we do not allow the network recognize classes by based on the same features. There are detailed descriptions below for chosen regularization techniques.

2.1.2.2.1L1 AND L2 REGULARIZATION L1 and L2 normalization are some of the most popular regularization techniques. The idea is to add an additional loss that keeps a network's weights as relativity small ($\ell_{total\_loss} = \ell_{loss(y,\hat{y})} + \lambda * \ell_{regularization\_term}$). Keeping them small allows the model to be simpler and fight to overfit. The goal is penalties for large weights because, in another case, the model is likely to learn the statistical noise in the training data. In that case, the model works unstable on new unseen data and becomes sensitive to changes to the input. Using the L1 or L2 normalization, we need to balance the model's final accuracy and low complexity - we can control it by setting the λ (often, it is as called the regularization rate parameter).

To calculate the L1 term, we need to sum the absolute values of the weights ($L1(w) = |w_1| + |w_2| + ... + |w_n|$). To calculate the L2 term (also known as weight decay), we need to sum the squared values of the weights ($L2(w) = {w_1}^2 + {w_2}^2 + ... + {w_n}^2$). For deep learning models, usually, the L2 norm is chosen. That is because L1 tends to set wights to be zero, while L2 does not - because the curve for L2 becomes flat near zero. In most cases, we do not want to reduce features (set some weights to 0) but keep features codependent.

2.1.2.2.2DROPOUT A dropout[229] is a technique where some neurons do not take part in a training phase - to be more precise, randomly selected neurons are ignored. In practice, a part of the input values is randomly set to 0 - usually, a parameter can be set to adjust

the probability that some inputs can drop out. The key here is not allowing the network to choose a limited number of neurons, which are crucial to successfully classifying. In that case, these neurons become too narrow a specialization. These important neurons may be dropped out during training, and in that case, the network should still deal with it. Thanks to the dropout technique, the importance of the neurons are more or less equally divided into all neurons. Hence, the network is more stabilized. In the testing phase, all neurons are set as activated.

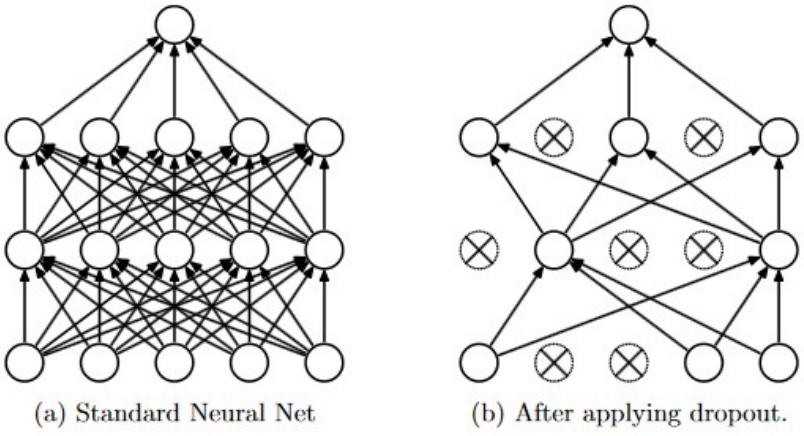


Figure 2.4: The dropout technique - idea from [229]

Nowadays, the dropout is not so popular to use. First, this technique effectively works with multi-weight dense networks, while nowadays, the classifier part on CNNs usually has only one layer - thanks to using techniques like global average pooling. Second, this technique does not work so well with convolutional layers. It happened because convolutions usually have fewer parameters. It can be important too to keep the most significant values relations for convolutions.

2.1.2.2.3DATA AUGMENTATION Data augmentation[218][170][170][219] is a technique that allows an increased number of training examples by modifying the already available samples. The key is changing the image so that the meaning of the image will not be changed, but the pixel values will - for the model perspective, it will be a new sample. The most straightforward ideas are rotating, flipping, scaling, cropping, or modifying color space by manipulating specific RGB channels. Thanks to this technique, the network during the training phase can see more differential examples, therefore, can better generalize each class representation. Well-used data augmentation can increase final classification up to a few percentage points. However, each modification should be well thought out. When the modification is too strong, the networks probably would not be able to focus on the most important features, and then this technique can even harm.

There are also more advanced techniques. One of them is mixing images[284][279] - get two images and mix them together by using proper proportions, usually 1:1. The important thing is that when we choose two different classes, the true label also should be changed, respectively. Another method is random erasing, where parts of images should be obscured - usually by changing the selected pixels to the value 0 or 1. The next method is using GAN or Neural style transfer to generate modifying or completely new images. We can also add some noise - random or even adversarial ones.

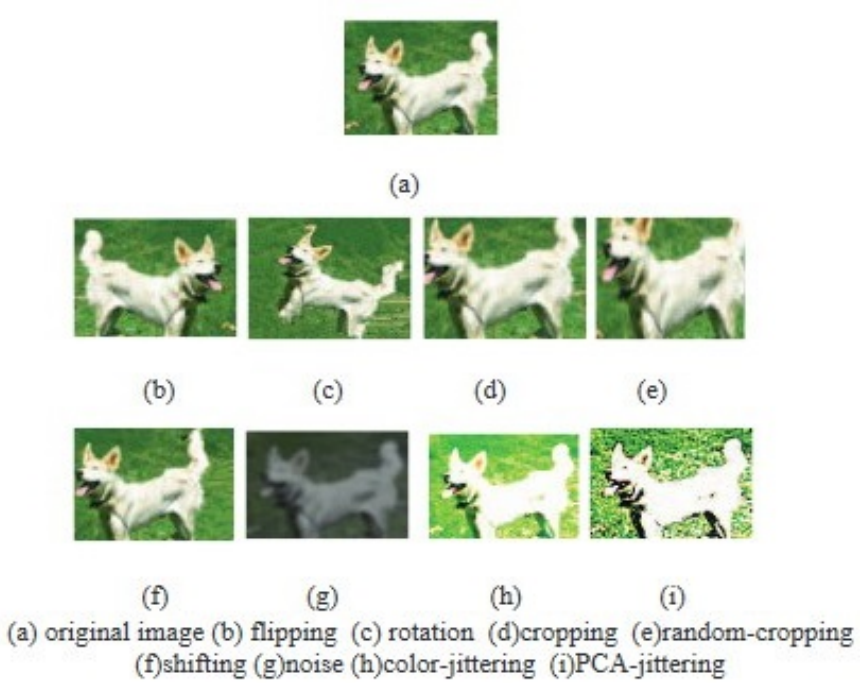


Figure 2.5: The example of data augmentation from [218]

Data augmentation is a successful regularization technique that can easily increase robustness and the ability to generalize the networks. The popular frameworks for deep learning have already implemented modules for basic methods.

2.1.2.2.4NORMALIZATION LAYERS The deep networks can be exposed to a problem of internal covariance shift. The models needs to constantly adapt to data distribution that changes in subsequent network layers. These layers have to adapt to a different distribution of input data values, which makes the learning process slower. Moreover, the gradient points to a way of changing parameters assuming that other layers don't change. In fact, we update all layers simultaneously, so the training is constantly chasing a moving target. The solution for this can be the normalization of input data for each layer. The batch normalization[104] is one of the first papers which suggest a solution for the above problem. Moreover, this approach makes input more smooth, so it points to the theory of "Loss and Gradient Smoothening". This theory assumes that by using "gradient descent", it is easier and faster to train equally distributed features.

$$\begin{aligned} \mu &= \frac{1}{n}\sum_{n}^{i=1} x_i \\ \sigma^2 &= \frac{1}{n}\sum_{n}^{i=1}(x_i-\mu)^2 \\ \hat{x}_i &= \frac{x_i-\mu}{\sqrt{\sigma^2}} \\ \hat{y}_i &= \gamma\hat{x}_i + \beta \equiv BN\gamma,\beta(x_i)) \end{aligned} \tag{2.7}$$

Batch normalization is a technique of normalization by subtracting the mean and dividing by the standard deviation calculated from mini-batch examples. The formula is presented in the equation 2.7. Moreover, there are additional learnable parameters γ

and β that are allowed to move and scale data to match values with data properties and make better predictions. The batch normalization is a powerful technique that speeds up the training process, makes input data well (Gaussian) distributed, makes the training phase more stable, and regularizes the network. The power of this technique is the fact that by using the initiation of network parameters randomly and allowing train only γ and β parameters, the network still can be well trained - it is well demonstrated in the paper "Training batchnorm and only batchnorm: On the expressive power of random features in cnns"[62]. On the other side, the authors of [66] critic this method and propose substitutes. However, the community still willingly uses BatchNormalization.

The batch normalization contributed to the uprising of other normalized methods as well. These other strategies are usually used when the classic BatchNormalization can work badly - i.e., for recurrent neural networks, when small mini-batches are used, or for specific tasks such as image generation. The popular technique is Layer Normalization[9], where authors focus on using it in recurrent neural networks to establish the hidden states. In contrast to Batch Normalization, this method does not become dependent on mini-batches, which was problematic in RNNs. The next one, Weight Normalization[203], allows decoupling the length of weight vectors in the deep models from their direction, consequently improving optimization by using popular optimizers like SGD. The alternative for Batch Normalization may be GroupNormalization[269], where authors divide channels into groups and normalize within each group using the mean and variance. This strategy is beneficial when the mini-batch size can not be large due to limitations (i.e., memory limit). Instance Normalization[254] is a technique where normalization is applied for each channel for each sample independently - handy for models for image generating.

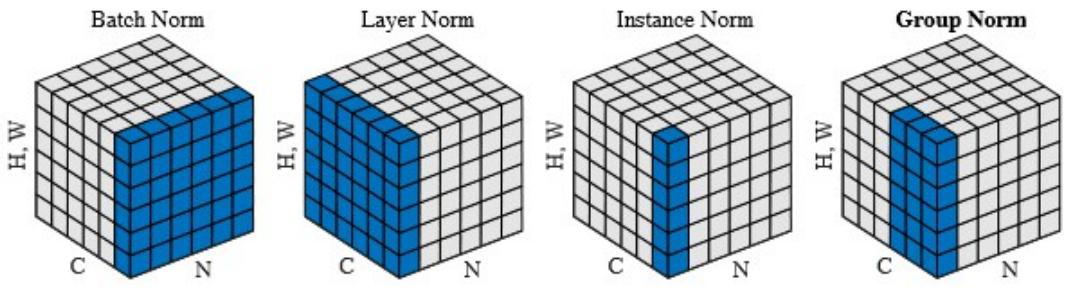


Figure 2.6: The normalization methods from [269]. The blue pixels are normalized by the same mean and variance.

2.1.2.3 *Activation Functions and Weight Initialization*

2.1.2.3.1ACTIVATION FUNCTIONS Activations functions are a key element in each artificial neural network. They transform and shape the final output of each artificial neuron. Using non-linearity functions allows the networks to compute non-trivial problems. There are many kinds of these functions divided depending on empirical performance, range, or possibility to continuously differentiable. The most popular activation functions are sigmoid, tanh, ReLU [163], Leaky ReLU [150], Swish [182], and Softmax. Their functions are pretested in the table 2.1.

[232] presents summary of them. The sigmoid and tanh functions used to be prone to the vanishing gradient problem while using them – nowadays are not commonly used. Moreover, the sigmoid function can "zigzag" during training because it is not zero

Name	**Function -** $f(x)$	**Derivative -** $f'(x)$
sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	$f(x)(1-f(x))$
tanh	$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	$1-f(x)^2$
ReLU	$\begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$	$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$
Leaky ReLU	$\begin{cases} 0.01x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$	$\begin{cases} 0.01 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$
Swish	$\frac{x}{1+e^{-x}}$	$\frac{1+e^{-x}+xe^{-x}}{(1+e^{-x})^2}$
Softmax	$\frac{e^{x_i}}{\sum_{c=1}^{C} e^{x_c}}$	-

Table 2.1: Commonly used activation functions

centered. The ReLU is the most popular choice for deep-learning algorithms. However, there can appear a problem called the "dying ReLU problem". Due to this problem, some modifications exist. The Leaky ReLU can be one of them with a simple solution. For really deep networks, the Swish function is willingly used. The Softmax function is used to estimate final predictions in classification tasks and the Transformers layer.

2.1.2.3.2WEIGHT INITIALIZATION The initial weights of the artificial neural networks should be random. Historically[73] a Gaussian or uniform distributions were used with small ranges. The initial values have an effect on the optimization procedure and on the ability of the network to generalize. The more sophisticated techniques which allow a "good start" was introduced. These new methods are slightly more effective during the training networks phase. They are designed based on the structure of the model, mainly based on used activation functions. For example, for sigmoid or tanh functions, the Xavier Weight Initialization[71] is used, and for ReLU-like functions, the He Weight Initialization.

Nowadays, the He Weight Initialization is often the default option. Here the weights are the Gaussian probability distribution - $G(\mu = 0, \sigma = \sqrt{2/n})$ where n is the number of inputs to the node.

2.2 Convolutional Neural Networks for Computer Vision

2.2.1 *Introduction*

Convolutional Neural Networks are primarily designed for the image classification problem. Nowadays, they are used in nearly all problems in computer vision and even for other types of data. The main idea is to transform the image from the pixel distribution to feature distribution using convolutional operations. Each layer transforms the distribution

more and more to be more separable by the given object classes. As a result, they generate high-level features from the input image, which can be easily used for the final classification.

2.2.2 *Popular Images Datasets*

There are many image datasets available publicly. However, only a few of them are used as a benchmark for image classification tasks. The most popular is the ImageNet 2012, which even nowadays is the main challengeable set among others. Nearly all best CNN models (or Transformers lately) are pre-trained on this dataset. Nowadays, there are opinions that the ImageNet is too small for modern solutions [231]. The community also found many lapses in it [17]. Due to the above, modern state-of-the-art models often use more extensive sets for training and only retrain themselves to the ImageNet 2012. On the other side, there are also small datasets like the MNIST, the SVHN, or the CIFARs used to quickly prototype new ideas or perform analysis in chosen areas in CNNs.

2.2.2.0.1MNIST The MNIST is a small dataset that contains 70.000 monochromatic images split into two subsets, "train" and "test" in a ratio 6:1. These images represent handwritten digits split equally into 10 categories. The size of each image is 28x28 pixels. Nowadays, it is not commonly used for simple tests due to small training times compared to other sets.

2.2.2.0.2SVHN The Street View House Numbers - SVHN is a "more complex" MNIST dataset. The images are colored, with bigger sizes (32x32 pixels), and they are from the real world. There are 73.257 digits for training, 26.032 digits for testing, and 531.131 additional ones. This dataset is not so commonly used. However, we used it in our experiments.

2.2.2.0.3CIFAR The CIFAR are two datasets the CIFAR-10 and the CIFAR-100 contain 60.000 images split respectively into 10 or 100 classes. There are two subsets, "train" and "test" split in a ratio 5:1. All classes are excluded in both datasets. The images are colored (RGB) with 32x32 pixels size. Humans can easily recognize the classes. For example, the CIFAR-10 contains the following categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The datasets - especially CIFAR-10 - are commonly used for prototyping new ideas.

2.2.2.0.4IMAGENET As mentioned before, the ImageNet [48] is the most popular benchmark dataset for image classification tasks. There are many versions of it. However, the ImageNet-2012 is the main one. All important models present their results with this dataset. The datasets contain more than 14 million images split into 20.000 categories. However, we focused only on data from the annual ImageNet Large Scale Visual Recognition Challenge (ILSCRC), where "only" 1.000 labels are used. To better assign images, the WordNet schema to categorize objects was used. The images are colored with high resolutions. In this work, we used two versions of this dataset from different years of the ILSCRC - the ImageNet-2012 as the main one and the ImageNet-2010 with chosen, excluded classes.

2.2.2.0.5PLACES365 The Places365[289] is an images dataset containing more than 10 million images comprising 400+ unique scene categories. The images are colored and high-resolution. The dataset contains three macro-classes: Indoor, Nature, and Urban. It is well-diversified - for instance, the class bedroom contains images from spare bedrooms,

teenage bedrooms, or romantic bedrooms, or the class coast can split into rocky coast, misty coast, or sunny coast.

2.2.2.0.6IMAGENET-O The ImageNet-O[92] is a specially designed dataset containing images different from the classic ImageNet. The images can be traded as natural adversarial images due to the high certainty of the returned results from popular CNN models. The authors' goal was to build datasets for challenging the models' robustness and Out-of-Distribution methods.

2.2.2.0.7JFT-300M There are many controversial with the JFT-300M [231] dataset. Over 300 million images allowed achieved great results by networks. However, Google (the authors) did not make it publicly available. The results based on this dataset are not reproducible. However, the community trusts Google enough to believe that set exists. The company shared the network's weights, which somehow proved the effectiveness of training models on that large number of images.

2.2.3 *CNN Layers*

2.2.3.1 *Convolutional Layer*

The following description is based on one of our paper[238]. The convolution operation is a central block for all CNNs. It allows extracting low-level features from images and using them later to recognize a specific object in mages. Thousands and sometimes millions of such operations have to be performed to classify an image into a given class. They are stacked together in layers allowing recognition of high-level features: starting from detecting edges, through textures, patterns, parts, and ending with objects. All of these features are propagated forward layer by layer to the classification part in CNNs. As a result, they can correctly classify images.

For a given input and filter/kernel, the convolution operation for point x, y can be calculated as the sum of the Hadamard product of the input sub-tensor with kernels. For more channels, the same operations have to be applied over all channels separately. The final output is an entrywise sum of them. Sub-tensor is point x, y and surrounding elements in that way, the point x, y is in the middle. The "sliding window" method is applied to use convolution over the whole image.

More formal, to calculate the convolution operation (without bias and with default parameters like stride or dilation rate) using the kernel K with size $(2n+1, 2n+1, n_C)$ (n_C refer to channels) for the input tensor I for a point (x, y) set convolution window size with the same size as the kernel $(2n+1) \times (2n+1) \times n_C$, assume the index of the center of convolution window to $(0, 0)$, place the center of the convolution window over the input point (x, y) and calculate the output for point (x, y) - see equation 2.8.

$$\text{conv}(I(x,y)) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} \sum_{k=1}^{n_C} I(x+i, y+j, k) K(i, j, k) \tag{2.8}$$

In practice, we use convolutional layers, which use convolutional operations to extract features from input tensors by making other tensors. Thanks to that, each layer better-generalized images passing from pixel level to global features. Parameters can easily control the size of the output tensor. The number of channels is controlled by repeatedly repeating the same operation but with different kernels. Other parameters can control

the width and height. The main one is kernel size, which aggregates the small part of the input tensor to be processed by convolutional operation. Usually, the odd number is set for both width and height to make convolutions with kernel sizes 1x1, 3x3, 5x5, and so on. Nowadays, the most popular variants are kernels 1x1 and 3x3. Another important parameter is the strides of the convolution along with the height and width. The other parameters are padding (adding extra padding to easily calculate the values of the edges) or dilatation (keep window size separated), the possibility to add bias, or choosing the activation function.

2.2.3.2 *Pooling Layer*

The other important layer is pooling. The main goal of this operation is to reduce the number of calculations by decreasing the width and height of the tensor. It uses the sliding window method (typically with size 2x2 and stride 2), where one number is chosen for each window. Usually, the maximum or average strategy was used to choose these numbers. When we use the typical configuration, the width and height is reduced by half each. Usually, there is one pooling operation after a series of convolutional layers.

Besides increasing the performance, the pooling operation can be a good influence on generalization by making the network more invariant to small translations of the input. Each strategy also has a different impact on the network. The average one seems to be a good choice because of trying to keep the whole "knowledge" from the network. The maximum one propagates only the crucial information further. The strategy is a hyper-parameter, and usually, the decision is made by network designers.

Nowadays, the pooling operations are not widespread, and reduction is used by other techniques like convolutional with extra stride.

2.2.3.3 *Fully Connected Layer*

The fully connected layer or dense layer is usually a classic FFNN. This part of the whole network is the classifier part - earlier layers are sometimes called feature generations. A few fully connected layers were used for many years, making the network prone to overfitting (due to the number of parameters). Using so many dense layers was necessary because all the last convolutional layer neurons were flatted, so the vector had a large size. There were numerous neurons, so the network needed more layers to "understand" this vast vector. Nowadays, thanks to an operation like global average pooling, we use only one layer (with the number of neurons equal to the number of classes), which significantly simplifies the model while keeping the same or even better results.

2.2.3.4 *Global Pooling*

Global Pooling operations are designed for converting the convolutional operation output (with 3d dimensionality) to vectors (1d dimensionality). They generate one response for every feature map. In practice, they replace fully connected layers in CNN, thanks to making connections between feature maps and categories. Moreover, these operations are parameters-free, making the network less prone to overfitting. Consequently, passing the resulting vector directly into only one dense layer (with the number of neurons equal to the number of classes) is possible. See the subsection 2.2.8.1 for more details.

2.2.4 *Important CNN models*

2.2.4.1 *LeNet-5*

The LeNet-5[130] model from 1998 was the beginning of CNNs. Lecun et al. proposed a simple (from today's point-of-view) network with two convolutional layers, a pooling layer, and fully-connected ones. The architecture of the model is presented in figure 2.7. They used it to classify small 32x32 pixels gray-scale images of handwritten characters. Although this network's basic mechanisms are still valid, the new approach was not noticed by the scientific community. Mainly because of not well-developed computer resources and consequently inability to make calculations on larger scale images.

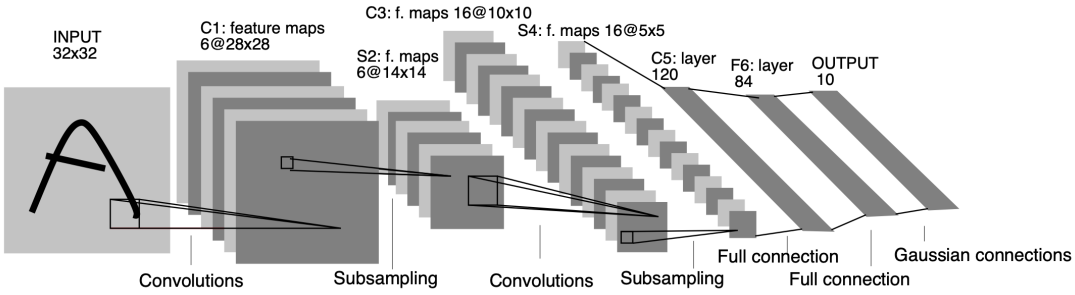


Figure 2.7: The LeNet-5 architecture from [130]

2.2.4.2 *AlexNet*

The AlexNet[123] model was a real breakthrough for image classification problems. It overcame more than ten percentage previous the state-of-the-art[205] solution in *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) in the top-5 ranking. It achieved 63.3% accuracy in the top-1 ranking and 84.6% accuracy in the top-5 ranking using 60 million parameters, which had to be trained. The AlexNet model became a base architecture for many future models over the years.

This model was based on LeNet-5 architecture but implemented many improvements. First, the authors used the ReLU activation function instead of a hyperbolic tangent (tanh), which allowed the training network a few times faster and helped with the vanishing gradient problem for deep networks. The next improvements were using max-pooling layers instead of average ones, using Local Response Normalization (which, as it turned out later, did not help), and the dropout method. The model works with input images 224×224×3-dimensional. The authors used data augmentation by extracting random patches and horizontal reflections to reduce overfitting. The architecture is shown in the figure 2.8.

The AlexNet model could be trained as a deep network because of using GPUs. This technology was developed mostly thanks to video games, and before, it was not possible to train in such a network in response time. Moreover, the authors needed two GPUs and six days to train the model, which was another challenge. Modification of the architecture by sharing weights between two sub-networks allowed running model using two GPUs simultaneously.

Another impressive contribution was using Euclidean distance on the feature vector(neurons from the last hidden layer) to find similar images in the dataset. Thus, Krizhevsky et al. showed how to use CNN in the image retrieval problem.

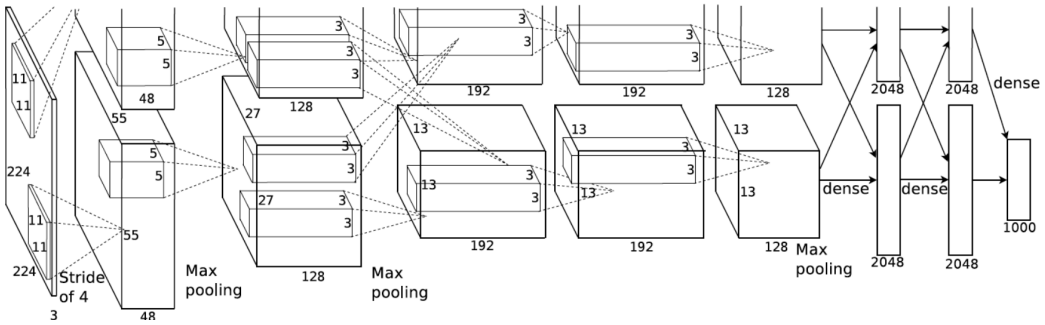


Figure 2.8: The AlexNet architecture from [123]

2.2.4.3 *ZFNet*

Although ZFNet[283] has not improved the accuracy of the CNN models, this paper significantly influenced the deep learning area. The authors modified the AlexNet model's hyper-parameters with, among others, changing kernel size from 11x11 to 7x7 for the first convolutional layer and stride from 4 to 2. Finally, this model achieved 85.3% accuracy in the top-5 ranking.

However, this model's most crucial contribution was a better understanding of how CNNs works using visualization methods. They noticed that the first convolutional layers could recognize simple patterns. The deeper layers of the network can recognize more complex features like, e.g., the eyes and noses of dogs, and finally, in full-connected layers, high-level features of the object in images. The authors proposed Deconvolutional Network to make their visualization.

Zeiler and Fergus also proposed an occlusion sensitivity method to show which fragment of the input image is crucial to classify the image successfully. This idea is the base for future class activation mapping methods.

In this paper, the transfer learning idea was also presented. Using knowledge from the first convolutional layers in a pre-trained network can also be applied to another image classification problem - working with another dataset. Only the last parts of models are required to retrain to work with new data successfully.

2.2.4.4 *Network In Network*

The "Network In Network" paper[139] presented two significant ideas. The first was convolution with a 1x1 filter size, which adds extra nonlinearity and decreases the number of channels in the network, thus reducing operations. The second contribution is Global Average Pooling, which reduces the number of neurons in fully-connected layers. The classification part - fully-connected layers - of CNN often takes up most of the memory of CNNs. Using a new pooling regularization technique allows for optimizing this part of the network.

2.2.4.5 *VGG*

The VGG[222] models are a group of CNN architectures. These are characterized by specific rules of how to create them. Usually reducing the width and height of the next layers' dimensions by half using pooling operations, the number of channels is doubled. Keeping this structure, the authors could make the deep network up to 19 convolutional

layers and achieved 74.5% accuracy in the top-1 ranking and 92.0% in the top-5 ranking using huge(even today) 144 million parameters.

The network used only filters with a small(3x3) receptive field. Using these convolutional layers with small size reduces the number of parameters and adds extra nonlinearity to the network. For example, it is possible to change one 5x5 convolution(with 25 parameters) with two 3x3 convolutions (with 18 parameters).

The authors proposed six different configurations and tested how each of them performs. The main conclusion is the importance of the network's depth. The deeper network is, the better results are given. Generally, ussually the VGG name refers to VGG-16 (model with 16 convolutional layers) or VGG-19(model with 19 convolutional layers). The deeper network than VGG-19 did not achieve better results due to the vanishing gradient problem.

2.2.4.6 *GoogLeNet / Inception*

Authors of the GoogLeNet[233] model proposed a few techniques that helped achieve better results. The most important one is the Inception module (presented in figure 2.9). The idea is to use many convolutional kernel sizes like 1x1, 3x3, or 5x5 on the same level and combine them. That allows looking at parts of the input tensor with different zooming and consequential extracting of various features. This architecture's other characteristics are using convolutions with 1x1 kernel size to reduce the number of operations and auxiliary classifiers to overcome vanishing gradient problems. The authors of the network tried to keep a balance between depth and width. That allowed achieving 69.8% in the top-1 ranking and 89.9% in the top-5 using only 5 million parameters.

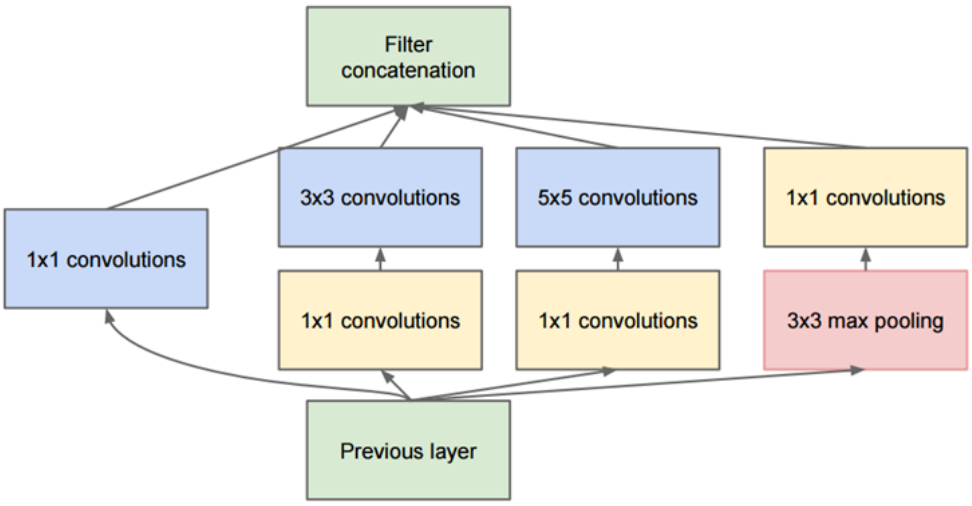


Figure 2.9: The inception block from [233]

2.2.4.7 *Inception v3*

The next versions 2 and 3 of Inception were introduced in the same paper[235]. The authors focused on further optimizations. There are convolutional operations with kernel 1x1 like in the previous version and changed convolutions with kernel size 5x5 to two 3x3. That allowed squeezing the weights from 25(5*5) to 18(3*3*2) and furthermore adding extra non-linearity. Next, convolutions with kernel size n*n could be changed with two with kernel size 1*n and n*1 - for convolution 3x3, weights will be changing from 9 to 6. To

keep the network wider, not deeper, the authors decided to split 1*n or n*1 blocks. In Inception v3, it was also used 7×7 factorized convolution. Using the above techniques allows proposing the three Inception modules(showed in figure 2.10) and the consequential the whole architecture.

The new Inception model also used MSprop optimizer, Batch Normalization, and Label Smoothing Regularization. The network became a standard for many years and was eagerly used, i.e., with a transfer-learning technique [236]. When combining all techniques, the model Inception v3 could achieve 78.8% accuracy of top-1 ranking and 94.4% of top-5.

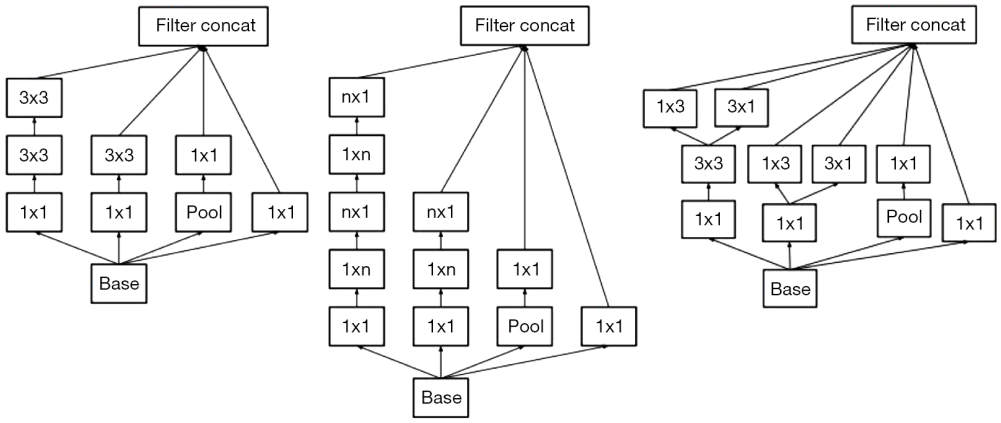


Figure 2.10: The three new inception blocks A(left), B(middle) and C(right) from [235]

2.2.4.8 *ResNet*

A real breakthrough was the ResNet[87] model, which started an era of the really deep neural network, even up to 1000 layers. Thanks to using residual connections (called also skip connections), overcoming the vanishing gradient problem could be possible. This idea allows for keeping the gradient and minimalizing the chance of vanishing it. The authors also proposed a bottleneck version of this idea with extra convolutions with kernel size 1x1, reducing the number of operations and adding additional nonlinearity. Figure 2.11 shows how these blocks look. The network achieved 78.57% accuracy in top-1 and 94.29% in top-5 ranking using 60 million parameters and 152-convolutional layers. Furthers models like [280] [234] [274] [100] improved this idea.

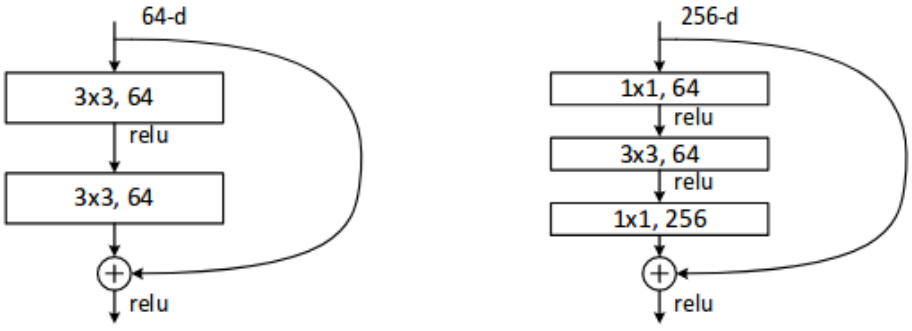


Figure 2.11: The ResNet blocks [87]

2.2.4.9 *Xception*

An interesting variation of the inception v3 model is Xception[42]. This architecture allows for creating wider CNNs thanks to Depthwise Separable Convolutions. They also used residual connections. Keeping a similar number of parameters as inception v3, a new model can perform better accuracy - 79.0% for top-1 ranking and 94.5% for top-5 ranking.

2.2.4.10 *Inception v4*

In 2016 Szegedy et al.[234] presented a new kind of inception architecture - inception v4. The main idea was further optimization of inception modules, changing hyperparameters of the network. The authors also decided to join the inception ideas with the residual connections[87] creating new models Inception-ResNet-v1 (based on inception v3) and Inception-ResNet-v2 (based on inception v4). Combining both techniques allowed for achieving 80.1% accuracy in top-1 ranking and 95.1% accuracy in top-5 ranking using 55.8 million parameters.

2.2.4.11 *ResNeXt*

The ResNeXt [274] paper proposed a highly modularized network architecture by modifying the ResNet's block. Figure 2.12 presented the main idea of this work: decreasing the number of channels in convolutions with kernel size 3x3 with the increasing number of those operations at the same level. By adding new parameters cardinality into the block(repeated convolutions on the same level), authors can increase the performance of the model 80.9% and 95.6% respectively, for the top-1 and top-5 ranking for ILSVRC data set. There are further developments of this idea like ScaleNet[136].

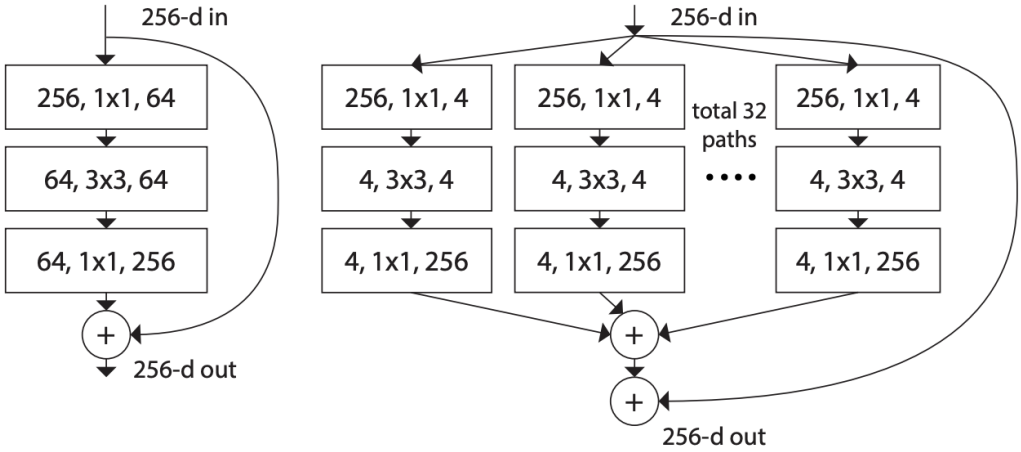


Figure 2.12: (Left) A block of ResNet (Right) A block of ResNeXt with roughly the same complexity. From: [274]

2.2.4.12 *DenseNet*

The residual connections [87] proposed by He et al. changed the way how to construct CNN. This idea was developed further, and one logical extension of it is the DenseNet model[100]. Additional skip connections (presented in the figure 2.13) were added not only for a single block but also for the next parts of the network. That allows strengthening

feature propagation, boosting feature reuse, reducing the number of parameters, and relieving the vanishing gradient problem. The specific architecture allows final layers to make a decision based on all feature maps in the network, not only the last ones as in the traditional approach. The DenseNet model can achieve 77.85% accuracy in the top-1 ranking and 93.88% accuracy in the top-5 ranking. There is a further development of that idea and used in different problems [109][292].

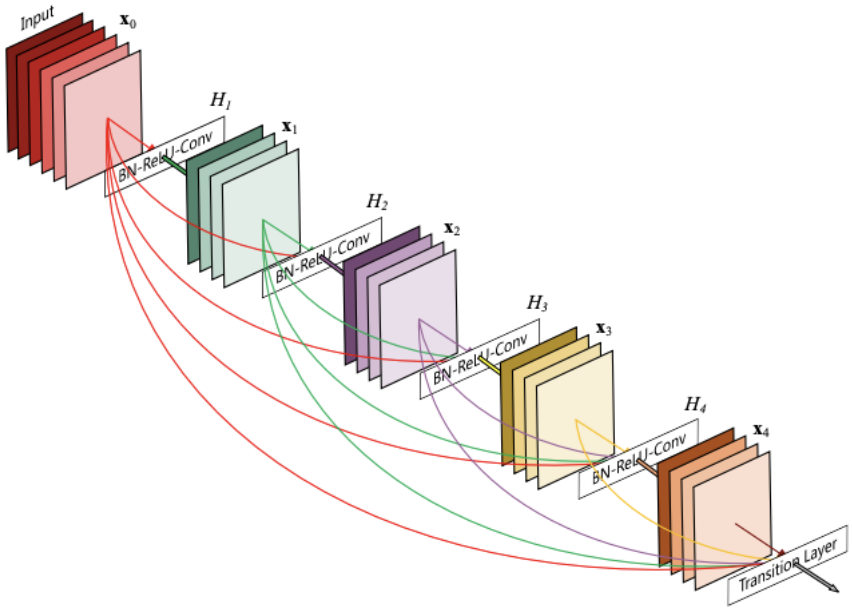


Figure 2.13: More robust feature propagation flow thanks to using additional residual connections in the DenseNet model [100]

2.2.4.13 *DPN*

Dual-path networks[39] are the next architecture that combines previous methods - in this case, ResNet[87] and DenseNet[87] - enjoying both sides' benefits. The characteristic of the ResNet network is feature reuse, and the DenseNet keeps exploring new ones. DPN models achieve high accuracy, small model size, low computational cost, and low GPU memory consumption by combining both ideas. The authors also tested an exciting idea to mix both maximum and mean global pooling described in [131]. It is worth noting that the input size of images will be increased even up to 320 x 320 px in models from this period. The DPN model can perform 81.38% accuracy for top-1 ranking and 95.77% for top-5 ranking with around 80 million parameters.

2.2.4.14 *PolyNet*

The PolyNet[286]is an extended version of Inception-ResNet-v2 [234]. Adding the next residual levels - in cascaded form - of the residual inception unit structure makes it possible to obtain better results. The new blocks are presented in figure 2.14, where F and G are abstract residual unit structure proposed in [234]. The model achieved 82.64% accuracy in the top-1 ranking and 96.55% in the top-5 ranking.

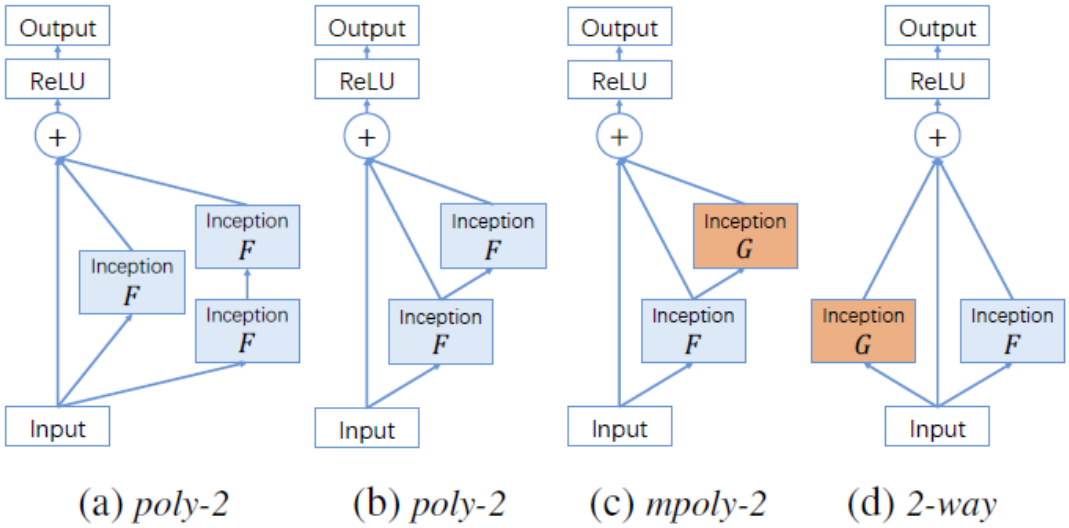


Figure 2.14: PolyNet blocks. F and G are abstract residual unit structure proposed in Inception-ResNet-v2 [234]. Source: [286]

2.2.4.15 *SENet*

Thanks to using an exciting new technique and modifying the exited earlier CNNs modules, the Squeeze-and-Excitation Network[99] model could achieve 83.12% and 96.42% accuracy for top-1 and top-5 scoring. That new idea was adding additional layers after the classic blocks of CNNs. First, the output for the classic block is squeezing, and global information is embedded. That operation allows squeezing global spatial information into a channel descriptor using global average pooling, generating channel-wise statistics. Next, the output from the classic block is combined with new embedding information. That idea can be used with any classic blocks - in the paper, the Inception and the ResNet blocks were chosen. Figure 2.15 shows the idea in detail.

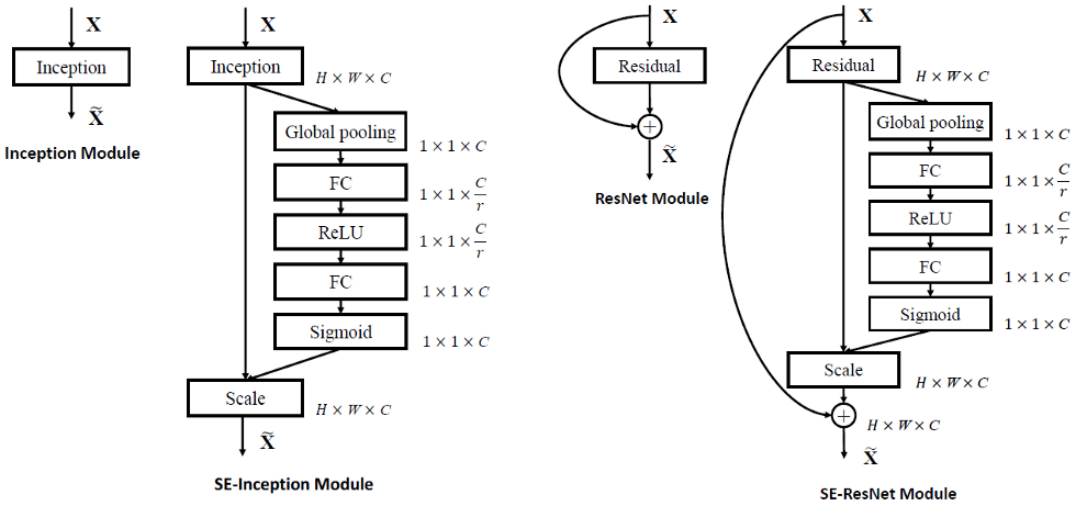


Figure 2.15: Example of using SENet idea of the Inception and the ResNet blocks. [99]

2.2.4.16 *NASNet*

The Google Brain teams introduced NASNet [293] model, where architectural building blocks learning for smaller datasets could be transferred and reused for the biggest ones like ImageNet. The model's architectures contain two types of blocks "normal cell" and "reduction cell". Both used convolutional cells to extract features, but "normal" keeps the feature map of the same dimension, and "reduction" reduces it by two factors. The specific structure of these blocks is searched by the reinforcement learning search method, so the authors do not predefine them. It was noticed that the convolutional cells discovered using the CIFAR-10 dataset generalize well also to the bigger one. The best model generated for ImagNet achieved 82.7% accuracy for top-1 ranking and 96.2% with top-5 ranking with 88.9 million parameters.

After running each sub-cell "normal" or "reduction," a new hidden state is put in memory. Using *Recurrent Neural Network* (RNN) and *Reinforcement Learning* (RL), a whole structure of a new type of cell is generated using all previous hidden states as input. That allowed generating a complicated "tree" with many operations such as classic convolutional, Depth-wise Convolution with different kernels, pooling, and identity blocks. An example of this "tree" is shown in figure 1.

A new regularization technique ScheduledDropPath is also an introduction. The main idea is to drop out of a path in the cell with some probability. That probability is linearly increased. Tests showed that this method significantly improved accuracy.

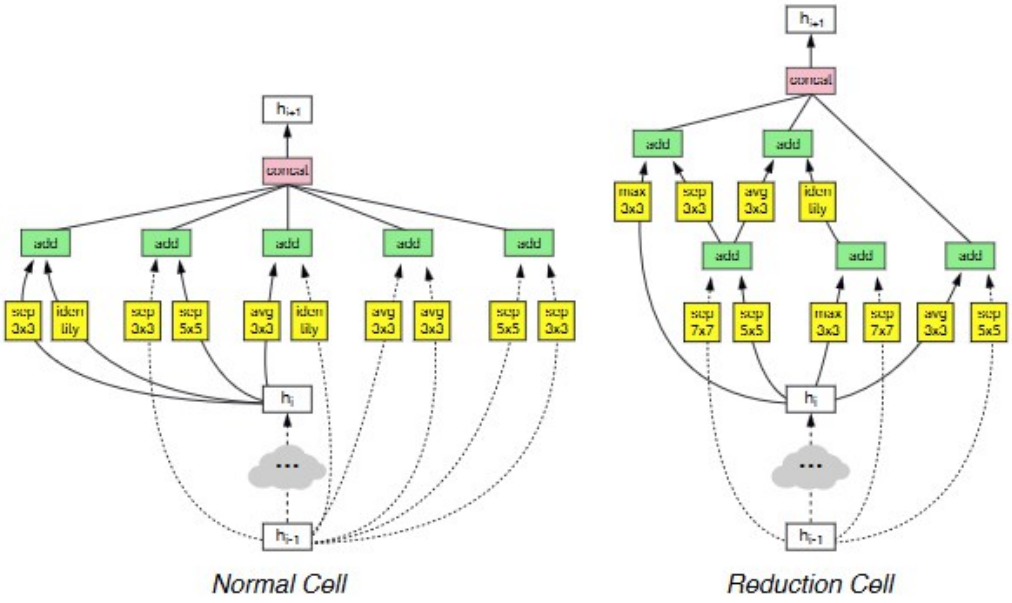


Figure 2.16: The NASNet cells generate for CIFAR-10. [293]

2.2.4.17 *AmoebaNet and GPipe*

Settings hyper-parameters for *Deep Neural Networks* (DNNs) is not trivial task. Researchers are looking for the best architecture testing many solutions. A team from Google Brain decided to extend previous techniques using another ML algorithm - the evolutionary algorithm - and, for the first time, surpasses hand designs by finding a new state-of-the-art solution. Using this idea, the model called AmoebaNet[184] allowed obtaining 83.9% and 96.6% for top-1 and top-5 accuracy with 469 million parameters.

The search space was limited and based on the model NASNet[293]. The evolutionary algorithm's goal has precisely to discover the architectures of the "normal" and "reduction" cells. The model was evolved on CIFAR-10 and then transferred to ImageNet. The example of the blocks is presented in figure 2.17. That kind of experience is, in practice, impossible to repeat for not very large corporations like Google due to resources limitation. There are further developments of this idea like PNASNet[142] which used a new Progressive Neural Architecture Search method.

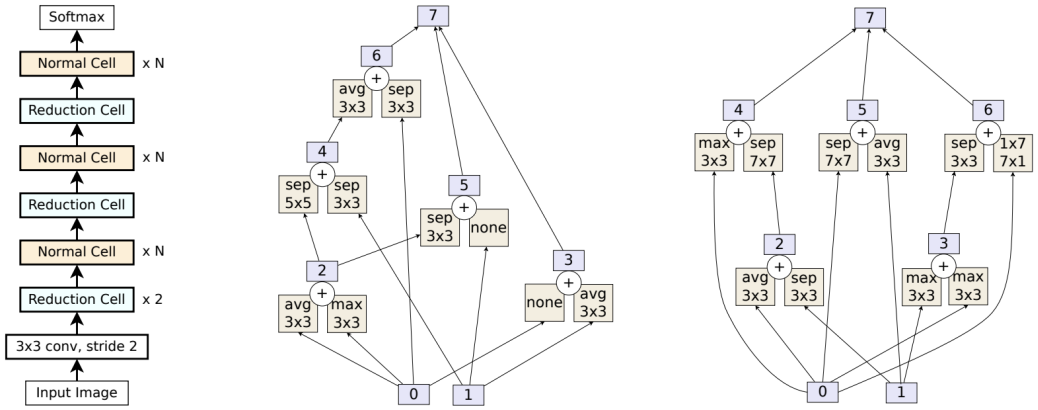


Figure 2.17: The AmoebaNet model for ImageNet. From left: overall model, normal cell, reduction cell. Source: [184]

The GPipe[102] is not a typical model but a new way to create pipeline parallelism. The authors trained a huge 557-million-parameter AmoebaNett[184] model and attained a top-1 accuracy of 84.4% and top-5 accuracy of 97.0%. The paper described how to adapt the parallelism approach where more machines can be used and split input data. The data can be split into micro-batches, and thanks to that, accelerators can operate in parallel.

2.2.4.18 *EfficientNet*

The authors of the EfficientNet[244] proposed a novel way how to construct CNNs. They deeply tested the effect of model scaling with width, depth, resolution, and compound all. Wider networks tend to capture more fine-grained features. Deeper networks can capture more complex features and generalize new tasks well. High-resolution input images can give more important information about the class, which has to be classified. Separate changing these hyper-parameters give no better results than not so much wider or deeper networks. The authors of EfficientNet proposed an easy-to-use method presenting how to scale all these parameters and finally achieved up to 84.4% accuracy of top-1 ranking and 97.1% of top-5 with 66 million parameters. The network's core block is MBConv, which is a combined block from MobileNetV2[206] with a Squeeze and Excite[99] block injected sometimes. A new active function is also used - Swish[182]

2.2.4.19 *NoisyStudent*

The ImageNet dataset has not had enough images for modern models. Researchers have started to use the external dataset for the training phase. For instance, using the ResNeXt-101[274] model trained with billions of social media images and using the transfer learning technique achieved 85.1% and 97.5% accuracy for top-1 and top-5 ranking in ILSVRC - details in [153].

The idea of using additional data was further developed. The NoisyStudent[273] model uses two sub-models: teacher and student. The teacher sub-model was trained on the classic training images and used to classify billions of unlabeled data. A new submodel - student - use labels for the classic training set and soft labels from the teacher for initially unknown data. The student sub-model has to be bigger than the teacher. The authors used EfficientNets[244] as their baseline models. The final model achieved 88.4% and 98.7% accuracy for top-1 and top-5 rankings with 480 million parameters.

2.2.4.20 *BiT*

The BigTransfer idea by Google Research teams shows how to transfer knowledge from pre-trained models to new datasets, even having only a few examples per class. The authors trained a vast network using datasets with up to 30 million images and transferred knowledge to classic datasets like ImageNet or CIFAR-10. In BigTransfer proposed simple rules on how to use their model on any sets, among others: use MixUp[284], prepare a new network with the new final layer, use BiT-HyperRule - heuristic for choosing hyperparameters for downstream fine-tuning. The bone model is based on ResNet152x4(4 times wider than the classic model) with some changes - like adding GroupNorm and Weight Standardisation. The transferred model to ILSVRC achieved 87.54% accuracy for top-1 ranking and 98.46% for top-5.

2.2.4.21 *FixEfficientNet*

In 2019 the Facebook AI Research team developed a new idea called FixResNet[249] for fixing the train-test resolution discrepancy. The authors claim that it can improve the performance of any CNN architecture. The key idea is to decrease the differences between training and testing data when using data augmentation models. For instance, using the crop method in the training phase, different parts of images are analyzed, while in the test phase, usually only the center part. The key to a new method is split into two-part: calibrating the object sizes by adjusting the crop size and adjusting statistics before the global pooling operation. The fine-tunning on the last layers has to be done to apply this method.

This method was applied to EfficientNet[244] and thanks to combine that two ideas the FixEfficientNet[250] appears. This model can achieve up to 88.5% accuracy in the top-1 ranking and 98.7% in the top-5 ranking keeping 480 million parameters.

2.2.5 *Mobile Models*

Deep learning solutions are more and more popular in normal-day applications. Because of that, some of the models have to be run on devices without powerful components like CPU, GPU, or RAM – for example, on smartphones or using AI platforms like NVIDIA Jetson, Google Coral, or Intel NCS. Some techniques - call them "model optimization" - help and allow the export of the models from dedicated AI servers to mobile or edge devices.

There are two main paths to optimize models – size reduction and latency reduction. The advantages of smaller models are smaller storage size, smaller download size, and less memory usage. Usually, we use the following strategy for this task: quantization, pruning, and clustering. The goal of latency reduction is to minimize the time needed to run the model for single data. Optimization of the models is a trade-off between size and speed v.s. final accuracy.

The popular strategy is quantization [106]. The idea is to reduce the model's size by using different data types to store network weights. Usually, we replace high-cost floating-point numbers (e.g., float32) with low-cost fixed-point numbers (e.g., int8/int16). As a result, we achieve a smaller model and faster computational speed. The quantization can be done during training or post-training. Of course, the post-training method is less effective, and the final accuracy can drop, but there is no data requirement.

The pruning[18] is the task of reducing the size of the network by removing parameters. The goal is to find the parameters which have the lowest impact on the final decision and remove them. First, a complex over-parameterized network is trained. Then we prune it based on come criteria - it can be done with zeros some wights of the model, making the network more sparse. The sparse model can be compressed more effectively. The clustering[224] gives similar benefits to pruning, so the model can be better compressed. The idea is to group the model's wights into clusters and then use centroid values instead of original values.

Popular deep learning libraries such as PyTorch or Tensorflow support making the above changes. These allow easy quantizing of the model, optimizing it, and saving in a special format supported by the library. The library supports popular devices like mobile phones with Android, iOS, or edge devices.

However, one of the most popular techniques is changing model architecture. By using models from the family of the MobileNet or the ShuffleNet, there is possible to train models with a small number of parameters and with good accuracy.

2.2.5.1 *MobileNet*

The MobileNet[97] is basically a simplified version of the Xception[42] architecture optimized for mobile applications. The key is using Depthwise Separable Convolution. First, the Depthwise convolution is used, and then another Pointwise convolution. The figure 2.18 well presented this. The Depthwise convolution is used for each channel separately. The convolutions with kernel size 1x1 are used to correlate outputs from all channels - we call it Pointwise convolution. The goal of this operation is to reduce the number of parameters and operations. For instance, where we compare classic convolution with kernel size 5x5 (stride = 1, no padding) where the input is Tensor with shape and the output is tensor with 256 channels, then we need 1228800 $(256 * 3 * 5 * 5 * 8 * 8)$ operations for classic convolution and 49152 $(3 * 5 * 5 * 8 * 8 + 256 * 1 * 1 * 3 * 8 * 8)$ for Depthwise Separable Convolution, so about 25 times fewer operations.

The MobileNet v2[206] introduced, among others, Inverted Residuals and Linear Bottlenecks blocks. The classic residual bottleneck uses convolutional with kernel size 1x1, next with size 3x3, and again with size 1x1. The purpose of that is to change the number of channels in the middle convolutional to reduce operations. We can present it as a "wide -> narrow -> wide" concept. The Inverted Residuals change it for the concept "narrow -> wide -> narrow". In MobileNet, thanks to using Deptwise Separable Convolutions, it is possible to make middle convolutional more effective and force it to work with a higher number of channels. The Linear Bottencks do not use the ReLU activation function after the last convolutional with kernel 1x1 before summing up with the input. Because ReLU ignores all negative numbers by not using it, we can increase performance. To be precise, the authors of this model used ReLU6, which can be denoted as $ReLU6(x) = min(max(0, x), 6)$. It helps with floating numbers when we use optimization techniques.

The MobileNet v3[96]introduced a few next improvements. The authors used squeeze-and-excitation blocks known from [99] with hard swish non-linearity ($h\text{-}swish(x) =$

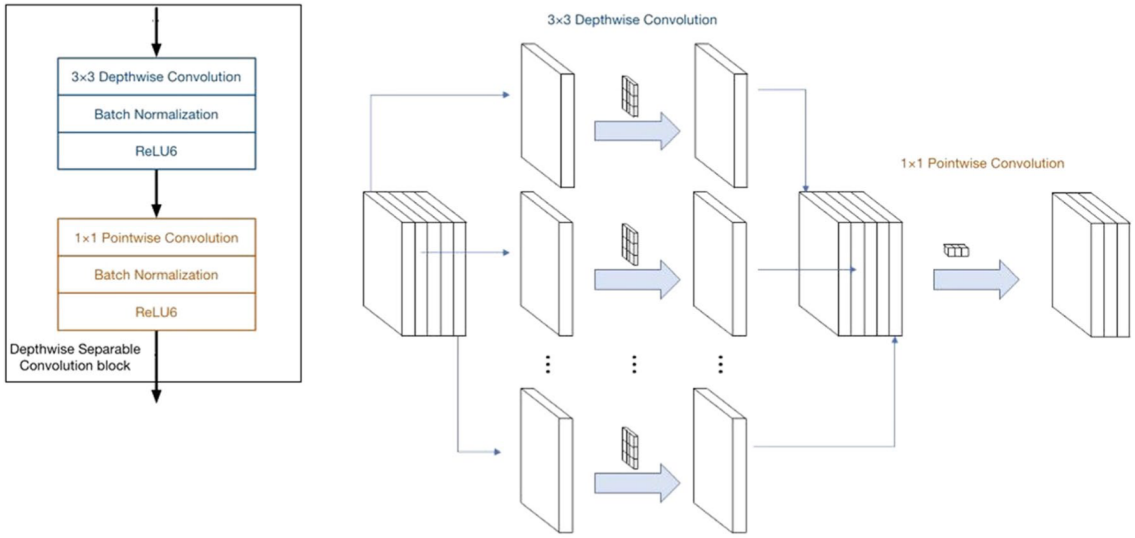


Figure 2.18: The Depthwise Separable Convolution used in the MobileNet – from [8].

$x\frac{ReLU(x+3)}{6}$) instead of sigmoid. The advantages of using squeeze-and-extraction was described in the previous sections. The new activation function was motivated by the advantages of using original swish and increased performance on optimized models. The authors of this version of the MobieNet also used an efficient last stage, where three expensive layers were dropped, with minimal influence of final accuracy.

2.2.5.2 *ShuffleNet*

The ShuffleNet[285] is the model which introduces the Channel Shuffle. The key idea is to use pointwise (using a kernel with size 1x1) grouped convolution (GConv), and next shuffle channels. The pointwise grouped convolution allows reducing the number of operations compared to the non-grouped version. However, to increase information flow, the outputs from each group have to be shuffle. It allows the model better fit to data within keeping advantages of used grouped convolutions. The idea Channel Shuffle is presented in figure 2.19. The authors use three units presented in figure 2.20. The depthwise convolution is donated as DWConv.

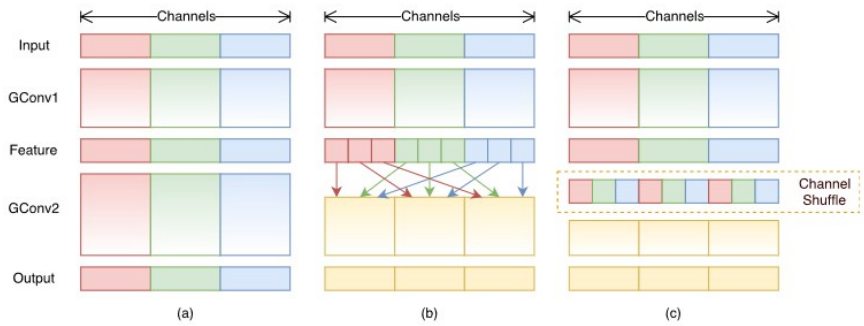


Figure 2.19: The idea of Channel Shuffle from [285]. **a)** Stacked convolutional layers with the same number of groups. The outputs are analyzed only within the group. **b)** Shuffling the features allow increased information flow when using GConv2. **c)** Equivalent for b) with using Channel Shuffle.

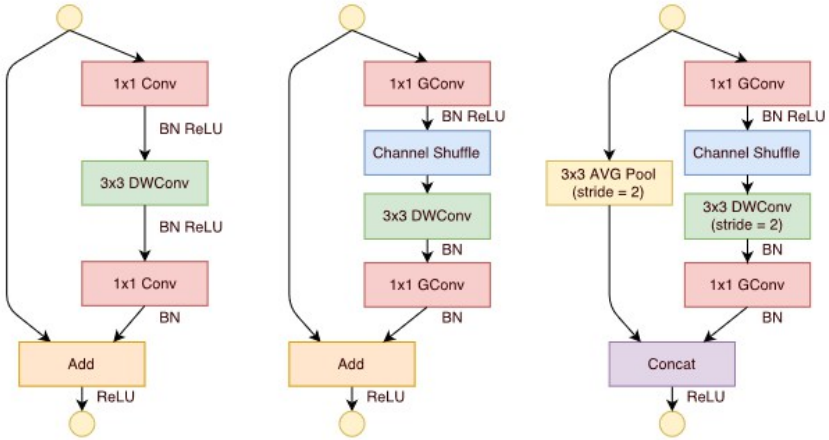


Figure 2.20: Three units used by the ShuffleNet from [285].

2.2.6 *Contrastive Learning*

Contrastive learning is a method of learning the representation of data where the goal is to keep similar inputs close to each other and dissimilar ones far - all in feature space. This method is one of the most powerful for working with unsupervised learning, although it can work with labeled data as well.

The authors of [43] proposed the contrastive loss, where the main idea is to work with pairs of inputs and increase the distance between different class inputs and minimize it for the same class. In [210] the authors proposed to use the Triplet Loss (see image 2.21), where three inputs are given the anchor and positive (same class) and negative (different class) examples. The goal is to minimize the distance in feature space between the anchor and positive example and maximize with an anchor and a negative one. Working with two pairs has a positive impact on learning the representation. In [223] generalized this approach to include comparison with multiple negative examples.

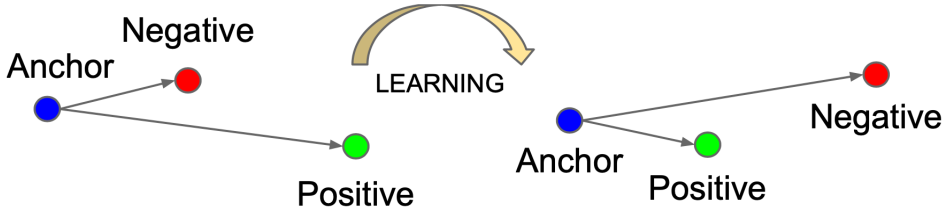


Figure 2.21: Triplet loss idea from [210]

Working with the unlabeled data required a different strategy. The key paper is SimCLR[35] where authors use a heavy data augmentation strategy on a single example and treat the pair as a positive example for contrastive learning loss. Any other pairs (two different - although augmentation - images) are treated as negative pairs. This method is further developed e.g., in [281] or [78].

2.2.7 *Detection, Segmentation and Key Points Estimation*

Object detection is a task where we need to find the specific object on an image and mark it by a bounding box (bbox). Segmentation is a task where we find the object and

mark all pixels belonging to it. Key points estimation is finding key points for the object - examples of semantic key points are "right shoulders", "left knees", or the "left brake lights of vehicles". A visual comparison is presented on the figure 2.22 (from [1]). We can also vary the segmentation problem on semantic segmentation, where each group of pixels marks the specific class, and instance segmentation, where the instances of the same specific class are separately marked.

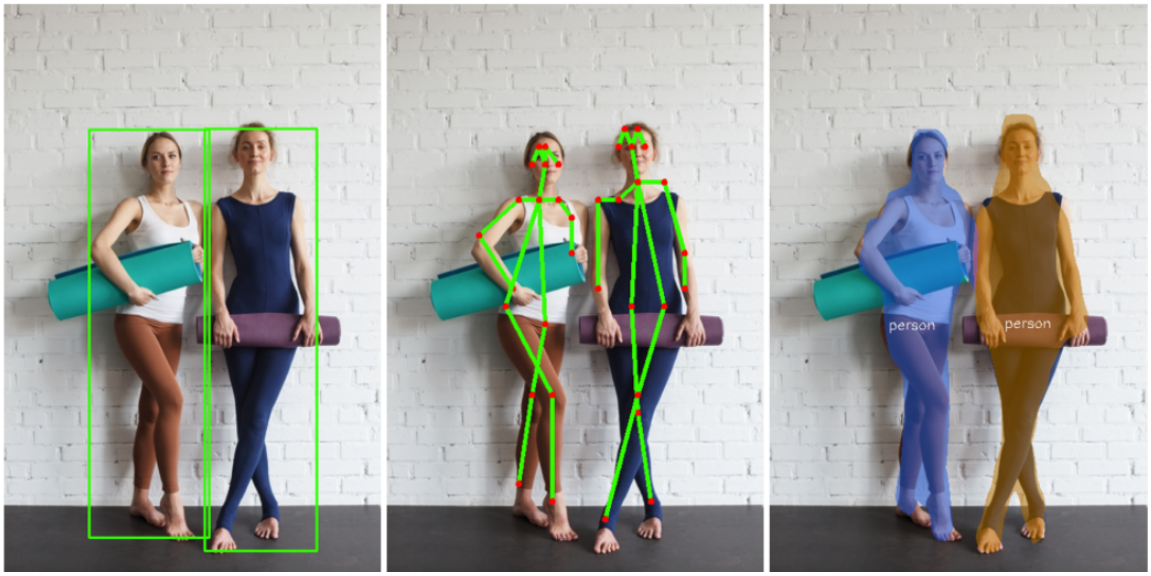


Figure 2.22: Presentation of object detection(left), key points estimation(center), and object segmentation(right)

Object detection is a vastly known problem. The R-CNN family is one of the first methods which tried to solve it. The R-CNN[70] is a strategy where used selective search[253] for generate about 2000 region proposal. Each of these is resized and passed through the CNN(then VGG). The network decides if the passed fragment of the image is known. Finally, the bounding-box regression is used to improve final detection. The next version - fast R-CNN[69] - overcame the biggest problem with the previous version – passing all-region proposals by the network independently. The authors proposed to use the RoI pooling layer. The idea is as follows. Each image is passed through by the CNN ones, and only the corresponding parts of values from the last convolutional operation are passed to the final classification part. The figure 2.23 described this well. Using this method, we can speed up the whole process because only classification is running multiple times. The biggest weakness is the region proposal method because many proposing areas are not worth analyzing. The faster R-CNN[190] solves this problem by using another network for proposing areas. It must be trained separately.

Another significant family for object detection is YOLO family[188][186][187][19][259]. The key idea is to design a network in that way to analyze the image and prepare bounding boxes at once – without using predefined region proposals. The image is resized, then divided into $S \times S$ gird. The goal of the CNN is to return the C class probability and prediction of B bounding boxes with confidence scores - all for each cell. In other words, the network return tensor with size $S \times S \times (B * 5 + C)$. The figure 2.24 demonstrates this. Non-maximal suppression[194] is used to fix multiple detections. The next versions focused on improving the speed and precision of detection - the authors, among others, added residual connections, allowed predicts at three scales (that help with detection of

[1] `https://github.com/hampen2929/pyvino`

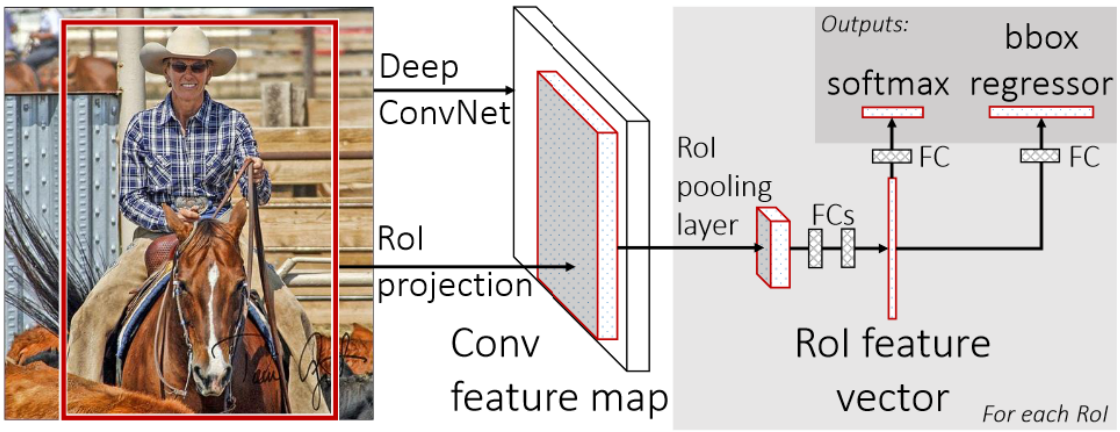


Figure 2.23: Fast R-CNN architecture - from [70]

small objects), improved classification with datasets with similar labels (like a person, a woman), and in [19] used mechanism for collecting feature maps from different stages.

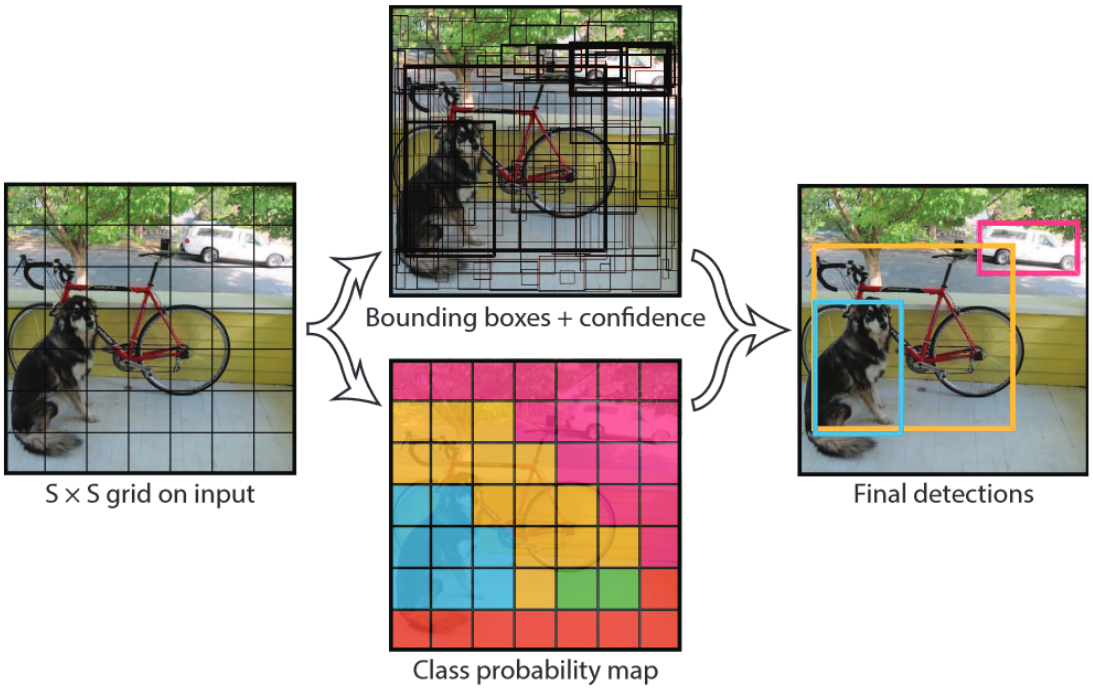


Figure 2.24: YOLO architecture - from [188]

Collecting feature maps from different stages is a new direction in object detection. One of the first implementations was proposed in [140] called Feature Pyramid Networks. The authors proposed a top-down pathway to fuse multi-scale features. It allows for better finding small objects by first "understanding" what is on the image and then propagating this information to the firsts layers. Later this method was further developed. In PANet [144] added ones more bottom-up path augmentation. In EfficientDet[245] proposed another modification (BiFPN) that helps achieve much better results in detection problems. The figure 2.25 visualize all the above approaches.

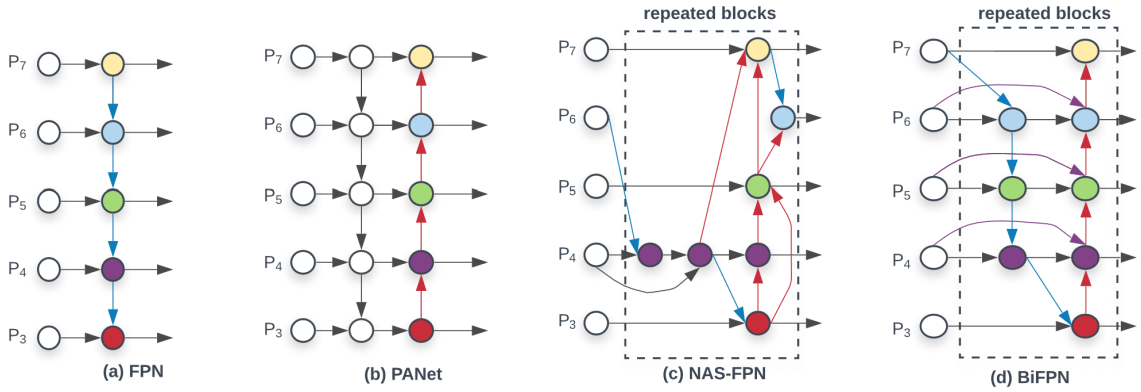


Figure 2.25: Development of Feature Pyramid Networks - from [245]

Segmentation and key points estimation used similar methods to object detections problem with some improvements specifically for these tasks.

The U-NET[192] is a popular method for segmentation. It used fully convolutional networks based on autoencoders[13]. In the U-NET, each corresponding block for the encoder and the decoder are also interconnected. There are many other improvements of this method, such as FusionNet[176] or DoubleU-NET[110]. The Mask R-CNN [88]is based on the R-CNN family, where authors used RoI align instead of RoI pooling - it used bilinear interpolation to generate new feature maps. They are passed to a convolutional subnetwork that returns binary mask and classification, which decide what class is on the object. The PANet - as we mentioned above - extends the FPN backbone and uses a similar approach as in the Mask R-CNN to create masks. Based on the YOLO idea, the YOLACT [21] was proposed.

Based on U-NET, there are the solutions for keypoints estimation like [164], where authors Stacked Hourglass Networks(similar to the U-NET) to generate a set of heatmaps responsible for the single detection point. The CPN[38] used the FPN idea for this task. By splitting the networks into two stages GlobalNet (localize "simple" keypoints) and RefineNet (localize "hard" - e.g., occluded keypoints) is a successful approach for multi-object keypoints detection. There are two main paradigms for multi-object detection: top-down and bottom-up. The top-down paradigm first detects the object and then performs single-object pose estimation for each detected object. The bottom-up paradigm[174][30][68] directly looks at all the keypoint positions and then tries to belong each of them to the object. The top-down way is more accurate and also costly due to an extra detection process of a single object.

2.2.8 *Image Retrieval*

In the context of deep learning, image retrieval is a system of finding the most similar image in the set of images based on the given query - also an image. The found image should be similar to the query in both: the meaning and style. Typically, we train the CNN network for a classification task and extract features for each image from the set using this network. These features are stored. Then the same extract method is used for the new query image. Next, all stored features are compared with the query's features using distance metrics like Euclidean or Cosine – the system return image, where the features have the minimal distance to each other, which means they are the most similar. A [36] is a good survey of the image retrieval problem using deep learning.

For the context of this research, the most crucial part is feature extraction. Before the deep-learning era, researchers tried to use hand-crafted features[128] to compare images and consequently solve image retrieval tasks. These methods are based on colors, brightness, edges, or simple shapes. Next, they used more complex methods like Scale-Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF), Histogram of Oriented Gradients (HOG) descriptor, Fisher Vector(FV), Vector of Locally Aggregated Descriptors (VLAD), or Bag-of-Words(BoW), often called Bag-of-Features in this context. Nowadays, we use features extracted from CNNs. The popular methods are described in the subsection 2.2.8.1.

The modern methods of image retrieval systems use contrastive learning. The Siamese Networks [119][177] is a popular solution. For example, the GEM used this approach - see figure 2.26. The Siamese Networks use two identical networks (with shared weights), two input images (i, j), and label Y with information that the images are matching or non-matching. The network is used for extracting features for each image - often also called the descriptor. In the training process, the authors of GEM used contrastive loss[43] presented in formula 2.9. The $\bar{f}(i)$ is $l2$-normalized GeM vector of image i, τ is a margin parameter that determines when pairs of unmatched images can be ignored by the loss – due to enough distance between pairs. The idea is simple - minimize distance for the matching example and maximize it for non-matching.

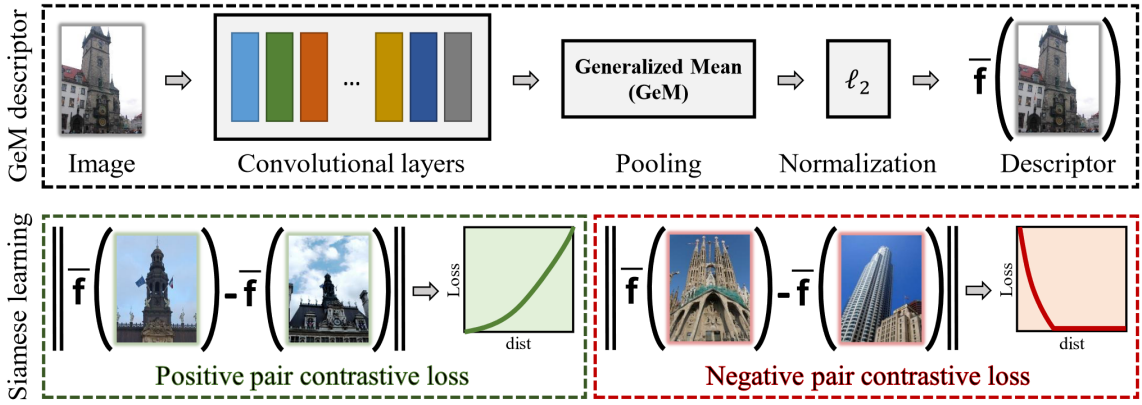


Figure 2.26: Contrastive learning used by GEM method– from [177].

$$\mathcal{L}(i,j) = \begin{cases} \frac{1}{2}\left\|\bar{f}(i) - \bar{f}(j)\right\|^2 & if\ Y(i,j) = 1 \\ \frac{1}{2}(max(0, \tau - \|\bar{f}(i) - \bar{f}(j)\|))^2 & if\ Y(i,j) = 0 \end{cases} \tag{2.9}$$

The above was further developed. One of the most known is using triple loss [262]. In that approach, the model optimizes matching and unmatching pairs simultaneously - see formula 2.10. j_p and j_n are respectively images matching(positive) and unmatching(negative) pairs for the image i. There are also less popular methods of contrastive learning like quadruplet loss[37], angular loss[261], or N-pair loss[223].

$$\mathcal{L}(i, j_p, j_n) = max(0, \tau + \|\bar{f}(i) - \bar{f}(j_p)\| - \|\bar{f}(i) - \bar{f}(j_n)\|) \tag{2.10}$$

2.2.8.1 *Feature Extraction*

2.2.8.1.1 NEURAL CODE One of the first image retrieval methods where CNN was used was "Neural codes"[11]. The authors got all values from one of the fully connected layers

and treated them as feature vectors. Authors use the architecture of CNN with three fully connected layers, and experiments showed the best results were given for the penultimate layer. The authors proposed a final process feature vector using $l2$ normalization, PCA compression with whitening[108], and again $l2$ normalization. That became a standard procedure in the next solutions. The paper has also proposed another strategy for reducing the vector size called "discriminative dimensionality reduction" - although it is not often used nowadays. The article presents exciting conclusions, e.g., the earlier layers betters perform with textures differences and later ones with high-level details. Although the results were not improved compared to the existing SOTA methods (VLAD, Fisher vectors), a new direction was developed.

2.2.8.1.2 GAP AND GMP The next method focused on getting features from the last convolutional layers using global pooling methods. There are two main variants: global average pooling(GAP) and maximum global pooling(GMP). It is a calculated mean or the maximum value picked over a matrix for each channel independently. For tensor T with shape (w, h, c), where w refer to width, h to height, and c to channels, we generate vector V with shape c) where each element is calculated as $\frac{1}{wh}\sum_{i=0}^{w}\sum_{j=0}^{h} T_{i,j,k}$ for GAP or $max_{i,j}^{w,h} T_{i,j,k}$ for GMP for each channel independently denoted as k. The more popular is GAP. It was first introduced in [139].

Scientists used different pooling methods like maximum, sum, or average pooling. Different methods have different advantages – for instance, average pooling is more invariant to the scale change since the maximum response of a feature map does not change abruptly with the scale change.

2.2.8.1.3 SPOC In 2015 it was a proposed idea[10] that expanded these methods. In this case, the sum pooling over each channel was used for the tensor from the final convolutional layer. According to the authors' experience - the essential parts of images usually are at the center of the picture. The authors decided to use additional coefficients - weights, which help focus on the center of the image - Gaussian weighting was used for this step. Next, the standard post-process should be done - using $l2$ normalization on the descriptor vector, PCA compression with whitening, and again $l2$ normalization.

Often the "SPoC" name is used for global average pooling - the sum of the averages is the sum of all the data, so the concept is still correct. For max pooling, the name MAC (Maximum Activations of convolutions) is used.

2.2.8.1.4 GEM The GEM (Generalized-Mean) [177] method is another generalized method of global pooling. GeM is a special layer that can be parametrized, and thanks to that, it can adjust itself. Adding the additional parameter p makes it possible to manipulate the pooling strategy. Setting parameter p to 1 can tread GEM as SPoC [10] - global average pooling, and to 2 as SQU - Gated square-root pooling [40] and for ∞ as maximum pooling. This parameter can be set manually or learned during the training phase. That allows for automating the process (learning parameters by back-propagation algorithm) and choosing the best strategy by the network itself. A three-stream Siamese network was also used for image retrieval tasks.

2.2.8.1.5 CROW Another modification of the global pooling strategy is cross-dimensional weighting(CroW) [112]. The features from last convolutional operation(T) is weighted channel-wise by a weight vector β and weighted location-wise by a weight matrix

α such as $T'_{i,j,k} = \alpha_{ij}\beta_k T_{i,j,k}$. Next, the sum-pooled is performed to aggregate T' features. Finally, the L2 normalization, PCA reduction, and again L2 normalization are performed.

After a series of experiments, the authors claimed: "It can be seen that spatial weighting tends to boost locations for which multiple channels are active relative to other spatial locations of the same image. This suggests that spatial weighting is a non-parametric and computationally cheap way to favor spatial locations for which features co-occur while also accounting for the strength of feature responses. These locations are more discriminative as there are combinatorially more configurations at mid-ranges of sparsity."[112].

2.2.8.1.6 R-MAC Next methods started to focus on local descriptors, focusing on specific regions in images, while globals ones treat images as a whole. It can be an important difference for tasks like image retrieval, where details are very significant. One of the first methods was R-MAC[247]. The idea is to extend maximum activations of convolutions over regionals of images. The input image is split into equal size regions separately for each chosen scale - making rigid grids. For each area, the MAC method is used to calculate the sub-feature vectors. Each sub-vectors is post-process with $l2$ normalization, PCA compression with whitening, and again $l2$ normalization. Finally, all of them are summed and also post-process with $l2$ normalization.

2.2.8.1.7 DIR The R-MAC [247] was further developed by using an additional Region Proposal Network [75]. This extra network offered regions called ROI (Region of Interest), which are more fitted to images, giving better results. The idea is primarily used in the faster R-CNN [190] method for object detection problems. Additionally, the DIR's authors used Siamese Neural Network to keep similar features closer and keep away parts that do not match together.

2.2.8.1.8 DELF The DeLF method proposes a complete framework for the Image Retrieval problem. The authors decided to focus on local descriptors. First, they trained CNN network, then added an attention module on top of the last convolutional layer, which ranks each subfragment of features. There are needed to retrain the attention module. The ranking defines which local descriptions are the most significant. They also used the receptive fields method and image pyramid method to generate specific points from CNN. Local descriptions direct reflect on a fragment of images thanks to the receptive field with the image pyramid method.

Later methods focused on improving these techniques. [103] is similar to DeLF. They extended the model to multiplicative and additive attention modules and aggregated them together using a VLAD descriptor.

2.2.8.1.9 MS-RMAC An interesting idea was presented in MS-RMAC [135]. The first layers are usually responsible for recognizing textures and simple shapes, and deeper layers are responsible for more and more high-level features. Using descriptors from different network parts is a reasonable choice to reflect the image's essence entirely. The authors proposed using the R-MAC method to extract feature vectors from each convolutional block and next transform them using PCA with a different number of principal components. The number of these components increases with the following parts of the network, so the sub-vector with global features is the most extended compared to others. Finally, all sub-vectors are merged.

2.2.8.1.10 GAP ALL Based on MS-RMAC we propose the use of our simple method. For each convolutional layer, we use GAP and then concatenate alls to one merged vector. Similar to MS-RMAC, using weights from the many layers allows for generating images features that are more resistant to deformation or scale change.

2.2.8.1.11 SCDA SCDA (Selective Convolutional Descriptor Aggregation)[265] is based on an activation features map[215]. First, the aggregation map $A_{i,j}$ is obtained as $\sum_{k=1}^{c} T_{i,j,k}$. For the aggregation map A, there are w, h summed activation responses corresponding to the positions w, h. Next, the mask map M of the same size as A is obtained as $M_{i,j}^{w,h} = 1$ if $A_{i,j} > \bar{a}; 0$ otherwise, where $\bar{a}$ is the mean value of all the positions in A. The final feature vector is selected based on the largest connected component of the mask map M.

2.2.8.1.12 OTHER METHODS An interesting method to compare multiple global descriptors was described in [111]. In [80] introduced a novel joint loss function with trainable parameters, which showed improvement in retrieval accuracy. Another interesting paper is [252] where authors used reinforcement learning and recurrent neural networks to train another network, which generates feature vector invariants of 16 transformations like size, rotation, color, or brightness.

2.3 Challenges

This document focuses on three challenges of current deep learning solutions: open-set classification, adversarial attacks, and trustworthiness AI.

2.3.1 *Out-of-Distribution Detection*

2.3.1.1 *OoD detection methods*

Openset classification, or Out-of-Distribution detection, is the problem of pointing to these input examples which do not belong to any training class and labeling them as an "unknown" class. Nowadays, machine learning solutions achieve spectacular accuracy on popular benchmarks datasets often overcome human precision in many tasks - e.g., in 2015, authors titled a paper "Surpassing Human-Level Performance on ImageNet Classification" [86]. However, most implemented solutions are not tested on how the machine works on never-seen data distribution. When they recognize that specific input is out of their scope and is unsuitable for working with that data, they should raise that issue. In another way, their results can be unpredictable and be a serious threat to customers.

OoD methods can be split into a variety of methodologies. The baseline method by [90] is based on the maximum value of posterior probability obtained as the softmax score. The ODIN method by [138] uses input preprocessing and temperature scaling in the softmax function to increase separability between in- and out-of-distribution samples. Posterior probabilities are then used to obtain confidence scores of prediction. Some authors propose to use multivariate Gaussian distributions as models of class-conditional distributions [132], [214], which leads to estimating the uncertainty of prediction using Mahalanobis distance. Other methods are based on the probability of inclusion (e.g., the EVM[195], OpenMax[16]) or non-parametric density-based(e.g., LOF[22], OSNN[156]). While the above methods use standard representations learned to discriminate within in-distribution classes, some recent works, e.g., [214], [242] propose to use contrastive learning-based representations

which aim to discriminate in- and OoD samples. A recent comprehensive survey of OoD approaches is given in [67].

An input sample x in d-dimensional feature space is classified by a closed-set classifier as $c_i = \arg\max_{c \in C} P(c|x)$, where $c_i \in \{c_1, c_2, \ldots, c_M\}$ are the categories known in training data and $P(c_k|x)$ are posterior probabilities of these categories. Open-set classifiers attempt to reject the sample x as unrecognized if it is reasonably far from the known data or if x is out-of-distribution concerning the training data X_i pertaining to class c_i.

Decision boundaries are constructed using the distribution of training data X_i only or taking training data into account for other categories. The rejection mechanism is commonly realized by thresholding on probability models of class membership or by thresholding on distance from the known data or density of the known data. To realize open-set recognition, methods use a threshold i.e., a sample x is classified as $c_i = \arg\max_{c \in C} P(c|x)$ providing $d(x, X_i) < \delta$, and is unrecognized for $d(x, X_i) \geq \delta$, where d is some measure of distance between the sample and known data.

All methods described below calculate confidence/openness scores denoted as $d_{name\_of\_the\_method}$.

2.3.1.2 *MaxSoftmax and MaxLogits*

2.3.1.2.1MAXSOFTMAX The MaxSoftmax method is based on [90]. The idea is relatively simple. We take the pre-trained model for the close-set classification and calculate the network's output after softmax operation for each sample in test and unknown sets. We expect the output from known examples to be much higher than the unknown ones.

While we usually take the original close-set classifier, we can generally process it as follows. For the given input x with regard to the known training data X, we need to build a neural network f which assigns a label as $\arg\max_{c \in C} f(x)_c$ to one of the known classes. By $f(x)_c$ we denoted classifier's output value for class c. Then we can calculate $d_{MaxSoftmax}$ according to the formula 2.11 where $softmax(x)_i = \frac{e^{x_i}}{\sum_{c \in C} e^{x_c}}$ for any given case.

$$d_{MaxSoftmax}(x) = \max_{\{c \in C\}} softmax(f(x))_c \tag{2.11}$$

2.3.1.2.2MAXLOGITS We also propose the MaxLogits method, which does not use the softmax function but works in the logits space. The $d_{MaxLogits}$ can be calculated as showed in the formula 2.12.

$$d_{MaxLogits}(x) = \max_{\{c \in C\}} f(x)_c \tag{2.12}$$

2.3.1.3 *Mahalanobis distance based methods*

The Mahalanobis distance[154] (do not confuse with openness score $d_{Mahalanobis}$) is the distance between a point and a distribution. Actually, it is a multivariate equivalent of the Euclidean distance to the mean of all cases in the given distribution. The difference between Euclidean and Mahalanobis distance is the fact that Mahalanobis transforms the variables into uncorrelated variables with variances equal to 1, and then calculates Euclidean distance. The uncorrelated variables are rescaled and shifted principal components based on the covariance matrix.

For the given X_c with regard to the known training data X for $c \in C$ class we calculate mean $\mu_c = \frac{1}{N_c} \sum_{x \in X_c} x$, and covariance matrix $\Sigma_c = \frac{1}{N_c} \sum_{x \in X_c} (x - \mu_c)(x - \mu_c)^\top$, where N_c is

number of examples for class c. Than we can calculate openness score $d_{Mahalanobis}$ according to the formula 2.13.

$$d_{Mahalanobis}(x) = -\min_{\{c \in C\}} \sqrt{(x-\mu_c)^\top \Sigma_c^{-1}(x-\mu_c)} \tag{2.13}$$

In this work, we define many variants of the Mahalanobis. The standard/main one - which we called the OoD score as $d_{Mahalanobis}$ - uses a different covariance matrix for each class. The next one is $d_{Euclidean}$ presented in the formula 2.14 which is equivalent to Euclidean distance to the mean of all cases in the given distribution. Another is $d_{SEuclidean}$ which is Euclidean with rescaled by variance $v_c = \frac{1}{N_c}\sum_{x \in X_c}(x-\mu_c)^2$ for each class - see the formula 2.15. We decided to also test MahalanobisUF proposed in the [132]. This variant uses common covariance matrix $\Sigma_c = \frac{1}{N}\sum\sum_{x \in X_c}(x-\mu_c)(x-\mu_c)^\top$ for all classes (the means are still individual for each class). The $d_{MahalanobisUF}$ is presented in the formula 2.16(it is similar to $d_{Mahalanobis}$).

$$d_{Euclidean}(x) = -\min_{\{c \in C\}} \sqrt{(x-\mu_c)^2} \tag{2.14}$$

$$d_{SEuclidean}(x) = -\min_{\{c \in C\}} \sqrt{\frac{(x-\mu_c)^2}{v_c}} \tag{2.15}$$

$$d_{MahalanobisUF}(x) = -\min_{\{c \in C\}} \sqrt{(x-\mu_c)^\top \Sigma^{-1}(x-\mu_c)} \tag{2.16}$$

2.3.1.4 *LOF (Local Outlier Factor)*

2.3.1.4.1LOF The Local Outlier Factor[22] method looks at the neighbors of a certain point to determine its density and then compares it to the other points. It calculates the local reachability density $LRD_k(x, X)$ of input x regarding the training subset X. LRD is described as an inverse of an average reachability distance between the given point, its k-neighbors, and their neighbors (for details, follow the [22]). K-neighbors ($N_k(x, X)$) includes a set of points that are in the circle of radius k-distance, where k-distance is the distance from the point to it's the farthest k^{th} nearest neighbor ($||N_k(x, X)|| >= k$). The LOF is formally described as the ratio of the average LRD of the k-neighbors of the point x to the LRD of the point - see equation 2.17.

$$d_{LOF}(x, X) = \frac{\sum_{y \in N_k(x,X)} LRD_k(y, X)}{||N_k(x, X)|| LRD_k(x, X)} \tag{2.17}$$

Original LOF was used for outlier detection inside a data set, but it can be easily opened to OoD detection. If the point is in-distribution, the ratio of the average LRD of neighbors is similar to the LRD of the point. For OoD data, the density of an outlier should be smaller than its neighbor densities. LOF requires specifying the number of analyzed neighbors k – we used the $k = 20$, the default value for most implementations. We tested both Euclidean and Cosine distances, although Euclidean was originally used.

2.3.1.4.2LOF_D The LOF builds a model for the whole training data set. However, the local reachability density can be different in each known class. Moreover, finding the nearest neighbors among a large number of data is a time-consuming problem. Therefore, we modified the original LOF by building a separate LOF model for each in-distribution class. For OoD detection/uncertainty quantification, we use the model LOF corresponding to the closest class identified by the close-set classifier denoted as c. We refer to this approach as LOF_D and can be denoted as shown in equation 2.18. The X_c is the subset of training X which contains all samples with known label c.

$$d_{LOF}(x, X_c) = \frac{\sum_{y \in N_k(x, X_c)} LRD_k(y, X_c)}{||N_k(x, X_c)||LRD_k(x, X_c)} \tag{2.18}$$

2.3.1.5 *Open-Set Nearest Neighbor*

The Open-Set Nearest-Neighbor (OSNN) is proposed in [156]. The method uses the advantages of density-based approaches. For the tested sample x we find the nearest neighbor t and another one u in that way that the classes for samples t and u have to be different. The d_{OSNN} is calculated as a ratio between distances $d(x, t)$ and $d(x, u)$ as shown in equation 2.19. We used as distance d both Euclidean and Cosine.

$$d_{OSNN}(x, t, u) = \frac{d(x, t)}{d(x, u)} \tag{2.19}$$

2.3.1.6 *MDistance*

The MDistance is our method. First, the fitting procedure is applied. The training data X is split into X_c, according to class c. x means distance md for all other samples in X_c is calculated and stored for each training sample. However, only these values are saved, where the class for the least distance and class for x are the same. Then the outstanding values are rejected (we applied the classic rejected approach based on mean plus 2.5 times the standard deviation). Next, the threshold t_c for the given class is calculated as the maximum stored value for each label. In the test phase, the $d_{MDistance}$ is calculated as shown in equation 2.20.

$$d_{MDistance}(x, X, t) = 1 - \min_{\{c \in C\}} \frac{md(x, X_c)}{t_c} \tag{2.20}$$

2.3.1.7 *Extreme Value Machine(EVM)*

The Extreme Value Machine(EVM)[195] is the method that uses Compact Abating Probability (CAP) for each class separately. To construct the CAP there is needed to create a radial inclusion function for each point $xinX$ for each class c independently. Given a fixed point $x \in X_c$, τ closest training examples from classes other than c are selected, denoted as t, and the margin distances from x to these examples are calculated as: $m_{ij} = \frac{\|x_i - t_j\|}{2}$ where i refers to N_c data and j to τ. Then the parametric model of the margin distance from x is estimated by fitting the Weibull distribution to the data $\{m_{i1}, m_{i2}, \ldots, m_{i\tau}\}$. The EVM is the semi-parametric model, which is justified by the Extreme Value Theory. The fitted Weibull model has described by two parameters λ_i and κ_i. The Weibull survival function

$(1 - CDF)$ can be denoted as: $\Psi(x_i, x) = e^{\frac{\|x_c - x\|}{\lambda_i}^{\kappa_i}}$. It can be interpreted as the CAP model of the decreasing probability of inclusion of the sample x in the class represented by the training example x_i. Now we can set the d_{EVM} as shown in equation 2.21. In our paper [258] we discuss the validity of the underlying assumptions.

$$d_{EVM}(x, X) = \underset{\{c \in C\}}{\arg\max} \left\{ \underset{\{x_k \in Xc\}}{\arg\max} \Psi(x_k, x) \right\} \tag{2.21}$$

Remarks on the EVM Implementation The implementation of the EVM[2] uses the libMR[3](provided by the authors of [209]) library for the Weibull model fitting. This library first uses the linear transformation and then returns the parameters (λ_i, κ_i) of the Weibull model fitted transformation data.

2.3.1.8 *OpenMax*

OpenMax[16] is a hybrid method. First, the method "opens" the classifier based on EVT (see subsection 2.3.1.7 and adds the new label "unknown". Next, the probability is calculated for each known label and one unknown,makes it easy to determine $d_{OpenMax}$ easily (see equation 2.22). Initially, the fitting phase is applied. The EVT models ρ for each class C are fitted by using the correctly classified training sample x. The EVT uses the Weibull distribution of the largest distance to the mean feature vector τ for each class. The authors proposed using FitHigh from the libMR library. The probability estimation with the rejection of unknown or uncertain input ϕ is applied during the testing phase. The results are normalized into a probability distribution together within the new unknown label.

$$d_{OpenMax}(x, \phi, \rho, \tau) = \underset{\{c \in C\}}{\arg\max}\, \phi_c(\rho_c(\|x - \tau_c\|) \tag{2.22}$$

2.3.1.9 *ODIN*

The ODIN proposed in [138] uses two approaches temperature scaling and input pre-processing. The temperature scaling is a variant of the SoftMax (see subsection 2.3.1.2) $S(x,t)_i = \frac{e^{\frac{x_i}{t}}}{\sum_{c \in C} e^{\frac{x_c}{t}}}$ - where the t is the temperature scaling value.

The input pre-processing approach adds small perturbations to the input image. This technique is based on the FGSM attack (see subsection 2.3.2.2.1). However, here, the goal is to increase the softmax score of any input, in opposition to the FGSM, where the goal is to decrease that value to fool the network. The authors claim that "the perturbations can have a stronger effect on the in-distribution images than on out-of-distribution images, making them more separable". The procedure is illustrated in equation 2.23 where $\hat{y}$ is the winning class. The ϵ is perturbation magnitude denoted later also as m. The ODIN can be denoted as $d_{ODIN}(x, t, m) = \max_{\{c \in C\}} S(f(\hat{x}_{\varepsilon=m}))_c$.

$$\hat{x} = x - \epsilon * sign(\nabla_x log S(x,t)_{\hat{y}}) \tag{2.23}$$

[2] https://github.com/EMRResearch/ExtremeValueMachine

[3] https://github.com/Vastlab/libMR

2.3.1.10 *Unified Framework*

The Unified Framework, proposed in [132], uses a few interesting approaches - see our schematic diagram of this method presented in figure 2.27. The main OoD method uses Mahalanobis distance with the common covariance matrix described in section 2.3.1.3. The OoD detection is based on a features ensemble approach. The features are extracted multi times from the different parts of the network. The idea here is that distinguishing between in-distribution and out-of-distribution examples can be more effective while using low-level, medium-level, or high-level features. The Unified Framework extracts features from ℓ parts of the network, and then uses regression $\sum_\ell \alpha_\ell * M_\ell(x)$, where $M_\ell(x)$ is the $d_{MahalanobisUF}(x)$ for each ℓ. To determine the values of the α, the Unified Framework uses parts of unknown data, which can be problematic in the real case because we do not know the distribution of unknown data. We discuss it further in chapter 3. Another trick is using the pre-processing of inputs similar to the ODIN approach: $\hat{x} = x - \epsilon * sign(\nabla_x M_\ell(x))$.

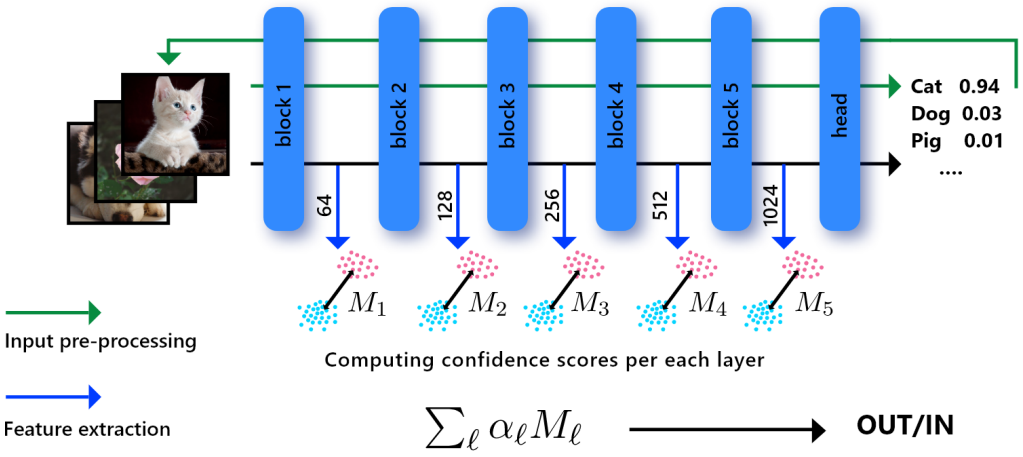


Figure 2.27: Method by [132], which uses the input pre-processing, with the Mahalanobis distance-based confidence score M_ℓ from feature ensemble. For hyperparameter tuning, some OoD examples are needed.

2.3.1.11 *Other methods*

2.3.1.11.1OUTLIER EXPOSURE Outlier Exposure[91] is a technique of showing some Out-of-Distribution data during the training process. The authors suggest that by showing it, the model can easier learn to build decision boundaries with respect to unknown data. They suggest that the method significantly improves detection performance and has a positive impact on the robustness of the network. The modification of the loss function can be denoted as $\mathcal{L}_{IN}(f(x_{IN}), y) + \mathcal{L}_{OoD}(f(x_{OoD}))$, wherein classic version both $\mathcal{L}_{IN}$ and $\mathcal{L}_{OoD}$ could be cross-entropy, but for OoD comparison to the uniform distribution. After fitting the procedure, the authors suggest using the classic baseline method (see subsection 2.3.1.2)

We noticed a problem when the OoD data presented during the training would be from different distributes than the during testing phase. The OoD system can even worsen the results while showing malicious examples. Because the OoD examples are fully unknown, we cannot be sure of the data distribution.

2.3.1.11.2ENERGY-BASED OOD DETECTION The authors of [145] proposed to use energy score in OoD detection problems. They suggest that this approach allows for distinguishing better In-Distribution and Out-of-Distribution samples than using the softmax scores. The best working technique with energy score is applied when the OoD examples are shown during the training and based on features, not logits.

2.3.1.11.3SSD The authors of SSD[214] combine two methods of contrastive learning to generate more robust features for OoD problem and Mahalanobis as OoD decision making. Authors suggest that using the contrastive learning idea, they can use unlabeled training data to fit the method. In this work, we do not focus on contrastive learning models. However, there is a trend that needs to be followed.

2.3.1.11.4GENERALIZED ODIN Generalized ODIN[98] is based on ODIN (see subsection 2.3.1.9) with two new strategies: decomposed confidence and the modified input preprocessing. Both, according to the authors, help in OoD detection performance without using any unknown examples.

2.3.1.11.5LEARNING CONFIDENCE In [50], the authors proposed additional output during the learning process called the confidence estimate output. This value can be used for OoD detection by using the simple threshold on this output. The confidence estimate is learned by additional loss based on the softmax output. When the network answers correctly, the confidence output should be increased. When incorrect, it should be decreased. The confidence estimate increases during the training process due to the overfitting of training data issues. An aggressive augmentation can deal with this problem because the network is not so confident when using it.

2.3.2 *Adversarial examples*

2.3.2.1 *Introduction*

Typical CNN models can be split into features extraction and final classification. The strength of CNN models lies in the first part, where complex maps of features based on many training images are created. State-of-the-art models use diverse techniques (some of them are listed above) to create better maps of features, based on which the CNN makes the final classification. This mode of operation makes CNNs vulnerable to so-called adversarial attacks, which generate special images with feature vectors similar to feature vectors for a specific class. Consequently, the CNN model (its classification part) is forced to produce a false answer – this is the idea of adversarial attacks.

Adversarial examples are specially prepared images that can fool the deep network. By adding a perturbation to the image with one class (from a human perspective), the network with high certainty claims the image belongs to another class. The deep network model is trained on thousands of images. This procedure allows the model to build decision boundaries for each image's class. The adversarial examples move the image outside the boundaries for a given class, often by minimizing the number of modifications on the image, for example, by replacing even one pixel.

There are two types of adversarial attacks – white and black[168]. White - the more popular - attacks assume that the CNN architecture is known and the attacker has access to the weights in the net. Black attacks assume that the attacker can only test final outputs

from the model while feeding modified input images, with no additional information about the model.

A good review of attacks and defenses is provided by [278] and [216]. The adversarial attacks can be very dangerous because CNN models are extensively used in commercial projects. For instance, robots that use visual recognition systems and work with people can threaten human health and even life[246]. Increasing the safety and reliability of CNN models seems to be crucial in modern AI.

2.3.2.2 *Attack methods*

2.3.2.2.1 FAST GRADIENT SIGN METHOD One of the fundamental methods is FGSM[74], where a new image x^{adv} is generated based on formula 2.24. The procedure is as follows. The image x with class y is passed through the network (defined by parameters θ), the gradient of a loss $J(\theta, x, y)$ is calculated for a given class, and it is added as small ϵ perturbation to the image. The goal is to add this small perturbation to the image, which does not change the original image too much (from the human perspective) but makes the network fooled. The figure 2.28 from the original paper well present it.

$$x^{adv} = x + \epsilon * sign(\nabla_x J(\theta, x, y)) \tag{2.24}$$

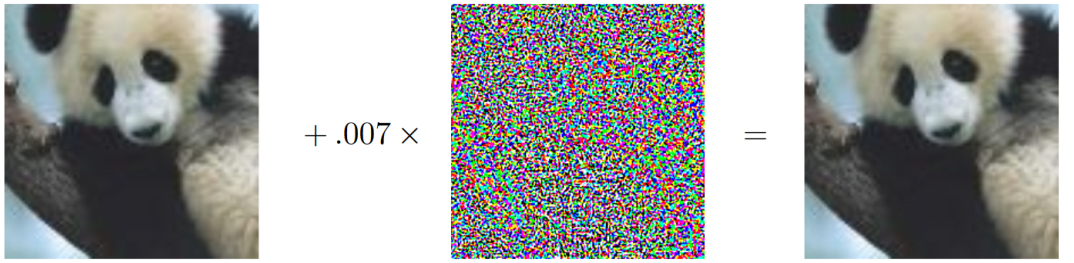


Figure 2.28: The example of FGSM from [74]

2.3.2.2.2 BASIC ITERATIVE METHOD The authors of BIM[124](also known as Iterative-FGSM) improve the FGSM attack by repeating of adding small perturbations multiple times. Moreover, a new image is clipped after each iteration to ensure that they are in an ϵ-neighborhood of the original one. The procedure can be demonstrated as shown in formula 2.25. The x_0^{adv} is the original image. The number of iterations has to be set heuristically. By setting α we can control how much the pixel values will be modified – in the paper $\alpha = 1$, so the values were changed by one at each interaction.

$$x_{N+1}^{adv} = Clip_{x,\epsilon}\{x_N^{adv} + \alpha * sign(\nabla_x J(\theta, x_N^{adv}, y))\} \tag{2.25}$$

2.3.2.2.3 PROJECTED GRADIENT DESCENT The Projected Gradient Descent (PGD)[152] is fundamentally the same method as the BIM attack. The only difference is with the start point – the PGD initializes to a random point in the ball of interest (decided by the

$L\infty$ norm) and does random restarts, while the BIM initializes to the original point. The method is also well known in optimization literature[25].

2.3.2.2.4CARLINI WAGNER ATTACK The authors of this approach[31] focused on solving the optimization problem of finding minimalized distance between two images x and $x+\delta$ for model C in that way that classification result will be different for both examples. The x is the static input image. To be more formal, see formula 2.26.

$$\begin{aligned} &minimize\ \mathcal{D}(x, x+\delta)\ such\ that \\ &C(x) = l,\ C(x+\delta) = t,\ x+\delta \in [0,1]^n \end{aligned} \tag{2.26}$$

The problem is not trivial due to the highly non-linear nature of deep models. The authors propose instead defining objective function f such that $C(x+\delta) = t$ if and only if $f(x+\delta) \leq 0$. The authors propose a list of possible f.

$$f(x') = max(max\{Z(x')_i : i \neq t\} - Z(x')_t, -k) \tag{2.27}$$

In the formula 2.27 is presented as the best among them – here $Z(x')_i$ is the logit for i class, and by changing k we can control how confident we want our adversarial attack to be misclassified. Now we can solve a new - much easier - optimization problem presented in the formula 2.28, where c is a positive suitably chosen constant.

$$\begin{aligned} &minimize\ \mathcal{D}(x, x+\delta) + c \cdot f(x+\delta)\ such\ that \\ &x+\delta \in [0,1]^n \end{aligned} \tag{2.28}$$

This problem can be now solved using, for example, the Adam optimizer.

2.3.2.2.5JACOBIAN-BASED SALIENCY MAP The JSMA[167] is the type of attack that minimizes the number of pixels that have to be changed to misclassify the image. This attack uses the gradient of the loss for every component of the input by the Jacobian matrix to extract the sensitivity direction. It uses saliency maps that show which part of the image is the most important during classification. Knowing that makes it possible to change the most significant pixels into others (corresponding with pixels from other classes). It allows increasing the likelihood of being different class. The process should be repeated until the network misclassifies the input image or the maximum number of pixels is reached (then the attack failure).

2.3.2.2.6DEEP FOOLED The Deep Fool[161] method is also the iteration approach. For this method, the distributions are calculated based on willingness to move the input across the decision boundaries with minimal changes. The first-order approximation of Taylor's expansion is used on a linear model to find these distributions.

2.3.2.2.7ONE-PIXEL ATTACK This method[230] is highly interesting due to changing only one pixel to make the network fooled. It uses a differential evolution algorithm to find which pixel should be changed. The candidate solutions contain information with x, y coordinates, and three RGB values. During each epoch, the population is randomly modified by a small factor, and the algorithm works until one of the candidates will be an adversarial attack. The crossover was not included in this approach. The fitness function is defined by the probabilistic label of the target class or the true label. It is an example of a black-box attack.

2.3.2.2.8OTHER METHODS There are many other attacks in the literature. In this work, we focused only on the classic ones described above. The authors of [24] or [246] proposed using special generated patches. These patches can be easily used in real-world cases. The authors claim that it is as easy as printing the patch and adding it to the scene. Then the classifier points out the wrong classes. It is hazardous because, for example, in [246] authors show how to generate that patch on popular ready-to-use models that are commonly used for human detection. If someone has a special print in their hand, the model is blind to it - see figure 2.29. The GAN[271] or U-NET[175] can also be used to attack the

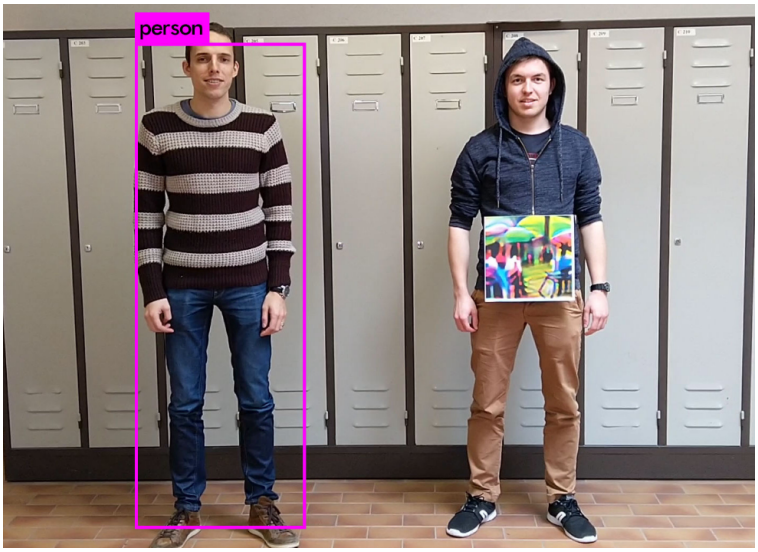


Figure 2.29: Example of adversarial patch from [246]

network. We can train the network to generate or modify the input image to fool the main classifier network. One of the first ideas in this group of attacks is [165], where authors used evolutionary algorithms for this – however, the attacked images from this paper do not look natural. Nowadays, there are interesting ideas to find real adversarial examples – the unmodified images that can fool models [92]. There are attempts to understand why such images can fool networks. In one of our research[240] we suggest the problem can lie in training images. For instance, in the ImageNet, there are 1300 miniskirt images in the training set, but only 55 are without a woman context. The model can learn spurious correlations, so any images with a woman's context can be wrongly classified as a miniskirt. Knowing these makes it possible to find natural adversarial examples – see figure 2.30

Figure 2.30: The example of are wrongly classified images because of existing spurious correlations. The captions below each image contains top-5 response from the ResNet-152 model. The image from [240]

2.3.2.3 *Defenses methods*

Several methods to defend against adversarial attacks have been proposed[278]. Unfortunately, all the methods can be effective only for some of the attacks. Some researchers focus on detecting them in the testing stage[149][157]. Other methods modify (squeeze) the input images to decrease the chance of adversarial attack. These methods include: feature squeezing[275] where the input image is modified by reducing the color bit depth or by using spatial smoothing, using JPEG transformation[58] which remove unwanted perturbation, using PixelCNN[225] based on a p-value calculated from a special network, methods based on autoencoders[79], a method of using specialists 1 ensemble [1], or a combination of several methods[89].

2.3.3 *Trustworthiness of AI*

Modern AI models have to be trustworthy. We can talk about that system only when the models are aware of out-of-distribution data. It is a necessary but not sufficient condition. This dissertation focused on this aspect, but the term of trustworthiness is broader.

High accuracy should not be the only criterion for evaluating modern ML models. Other evaluation metrics are significant, especially in systems in which human lives depend. The complex ML models, like deep learning models, are systems called often "black boxes". It is hard to receive any additional information about the decision from the model, especially "why" it was decided in that way. As Bodria et al. wrote in their survey[20] "modern, complex machine learning models hide the logic of their internal process (...) potential issues inherited by training on biased data".

There are many biased systems - for example, the COMPAS[198] system is a recidivism-risk scoring model biased on human races, or the Amazon recruiting system was biased on gender[120]. The authors of [26] also see the problem of biased datasets (in this paper on gender too). The typical example of the biased dataset in computer vision is the classification of huskies and wolves, where models learned to distinguish the classes by background[191]. It again confirms that the problem often lies in data. We suggested verifying the data [240] used for training the model. However, the deep learning model does not always learn features that we humans want them to learn. So the additional evaluation criteria are crucial here.

It is also a problem of law. For example, the European Union General Data Protection Regulation (GDPR) sees the problem and wants to give the "right to explanation" of all decisions made or supported by automated or artificially intelligent algorithms[41]. There is no consensus in research on how to measure the interpretability, or for whom the explanation should be generated (ie. ML specialists, end-users, lawyers) [266][7][29][141][53].

The trustworthiness also connects with the interpretability of the models. There are many ideas on how to try to explain how the network works[204][20]. In the computer vision field, the papers focus mainly on the show which part of the image is crucial for a decision. The most popular methods are in the family of saliency maps generation - the special map where each pixel is highlighted according to its contribution to classification. The popular methods are Guided Backprop[227], Grad-cam[215], Grad-cam++[32] or Score-cam[260]. The local interpretable model agnostic explanation - LIME[191] - also can help to understand how models see our world. This approach focuses on differences locally. The Concept Attribution is another technique where the model is tested on sensitivity for "concepts," i.e., how the shape of stripes contributes to zebra classification[117]. Another

group of methods is called "Prototypes" [116][34][81] where the similarity to a batch of ideal examples are analyzed to explain the model – it is possible also to add "criticism" (negative examples). There is also possible to create models which already can be interpretable "explanation by design" - so no "open the black boxes" [197][199].

Although the above methods have been known for years, we still have many problems to solve. There is no one the best method to verify the trustworthiness of the models. Additional research is necessary to understand better how the data influence the final model, how to verify the decision-making of the final model itself, and how to explain its output of it to humans.

2.3.3.1 *Robustness*

In the context of trustworthiness, one of the main concepts is robustness[288][5][276][263]. Following one of our papers [239]: "It is the characteristic that describes how effective the network is while being tested on the new independent dataset. This characteristic is broadly defined. It should enclose the factors that may influence of model's final prediction – for example: by checking the influence of existing biases or spurious correlations, changing the context (i.e., background) or image properties (i.e., image brightness contrast), or by using adversarial attacks or applying distortions to the image.".

Part II

Research

3

Research

3.1 Introduction and Chapter Plan

This chapter presents our research. The main thesis of this dissertation is that the safety and trustworthiness of AI models can be assured only when the models are aware that the unknown data exist - therefore, we showed and thoroughly researched the Out-of-Distribution detection techniques. Still, there are no clear conclusions and recommendations in the literature on which methods are most successful for a given recognition problem. Furthermore, our research suggest (and recent literature[243]) that the best method depends on pairs of known and unknown samples. However, the OoD samples are unknown by their nature. In this thesis, we looked at OoD from a practical perspective. The safety and trustworthiness models must work as open-set classification – we focused on how to force classic close-set models to be OoD aware.

First, we demonstrated the complexity of OoD detection. We pointed out how many decisions were to be taken – among others, we needed to choose the CNN model, how to extract features from the network, or how to evaluate the results. We analyzed the factors which have the most significant impact on the performance of OoD. Starting with a simple example, we were adding successively new variants like unknown datasets, OoD methods, and CNN models. It showed that there is no one perfect OoD method – the approach should be chosen based on pairs of known and unknown data. This situation is unacceptable in a real case problem because we do not know the unknown data. We also tested the OoD problem in the context of large-scale images, which is rare in the literature, but necessary to examine to enable methods in modern applications. We verified research hypotheses like if the better model in close-set problems is better for the OoD detection. We also promoted the LOF-based methods, which are rarely used in the OoD problem with deep learning. However, they achieved good, stable results and, in our opinion, should be presented in benchmark comparison. The detailed plan of the section 3.2 is as follows.

- In the subsection 3.2.1, we demonstrated a baseline OoD detection problem based on the known CIFAR-10 dataset and the ResNet-101 model. We analyzed the factors that impact OoD efficiently, showed how many decisions must be taken for the simple open-set approach, and explained the discussed task in detail. We need to choose, among others: unknown examples for evaluation, evaluation metrics, the feature extraction method, the feature post-processing method, and finally, the OoD method itself. Each of them significantly influences of OoD detection possibly yielding contradicting results and conclusions.

- In the subsection 3.2.2, we extended our baseline problem with two new CNN models and the Mahalanobis OoD method. The achieved results were the starting point for our research. We showed that the OoD method is not the only essential factor for detecting unknown samples.

- In the subsection 3.2.3, we performed in-depth analysis of OoD detection on one example CNN model - we choose the ResNet-101. We presented results for many different OoD methods, which can be split into distance-based (the Mahalanobis,

the MDistance), density-based (the LOF, the LOF_D, and the OSNN), based on the models of the probability of inclusion (the OpenMax), and based on maximum logits(the MaxSoftmax, MaxLogits). It allowed for a better understanding of each OoD method and has opened up the field for further research.

- In the subsection 3.2.4 we tested the OoD problem in the context of large-scale images. By large-scale data, we mean, i.e., a large number of training samples, a high number of known classes, and high image resolutions. Unfortunately, in most literature benchmarks, authors tested their solution based only on small-scale data. We used ImageNet as known data and the ImageNet-O and the Places365 as unknown data. The ImageNet-O contains natural adversarial examples, so we challenged the OoD methods. Moreover, we tested many popular CNNs architectures, which allowed us to check hypothesis that the better CNN model, the better efficiency of OoD detection, and we got inconclusive results.

In next section, we focused on a profound analysis of the popular OoD methods (Mahalanobis, Simple Unified Framework, and EVM) and their limitations and assumptions. This analysis allowed a better understanding of those methods and confirmed the results from the first section that no method is perfect. Each method could work well only for specific assumptions, which sometimes cannot be met working in high-dimensional features by using deep learning models. The detailed plan of the section 3.3 is as follows.

- In subsection 3.3.1 we focused on the Mahalanobis distance, which is one of the most popular OoD methods recently. We discussed why the OoD problems with large signal-to-noise ratios may still rely on Mahalanobis distance. Beyond classic evaluation on benchmark datasets, we performed many simulations testing the sensitivity of the Mahalanobis method to the dimensionality of data, size of training data, and shift/difference between known and unknown data. We showed and compared different variants of this method.

- Next, in the subsection 3.3.2, we focused on analyzing the popular OoD detection method proposed by Lee et al. in the paper "A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks". This method is one of the state-of-the-art methods for the OoD problem. We showed the strengths of this approach and showed that using the Mahalanobis is not the main one. We discuss each of the major concepts used in this paper.

- In the subsection 3.3.3 we analyzed the EVM. We discussed the EVM's theoretical assumptions and claimed that they are not always fulfilled in the context of CNNs and some popular datasets. The promoted Wailbull parametric distribution, in reality, has a minor influence on the final method.

One of the hypotheses was that current, widely used CNNs models do not generate enough robust features. That means that these networks can not well map images from the image space into features space, contrary to human intuition. Good examples are adversarial images where CNNs generate different features for very similar images. Similar, as we have shown later, images entirely different from the human perspective (i.e., dog and house number) are very close in feature space. Generating robust features seems crucial because good mapping distinguishes the known and unknown samples better.

In the next section, we focused on data representations and suggested that OoD methods depend more likely to feature space than the method itself. OoD results are more sensitive for the CNN model, feature extraction method, hyperparameters of fitting the network,

the input preprocessing, and others. The goal of CNNs is to transform the input data from image space into feature space. However, CNNs mismatch semantic and feature space similarity for many samples. In that case, any OoD strategy may fail. We demonstrated this phenomenon. As one of our contributions, we showed that the proper features obtained strategy might follow to improve OoD results. The detailed plan of the section 3.4 is as follows.

- In the subsection 3.4.1 we showed the influence of the feature extraction on OoD detection. We checked if the OoD detection algorithms could build better decision boundaries when using other methods for obtaining features from CNNs than classic GAP. It is one of our main contributions. We proposed to use a new hyperparameter (feature extraction strategy), which, when properly chosen, allowed boosting final results by a few percentage points. It can increase OoD efficiency due to various approaches that can focus on different components (e.g., on edges, patterns, or whole objects), so for different pairs of ID and OoD, different features can be useful to separate data.

- Many OoD detection methods can work poorly in high-dimensional space due to the curse of dimensionality. In the subsection 3.4.2, we focused on reducing the feature vector. We checked how much that vector can be reduced by using the popular technique (L2 normalization -> PCA reduction -> whitening -> L2 normalization) and how much it influenced the OoD evaluation.

- In subsection 3.4.3, we checked the influence of the image augmentation strategy on the model's robustness in close-set classification and OoD detection problems. The networks build their decision boundaries during the learning process, so the proper strategy should be substantial. We wanted to test which augmentation strategy can help to improve OoD results.

- In the subsection 3.4.4, we noticed that the models with the same architectures and similar hyperparameters with similar close-set accuracy could achieve completely different results. We also tested the stability of OoD detection during the learning process. That suggested that the CNN model state could affect the results very strongly, and in consequence, it can spark an entire field of discussion in context of the literature results.

- We analyzed the hypothesis that there is a mismatch between image and features representations in the subsection 3.4.5. We believe that the characteristics of the training dataset cause this mismatch – different models and feature extraction methods achieved similar results on the same data. Using this observation, we could generate subsets of pair (in-distribution and out-of-distribution) data, which were extremely easy or hard for the OoD detection methods.

- In the last subsection 3.4.6, we analyzed OoD detection as the detector of an adversarial attack. The thesis was that adversarial images move attacked images towards another class group in the feature space. However, the close-set decision boundaries are broader than that created by the OoD methods. So we suggested that OoD approaches should detect the attacked image. Moreover, the adversarial attack methods really attack the features based on the GAP approach, so using the OoD with other feature extraction should help the detection problem. We tested that hypothesis too.

.

3.1.1 *Research Limitations and Assumptions*

The problem of Out-of-Distribution examples detection is vast. Because of that, we focused on specific areas in our research. The main goal of OoD detection is to separate (never seen before) images into two groups, known (in-distribution) and unknown (out-of-distribution), based on two pieces of information. The first is the set of example images split according to labels/categories (called the training set). The second is the close-set model designed to classify images to one of these labels. Next, for most of our research, we made the following assumptions:

1. Models could be based on CNN architectures pre-trained for close-set classification problems. We did not want to modify or retrain the models. However, we still could return any information from inside the model as features, logits, or information about the chosen class. The motivation here was covering a classic, practical case where the solid classification system can already exist, so the goal would be to extend that system to OoD detection – without, often costly, modification in the life cycle/pipeline of model generation. The exception to this were studies shown in subsections 3.4.3 and 3.4.4.

2. It was possible to use only three subsets of data: train, test, and unknown. The train and test should have the same data distribution. The unknown should contain OoD images. We have worked on benchmark datasets like the CIFAR-10 or the ImageNet, so we could use a training subset of images to fit the model and the OoD detection algorithm. Next, we could use test-known data and the same number of unknown images from another dataset as OoD, like the CIFAR-100 or the Places365. Both were used to evaluate the model. The motivation for that limit was that we did not want the network to see any unknown images. We assumed that all decision boundaries should be based on in-distribution data.

Based on the above assumptions, we resigned from OoD methods like these based on contrastive learning (retraining the network) or outlier exposure (retraining the network and using extra data "known unknown examples"). In our resarch, we focused on such problems like: model architecture, feature extraction method, the number of training examples, near or far OoD examples, low- and large-scale datasets, variety of the different number of labels in in-distribution data resolution, using post-processing of features and many others.

3.2 Complexity of the OoD Problem

This section focused on our preliminary research on the Out-of-Distribution detection problem. We showed how complex that problem is and how many decisions must be taken, and their effect on the final results. Our goal was showed that there is no one perfect method, and still, there are many areas to develop further. We demonstrated step by step how to perform OoD detection. We tested the classic methods on popular benchmark datasets. We showed, among other conclusions, that well-known but unpopular (in the context of deep learning) methods based on LOF achieved promising results. Moreover, we tested the OoD methods in high-resolution images, which is still rare in literature.

3.2.1 *Simple Example to Demonstrate the Complexity of OoD Detection Problem*

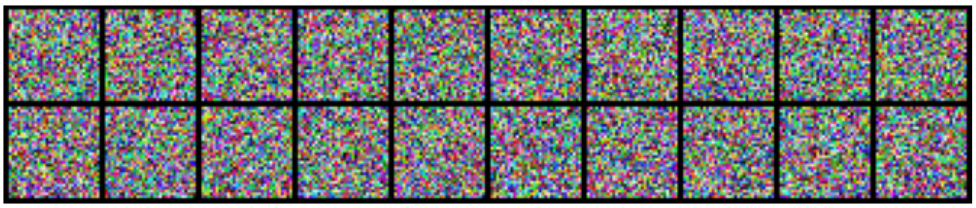
Section Objectives In this subsection, we wanted to demonstrate a simple example of an Out-of-Distribution detection problem. It should allow us to demonstrate how complex and vast OoD detection is. We analyzed the factors, which have the most significant impact on the performance of OoD. Our example was based on the CIFAR-10 (as the known data) and three other sets (as the unknown data). We used the popular ResNet-101 model.

We formulated a problem as proposing a method to distinguish in-distribution from the CIFAR-10 and out-of-distribution data. The CIFAR-10 contains 60,000 32x32 color images divided into 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. There are 5,000 images per class in the training set and 1,000 in the test set. We needed to define a set of unknown examples. There was the first difficulty here because we did not know what was unknown. The literature has suggested finding near or far images as unknown examples. Near means that the images should be strongly similar to training examples, and far, they should be entirely different. Although we argued above in the following sections, we used the CIFAR-100 as the near-OoD example for our experiments. The CIFAR-100 contains similar images as the CIFAR-10 but with different classes. The SVHN was used as the far-OoD example. The SVHN contains real-world images of Street View House Numbers from Google Street View - they are easily distinguishable for humans compared to CIFARs ones. Moreover, we used extra-far-OoD - the Noise. The Noise is randomly generated images - a theoretically straightforward task to identify these. To evaluate the OoD methods, we used the testing partitions of the in-distribution datasets and the given out-of-distribution dataset, with a 1:1 proportion of known and unknown samples. See the figure 3.1 to visually compare the sets.

Next, we needed to define metrics to evaluate OoD methods. In many OoD benchmarks, the most noteworthy ones are DTACC, AUC, and AUPR. These metrics assume the binary classification problem.
They move the decision boundaries and see how known and unknown data are separable. The detection accuracy (DTACC) defines the ratio of correct classification of the test and unknown examples to all examples for the best value of the internal threshold. The internal threshold defines how to deal with the single sample - if it should be treated as known or unknown. The Area Under Receiver Operating Characteristic curve (AUROC - or AUC as we denote in all tables) is the area under the false positive rate against the true positive rate curve. It defines the OoD method's ability to discriminate between test examples and OoD examples. The AUPR is defined as the Area Under the Precision and Recall curve. We can split it into two sub-metric AUPR IN or AUPR OUT, where "positive examples" are from the test or unknown examples. We denoted AUPR as the mean from both due to equal samples in both sets. The higher the values, the better the OoD method was for all metrics.

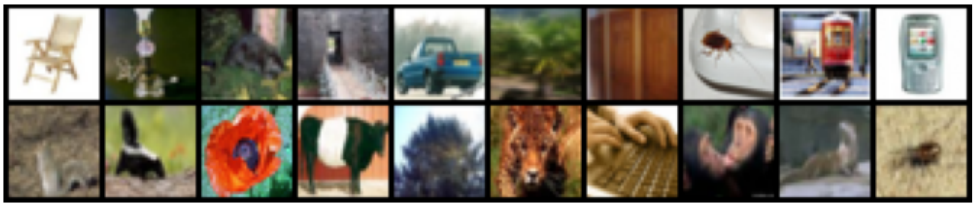
Those ideas are not well-suitable for local analysis. Moreover, based on them is hard to set the decision boundaries in real-practice problems when we do not know any of the OoD data. Due to the above, we also considered the metric True Negative Rate at 95% True Positive Rate (TNR at TPR 95%). It can be interpreted as the probability of correctly classifying the Out-of-Distribution examples when the In-Distribution (test) samples are classified as high as 95%. There are other variations, like TNR at TPR 99% or FPR at TPR 95%, but the main idea is still the same. The decision is set based only on the in-Distribution data.

(a) The CIFAR-10(known)



(b) The Noise

(c) The SVHN



(d) The CIFAR-100

Figure 3.1: Examples of images from known and unknown datasets used in experiments.

To show the complexity of the OoD detection, we made the following decisions. Firstly, we needed to prepare a pre-trained model fitted on training examples. So, we needed to choose architecture and hyperparameters. Intuition can suggest that the better the CNN model, the better the "understanding" of images (more robust features extracted from the network), so the better the efficiency of OoD detection should be. However, our study showed that is not always true, but we discussed it later. We chose the ResNet-101 architecture (it achieved 94.79% accuracy on the CIFAR-10) in our simile example due to its popularity. Next, we needed to choose the OoD method itself. We used MaxSoftmax[90] due to the simplification of this approach. The next step was choosing a features extraction approach from the model. We used a logits from the pre-trained model (due to MaxSoftmax). In this simple case, we resigned from using a feature post-processing method.

Finally, we could test the whole OoD approach. The results are presented in table 3.1. We achieved the expected results - the further the unknown examples, the better the results. The only surprising thing is the relatively low metrics values for the Noise. That task seems extremely easy for humans – however, 13.37% images from the test or the unknown subsets were misclassified.

Table 3.1: Example of OoD detection for the CIFAR-10 as the in-distribution data. We used the ResNet-101 model and MaxSoftmax method.

Out-dist	DTACC	AUC	AUPR	TNR at TPR 95%
Noise	86.63	91.41	90.08	37.94
SVHN	83.77	88.93	87.40	32.11
CIFAR-100	79.40	84.73	82.58	25.46

Section Summary We demonstrated how complex the OoD detection is. We formulated the problem and proposed a method to distinguish in-distribution and out-of-distribution data. For a given model and the known images (split into train and test subsets), we needed to choose: unknown examples for evaluation, evaluation metrics, the feature extraction method, the feature post-processing method, and finally, the OoD method itself. Each of them significantly influences the OoD detection, possibly yielding contradicting results and conclusions. Although the goal was to open an already fitted, close-set model, it may be worth considering changing the architecture or hyperparameters of the model too. The following subsections show that each decision can significantly influence the final results.

3.2.2 *Extending the Simple Example by Adding New CNNs Models and the Mahalanobis Method*

Section Objectives We decided to develop further our OoD example. We added a new OoD method based on the Mahalanobis distance, which worked with features (not logits) and new CNN models. The goal was to demonstrate some unexpected results and draw further conclusions.

We decided to extend the baseline example to show challenges for OoD detection. We added a new method based on Mahalanobis distance (see more details section 2.3.1.3 and two new CNN models with much simpler and more complex architectures. The simple model used only two convolutional layers and achieved 79.45% accuracy. The complex architecture was based on PyramidNet-272[84] and Divide and Co-training[287], which is one of the state-of-the-art solutions for the CIFAR-10. It was called SplitNet-PyramidNet-272, and it achieved 98.71% accuracy. We used the standard approach as the feature extraction for Mahalanobis using a vector from the Global Average Pooling (GAP) layer. Whereas for the model SplitNet-PyramidNet-272, we used the mean of GAP from all sub-networks due to the specific model's architecture.

The results are presented in table 3.2. We can see general rules like (1) again the further the unknown examples, the better the results, (2) the better CNN model, the better results, (3) the Mahalanobis returns better results than the much simpler method MaxSoftmax.

However, we observed results contrary to these rules. The GAP features generated by Simple CNN for the Noise were much more likely separable from known examples than from the ResNet-101 model when using the Mahalanobis method. The Simple CNN could return good results only for the extra-far-OoD as the Noise example and only for the Mahalanobis method. It did not work well for OoD detection for other datasets and methods. See the TNR at TPR 95% metric. Only the Noise could be treated as the solved

Table 3.2: Extended the simple examples of OoD detection for the CIFAR-10 as the in-distribution data. We used the three models: Simple CNN, the ResNet-101, and the SplitNet-PyramidNet-272 with two OoD methods: MaxSoftmax and Mahalanobis.

Out-dist	Model	Method	DTACC	AUC	AUPR	TNR at TPR 95%
Noise	Simple CNN	MaxSoftmax	53.80	44.30	48.33	0.73
		Mahalanobis	99.66	99.87	99.77	100.00
	ResNet-101	MaxSoftmax	86.63	91.41	90.08	37.94
		Mahalanobis	89.98	94.99	94.47	63.06
	SplitNet-PyramidNet-272	MaxSoftmax	97.19	98.09	96.76	99.22
		Mahalanobis	99.97	99.98	99.98	100.00
SVHN	Simple CNN	Mahalanobis	58.59	60.13	61.35	9.20
		MaxSoftmax	76.06	81.56	80.05	16.00
	ResNet-101	MaxSoftmax	83.77	88.93	87.40	32.11
		Mahalanobis	87.66	93.91	93.73	57.87
	SplitNet-PyramidNet-272	MaxSoftmax	94.52	97.51	97.11	93.77
		Mahalanobis	96.05	98.74	98.58	96.99
CIFAR-100	Simple CNN	MaxSoftmax	69.40	75.18	74.34	15.63
		Mahalanobis	56.64	59.10	58.79	14.67
	ResNet-101	MaxSoftmax	79.40	84.73	82.58	25.46
		Mahalanobis	84.47	91.21	90.97	46.03
	SplitNet-PyramidNet-272	MaxSoftmax	85.69	90.61	88.74	72.67
		Mahalanobis	89.00	95.09	94.89	78.08

problem (still 5% of known examples are rejected). The widely spread model, as the ResNet-101, returned relatively low results too. Even when using the Mahalanobis with the SplitNet-PyramidNet-272 for the SVHN as the unknown, we achieved "only" 98.74% AUC and 96.99% TNR at TPR 95%. We used the word "only", because the task is trivial for humans - see again figure 3.1 to compare the CIFAR-10 and the SVHN subsets. Here we used the SOTA model for the CIFAR-10 and the popular OoD technique, and still, the method failed for many images.

The above results were unexpected, and they were the starting point for our research. In next section, we discussed and criticized many assumptions and popular approaches used in the OoD detection problem.

Section Summary We extended our results by adding a new method based on the Mahalanobis distance and new models, simple and SOTA. Our tests showed that the OoD performance depends on many factors, and the OoD method itself is not the essential in this case.

3.2.3 *Comprehensive Comparison of OoD Methods for the Resnet-101*

Section Objectives We decided to develop our example further to see more contradicting results. We focused on deep analysis of OoD detection on one example CNN model - we choose the ResNet-101. We extend OoD methods by LOF, OpenMax, OSNN, and our custom approaches LOF_D, MDistance, and MaxLogits. For methods which need to use distances, we used both Eudlicean and Cosine distances. To compare with SOTA results, we showed the results also for ODIN and Unified Framework. We used the same datasets as in the previous section.

We decided to present the results for the ResNet-101 for different OoD methods - see table 3.3. The goal was to show that different approaches achieved different results with no clear pattern – or even when such patterns seemed to be visible, we argued them in later parts of this work. We added the following OoD methods: the LOF(Local Outlier Factor) and our modification the LOF_D (described in section 2.3.1.4), our MaxLogits (described in section 2.3.1.2), our MDistance (described in section 2.3.1.6), the OpenMax (described in section 2.3.1.8) and OSNN (described in section 2.3.1.5). The methods which needed to use distance used Euclidean or Cosine one. Each method used the same features inputs (based on the GAP). The MaxLogits and the MaxSoftmax use the original CNN classifier to obtain logits. The above OoD methods work in different ways. They can be split into distance-based (the Mahalanobis, the MDistance), density-based (the LOF, the LOF_D, and the OSNN), based on the models of the probability of inclusion (the OpenMax), and based on maximum logits (the MaxSoftmax, MaxLogits) methods. Using different approaches allowed for overall analysis and basic understanding of each OoD method.

Moreover, we used two additional OoD methods ODIN (described in section 2.3.1.9) and Unified Framework (described in section 2.3.1.10). We used original codes (outside resources) for these methods. In this section, the main goal was to compare the OoD methods without using the additional approaches (that those methods do). Due to the above, we denoted these methods by *. We showed results for the ODIN and the Unified Framework for a more comprehensive comparison. Note that ODIN is based on the MaxSoftmax. However, it used the input preprocessing approach. The Unified Framework uses some unknown examples to calibrate regression weights, and it uses features from different parts of the network (not only GAP as other tested methods). This method is based on the Mahalanobis and also uses input preprocessing. We suggested (and verified in the next sections) that the results for these methods could be well due to the use of additional approaches - not the OoD methods themselves.

Table 3.3: Results for different OoD methods based on the CIFAR-10 as the in-distribution data with using the ResNet-101 and the GAP. We tested the following methods: distance-based (the Mahalanobis, the MDistance), density-based (the LOF, the LOF_D, and the OSNN), based on the models of the probability of inclusion (the OpenMax), and based on maximum logits(the MaxSoftmax, MaxLogits) and SOTA methods ODIN and Unified Framework.

Out-dist	Method	DTACC	AUC	AUPR	TNR at TPR 95%
Noise	LOF_{Cos}	87.64	93.39	93.00	52.61
	LOF_{Euc}	87.77	92.85	91.32	47.71

Table 3.3: Results for different OoD methods based on the CIFAR-10 as the in-distribution data with using the ResNet-101 and the GAP. We tested the following methods: distance-based (the Mahalanobis, the MDistance), density-based (the LOF, the LOF_D, and the OSNN), based on the models of the probability of inclusion (the OpenMax), and based on maximum logits(the MaxSoftmax, MaxLogits) and SOTA methods ODIN and Unified Framework.

Out-dist	Method	DTACC	AUC	AUPR	TNR at TPR 95%
	LOF_D_{Cos}	87.54	93.02	89.13	36.27
	LOF_D_{Euc}	87.77	92.70	88.24	37.73
	Mahalanobis	89.98	94.99	94.47	63.06
	MaxLogits	88.72	93.50	91.78	53.26
	MaxSoftmax	86.63	91.41	90.08	37.94
	$MDistance_{Cos}$	82.91	88.14	86.15	29.18
	$MDistance_{Euc}$	82.69	87.94	85.92	30.31
	$OpenMax_{Cos}$ a=3, t=200	84.70	87.52	75.01	0.00
	$OpenMax_{Euc}$ a=3, t=200	84.70	87.71	75.36	0.00
	$OSNN_{Cos}$	86.00	91.17	90.21	35.89
	$OSNN_{Euc}$	86.05	91.03	89.87	35.33
	ODIN*	97.00	99.00	99.00	99.00
	Unified Framework*	100.00	100.00	99.00	100.00
SVHN	LOF_{Cos}	86.22	92.39	92.03	48.96
	LOF_{Euc}	86.08	91.81	90.71	45.64
	LOF_D_{Cos}	86.24	92.05	88.28	34.79
	LOF_D_{Euc}	86.10	91.60	87.28	36.98
	Mahalanobis	87.66	93.91	93.73	57.87
	MaxLogits	84.81	90.11	88.18	39.58
	MaxSoftmax	83.77	88.93	87.40	32.11
	$MDistance_{Cos}$	81.28	86.53	84.34	25.30
	$MDistance_{Euc}$	80.06	84.71	81.72	19.15
	$OpenMax_{Cos}$ a=3, t=200	83.14	86.52	74.30	0.00
	$OpenMax_{Euc}$ a=3, t=200	83.36	86.83	74.73	0.00
	$OSNN_{Cos}$	84.06	89.77	88.78	32.37
	$OSNN_{Euc}$	84.16	89.69	88.56	31.59
	ODIN*	83.00	89.00	89.00	51.00
	Unified Framework*	94.00	98.00	98.00	94.00
CIFAR-100	LOF_{Cos}	82.31	88.96	88.45	37.07
	LOF_{Euc}	81.69	88.07	86.81	34.31
	LOF_D_{Cos}	82.23	88.77	85.28	26.35
	LOF_D_{Euc}	81.63	87.94	83.99	27.75

Table 3.3: Results for different OoD methods based on the CIFAR-10 as the in-distribution data with using the ResNet-101 and the GAP. We tested the following methods: distance-based (the Mahalanobis, the MDistance), density-based (the LOF, the LOF_D, and the OSNN), based on the models of the probability of inclusion (the OpenMax), and based on maximum logits(the MaxSoftmax, MaxLogits) and SOTA methods ODIN and Unified Framework.

Out-dist	Method	DTACC	AUC	AUPR	TNR at TPR 95%
	Mahalanobis	84.47	91.21	90.97	46.03
	MaxLogits	79.63	84.99	82.44	27.19
	MaxSoftmax	79.40	84.73	82.58	25.46
	MDistance $_{\text{Cos}}$	77.66	82.87	80.10	22.05
	MDistance $_{\text{Euc}}$	77.38	82.68	80.14	23.71
	$\text{OpenMax}_{\text{Cos}}$ a=3, t=200	79.31	83.51	71.90	0.00
	$\text{OpenMax}_{\text{Euc}}$ a=3, t=200	79.41	83.66	72.29	0.00
	OSNN_{Cos}	79.81	86.02	84.92	26.17
	OSNN_{Euc}	79.56	85.72	84.46	25.73
	ODIN*	79.00	85.00	83.00	42.00
	Unified Framework*	83.00	89.00	90.00	47.00

As we mentioned, the conclusions based on analyzing table 3.3 can be inaccurate. However, some of the following ones are based on our more complete understanding described in the next sections. The Unified Framework achieved the best results with nearly "solving" two tasks for the Noise and SVHN. However, this method "sees" some unknown examples during training add it uses features from different parts of the network, which can be better separable the data than the GAP. For the CIFAR-100, the classic Mahalanobis achieved even better results without using any of the OoD data. The ODIN method usually works well for extra-far-OoD examples. The key methods seemed to be based on the LOF, Mahalanobis, and Logits. However, we showed (in the following sections) examples where also other OoD methods could achieve good results. Interestingly, the TNR at TPR 95%, where the OpenMax completely failed on all tasks, and all methods, including Unified Framework, failed on the near-OoD task based on the CIFAR-100. MaxLogits was better than the MaxSoftmax. The poor results for the Noise are unexpected, but they could be specific to the model. The input preprocessing might force the data to be more separable. There was no clear advantage to using Euclidean or Cosine distance.

The LOF-based achieved good and stable results. One of the main conclusions from this work is to promote this method. Those methods are not commonly used in the literature, giving way to the Mahalanobis. They are based on density, making them less dependent on high-dimensional curiosity. The similar method OSSN always worked worst in our experiments.

Section Summary We tested many methods for OoD results based on the ResNet-101 model. No method has managed to overcome the others. The most stable methods were Mahalanobis, LOF-based, and MaxLogits. They worked using a different methodology,

which suggests that the OoD methods could not be the key to OoD problems. We argued that there are other, more important, factors which make the features more or less separable, like the model, the data, or feature extraction strategy. The good results achieved for Unified Framework methods suggested that too. This approach used techniques that made the data more distinguishable, although the Mahalanobis distance was used as OoD method.

3.2.4 *Applying OoD Methods to Large-Scale Images*

Section Objectives By large-scale data, we mean, i.e., a large number of training samples, a high number of known classes, and high image resolutions. The new OoD methods proposed by literature are not usually tested on large-scale data. In this section, we would like to verify if the OoD approaches can be used on those images by models trained on the ImageNet dataset. Moreover, because there are many publicly available pre-trained models for this dataset, we would like to check the hypothesis that the better the close-set model, the better OoD detection efficiency. We used two datasets as OoD: ImageNet-O and Places365. The ImageNet-O contains natural adversarial examples, so we checked incidentally how the OoD methods performed on these data.

We decided to test high-resolution images in OoD detection problems. For this purpose, we used the pre-trained CNN models on the ImageNet dataset. In table 3.4 we presented them together with top-1 and top-5 accuracy. The models use different architecture with different ideas developed over the years of developments of CNNs. We used the models with different levels of accuracy to test the hypothesis that the better model, the better OoD detection. To simplify the calculations for the ImageNet variants, in our experiments we used only the 50 first classes from the ImageNet as known data. We note that the all models were trained on all 1000 classes.

In table 3.5 we showed the results for two datasets ImageNet-O and Places365. For the AlexNet, the achieved results were poor for most methods for both datasets – only LOF_{Euc} for ImageNet-O worked quite well (94% of AUC, 75.4% of TNR at TPR 95%). For the next model ResNet-18 the conclusions are similar - most methods worked poorly - but the $\text{LOF\_D}_{\text{Euc}}$ solved the problem of detection OoD forImageNet-O (both metrics were above 99%). The second best method was Mahalanobis. For the next one, the VGG-16, the LOF_D based method was above others and achieved quite good results for both datasets. For the next one, the Inception v3, the $\text{LOF\_D}_{\text{Cos}}$ solved the problem for the ImageNet-O. There was quite a different result for the $\text{LOF\_D}_{\text{Euc}}$ for this dataset when the TNR at TPR 95% dropped significantly. The Places365 was too hard a problem for this model – only the Mahalanobis achieved acceptable results. For the ResNet-101 model, LOF-based methods and Mahalanobis were the best, similar for the DenseNet-161, but here the $\text{MDistance}_{\text{Cos}}$ achieved good results too, and the metric TNR at TPR 95% for LOF_D approaches dropped. The WideResNet-101-2 with LOF_D approaches achieved the best results for the ImageNet-O over all others. For Places365, it achieved good results too. The EfficientNet achieved stable results at a high level for both datasets.

We can see that the MaxLogits and Max Softmax failed for the ImageNet-O dataset – it is expected because it was designed to fool the close-set classifiers. However, for the Places365 they achieved poor results too. The most stable methods were LOF_Ds. The Mahalanobis seemed not to work as well as in low-resolution data.

Table 3.4: Accuracy of the Imagenet Models

Name	Acc top-1	Acc top-5
AlexNet	56.52	79.07
ResNet-18	69.76	89.08
VGG-16	71.59	90.38
Inception v3	77.29	93.45
ResNet-101	77.37	93.55
DenseNet-161	77.14	93.56
WideResNet-101-2	78.85	94.28
EfficientNet-B3	81.1	-

Table 3.5: Results for different OoD methods based on the high-resolution images. We used the ImageNet as known and the ImageNet-O and the Places365 as unknown data.

Out-dist	Model	Method	DTACC	AUC	AUPR	TNR at TPR 95%
ImageNet-O	AlexNet	LOF_{Cos}	84.16	92.01	91.78	64.10
		LOF_{Euc}	86.84	94.09	93.91	75.47
		LOF_D_{Cos}	90.07	81.13	77.59	0.10
		LOF_D_{Euc}	89.52	89.69	89.56	42.65
		Mahalanobis	81.63	88.10	87.39	61.47
		MaxLogits	65.89	68.70	67.76	6.85
		MaxSoftmax	59.88	62.20	62.14	8.14
		$MDistance_{Cos}$	59.83	62.20	59.56	9.68
		$MDistance_{Euc}$	50.12	35.75	41.01	1.64
		$OSNN_{Cos}$	62.88	67.01	66.80	10.18
		$OSNN_{Euc}$	62.88	67.05	66.64	9.43
	ResNet-18	LOF_{Cos}	88.75	95.47	95.47	76.51
		LOF_{Euc}	87.29	94.27	94.20	71.80
		LOF_D_{Cos}	95.01	93.78	91.42	33.56
		LOF_D_{Euc}	99.60	99.68	99.56	99.20
		Mahalanobis	92.55	97.56	97.46	88.38
		MaxLogits	68.47	72.57	70.47	8.89
		MaxSoftmax	58.34	56.99	57.16	5.21
		$MDistance_{Cos}$	86.87	91.55	89.35	48.41
		$MDistance_{Euc}$	55.56	44.29	48.58	0.65
		$OSNN_{Cos}$	75.15	80.22	78.39	14.50
		$OSNN_{Euc}$	73.11	78.90	77.49	16.63
	VGG-16	LOF_{Cos}	81.28	88.86	88.88	43.00

Table 3.5: Results for different OoD methods based on the high-resolution images. We used the ImageNet as known and the ImageNet-O and the Places365 as unknown data.

Out-dist	Model	Method	DTACC	AUC	AUPR	TNR at TPR 95%
		LOF_{Euc}	92.35	97.22	96.99	89.03
		LOF_D_{Cos}	96.08	98.84	98.82	93.69
		LOF_D_{Euc}	96.67	95.14	92.58	57.65
		Mahalanobis	86.17	92.19	91.46	74.63
		MaxLogits	73.68	78.91	76.93	11.87
		MaxSoftmax	60.50	61.75	61.95	7.20
		$MDistance_{Cos}$	84.16	89.91	87.56	50.20
		$MDistance_{Euc}$	50.00	24.83	36.31	1.54
		$OSNN_{Cos}$	72.89	79.34	77.75	20.51
		$OSNN_{Euc}$	74.06	79.73	78.00	18.77
	Inception v3	LOF_{Cos}	76.59	82.21	81.70	19.56
		LOF_{Euc}	83.22	89.56	89.02	40.07
		LOF_D_{Cos}	99.45	99.53	99.22	99.06
		LOF_D_{Euc}	95.73	92.09	87.67	0.15
		Mahalanobis	89.18	94.83	94.28	68.52
		MaxLogits	72.09	75.52	70.77	22.19
		MaxSoftmax	64.72	67.64	64.56	13.55
		$MDistance_{Cos}$	84.46	85.42	81.93	7.85
		$MDistance_{Euc}$	59.26	51.88	54.33	2.08
		$OSNN_{Cos}$	78.80	84.50	82.75	20.51
		$OSNN_{Euc}$	82.35	88.72	87.54	38.98
	ResNet-101	LOF_{Cos}	88.98	95.75	95.75	75.12
		LOF_{Euc}	90.99	97.18	97.18	83.76
		LOF_D_{Cos}	89.60	95.57	95.79	71.15
		LOF_D_{Euc}	91.33	91.50	89.77	28.55
		Mahalanobis	92.48	97.42	97.31	86.10
		MaxLogits	50.00	15.12	33.06	0.20
		MaxSoftmax	50.00	40.27	43.25	0.50
		$MDistance_{Cos}$	84.38	89.63	87.02	34.56
		$MDistance_{Euc}$	58.81	57.99	56.68	4.52
		$OSNN_{Cos}$	76.89	82.82	81.23	21.65
		$OSNN_{Euc}$	77.75	84.08	82.75	27.51
	DenseNet-161	LOF_{Cos}	94.46	98.55	98.54	93.79
		LOF_{Euc}	95.11	98.80	98.71	95.03
		LOF_D_{Cos}	96.75	94.05	89.94	0.10

Table 3.5: Results for different OoD methods based on the high-resolution images. We used the ImageNet as known and the ImageNet-O and the Places365 as unknown data.

Out-dist	Model	Method	DTACC	AUC	AUPR	TNR at TPR 95%
		LOF_D_{Euc}	96.77	94.05	89.98	0.55
		Mahalanobis	94.94	98.40	98.24	94.59
		MaxLogits	59.66	61.36	58.41	9.53
		MaxSoftmax	55.36	54.77	53.48	5.96
		$MDistance_{Cos}$	92.58	96.32	95.23	85.95
		$MDistance_{Euc}$	65.32	67.51	65.63	6.40
		$OSNN_{Cos}$	79.97	86.23	84.68	28.65
		$OSNN_{Euc}$	80.69	86.69	84.95	26.76
	WideResNet-101-2	LOF_{Cos}	90.39	96.17	95.99	77.46
		LOF_{Euc}	95.33	98.90	98.82	95.13
		LOF_D_{Cos}	99.55	99.76	99.58	99.26
		LOF_D_{Euc}	99.55	99.58	99.25	99.26
		Mahalanobis	92.25	97.14	96.92	85.90
		MaxLogits	64.20	67.71	65.48	13.95
		MaxSoftmax	60.50	62.80	60.99	11.82
		$MDistance_{Cos}$	94.74	97.54	96.71	93.64
		$MDistance_{Euc}$	86.92	91.16	89.06	38.58
		$OSNN_{Cos}$	84.76	90.31	88.53	36.44
		$OSNN_{Euc}$	85.33	90.69	88.81	39.42
	EfficientNet-B3	LOF_{Cos}	80.88	90.44	90.95	57.70
		LOF_{Euc}	97.64	99.58	99.56	98.16
		LOF_D_{Cos}	97.77	96.31	93.02	99.06
		LOF_D_{Euc}	97.79	98.32	97.93	99.20
		Mahalanobis	94.39	97.81	97.57	90.52
		MaxLogits	60.85	58.38	57.37	16.29
		MaxSoftmax	63.68	64.89	61.95	15.04
		$MDistance_{Cos}$	95.93	97.48	95.91	96.33
		$MDistance_{Euc}$	76.61	81.65	80.19	18.92
		$OSNN_{Cos}$	86.62	92.16	90.52	44.89
		$OSNN_{Euc}$	90.19	94.81	93.26	70.36
Places365	AlexNet	LOF_{Cos}	69.79	76.26	75.67	20.51
		LOF_{Euc}	76.56	83.08	81.30	36.84
		LOF_D_{Cos}	78.43	69.90	64.67	10.62
		LOF_D_{Euc}	73.36	67.44	66.39	41.06
		Mahalanobis	70.33	74.29	72.21	32.82

Table 3.5: Results for different OoD methods based on the high-resolution images. We used the ImageNet as known and the ImageNet-O and the Places365 as unknown data.

Out-dist	Model	Method	DTACC	AUC	AUPR	TNR at TPR 95%
		MaxLogits	77.33	83.87	82.63	24.53
		MaxSoftmax	71.43	77.82	76.96	22.24
		$\text{MDistance}_{\text{Cos}}$	60.55	63.98	61.90	12.02
		$\text{MDistance}_{\text{Euc}}$	50.00	42.52	44.22	1.39
		OSNN_{Cos}	63.93	68.49	67.79	9.68
		OSNN_{Euc}	63.95	68.03	67.19	9.43
	ResNet-18	LOF_{Cos}	82.15	88.70	87.88	34.76
		LOF_{Euc}	74.11	82.01	81.77	38.93
		$\text{LOF\_D}_{\text{Cos}}$	95.08	92.72	91.74	23.48
		$\text{LOF\_D}_{\text{Euc}}$	80.56	75.11	72.18	42.05
		Mahalanobis	86.10	92.87	92.69	69.12
		MaxLogits	81.58	88.33	87.23	45.33
		MaxSoftmax	74.68	82.05	80.54	30.39
		$\text{MDistance}_{\text{Cos}}$	73.68	78.05	74.92	9.14
		$\text{MDistance}_{\text{Euc}}$	52.66	43.93	47.38	2.28
		OSNN_{Cos}	74.63	79.87	78.02	14.25
		OSNN_{Euc}	71.97	76.91	75.49	13.60
	VGG-16	LOF_{Cos}	71.65	75.35	74.46	10.62
		LOF_{Euc}	84.04	89.58	87.99	65.64
		$\text{LOF\_D}_{\text{Cos}}$	95.80	95.79	94.61	62.86
		$\text{LOF\_D}_{\text{Euc}}$	96.52	98.29	98.25	75.92
		Mahalanobis	77.56	81.12	78.49	54.57
		MaxLogits	86.69	92.61	92.06	58.14
		MaxSoftmax	78.23	85.85	85.30	40.52
		$\text{MDistance}_{\text{Cos}}$	83.59	89.73	87.87	48.21
		$\text{MDistance}_{\text{Euc}}$	51.24	37.66	43.02	1.19
		OSNN_{Cos}	75.02	81.34	79.92	21.15
		OSNN_{Euc}	74.68	79.89	78.30	18.72
	Inception v3	LOF_{Cos}	70.38	73.66	72.97	9.53
		LOF_{Euc}	79.32	85.97	84.65	30.98
		$\text{LOF\_D}_{\text{Cos}}$	64.65	42.96	52.85	28.30
		$\text{LOF\_D}_{\text{Euc}}$	90.99	87.28	82.66	11.67
		Mahalanobis	88.60	94.61	93.95	65.49
		MaxLogits	81.21	86.71	84.00	37.73
		MaxSoftmax	74.53	80.79	79.20	27.36

Table 3.5: Results for different OoD methods based on the high-resolution images. We used the ImageNet as known and the ImageNet-O and the Places365 as unknown data.

Out-dist	Model	Method	DTACC	AUC	AUPR	TNR at TPR 95%
		$\text{MDistance}_{\text{Cos}}$	63.03	62.27	61.10	1.89
		$\text{MDistance}_{\text{Euc}}$	50.00	40.57	43.50	2.38
		OSNN_{Cos}	79.44	84.99	83.33	22.59
		OSNN_{Euc}	82.72	88.93	87.80	38.33
	ResNet-101	LOF_{Cos}	86.40	93.80	93.86	63.76
		LOF_{Euc}	86.30	93.49	93.41	73.96
		$\text{LOF\_D}_{\text{Cos}}$	91.62	95.06	95.37	64.36
		$\text{LOF\_D}_{\text{Euc}}$	91.58	94.20	93.56	56.92
		Mahalanobis	82.55	90.06	89.36	45.08
		MaxLogits	60.34	62.99	60.71	9.88
		MaxSoftmax	70.64	75.35	73.06	14.84
		$\text{MDistance}_{\text{Cos}}$	64.22	66.36	64.25	4.32
		$\text{MDistance}_{\text{Euc}}$	50.78	42.01	45.06	1.56
		OSNN_{Cos}	80.52	86.61	85.45	28.84
		OSNN_{Euc}	80.68	87.13	86.28	35.32
	DenseNet-161	LOF_{Cos}	91.26	96.98	97.00	83.20
		LOF_{Euc}	90.61	96.52	96.46	85.40
		$\text{LOF\_D}_{\text{Cos}}$	94.81	93.92	91.57	26.66
		$\text{LOF\_D}_{\text{Euc}}$	96.97	94.21	90.31	0.65
		Mahalanobis	91.46	96.99	96.91	83.76
		MaxLogits	77.28	82.99	80.53	39.47
		MaxSoftmax	74.33	81.46	79.40	33.71
		$\text{MDistance}_{\text{Cos}}$	88.36	92.74	90.89	48.21
		$\text{MDistance}_{\text{Euc}}$	50.10	28.03	37.64	0.45
		OSNN_{Cos}	80.44	86.44	84.83	27.76
		OSNN_{Euc}	77.75	84.24	82.46	23.88
	WideResNet-101-2	LOF_{Cos}	86.67	92.70	92.14	49.70
		LOF_{Euc}	94.44	98.59	98.49	92.40
		$\text{LOF\_D}_{\text{Cos}}$	95.60	94.57	90.20	95.98
		$\text{LOF\_D}_{\text{Euc}}$	96.77	97.22	96.08	98.21
		Mahalanobis	92.97	97.37	97.07	85.15
		MaxLogits	77.93	83.56	81.74	38.48
		MaxSoftmax	77.26	84.58	82.84	43.35
		$\text{MDistance}_{\text{Cos}}$	91.81	94.88	93.34	66.63
		$\text{MDistance}_{\text{Euc}}$	71.57	76.20	73.88	11.92

Table 3.5: Results for different OoD methods based on the high-resolution images. We used the ImageNet as known and the ImageNet-O and the Places365 as unknown data.

Out-dist	Model	Method	DTACC	AUC	AUPR	TNR at TPR 95%
		$OSNN_{Cos}$	84.73	90.22	88.67	36.10
		$OSNN_{Euc}$	84.28	90.29	88.55	38.43
	EfficientNet-B3	LOF_{Cos}	68.35	73.57	74.90	20.31
		LOF_{Euc}	97.17	99.56	99.52	98.66
		LOF_D_{Cos}	98.06	96.67	95.00	99.60
		LOF_D_{Euc}	97.29	97.60	96.66	98.46
		Mahalanobis	95.18	97.87	97.60	89.47
		MaxLogits	75.20	78.21	75.34	41.71
		MaxSoftmax	77.83	84.54	82.31	45.33
		$MDistance_{Cos}$	95.75	96.37	94.36	81.68
		$MDistance_{Euc}$	62.59	64.04	60.90	8.54
		$OSNN_{Cos}$	85.22	91.40	89.87	40.64
		$OSNN_{Euc}$	89.13	94.54	93.18	66.73

Section Summary The OoD methods can work well on large-scale images. The methods based on Mahalanobis worked slightly worse than other methods working on a large scale. The methods based on logits worked poorly. It is probably connected with the number of classes being high - 1000 in our experiments. Our experiments have suggested using LOF-based methods. We verified the hypothesis that the better the CNN model, the better the OoD detection efficiency. However the results was inconclusive. The trend is visible for methods like Mahalanobis and OSNNs, but there is no direct dependency on other methods. Moreover, we tested how OoD approaches worked with the ImageNet-O dataset, which was designed to fool the networks – generally, the LOFs and Mahalanobis worked quite well on this dataset.

3.3 Analysis of Assumptions of Chosen Ood Methods

In section, we focused on analyzing assumptions of Mahalanobis and EVM methods. Both parametric approaches can achieve good results on the benchmarks dataset, so they are willingly used. We decided to verify the basic assumptions of each method: (1) how the Mahalanobis works in high-dimensional space, (2) why Simple Unified Framework works so well and if it is due to Mahalanobis, and (3) when the parametric model in EVM is not appropriate due to unsatisfied assumptions of the Extreme Value Theorem.

3.3.1 *Discussion on the Mahalanobis*

Section Objectives This section analyzed the instability of parametric estimates of Multivariate Normal Distribution(MVN) density by Mahalanobis distance in high-dimensional features generated from the CNNs. We discussed why the far OoD problems with large signal-to-noise ratios may still rely on Mahalanobis distance. Morover, we compared different modifications (see section 2.3.1.3) of this method. Part of this research is based on one of our papers[151].

First, we decided to test how the number of training samples n influences the distribution of the Mahalanobis distance. We used the simulation data for this problem.

We generated two (training and testing) in-distribution subsets with n samples with d dimensional feature space using the MultiVariate Normal distribution (MVN). The used MVN has the mean at $[0]_d$ and uncorrelated variables with variance 1. As out-of-distribution subset, we used a cluster with mean at $[\frac{r}{\sqrt{d}}]_d$, uncorrelated, with variance 1. We generated 3 OoD subsets (we called them scheme s): (1) we changed means along only one ax, (2) we changed means along $\frac{d}{2}$ axis, and (3) we changed means for all d axis. In next parts the three schemes can be denoted as $ood1(s = 1)$, $ood2(s = 2)$, and $ood3(s = 3)$. The example of data in 3-dimensional space is presented in figure 3.2.

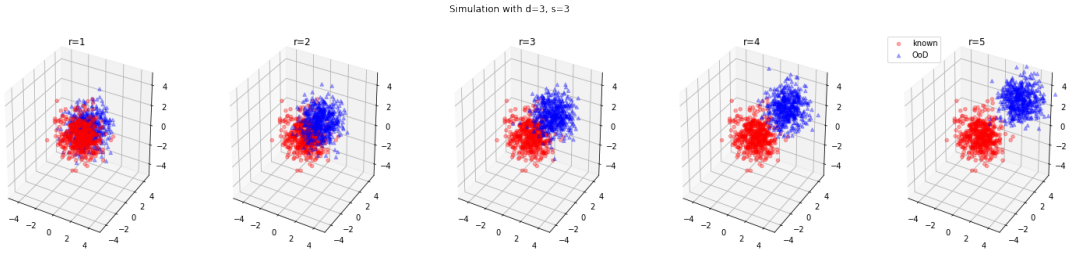


Figure 3.2: The example of the simulation data

To quantify the dissimilarity between groups od data, we use the measure $\Delta(group1, group2) = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{s_1^2 + s_2^2}}$, where $\bar{X}_i$ and s_i are the mean and its standard error in group i. Note that this measure is used as the test statistic in Welch's t-test.

See the figure 3.3 where we showed results of our experiments on simulation data with $d = 2000$, $r = 8$ and with $n = 5000$ and $n = 50000$. The values for d and n correspond to the real case based on the CIFAR-10 and the ResNet-101. Generally, we observed that the Mahalanobis distance could not distinguish in-distribution and OoD data. However, large training samples lead to more robust models of in-distribution data (difference between train and test data decreases) and better separability of in- and OoD data (difference between test and OoD increases). See dissimilarity between distances: $\Delta(test, train) = 708$ $(n = x)$, 204 $(n = x)$ denoted as red arrow in the figure and $\Delta(ood1, test) = 39$ $(n = x)$, 154 $(n = x)$ denotes as blue arrow. Hence, with increasing the number of samples, the model stabilizes and better distinguishes known and unknown data.

Next, we checked the influence of shifting r - see the figure 3.4. Enough far OoD data (e.g., for $r = 32$) seemed to be no influence for the inaccuracy in MVN model estimation: OoD examples were significantly more distant from the model than the test in-distribution data. We argued that this explains the success or failure of Mahalanobis distance-based OoD detection in CNN benchmarks.

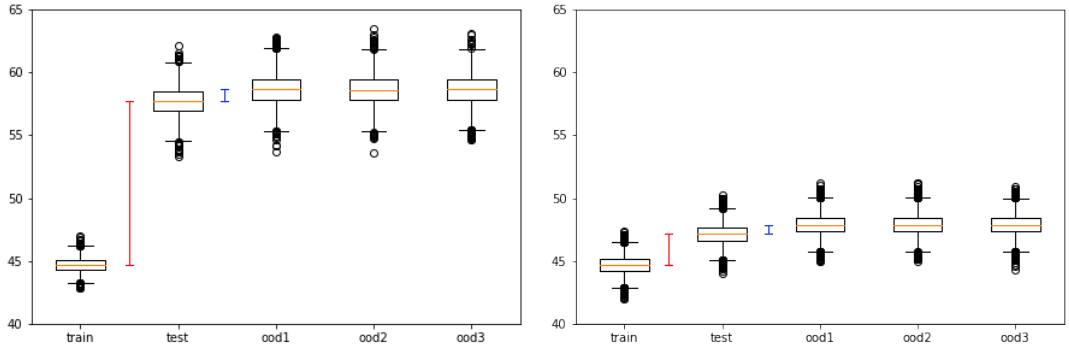


Figure 3.3: Distribution of Mahalanobis distance of train, test and OoD samples to the MVN model fitted to the train data, dimensionality $d = 2000$, OoD shift $r = 8$, sample size $n = 5000$ (left), $n = 50000$ (right).

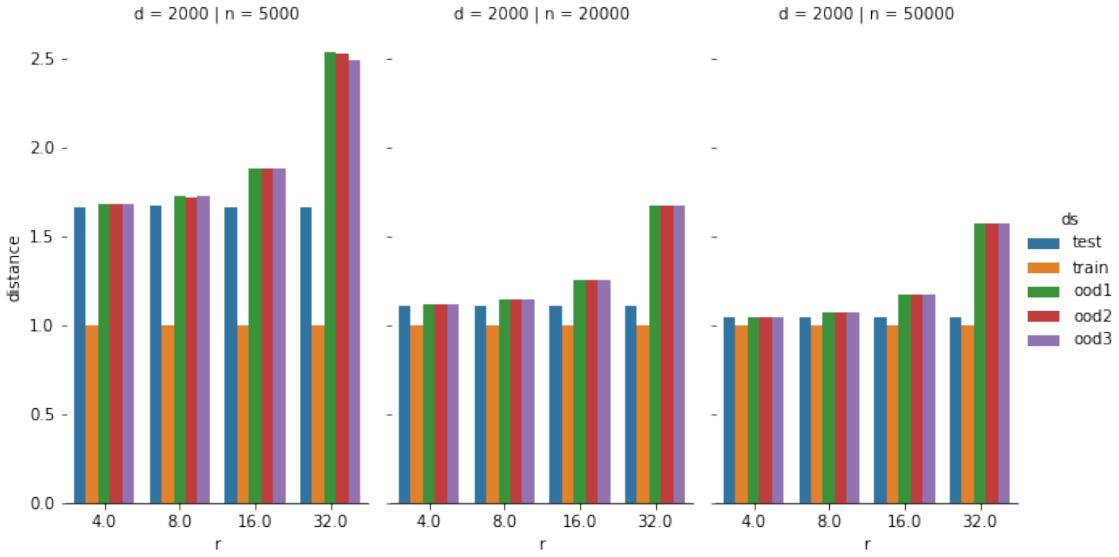


Figure 3.4: Comparing Mahalanobis distances of train, test and OoD samples, as a function of sample size n, OoD shift r, for dimensionality $d = 2000$. Note that the distance is the squared Mahalanobis distance divided by d.

Next, we tested the influence of the feature correlation. See the result in figure 3.5 with uncorrelated features (left) and 1000 features correlated with coefficient 0.5, both for dimensionality $d = 2000$, OoD shift $r = 8$, sample size $n = 20000$. See dissimilarity between distances: $\Delta(test, train) = 330$ (for both) denoted as red arrow in the figure, and $\Delta(ood1, test) = 95$ (uncorrelated), 1278 (correlated) denoted as blue arrow. Note that with correlated data, distance of different outlier groups $ood1, 2, 3$ to the model is significantly different, e.g., $\Delta(ood1, ood2) = 122$ (correlated) denoted as greenarrow, whereas for uncorrelated differences were not as much significant. We observed that when the correlation of the distribution data and the OoD data vary, the separability of the OoD samples improves.

We tested the Mahalanobis-based OoD detection methods in real-data benchmarks datasets. We used the ResNet-101 (fitted on the CIFAR-10) and the DenseNet-169 (fitted on the CIFAR-100). We used near and far OoD examples: the Noise, the SVHN, and the CIFAR-10/CIFAR-100. We tested the four variants of the Mahalanobis. The LOF_{Euc} is shown as a reference non-parametric method.

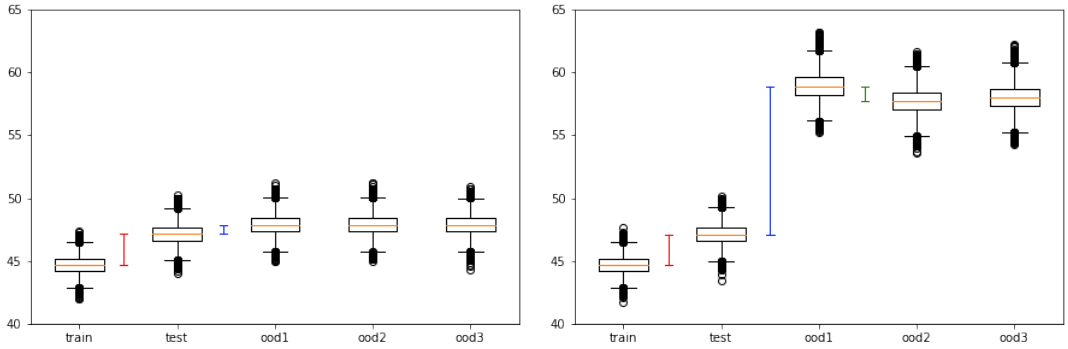


Figure 3.5: Effect of correlated features: distribution of Mahalanobis distance of train, test and OoD samples to the MVN model, for dimensionality $d = 2000$, OoD shift $r = 8$, sample size $n = 20000$, uncorrelated features (left); 1000 features correlated with coefficient 0.5. (right).

The comparison, presented in the table 3.6, did not give straightforward conclusions. There was no best or worst method. For example, (1) *Euclidean* failed for the DenseNet-169 and the SVHN and outperformed others for the CIFAR-10 and the same model, (2) *SEuclidean* failed for the ResNet-101 and the Noise and outperformed for the SVHN and the DenseNet-169, (3) the *MahalanobisUF* failed in the CIFAR-10 and the DenseNet-169, and (4) *Mahalanobis* failed in the SVHN and the DenseNet-169.

Section Summary This section focused on the in-depth analysis of the Mahalanobis approaches. We showed, by using the simulation data, how the instability of MVN density estimates leads to the intrinsic limitation of this method: near OoD samples cannot be distinguished from known data. For fixed dimensionality of features and the training data size, we can estimate the minimum distance from known data beyond which outliers are detectable. We showed that this distance decreases with the growing number of training samples.

We tested different variants of Mahalanobis distance in benchmark datasets. We showed that none of these methods could be treated as the universal approach. The performance depended on the data and CNNs models.

Table 3.6: The comparison of analysed OoD methods for the CIFAR-10 and the CIFAR-100 .

In-dist (Model)	Out-dist	Method	DTACC	AUC	AUPR	TNR at TPR 95%
CIFAR-10 (ResNet-101)	Noise	Euc	96.21	98.81	98.56	97.12
		SEuc	94.71	94.57	91.24	45.92
		MahUF	100.00	100.00	100.00	100.00
		Mah	100.00	100.00	100.00	100.00
		LOF_{Euc}	99.30	99.90	99.89	100.00
	SVHN	Euc	86.75	93.00	92.52	55.03
		SEuc	82.57	86.59	83.85	13.55
		MahUF	83.26	91.13	90.96	53.84
		Mah	82.25	89.33	88.77	41.34
		LOF_{Euc}	86.43	93.00	92.73	57.80
	CIFAR-100	Euc	80.69	87.38	86.49	41.64
		SEuc	80.19	86.30	84.84	35.75
		MahUF	71.38	78.18	77.40	24.07
		Mah	78.55	85.91	85.16	37.59
		LOF_{Euc}	79.49	87.22	86.74	47.84
CIFAR-100 (DenseNet-169)	Noise	Euc	98.02	98.99	98.32	99.98
		SEuc	100.00	100.00	100.00	100.00
		MahUF	100.00	100.00	100.00	100.00
		Ma	100.00	100.00	100.00	100.00
		LOF_{Euc}	95.81	95.91	92.99	81.05
	SVHN	Euc	70.65	75.21	73.95	12.24
		SEuc	78.41	85.85	85.07	37.30
		MahUF	74.30	81.82	80.93	29.11
		Mah	75.73	81.31	79.74	19.37
		LOF_{Euc}	76.86	83.46	81.96	24.48
	CIFAR-10	Euc	68.39	73.29	71.19	15.48
		SEuc	66.40	69.77	68.07	9.30
		MahUF	62.22	65.65	63.97	8.05
		Mah	65.98	69.42	68.33	10.08
		LOF_{Euc}	68.28	73.36	71.71	13.51

3.3.2 *Discussion on A Simple Unified Framework for OoD*

Section Objectives A Simple Unified Framework for Detecting Out-of-Distribution Sam-

ples[132] is one of the most popular methods for OoD detection. In this section, we used "UF" as the name for this method. In practice, the UF's authors used a few concepts: features ensemble, input preprocessing, and the Mahalanobis distance (see details in section 2.3.1.10). The authors focused mainly on the Mahalanobis distance, while we argue that this method's efficiency leads in the first two concepts. We also tested the approach by replacing the Mahalanobis with the LOF-based methods. Part of this research is based on one of our pre-print papers[241].

Our goal was to see what is essential in achieving that good results by the UF. Most papers that cited this work suggest that is Mahalanobis distance. We showed in the previous section 3.3.1 that there is no one variant of the Mahalanobis that can be treated as a better approach compared to others. We also tested the variant proposed in the UF with a common covariance matrix(called MahalanobisUF). Moreover, we showed that this group of methods could be unstable in multidimensional space. The authors of UF noticed this too but left without further comment (see a screenshot of their table presented in figure 3.6). In that table, we can see that the UF without concepts like features enabled or input preprocessing did not work significantly well. Therefore, we suggest that replacing the Mahalanobis with another method should work similarly, and thereby the strength of the UF is not the using the Mahalanobis distance itself.

Method	Feature ensemble	Input pre-processing	TNR at TPR 95%	AUROC	Detection accuracy	AUPR in	AUPR out
Baseline [13]	-	-	32.47	89.88	85.06	85.40	93.96
ODIN [21]	-	-	86.55	96.65	91.08	92.54	98.52
Mahalanobis (ours)	-	-	54.51	93.92	89.13	91.56	95.95
	-	✓	92.26	98.30	93.72	96.01	99.28
	✓	-	91.45	98.37	93.55	96.43	99.35
	✓	✓	**96.42**	**99.14**	**95.75**	**98.26**	**99.60**

Table 1: Contribution of each proposed method on distinguishing in- and out-of-distribution test set data. We measure the detection performance using ResNet trained on CIFAR-10, when SVHN dataset is used as OOD. All values are percentages and the best results are indicated in bold.

Figure 3.6: Table from [132]. It shows the influence of used concepts in UF.

We choose the LOF-based methods as a replacement for the Mahalanobis. We chose them due to results performed in experiments based on the simulation data (see section 3.3.1). We summarized the performance of Mahalanobis distance-based and LOF-based OoD detectors as a function of dimensionality d. We observed that for $d = 1000$ with 1000 training points per class, the Mahalanobis procedure no longer detected outliers (AUCROC $\approx 50\%$), while LOF was more reliable (AUCROC $> 80\%$). See the figure 3.7.

We tested numerous pairs (in-distribution and out-of-distribution) using the UF to see the influence of the input preprocessing. This concept is computationally complex and requires parameter-tuning (the magnitude of perturbation). Our experiments suggested that the ODIN-like input perturbations have negligible (or even negative) impact on separating the in- and out-of-distribution samples for LOF-based methods. For instance, for experiments with CIFAR-10, perturbation equal to zero was always selected as the best on validation data.

To verify the sentence that "the strength of the UF is not the using the Mahalanobis distance itself", we tested the UF with MahalanobisUF (original concept), LOF_{Euc}, and $\text{LOF\_D}_{\text{Cos}}$. We did not use the input pre-processing method in our LOF-based methods.

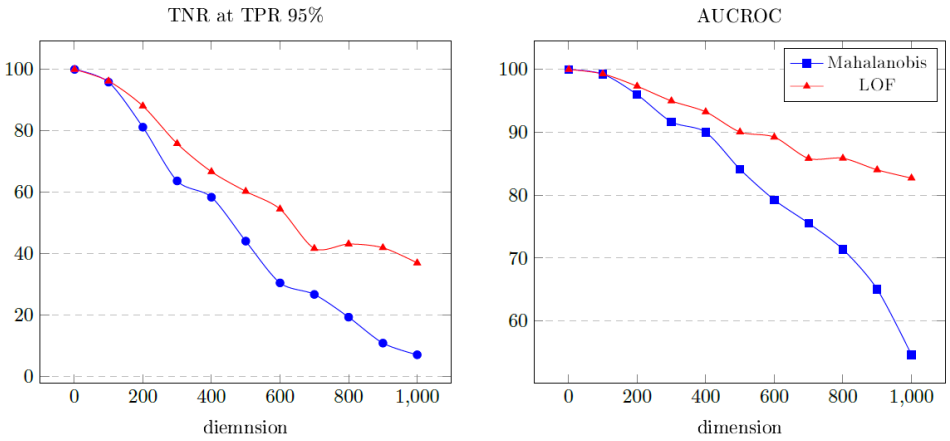


Figure 3.7: Comparison of the TNR at TPR 95 % and the AUC for Mahalonobis and LOF on simulation data. By increasing the complexity of the problem by expanding the input data dimension, the LOF method is much more stable and achieves better results. The dimension above 1,000 is common in the lasts layers of CNNs.

The results are presented in table 3.7. We observed that on difficult OoD problems (i.e., the CIFAR-10 as in-distribution vs. the CIFAR-100 as OoD; the ImageNet as in-distribution vs. the ImageNet-2010 as OoD – difficult problems for the ResNet-101), our LoF-based procedure outperforms the Mahalanobis distance-based method. We noticed that for the OoD benchmarks which seemed to be easier, i.e., the AUROC obtained by the methods 98% - 99%, (these benchmarks include: CIFAR-10 vs. SVHN, ImageNet vs. ImageNet-O or Places365, ImageNet vs. ImageNet-2010 (for EfficientNet-B3)), our modification achieved similar results. Notice that LOF-based methods had no input perturbation hyperparameter, fine-tuned for a specific study in the Mahalanobis method.

Finally, we wanted to discuss the features ensemble approach. The idea seems reasonable to us. The features obtained from the earlier layers of the network can better separate the unknown examples. However, we see some limitations here.

Firstly, to fit the weights, there is a need to see some out-of-distribution examples. The authors of UF propose to use the OoD from the same distribution. For instance, for the CIFAR-10 vs. the SVHN, they fit weights on the subset of SVHN, and the CIFAR-10 vs. the CIFAR-100 they fit them on the subset of the CIFAR-100. In real problems, the OoD data is entirely unknown. Of course, there are methods like Outlier Exposure[91], but it is hard to compare them with others that do not see any unknown data. In our opinion, this is the main reason why the UF method outperforms others.

Secondly, the UF method assumed that the features obtained from the earlier layers can be the assignment to a given class. However, the first layers of the network are not "designed" for classify the data, so using Mahalanobis boundaries based on the final classes can be a false idea.

Section Summary In this section, we focused on a deep analysis of the UF method. This approach is one of the state-of-the-art method. However, in our opinion, the method's strength lies in the "Outlier Exposure" procedure during the weights fitting. The Mahalanobis itself is a solid approach, but the LOF-based are equally good or even better. The concept of input pre-processing was not valuable in many cases.

Table 3.7: The comparison of [132] with replacing the Mahalanobis with LOF-based approaches. For the two most challenging problems, the CIFAR-10 vs. the CIFAR-100 and the ImageNet vs. the ImageNet-2010 for the ResNet-101, our approach outperforms [132].

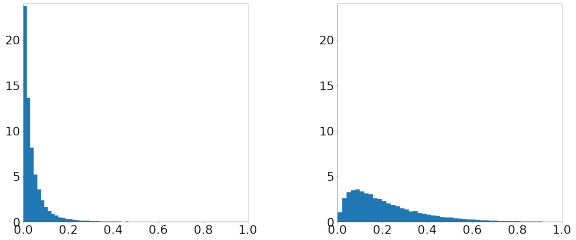
In-dist (Model)	Out-dist	Method	DTACC	AUC	AUPR	TNR at TPR 95%
CIFAR-10 (ResNet-101)	CIFAR-100	UF	83.18	90.47	90.12	53.77
		UF+LOF_{Euc}	85.24	92.41	92.31	63.91
		UF+LOF_D_{Cos}	85.89	93.16	93.11	65.92
	SVHN	UF	97.14	99.46	99.40	99.02
		UF+LOF_{Euc}	99.25	99.95	99.94	99.97
		UF+LOF_D_{Cos}	98.77	99.9	99.9	99.74
ImageNet (ResNet-101)	ImageNet-2010	UF	87.75	93.70	93.24	71.60
		UF+LOF_{Euc}	87.28	94.23	94.16	74.50
		UF+LOF_D_{Cos}	84.45	92.31	92.23	66.15
	ImageNet-O	UF	98.73	99.88	99.85	99.93
		UF+LOF_{Euc}	98.77	99.91	99.88	99.67
		UF+LOF_D_{Cos}	98.40	99.77	99.74	99.33
	Places365	UF	97.75	99.64	99.62	99.40
		UF+LOF_{Euc}	97.32	99.69	99.68	98.65
		UF+LOF_D_{Cos}	97.45	99.66	99.65	98.90
ImageNet (EfficientNet-B3)	ImageNet-2010	UF	93.65	98.11	97.98	91.75
		UF+LOF_{Euc}	93.35	98.08	98.03	91.05
		UF+LOF_D_{Cos}	88.75	95.44	95.48	77.75
	ImageNet-O	UF	99.27	99.97	99.93	99.93
		UF+LOF_{Euc}	99.63	99.98	99.95	99.94
		UF+LOF_D_{Cos}	98.62	99.87	99.85	99.25
	Places365	UF	99.48	99.97	99.95	100.0
		UF+LOF_{Euc}	99.55	99.99	99.96	99.95
		UF+LOF_D_{Cos}	99.03	99.89	99.87	99.85

3.3.3 *Discussion on Extreme Value Machine(EVM)*

Section Objectives This subsection focused on analyzing the Extreme Value Machine for OoD detection. Although the method was not used in the other parts of this work, it was noticed by the scientific community. As the authors claim, this approach was justified by theoretical research. We argued with the assumption in the context of OoD usage in deep learning and computer vision. In this section, we empirically verified it. This research is based on one of our papers[258].

The Extreme Value Machine(EVM)(see section 2.3.1.7) is method which used Compact Abating Probability (CAP). To construct the CAP there is needed to create a radial inclusion function for each point x. Distances from each point x and closest τ points (with a different class than x) are calculated. Then the parametric model of the margin distances are estimated by fitting the Weibull distribution. This step is justified by the authors by the Extreme Value Theory and is later analyzed in terms of the validity of the underlying assumptions. We empirically verified this by using the Kolmogorov-Smirnov goodness of fit test, with the null hypothesis that the margin distances have the Weibull distribution estimated by the EVM implementation (see details in section 2.3.1.7).

For all experiments, we used the ResNet-101, which was trained from scratch for the classification task. It achieved 95.15% final accuracy. The CIFAR-10 was used as known data.



(a) the CIFAR-10 without normalization

(b) the CIFAR-10 with normalization

Figure 3.8: Histograms of Weibull goodness-of fit test p-values for raw and standardized CIFAR-10 features - form our [258].

For mean p-values of Kolmogorov-Smirnov tests for all training examples in each data set, we achieved 0.043452 for the CIFAR-10 and 0.215405 for the CIFAR-10 with standardized values (by using the popular $z-$score normalization with the mean and standard deviation for each variable estimated on the train data set). The detailed analysis of p-values for the CIFAR-10 data set is shown on histograms in figure 3.8. We could notice that normalization of CIFAR-10 data changes the distribution of margin distances: for a majority of training examples in the raw dataset, the Weibull model was not fit the data (most of p-values $< 5\%$), whereas for standardized data the Weibull model was appropriate (most of p-values $> 5\%$). It led to improved performance of the EVM in OoD - e.g., with the CIFAR-100 as unknown data (AUC increased from 79.64% to 87.95%).

This (and other presented in [258]) analysis proved that margin distances do not follow the Weibull distribution in many real datasets, contrary to the theoretical justification given in [195] (Theorem 2). The justification given in [195] is grounded on the Fisher-Tippett-

Gnedenko (or Extreme Value) Theorem. Hence the underlying assumption needed for the margin distances to follow the Weibull distribution is that they can be treated as the maximum from a series of random variables, which was not shown, but only postulated in [195].

Table 3.8: AUC for the original EVM and its modifications based on other parametric models for the the CIFAR-10 with using GAP as the feature extraction

Out-dist	Stand.	EVM	Weib min	Normal	Gamma	ECDF	LOF
MNIST	False	89.78	**95.73**	92.67	87.29	85.97	**98.46**
	True	97.24	**97.89**	97.87	97.24	97.15	**98.26**
CIFAR-100	False	79.64	**88.79**	82.86	77.25	75.17	**88.87**
	True	**87.96**	83.25	87.84	87.04	81.71	**89.34**

We could conclude that the Weibull distribution is not the key to the EVM performance. To confirm this, we substituted the Weibull distribution by some other distributions and repeated the previous OoD detection experiments using this modified EVM. More specifically, we followed the EVM algorithm as described in the original paper [195], but fitted the parametric model directly to the margin distances, and not to the transformed. We tried four alternatives: the Weibull Minimum Extreme Value (Weib min), Normal, Gamma, and empirical CDF (ECDF). The achieved AUC values were compared with the original EVM - see table 3.8. We also added the LOF_{Euc} method due to comparison, which achieved much better results in all tested tasks. These results suggested that the parametric distribution type, as well as transformation applied by the libMR (see details in 2.3.1.7), have a minor influence on the EVM performance. None of the analyzed distributions outperformed other types (see th full comparison in [258]). The assumption of the Weibull distribution was not essential for the performance of the EVM.

Section Summary We showed that the EVM's theoretical assumptions were not fulfilled in the context of CNNs and some popular datasets. Margin distances often did not follow the Weibull distribution. Moreover, the type of parametric distribution had an overall minor influence on the EVM. In comparison with the LOF method, the EVM achieved much worse results.

3.4 The Influence of Features on OoD Detection

The CNN classification models are fitted on the training data, which allows learning patterns of what each type of category looks like. The images are transformed from pixel space into feature space. Step by step, the convolutional blocks make a set of pixels transformed into features that are more and more separable by class type. So finally, the close-set classifier can distinguish them and make the classification.

All popular OoD algorithms build and estimate decision boundaries based on feature space distribution. The research described in this section focuses on the features and what gives them their final form. First, we concentrated on feature extraction methods from the CNNs - different from classic Global Average Pooling. Various approaches that can focus on different components (e.g., on edges, patterns, or whole objects), so for different pairs of ID and OoD, different features can be useful to separate data. Next, we focused on feature

normalization and reduction. Both ideas are based on the solutions for Image Retrieval problems, but there are novel for OoD ones. Further, we check the influence of images augmentations. Different augmentation strategies allow the network to learn different patterns, which can help distinguish the out-of-disturbed examples. Next, we checked how the OoD algorithms are sensitive to changing their weights during the training process. Finally, we used the above conclusions to generate the hard dataset for nearly all OoD detection methods, and we tested OoD approaches on the problem of adversarial attacks detection.

3.4.1 *Effect of the Feature Extraction Method on OoD Detection*

Section Objectives Usually, the features from deep networks are extracted using Global Average Pooling (GAP) on the last convolutional layer. Next, these are passed to the classifier to propose an output class. In most techniques for OoD detection, these features are the basis for distinguishing between in-distribution and out-of-distribution. In this subsection, we focused on checking the influence of different approaches for feature extraction. The idea is taken from the image retrieval problem, where many papers have focused on finding the proper strategy for obtaining global and local descriptors from the images. Our research hypothesis is that the OoD detection algorithms could build better decision boundaries by changing the features extractor approach. It can increase OoD efficiency due to various approaches that can focus on different components (e.g., on edges, patterns, or whole objects), so for different pairs of ID and OoD, different features can be useful to separate data.

The Global Average Pooling (GAP) is the most popular feature extraction. Using it in classic CNNs reduces the number of parameters in the classification part and limits the risk of overfitting. These features are often called global ones due to good generalization ability. However, these are not always well suitable for all ML problems - for example, in the image retrieval tasks, the well-chosen strategy can be key for finding similar images, so other methods have been developed. We decided to check the influence of the features obtaining strategy in OoD detection problems as well. The hypothesis is that by changing the features could allow building better decision boundaries to separate the known and unknown examples better. These features could be from a different distribution than GAP (used for the classifier), so new features are less likely to be overfitted.

We compared using different feature extraction in OoD detection based on the ResNet-101 fitted on the CIFAR-10, and the ImageNet, respectively. We focused on various methods: focusing on local (object details) and/or global (whole object) descriptions and low-level (e.g., shapes, textures) and/or high-level (whole image meaning) features. We used the following feature extraction methods: CroW, GAP, GAP_All, GMP, and SCDA (see section 2.2.8.1 for more details). The GAP is the classic approach. The GAP_All used concatenated GAP from all main layers from the ResNet. This method's motivation is to also focus on the low-level features, which focus more on shapes, edges, object parts rather than whole objects. Some OoD examples can be distinguished from the in-distribution in low- or middle-level. The GMP (Global Maximum Pooling) or Maximum activations of convolutions (MAC) is an alternative for the GAP. However, it is focused on the most significant parts of the last convolutional layer. The CroW is a non-parametric weighting and aggregation scheme. Furthermore, the last method that we used is the SCDA. This approach focuses more on the local features. There are many other feature extraction

methods. However, we would like to compare only a few in this section – these were quick to adapt and did not require to retrain the model.

The tables 3.9, 3.10 and 3.11 presented the results for the CIFAR-10 as known data and the Noise, the SVHN, and the CIFAR-100 as OoD - respectively. The MaxLogits and MaxSoftmax were used for comparison purposes – these methods do not use features but logits. We could retrain the classifier on new features, but that is out of the scope of this section. Generally, changing the feature extraction strategy helped improve OoD detection for all sets compared to the GAP. However, there was no one best method. It depended on near- or far- OoD examples. For the Noise and the CroW and SCDA with Mahalanobis achieved the best results. For the SVHN, all methods worked similarly – however, the GAP_All and GMP were the best. And for the CIFAR-100 GMP was the best, however, the GAP worked quite well too. The GMP seems to work the most stable among other methods. LOF_D_{EUC} seemed to not work for the CroW and GAP_All for some reason. The most stable OoD method over the different features seemed to be Mahalanobis.

We decided to perform similar experiments but on high-resolution images based on the ResNet-101 model trained for the ImageNet. In tables 3.12 and 3.13 we presented results for the ImageNet-O and the Places365 respectively. Again, well-chosen feature extraction allowed a significant boost in the final results. For example, for ImageNet-O, when we used classic GAP, the best method - the Mahalanobis - achieved 97.42% of AUC and 86.10% of TNR at TPR 95% while changing the way of obtaining features to GMP or SCDA, the results increased over 99% for both results for LOF-based approach. A similar effect for the Places365. The best method for the GAP - the LOF_D_{Cos} achieved 95.06% of AUC and 64.36% of TNR at TPR 95%. When the features were changed to CroW, GMP, or SCDA, the results increased for both metrics to get closer to 100%. There are some unexpected results here. The LOF_D_{Cos} did not work well with the SCDA for the ImageNet-O and perfectly well for the Places365. Similar, the LOF_D_{Euc} for GMP. The Mahalanobis did not usually work well on high-dimensional images for different feature extraction methods. The GAP_All made LOF_Ds failed.

Section Summary Our experiments showed that changing the feature extraction methodology can significantly improve OoD detection efficiency. To the best of our knowledge, there is a lack of similar research in the literature. Although [132] used the "feature ensemble" method (similar to our GAP_All, but calculated independently - not concatenated as our) and the authors claimed that it allowed an increase in the efficiency, but it was still based on the GAP.

Changing feature obtaining strategy into GMP, GAP_All, CroW, or SCADA, there is possible to boost the OoD method. There are no clear guidelines on which method should be used, but our experiments showed that the results were better or similar when we changed the extractor, so we recommend doing it. We proposed to use a new hyperparameter (feature extraction strategy), which, when properly chosen, allowed boosting final results by a few percentage points. We proposed to consider various methods: focusing on local (object details) and/or global (whole object) descriptions and low-level (e.g., shapes, textures) and/or high-level (whole image meaning) features – as we have shown above.

Table 3.9: Comparision of the OoD and feature extraction methods based on the the ResNet-101 model and the **Noise** as OoD data.

Feature extraction	Method	DTACC	AUC	AUPR	TNR at TPR 95%
GAP	LOF_{Cos}	87.64	93.39	93.00	52.61
	LOF_{Euc}	87.77	92.85	91.32	47.71
	LOF_D_{Cos}	87.54	93.02	89.13	36.27
	LOF_D_{Euc}	87.77	92.70	88.24	37.73
	Mahalanobis	89.98	94.99	94.47	63.06
	MaxLogits	88.72	93.50	91.78	53.26
	MaxSoftmax	86.63	91.41	90.08	37.94
	$OSNN_{Cos}$	86.00	91.17	90.21	35.89
	$OSNN_{Euc}$	86.05	91.03	89.87	35.33
CroW	LOF_{Cos}	97.04	97.85	96.65	99.09
	LOF_{Euc}	98.70	99.57	99.44	100.00
	LOF_D_{Cos}	96.34	97.24	91.09	91.08
	LOF_D_{Euc}	97.15	97.14	90.22	0.00
	Mahalanobis	99.83	100.00	99.99	100.00
	$OSNN_{Cos}$	97.38	98.11	97.47	99.72
	$OSNN_{Euc}$	97.75	98.35	97.82	99.98
GAP_All	LOF_{Cos}	86.00	91.94	90.84	47.67
	LOF_{Euc}	86.23	91.77	90.29	45.76
	LOF_D_{Cos}	85.93	91.51	86.05	0.00
	LOF_D_{Euc}	86.20	91.51	85.94	0.00
	Mahalanobis	91.44	94.81	93.25	59.37
	$OSNN_{Cos}$	85.61	90.77	89.62	37.00
	$OSNN_{Euc}$	85.89	90.97	89.75	37.00
GMP	LOF_{Cos}	90.25	95.23	95.02	64.64
	LOF_{Euc}	90.61	94.37	92.28	58.37
	LOF_D_{Cos}	90.09	94.66	90.59	46.29
	LOF_D_{Euc}	90.42	94.20	89.40	46.68
	Mahalanobis	94.86	98.85	98.85	94.04
	$OSNN_{Cos}$	87.72	92.29	91.29	41.61
	$OSNN_{Euc}$	87.72	92.11	91.06	40.42
SCDA	LOF_{Cos}	97.72	98.90	98.44	99.86
	LOF_{Euc}	97.70	98.92	98.54	99.90
	LOF_D_{Cos}	97.25	97.31	92.43	68.18
	LOF_D_{Euc}	97.10	97.27	92.78	98.73
	Mahalanobis	99.05	99.73	99.71	100.00
	$OSNN_{Cos}$	97.44	98.36	97.78	99.69
	$OSNN_{Euc}$	97.47	98.58	98.22	99.78

Table 3.10: Comparision of the OoD and feature extraction methods based on the the ResNet-101 model and the **SVHN** as OoD data.

Feature extraction	Method	DTACC	AUC	AUPR	TNR at TPR 95%
GAP	LOF_{Cos}	86.22	92.39	92.03	48.96
	LOF_{Euc}	86.08	91.81	90.71	45.64
	LOF_D_{Cos}	86.24	92.05	88.28	34.79
	LOF_D_{Euc}	86.10	91.60	87.28	36.98
	Mahalanobis	87.66	93.91	93.73	57.87
	MaxLogits	84.81	90.11	88.18	39.58
	MaxSoftmax	83.77	88.93	87.40	32.11
	$OSNN_{Cos}$	84.06	89.77	88.78	32.37
	$OSNN_{Euc}$	84.16	89.69	88.56	31.59
CroW	LOF_{Cos}	84.58	91.05	90.37	46.66
	LOF_{Euc}	83.55	89.76	88.81	42.28
	LOF_D_{Cos}	84.62	91.01	86.25	46.41
	LOF_D_{Euc}	83.75	89.78	84.51	0.00
	Mahalanobis	84.16	90.87	90.20	48.66
	$OSNN_{Cos}$	83.38	89.89	88.89	39.61
	$OSNN_{Euc}$	82.59	89.18	88.15	38.73
GAP_All	LOF_{Cos}	86.67	92.55	91.76	50.05
	LOF_{Euc}	86.44	92.17	91.19	48.12
	LOF_D_{Cos}	86.53	91.97	86.47	0.00
	LOF_D_{Euc}	86.42	91.76	86.20	0.00
	Mahalanobis	90.00	95.10	94.17	65.45
	$OSNN_{Cos}$	84.81	90.08	88.88	34.49
	$OSNN_{Euc}$	84.72	90.04	88.77	33.58
GMP	LOF_{Cos}	87.17	93.31	93.09	55.57
	LOF_{Euc}	87.56	92.75	91.06	50.58
	LOF_D_{Cos}	87.25	92.85	88.99	39.51
	LOF_D_{Euc}	87.46	92.56	87.99	41.49
	Mahalanobis	87.03	94.69	94.82	70.91
	$OSNN_{Cos}$	84.56	89.83	88.67	34.45
	$OSNN_{Euc}$	84.67	89.76	88.60	33.88
SCDA	LOF_{Cos}	80.83	87.76	86.95	45.65
	LOF_{Euc}	79.61	86.76	85.88	42.43
	LOF_D_{Cos}	80.67	87.39	83.50	30.35
	LOF_D_{Euc}	79.47	86.59	83.02	37.60
	Mahalanobis	82.14	88.94	87.82	45.14
	$OSNN_{Cos}$	79.64	87.22	86.42	36.55
	$OSNN_{Euc}$	79.61	87.17	86.44	36.25

Table 3.11: Comparision of the OoD and feature extraction methods based on the the ResNet-101 model and the **CIFAR-100** as OoD data.

Feature extraction	Method	DTACC	AUC	AUPR	TNR at TPR 95%
GAP	LOF_{Cos}	82.31	88.96	88.45	37.07
	LOF_{Euc}	81.69	88.07	86.81	34.31
	LOF_D_{Cos}	82.23	88.77	85.28	26.35
	LOF_D_{Euc}	81.63	87.94	83.99	27.75
	Mahalanobis	84.47	91.21	90.97	46.03
	MaxLogits	79.63	84.99	82.44	27.19
	MaxSoftmax	79.40	84.73	82.58	25.46
	$OSNN_{Cos}$	79.81	86.02	84.92	26.17
	$OSNN_{Euc}$	79.56	85.72	84.46	25.73
CroW	LOF_{Cos}	81.87	88.51	87.74	44.88
	LOF_{Euc}	81.51	88.04	87.12	44.45
	LOF_D_{Cos}	81.89	88.31	83.46	42.81
	LOF_D_{Euc}	81.66	87.79	82.21	0.00
	Mahalanobis	82.34	89.30	88.65	46.37
	$OSNN_{Cos}$	82.06	88.75	87.67	40.64
	$OSNN_{Euc}$	82.34	88.61	87.41	41.08
GAP_All	LOF_{Cos}	82.42	88.78	87.90	36.32
	LOF_{Euc}	82.02	88.20	87.06	34.45
	LOF_D_{Cos}	82.35	88.41	83.53	0.00
	LOF_D_{Euc}	81.89	87.93	82.98	0.00
	Mahalanobis	83.58	90.10	89.03	40.77
	$OSNN_{Cos}$	80.74	86.48	85.11	26.90
	$OSNN_{Euc}$	80.58	86.34	84.91	26.34
GMP	LOF_{Cos}	83.89	90.33	90.04	43.83
	LOF_{Euc}	83.80	89.68	88.06	38.65
	LOF_D_{Cos}	83.88	90.01	86.38	30.36
	LOF_D_{Euc}	83.87	89.53	85.33	30.46
	Mahalanobis	84.31	92.26	92.27	60.83
	$OSNN_{Cos}$	80.96	86.73	85.34	27.61
	$OSNN_{Euc}$	80.84	86.57	85.18	27.35
SCDA	LOF_{Cos}	79.87	86.82	86.15	44.79
	LOF_{Euc}	79.45	86.61	85.94	44.05
	LOF_D_{Cos}	79.94	86.68	82.70	35.26
	LOF_D_{Euc}	79.50	86.50	82.73	42.58
	Mahalanobis	81.77	88.74	87.92	46.14
	$OSNN_{Cos}$	80.92	87.84	86.80	39.00
	$OSNN_{Euc}$	80.95	87.72	86.70	39.41

Table 3.12: Comparision of the OoD and feature extraction methods based on the the ResNet-101 model and the **ImageNet-O** as OoD data.

Feature extraction	Method	DTACC	AUC	AUPR	TNR at TPR 95%
GAP	LOF_{Cos}	88.98	95.75	95.75	75.12
	LOF_{Euc}	90.99	97.18	97.18	83.76
	LOF_D_{Cos}	89.60	95.57	95.79	71.15
	LOF_D_{Euc}	91.33	91.50	89.77	28.55
	Mahalanobis	92.48	97.42	97.31	86.10
	MaxLogits	50.00	15.12	33.06	0.20
	MaxSoftmax	50.00	40.27	43.25	0.50
	$OSNN_{Cos}$	76.89	82.82	81.23	21.65
	$OSNN_{Euc}$	77.75	84.08	82.75	27.51
CroW	LOF_{Cos}	92.65	97.91	97.94	88.18
	LOF_{Euc}	93.92	98.46	98.46	92.40
	LOF_D_{Cos}	96.87	96.69	95.16	90.32
	LOF_D_{Euc}	97.44	96.57	94.44	99.26
	Mahalanobis	94.27	98.03	97.99	92.40
	$OSNN_{Cos}$	79.82	85.83	83.47	27.36
	$OSNN_{Euc}$	82.89	88.60	86.81	33.86
GAP_All	LOF_{Cos}	87.69	94.83	94.86	71.15
	LOF_{Euc}	89.82	96.51	96.51	81.73
	LOF_D_{Cos}	89.30	85.41	83.08	12.02
	LOF_D_{Euc}	91.39	92.17	92.01	42.95
	Mahalanobis	93.47	97.91	97.78	90.02
	$OSNN_{Cos}$	76.59	82.84	81.65	22.54
	$OSNN_{Euc}$	77.21	83.60	81.96	24.43
GMP	LOF_{Cos}	92.13	97.51	97.50	85.10
	LOF_{Euc}	92.43	97.63	97.59	88.28
	LOF_D_{Cos}	96.87	99.75	99.58	99.20
	LOF_D_{Euc}	96.87	94.48	91.77	4.82
	Mahalanobis	75.87	82.38	81.24	30.44
	$OSNN_{Cos}$	81.08	86.84	85.03	22.59
	$OSNN_{Euc}$	82.03	87.86	86.08	30.88
SCDA	LOF_{Cos}	90.64	96.79	96.83	80.88
	LOF_{Euc}	91.86	97.41	97.41	87.59
	LOF_D_{Cos}	49.98	0.71	30.78	0.20
	LOF_D_{Euc}	96.85	99.65	99.44	99.20
	Mahalanobis	78.60	85.57	84.66	35.45
	$OSNN_{Cos}$	77.93	83.43	81.33	18.77
	$OSNN_{Euc}$	82.47	87.84	86.07	29.00

Table 3.13: Comparision of the OoD and feature extraction methods based on the the ResNet-101 model and the **Places365** as OoD data.

Feature extraction	Method	DTACC	AUC	AUPR	TNR at TPR 95%
GAP	LOF_{Cos}	86.40	93.80	93.86	63.76
	LOF_{Euc}	86.30	93.49	93.41	73.96
	LOF_D_{Cos}	91.62	95.06	95.37	64.36
	LOF_D_{Euc}	91.58	94.20	93.56	56.92
	Mahalanobis	82.55	90.06	89.36	45.08
	MaxLogits	60.34	62.99	60.71	9.88
	MaxSoftmax	70.64	75.35	73.06	14.84
	$OSNN_{Cos}$	80.52	86.61	85.45	28.84
	$OSNN_{Euc}$	80.68	87.13	86.28	35.32
CroW	LOF_{Cos}	91.00	96.60	96.63	75.68
	LOF_{Euc}	93.18	97.87	97.89	91.40
	LOF_D_{Cos}	97.54	96.90	95.46	99.68
	LOF_D_{Euc}	97.50	97.11	95.08	99.52
	Mahalanobis	90.74	95.88	95.41	76.31
	$OSNN_{Cos}$	82.44	88.47	86.66	34.12
	$OSNN_{Euc}$	81.70	88.05	86.31	32.68
GAP_All	LOF_{Cos}	85.96	93.67	93.77	62.32
	LOF_{Euc}	86.32	93.90	93.84	75.44
	LOF_D_{Cos}	91.32	90.19	87.49	19.72
	LOF_D_{Euc}	91.30	96.68	96.78	80.24
	Mahalanobis	85.18	92.19	91.74	57.35
	$OSNN_{Cos}$	80.66	87.13	86.17	28.32
	$OSNN_{Euc}$	80.04	86.95	85.93	32.80
GMP	LOF_{Cos}	92.44	97.84	97.86	86.76
	LOF_{Euc}	94.30	98.73	98.71	93.24
	LOF_D_{Cos}	97.56	97.64	96.84	99.68
	LOF_D_{Euc}	97.54	97.92	97.47	99.64
	Mahalanobis	81.63	88.06	86.89	38.18
	$OSNN_{Cos}$	80.80	86.83	85.15	23.36
	$OSNN_{Euc}$	82.08	88.07	86.21	29.48
SCDA	LOF_{Cos}	90.62	96.89	96.94	79.28
	LOF_{Euc}	93.40	98.40	98.42	91.36
	LOF_D_{Cos}	97.58	97.79	97.22	99.68
	LOF_D_{Euc}	97.58	97.17	95.91	99.68
	Mahalanobis	83.19	89.91	88.93	45.93
	$OSNN_{Cos}$	80.88	86.81	85.19	25.44
	$OSNN_{Euc}$	82.70	88.65	87.08	33.04

3.4.2 *Effect of the Feature Reduction*

Section Objectives

As we have shown above, the many OoD detection methods can work poorly in high-dimensional space due to the "curse of dimensionality". This section focused on verifying if the standard technique of reduction features (used mainly in the image retrieval problem) also works in the context of OoD data. The idea of transforming features is relatively simple: first, the L2 normalization is used, next, the PCA with whitening, and again L2 normalization. This procedure should promote retrieval accuracy and reduce computational costs. We tested this method in the context of OoD detection based on the ResNet-101 fitted on the CIFAR-10 model using 2048 length GAP features.

The features modification with using L2 normalization -> PCA reduction -> whitening -> L2 normalization is described in many papers about image retrieval problems[247][75],[166]. PCA is used to reduce feature dimensionality, whitening[108] to down-weight co-occurrence between features, and all L2 normalizations are used to achieve unit length. All above should reduce the redundancy in features, speed up the calculation, and possibly improve results.

Our research focused on using the above procedure on GAP features obtained from the ResNet-101 model fitted on the CIFAR-10 dataset. These features contained raw 2048-length vectors. We modified them by using L2->PCA with whitening->L2, reducing dimensionally into n. We tested different variants of n: 3, 64, 128, and 512 length vectors. Next, we verified the OoD detection results by fitting the OoD algorithms with modified features.

In tables 3.14, 3.15 and 3.16 we presented the results of our experiments for the Noise, the SVHN, and the CIFAR-100 as OoD data. Thanks to normalization and whitening, the Euclidean and Cosine distance difference is not noticeable. Generally achieved results are slightly worse for the Mahalanobis, the MDistance, and the OSNN, but LOFs nearly equal or even slightly better. For instance, for the Noise, the raw features of the Mahalanobis achieved 94.99% and with PCA reduction into 128 length vectors into 92.54%, for the SVHN 93.91% (raw) and 89.47% (PCA n=128), and for the CIFAR-100 91.21% (raw) and 86.15% (PCA n=128). In contrast, for the LOF_{Cos} for the Noise, this OoD method achieved 93.39% of AUC for raw features, and 94.45% for PCA with a reduction into 128 length vectors (better), for the SVHN 92.39% (raw) and 91.86% (PCA n=128), and for the CIFAR-100 88.86% (raw) and 86.82% (PCA n=128), but already 88.02% for PCA with reduction into 64 length vectors.

We also compared the reduction level by PCA. The PCA n=3 was proposed for possible visualization purposes, but the results were poor. The PCA n=64 and PCA n=128 work significantly well – the results are similar to results based on the raw data (on the scope we wrote about in the previous paragraph). PCA n=512 works noticeably worse than raw features, which was unexpected.

Table 3.14: Compare different level of features reduction with using L2 normalization -> PCA into *n* dimensional with whitening -> L2 normalization – based on the ResNet-101 fitted on the CIFAR-10 and the **Noise** as OoD.

Feature modification	Method	DTACC	AUC	AUPR	TNR at TPR 95%
PCA n=3	LOF_{Cos}	74.38	79.50	77.56	36.77
	LOF_{Euc}	74.25	79.34	77.42	36.86
	LOF_D_{Cos}	79.45	84.54	81.11	43.38
	LOF_D_{Euc}	79.44	84.41	80.94	43.73
	Mahalanobis	76.42	82.92	81.26	34.17
	$MDistance_{Cos}$	76.50	83.09	81.73	33.11
	$MDistance_{Euc}$	76.73	83.41	82.05	34.68
	$OSNN_{Cos}$	67.77	72.47	70.51	12.68
	$OSNN_{Euc}$	67.77	72.47	70.51	12.68
PCA n=64	LOF_{Cos}	89.69	94.73	93.76	65.38
	LOF_{Euc}	89.71	94.73	93.76	65.33
	LOF_D_{Cos}	90.08	93.58	90.22	53.77
	LOF_D_{Euc}	90.05	93.58	90.22	53.92
	Mahalanobis	85.55	90.41	88.78	32.07
	$MDistance_{Cos}$	78.50	83.44	80.54	20.66
	$MDistance_{Euc}$	78.75	83.99	81.28	23.23
	$OSNN_{Cos}$	85.30	90.51	89.39	35.54
	$OSNN_{Euc}$	85.30	90.51	89.39	35.54
PCA n=128	LOF_{Cos}	89.52	94.45	92.90	63.39
	LOF_{Euc}	89.51	94.45	92.90	63.41
	LOF_D_{Cos}	89.38	92.97	89.55	52.21
	LOF_D_{Euc}	89.37	92.99	89.59	52.00
	Mahalanobis	87.52	92.54	91.31	46.35
	$MDistance_{Cos}$	76.22	80.73	77.52	15.73
	$MDistance_{Euc}$	76.66	81.31	78.12	17.98
	$OSNN_{Cos}$	86.58	91.62	90.43	40.29
	$OSNN_{Euc}$	86.58	91.62	90.43	40.29
PCA n=512	LOF_{Cos}	84.00	90.81	89.50	55.07
	LOF_{Euc}	83.98	90.81	89.50	54.96
	LOF_D_{Cos}	86.65	91.75	88.50	57.98
	LOF_D_{Euc}	86.62	91.75	88.50	58.01
	Mahalanobis	87.88	93.44	93.06	53.16
	$MDistance_{Cos}$	75.52	78.66	74.74	6.14
	$MDistance_{Euc}$	74.89	77.97	73.91	5.90

Table 3.14: Compare different level of features reduction with using L2 normalization -> PCA into *n* dimensional with whitening -> L2 normalization – based on the ResNet-101 fitted on the CIFAR-10 and the **Noise** as OoD.

Feature modification	Method	DTACC	AUC	AUPR	TNR at TPR 95%
	$OSNN_{Cos}$	82.25	88.20	86.73	33.53
	$OSNN_{Euc}$	82.25	88.20	86.73	33.53
RAW n=2048	LOF_{Cos}	87.64	93.39	93.00	52.61
	LOF_{Euc}	87.77	92.85	91.32	47.71
	LOF_D_{Cos}	87.54	93.02	89.13	36.27
	LOF_D_{Euc}	87.77	92.70	88.24	37.73
	Mahalanobis	89.98	94.99	94.47	63.06
	$MDistance_{Cos}$	82.91	88.14	86.15	29.18
	$MDistance_{Euc}$	82.69	87.94	85.92	30.31
	$OSNN_{Cos}$	86.00	91.17	90.21	35.89
	$OSNN_{Euc}$	86.05	91.03	89.87	35.33

Table 3.15: Compare different level of features reduction with using L2 normalization -> PCA into *n* dimensional with whitening -> L2 normalization – based on the ResNet-101 fitted on the CIFAR-10 and the **SVHN** as OoD.

Feature modification	Method	DTACC	AUC	AUPR	TNR at TPR 95%
PCA n=3	LOF_{Cos}	73.19	78.48	76.54	33.98
	LOF_{Euc}	73.13	78.30	76.38	33.76
	LOF_D_{Cos}	78.44	83.53	80.13	40.08
	LOF_D_{Euc}	78.42	83.37	79.96	40.58
	Mahalanobis	75.22	81.56	79.68	30.96
	$MDistance_{Cos}$	75.17	81.73	80.12	29.87
	$MDistance_{Euc}$	75.74	82.34	80.83	32.06
	$OSNN_{Cos}$	67.17	71.56	69.31	12.04
	$OSNN_{Euc}$	67.17	71.56	69.31	12.04
PCA n=64	LOF_{Cos}	86.26	92.34	91.47	53.94
	LOF_{Euc}	86.27	92.36	91.49	53.85
	LOF_D_{Cos}	86.71	91.53	88.20	44.88
	LOF_D_{Euc}	86.72	91.54	88.21	45.07
	Mahalanobis	82.67	87.94	86.10	25.74
	$MDistance_{Cos}$	73.61	77.45	73.32	13.88

Table 3.15: Compare different level of features reduction with using L2 normalization -> PCA into *n* dimensional with whitening -> L2 normalization – based on the ResNet-101 fitted on the CIFAR-10 and the **SVHN** as OoD.

Feature modification	Method	DTACC	AUC	AUPR	TNR at TPR 95%
	$\text{MDistance}_{\text{Euc}}$	74.10	78.40	74.60	14.93
	OSNN_{Cos}	82.29	88.06	86.73	29.99
	OSNN_{Euc}	82.29	88.06	86.73	29.99
PCA n=128	LOF_{Cos}	85.80	91.86	90.75	51.03
	LOF_{Euc}	85.82	91.87	90.76	51.12
	$\text{LOF\_D}_{\text{Cos}}$	85.83	90.62	87.26	42.81
	$\text{LOF\_D}_{\text{Euc}}$	85.86	90.64	87.30	42.64
	Mahalanobis	83.91	89.47	87.91	33.75
	$\text{MDistance}_{\text{Cos}}$	74.41	79.02	75.93	16.36
	$\text{MDistance}_{\text{Euc}}$	74.97	79.80	76.80	17.99
	OSNN_{Cos}	83.17	88.91	87.60	32.90
	OSNN_{Euc}	83.17	88.91	87.60	32.90
PCA n=512	LOF_{Cos}	77.97	85.36	83.90	39.15
	LOF_{Euc}	77.95	85.36	83.90	39.02
	$\text{LOF\_D}_{\text{Cos}}$	81.18	87.44	84.23	43.32
	$\text{LOF\_D}_{\text{Euc}}$	81.17	87.44	84.23	43.15
	Mahalanobis	83.84	89.85	89.08	37.37
	$\text{MDistance}_{\text{Cos}}$	73.44	76.70	72.35	9.34
	$\text{MDistance}_{\text{Euc}}$	73.02	76.27	71.81	9.04
	OSNN_{Cos}	79.00	85.04	83.45	28.80
	OSNN_{Euc}	79.00	85.04	83.45	28.80
RAW n=2048	LOF_{Cos}	86.22	92.39	92.03	48.96
	LOF_{Euc}	86.08	91.81	90.71	45.64
	$\text{LOF\_D}_{\text{Cos}}$	86.24	92.05	88.28	34.79
	$\text{LOF\_D}_{\text{Euc}}$	86.10	91.60	87.28	36.98
	Mahalanobis	87.66	93.91	93.73	57.87
	$\text{MDistance}_{\text{Cos}}$	81.28	86.53	84.34	25.30
	$\text{MDistance}_{\text{Euc}}$	80.06	84.71	81.72	19.15
	OSNN_{Cos}	84.06	89.77	88.78	32.37
	OSNN_{Euc}	84.16	89.69	88.56	31.59

Table 3.16: Compare different level of features reduction with using L2 normalization -> PCA into *n* dimensional with whitening -> L2 normalization – based on the ResNet-101 fitted on the CIFAR-10 and the **CIFAR-100** as OoD.

Feature modification	Method	DTACC	AUC	AUPR	TNR at TPR 95%
PCA n=3	LOF_{Cos}	69.78	73.92	71.73	28.35
	LOF_{Euc}	69.68	73.76	71.60	28.16
	LOF_D_{Cos}	74.03	78.52	75.04	32.60
	LOF_D_{Euc}	73.91	78.36	74.88	33.29
	Mahalanobis	71.62	77.45	75.47	25.86
	$MDistance_{Cos}$	71.30	77.78	76.15	24.64
	$MDistance_{Euc}$	71.94	78.50	76.96	26.35
	$OSNN_{Cos}$	65.33	69.19	66.78	11.44
	$OSNN_{Euc}$	65.33	69.19	66.78	11.44
PCA n=64	LOF_{Cos}	81.74	88.02	86.62	38.74
	LOF_{Euc}	81.77	88.05	86.65	38.74
	LOF_D_{Cos}	82.46	87.77	84.34	30.89
	LOF_D_{Euc}	82.46	87.79	84.36	30.95
	Mahalanobis	78.95	84.31	82.27	22.22
	$MDistance_{Cos}$	71.84	75.06	70.50	12.53
	$MDistance_{Euc}$	71.73	75.17	70.59	12.99
	$OSNN_{Cos}$	78.63	84.39	82.59	24.24
	$OSNN_{Euc}$	78.63	84.39	82.59	24.24
PCA n=128	LOF_{Cos}	80.61	86.82	84.93	34.95
	LOF_{Euc}	80.56	86.83	84.95	34.99
	LOF_D_{Cos}	80.76	86.09	82.52	28.80
	LOF_D_{Euc}	80.74	86.10	82.56	28.67
	Mahalanobis	80.30	86.15	84.44	26.93
	$MDistance_{Cos}$	71.43	74.77	70.32	12.98
	$MDistance_{Euc}$	72.34	75.96	71.74	14.81
	$OSNN_{Cos}$	79.63	85.21	83.40	25.85
	$OSNN_{Euc}$	79.63	85.21	83.40	25.85
PCA n=512	LOF_{Cos}	71.09	77.62	75.73	23.84
	LOF_{Euc}	71.09	77.62	75.73	23.78
	LOF_D_{Cos}	74.66	80.97	77.73	28.18
	LOF_D_{Euc}	74.64	80.96	77.74	28.10
	Mahalanobis	79.57	85.39	83.98	27.85
	$MDistance_{Cos}$	72.60	75.58	70.98	8.77
	$MDistance_{Euc}$	72.17	74.98	70.23	8.04

Table 3.16: Compare different level of features reduction with using L2 normalization -> PCA into *n* dimensional with whitening -> L2 normalization – based on the ResNet-101 fitted on the CIFAR-10 and the **CIFAR-100** as OoD.

Feature modification	Method	DTACC	AUC	AUPR	TNR at TPR 95%
	$OSNN_{Cos}$	74.83	80.75	78.82	23.41
	$OSNN_{Euc}$	74.83	80.75	78.82	23.41
RAW n=2048	LOF_{Cos}	82.31	88.96	88.45	37.07
	LOF_{Euc}	81.69	88.07	86.81	34.31
	LOF_D_{Cos}	82.23	88.77	85.28	26.35
	LOF_D_{Euc}	81.63	87.94	83.99	27.75
	Mahalanobis	84.47	91.21	90.97	46.03
	$MDistance_{Cos}$	77.66	82.87	80.10	22.05
	$MDistance_{Euc}$	77.38	82.68	80.14	23.71
	$OSNN_{Cos}$	79.81	86.02	84.92	26.17
	$OSNN_{Euc}$	79.56	85.72	84.46	25.73

Section Summary The feature reduction is successfully used in many image retrieval solutions and can be applied in OoD problems when using LOF-based methods. However, it should be avoided for other OoD methods. Our experiments suggest using reduction to around 128 components. We did not make performance tests, but we expected to reduce training and usage time and memory – with 128 components, the reduction was 16 times.

3.4.3 *An Impact of Data Augmentation Techniques on the Robustness and OoD Detection*

Section Objectives Images augmentation is the technique of changing training data to increase their diversity. The various strategies allow networks to learn different patterns and, consequently, different features. We hypothesize that by choosing the proper strategy, the networks can increase the robustness and capabilities of OoD detection. Both are strongly correlated due to robust models generating robust features allowing for better distinction between known and unknown samples. We trained CNNs models with 21 different augmentations to verify the above statement. For checking robustness, we prepared 10 types of distortions for the same test subset, and we checked the influence of the models on the final accuracy. For checking the OoD we performed classic tests described in previous sections by using the models with diverse augmentations. Part of this research focused on robustness is based on one of our papers[239].

Augmentation[219] is a popular technique in deep learning to increase training data diversity. There are many different approaches. The typical ones are, among others, changing the brightness, rotation, or applying horizontal flip. However, there are also specific applications for instance, in [65] authors focus on the influence of color

augmentation for skin image analysis, in [237] on the influence of the images color spaces, in [63] on using GAN to modify the image, or in [45] on automatically search proper augmentation policies.

First, we have checked the influence of the augmentation of the robustness of the models. The input data significantly influence the final shape of the features obtained by CNNs. There are papers[185][76], which deal with similar problems too. However, we focused on different type of distortions (described below), and they focused on distortions based on the ℓ_2- and ℓ_∞-norm-bounded perturbations[44].

We trained CNN models, the ResNet-101 on the CIFAR-10 and the MobileNet v2 on the CIFAR-100 datasets. We set the same hyperparameters for all models except the augmentation strategy. We normalized all images to a uniform mean of 0 and a standard deviation of 1. We tried the following augmentation techniques (mostly we used Albumentations[28] library): None, Affine, CLAHE, CoarseDropout, ColorJitter, CropAndPad, CutMix[279], FancyPCA, GaussNoise, GridDistortion, HorizontalFlip, HueSaturationValue, MedianBlur, MixUp[284], RandomBrightnessContrast, RandomGridShuffle, RandomShadow, Solarize, VerticalFlip, Complex 1 (mix of HorizontalFlip, RandomBrightnessContrastm, ShiftScaleRotate, ImageCompression, HueSaturationValue), and Complex 2 (mix of HorizontalFlip, ShiftScaleRotate, Blur, OpticalDistortion, GridDistortion, HueSaturationValue). See the details of each method in the library documentation [1].

Detailed parameters for each augmentation are presented below. For CropAndPad, we used $px = (-3, 3)$, for CoarseDropout $max\_holes = 5$, $max\_height = 3$, and $max\_width = 3$, for GaussNoise $var\_limit = (15.0, 80.0)$, for MedianBlur $blur\_limit = 3$, for Complex 1 ShiftScaleRotate with $rotate\_limit = 15$, $scale\_limit = 0.10$, ImageCompression with value 80, and for Complex 2 ShiftScaleRotate with $shift\_limit = 0.05$, $scale\_limit = 0.05$, and $rotate\_limit = 5$, Blur with $blur\_limit = 3$. For all others approaches, we used Albumentations's default parameters. For MixUp, we used the publicly available code from the authors of [284] with default configuration, and for CutMix this code[2] with followig parameters $beta = 1.0$ and $num\_mix = 6$.

Figure 3.9 showed the proposed augmentations for the CIFAR-10. By default, the augmentation is applied with 50% probability for most methods, but the image showed examples with artificially increased that probability to 1.0. The above augmentation methods can be grouped in many ways. For instance, by spatial-level transforms (i.e., Affine, CropAndPad, VerticalFlip), by color manipulations (i.e., ColorJitter, GaussNoise, RandomBrightnessContrast, HueSaturationValue), by adding minor distortion (i.e., CoarseDropout, GridDistortion, GaussNoise, GridDistortion, RandomGridShuffle), by mixing different images (CutMix and MixUp), by "weather" augmentation (i.e., RandomShadow, Solarize), by complexity (Complex 1, Complex 2).

Overall, we had 21 CNN models for the ResNet-101 fitted on the CIFAR-10 and 21 MobileNet v2 models fitted on the CIFAR-100.

We measured robustness as the accuracy of the test subset with applied variety distortions. By distortions, we mean applying strong augmentation so that humans could mostly recognize objects on the images, despite adding severe distortion. We tested the following methods: GaussNoise, Rotate, Blur, Downscale, Cutout, Hue, RandomBrightness, OverlayImage, ToGray. See the examples of the above in figure 3.10. The experiments of checking robustness consisted of testing the accuracy of the data entirely modified by the above strong augmentation. We tested all 21 models for ResNet-101 and MobileNet v2.

[1] https://albumentations.ai/docs/api_reference/full_reference/

[2] https://github.com/ildoonet/cutmix

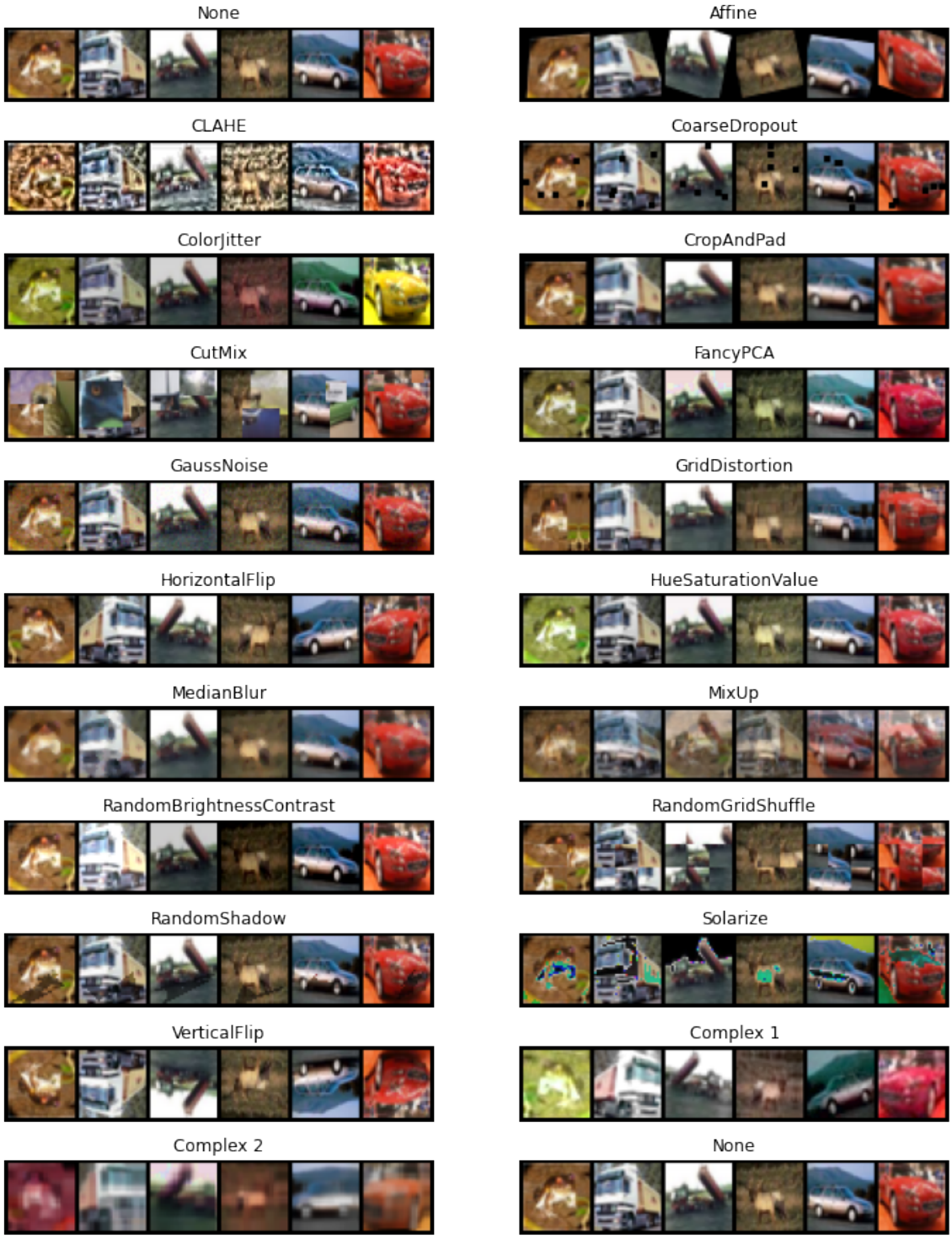


Figure 3.9: The examples of augmentations that we used. By default, we applied the augmentation with 50% probability.

Detailed parameters for the Albumentations library for each strong augmentation(distortions) are presented below. For GaussNoise $var\_limit = (0.035, 0.035)$, for Rotate one of the $limit = [90, 90]$ or $limit = [270, 270]$, for Blur $blur\_limit = [4, 4]$, for Downscale $scale\_min = 0.4$ and $scale\_max = 0.4$, for Cutout we used twice Cutout methods with diffrents configurations $num\_holes = 5$, $max\_h\_size = 10$, $max\_w\_size = 3$ and $num\_holes = 5$, $max\_h\_size = 3$, $max\_w\_size = 10$ for Hue one of the

$hue\_shift\_limit = (20, 20)$ or $hue\_shift\_limit = (-20, -20)$ both with $at\_shift\_limit = 0$ and $val\_shift\_limit = 0$, for RandomBrightness one of the $limit = (0.5, 0.5)$ or $limit = (-0.5, -0.5)$. For OverlayImage, one of the five emoticons 12px by 12px image was overlaid (about 14% of the image area). For all complex images, the probability of applied augumentation was set to 1.0. Others parameters were set as default.

The results of our robustness test are shown in table 3.17. The column *None* presents the classic test accuracy without any images transformations. All augmentation techniques for the ResNet-101 fitted on the CIFAR-10 significantly increase the final results from the 75% (no augmentation) up to 85% (least MedianBlur) or 94% (most Affine and CropAndPad). The MobileNet v2 fitted on the CIFAR-100 model achieved 64% without augmentation. Up to 9 methods did not improve results. The best method achieved up to 9p.p. better results (Affine).

We can see that generally, CropAndPad with Affine returned the best results. The above suggests that essential augmentation is changing the object's position. HorizontalFlip (92% for the CIFAR-10 and 69% for the CIFAR-100) achieved a similarly good result, which confirms above. The methods that change a single pixel's colors did not achieve the best results - see, i.e., ColorJitter, FancyPCA, HSV, or RandomBrightnessConstrast. It suggests that images are varied naturally enough, and only minor changes in pixels may be applied. The distortion and mixing of the two images group achieved not bad results.

In the rest of the columns of table 3.17, we presented the results of robustness analysis based on the final accuracy on distorted images.

The Hue was the least effective approach - the networks seemed to be relatively robust on it. We hypothesize the same as above: the objects in both datasets are varied on pixel-level naturally enough. A similar case was for another pixel-level change as ToGray. Downscale was the more complex distortion for the humans and CNN models. The well-known fact that the networks are not robust for rotation[133] was also observable in our experiments. All augmentation methods nearly equally failed. The most variation distractions were Cutout and Blur. The good augmentations were similar to this distortion, so CoarseDropout and CutMix for Cutout and Complex 2 and MedianBlur for Blur. The Affine or CropAndPad were good alternative methods.

There are other notable results. For Downscale, the no augmentation achieved the best results for CIFAR-10 ResNet-101. The RandomBrightnessContrast worked well only for RandomBrightness distortion. CutMix generally worked well for all distortions, especially for OverlayImage. Complex augmentation is a good direction for increasing robustness – however, they were not as significant as we expected. A simple and popular approach - CropAndPad - achieved outstanding results.

Figure 3.10: The example of distorted images used for robustness evaluation. We set parameters so that humans could recognize the correct class of the image in most cases.

Table 3.17: The robustness analysis based on non-modified and strongly distorted images for the CIFAR-10 and the CIFAR-100 datasets. We presented an evaluation of two different CNN models.

Model	None	Blur	Cutout	Downscale	GaussNoise	Hue	OverlayImage	RandomBrightness	Rotate	ToGray	Avg

Table 3.17: The robustness analysis based on non-modified and strongly distorted images for the CIFAR-10 and the CIFAR-100 datasets. We presented an evaluation of two different CNN models.

Aug Model	None	Blur	Cutout	Downscale	GaussNoise	Hue	OverlayImage	RandomBrightness	Rotate	ToGray	Avg
None	75%	49%	32%	**61%**	34%	70%	55%	29%	28%	66%	50%
Affine	**94%**	59%	81%	32%	46%	90%	80%	63%	**41%**	**88%**	67%
CLAHE	88%	56%	34%	25%	52%	85%	66%	66%	35%	82%	59%
CoarseDrop	89%	36%	**82%**	24%	49%	85%	71%	58%	34%	85%	61%
ColorJitter	87%	37%	32%	22%	51%	87%	65%	63%	31%	85%	56%
Crop&Pad	**94%**	71%	73%	33%	52%	90%	74%	64%	**41%**	87%	**68%**
CutMix	92%	27%	81%	24%	29%	87%	**87%**	62%	39%	84%	61%
FancyPCA	88%	29%	33%	20%	47%	87%	68%	63%	32%	85%	55%
GaussNoise	88%	43%	33%	28%	51%	85%	67%	56%	32%	83%	57%
GridDist	93%	69%	60%	53%	48%	88%	73%	63%	38%	85%	67%
HorizFlip	92%	32%	39%	29%	52%	89%	73%	64%	37%	87%	59%
HSV	87%	29%	32%	21%	44%	87%	64%	61%	32%	85%	54%
MedianBlur	85%	76%	37%	46%	52%	80%	65%	55%	31%	78%	61%
MixUp	87%	49%	37%	37%	43%	84%	69%	61%	31%	77%	58%
RBrighCon	88%	39%	29%	20%	61%	85%	67%	74%	34%	84%	58%
RGridShuf	91%	24%	69%	21%	37%	86%	80%	61%	40%	85%	59%
RShadow	88%	37%	59%	25%	40%	84%	71%	53%	34%	83%	57%
Solarize	88%	37%	48%	21%	41%	84%	71%	53%	34%	83%	56%
VerticalFlip	88%	27%	32%	28%	39%	83%	66%	56%	40%	81%	54%
Complex 1	93%	67%	42%	38%	**65%**	**92%**	70%	**80%**	34%	**88%**	67%
Complex 2	93%	**89%**	48%	58%	52%	**92%**	70%	63%	34%	85%	**68%**
Avg	89%	47%	48%	32%	47%	86%	70%	60%	35%	83%	60%
The MobileNetV2 on the CIFAR-100											
None	64%	23%	11%	8%	19%	53%	25%	32%	24%	44%	30%
Affine	**71%**	33%	50%	13%	17%	57%	34%	28%	**30%**	45%	**38%**
CLAHE	64%	28%	15%	13%	23%	53%	25%	34%	23%	42%	32%
CoarseDrop	66%	23%	**55%**	11%	20%	54%	27%	32%	24%	44%	36%
ColorJitter	63%	18%	9%	9%	25%	62%	24%	36%	21%	48%	32%
Crop&Pad	70%	43%	43%	13%	18%	57%	31%	32%	29%	45%	**38%**
CutMix	70%	20%	53%	5%	14%	55%	**58%**	31%	**30%**	46%	**38%**

Table 3.17: The robustness analysis based on non-modified and strongly distorted images for the CIFAR-10 and the CIFAR-100 datasets. We presented an evaluation of two different CNN models.

Aug Model	None	Blur	Cutout	Downscale	GaussNoise	Hue	OverlayImage	RandomBrightness	Rotate	ToGray	Avg
FancyPCA	64%	19%	11%	11%	22%	62%	24%	35%	21%	**50%**	32%
GaussNoise	62%	28%	12%	17%	28%	52%	25%	29%	22%	42%	32%
GridDist	68%	41%	32%	33%	17%	53%	26%	30%	29%	41%	37%
HorizFlip	69%	23%	14%	11%	20%	57%	30%	34%	27%	47%	33%
HSV	63%	18%	10%	10%	22%	62%	26%	33%	21%	49%	31%
MedianBlur	63%	36%	16%	23%	18%	51%	27%	30%	23%	39%	33%
MixUp	68%	29%	16%	19%	21%	57%	33%	35%	25%	46%	35%
RBrighCon	64%	24%	12%	10%	27%	53%	24%	**46%**	23%	43%	33%
RGridShuf	67%	16%	28%	7%	14%	54%	39%	28%	28%	42%	32%
RShadow	65%	23%	35%	8%	19%	53%	38%	29%	25%	43%	34%
Solarize	64%	24%	24%	9%	18%	53%	37%	27%	24%	44%	32%
VerticalFlip	64%	20%	12%	9%	16%	51%	22%	29%	29%	43%	29%
Complex 1	67%	40%	13%	17%	**32%**	**66%**	28%	44%	24%	49%	**38%**
Complex 2	66%	**60%**	20%	**36%**	19%	63%	24%	30%	23%	42%	**38%**
Avg	66%	28%	23%	14%	20%	56%	30%	33%	25%	45%	34%

We checked the influence of augmentation for OoD detection – see table table 3.18. We tested two variants: the MobileNet v2 with the SVHN as the OoD and the ResNet-101 with the CIFAR-100 as the OoD, both trained on the CIFAR-10. We examined three OoD methods Mahalanobis, MaxSoftmax, and LOF_{Euc}. We can see that the influence of OoD is significant. For AUC the differences were up to even 20 p.p. (see ResNet-101 for MaxSoftmax). The outstanding method is the MixUp. Following the authors of this method[284] "mixup leads to decision boundaries that transition linearly from class to class, providing a smoother estimate of uncertainty". Interestingly, the similar method - CutMix was not working so well. The OoD methods did not influence the results as much as the augmentation strategy, suggesting that the feature's shape is more important than the OoD method. However, it seems, that there were methods that were "designed" for a particular OoD approach, like Complex ones, which worked better for Mahalanobis. The more general conclusion can also be that the more robust the model, the better OoD detection. Both are related.

Table 3.18: The OoD analysis based on different augumentation startegy. We tested the MobileNet v2 with the SVHN as the OoD and the ResNet-101 with the CIFAR-100 as the OoD, both trained on the CIFAR-10.

Model	Mahalanobis		MaxSoftmax		LOF_{Euc}	
	AUC	TNR at TPR 95%	AUC	TNR at TPR 95%	AUC	TNR at TPR 95%
Model: MobileNetV2 – OoD: SVHN						
None	89.2%	37.0%	81.0%	23.8%	83.7%	29.0%
Affine	90.4%	48.2%	85.9%	30.5%	78.7%	25.3%
CLAHE	89.0%	41.0%	83.0%	25.2%	81.2%	26.3%
CoarseDropout	89.6%	43.0%	82.8%	24.8%	82.6%	27.4%
ColorJitter	89.7%	41.6%	83.3%	25.8%	84.8%	30.8%
CropAndPad	90.1%	42.6%	86.5%	32.0%	80.8%	24.7%
CutMix	87.5%	43.9%	79.5%	30.5%	83.3%	29.4%
FancyPCA	90.4%	43.0%	82.4%	24.5%	84.7%	32.1%
GaussNoise	89.1%	40.2%	82.9%	24.8%	83.9%	29.6%
GridDistortion	90.5%	47.1%	86.8%	32.4%	81.0%	27.9%
HorizontalFlip	91.4%	47.3%	84.7%	28.8%	84.5%	32.7%
HSV	89.4%	39.2%	82.7%	25.2%	83.8%	29.4%
MedianBlur	90.2%	43.0%	82.2%	24.2%	83.4%	30.6%
MixUp	**96.6%**	**75.7%**	**96.3%**	**69.8%**	**98.4%**	**91.1%**
RBrightnessContrast	90.3%	44.1%	83.5%	26.5%	84.4%	29.9%
RGridShuffle	87.6%	40.3%	83.6%	25.8%	79.7%	25.5%
RShadow	90.3%	42.6%	83.4%	26.0%	85.5%	33.1%
Solarize	88.6%	38.8%	81.6%	23.2%	79.7%	23.3%
VerticalFlip	88.0%	43.3%	81.8%	23.1%	79.2%	26.7%
Complex 1	91.5%	52.6%	87.8%	35.6%	85.3%	38.0%
Complex 2	91.3%	52.5%	86.8%	31.1%	80.8%	28.6%
Model: ResNet-101 – OoD: CIFAR-100						
None	78.2%	30.4%	68.9%	11.9%	68.5%	17.2%
Affine	90.9%	58.3%	83.8%	26.5%	82.9%	34.7%
CLAHE	87.9%	47.8%	77.8%	17.6%	83.2%	35.1%
CoarseDropout	89.7%	51.5%	77.0%	16.2%	83.3%	33.3%
ColorJitter	85.7%	41.1%	76.8%	17.9%	81.5%	29.3%
CropAndPad	**92.7%**	61.1%	84.3%	27.8%	83.8%	35.1%
CutMix	85.9%	36.3%	82.0%	30.9%	80.5%	20.3%
FancyPCA	87.2%	39.0%	76.8%	16.3%	81.5%	26.8%
GaussNoise	87.5%	41.8%	78.0%	19.4%	82.5%	30.0%
GridDistortion	90.6%	52.2%	84.1%	27.7%	82.9%	33.4%

Table 3.18: The OoD analysis based on different augumentation startegy. We tested the MobileNet v2 with the SVHN as the OoD and the ResNet-101 with the CIFAR-100 as the OoD, both trained on the CIFAR-10.

Model	Mahalanobis		MaxSoftmax		LOF_{Euc}	
	AUC	TNR at TPR 95%	AUC	TNR at TPR 95%	AUC	TNR at TPR 95%
HorizontalFlip	90.4%	53.0%	82.6%	24.8%	84.5%	36.4%
HSV	85.4%	33.1%	76.3%	15.5%	80.2%	22.6%
MedianBlur	86.3%	39.7%	75.1%	15.1%	80.5%	25.4%
MixUp	88.1%	34.9%	**90.9%**	**38.3%**	**90.6%**	**41.4%**
RBrightnessContrast	89.2%	49.3%	76.9%	16.7%	83.1%	33.6%
RGridShuffle	88.2%	45.7%	80.4%	21.4%	81.3%	29.4%
RShadow	87.5%	43.2%	76.2%	17.2%	81.9%	30.9%
Solarize	87.9%	44.8%	75.7%	16.3%	80.8%	28.6%
VerticalFlip	86.5%	44.2%	77.1%	17.8%	81.3%	31.2%
Complex 1	91.1%	55.2%	84.4%	28.5%	85.2%	37.5%
Complex 2	91.8%	**61.1%**	82.6%	26.5%	83.7%	39.4%

Section Summary The most effective augmentation strategy was a modification of the object's position. Modification of pixels' color values did not lead to a significant influence on the effectiveness of the network. The robustness and OoD detection seemed to be correlated. The better robustness of the model, the better OoD detection efficiency. The variety of augmentations may help achieve more stable results. The idea of mixing images improves robustness, but above all, it is beneficial for OoD detection.

3.4.4 *Testing the Sensitivity of the OoD Detection Based on the CNN Model State*

Section Objectives We observed some not-intuitive behaviors during our experiments. CNN models with the same architecture, similar close-set accuracy, but different initial hyperparameters achieved distinguishable OoD performances. We decided to explore this phenomenon in detail in this section. Moreover, we tested the stability of OoD detection during the learning process.

To investigate the phenomenon described above, we trained two ResNet-101 CNN models (exact same architecture) fitted on the CIFAR-10 dataset. The first model (called A) achieved 94,79% of accuracy. It was fitting for 300 epochs using the MultiStepLR (for epochs: 175, 225, 275, and gamma=0.1) as a learning rate scheduler. The second model (called B) achieved 94,65% of accuracy. It was fitting for 100 epochs and with ReduceLROnPlateau(patience=5, min_lr=1e-06) as learning rate scheduler. For both we used SGD (lr=0.01, momentum=0.9, weight_decay=0.0005), the CrossEntropy as loss function, and the same augmentation strategy. For both, we used the same Python library.

In table 3.19 we presented a comparison of OoD results for both models A and B. We can notice that model B is much more effective in recognizing the Noise. Model A worked not well here. The best approach, the Mahalanobis, achieved only 94.99% of AUC and 63.06% of TNR at TPR 95%. Notice the simplicity of the above problem. However, for the SVHN and the CIFAR-100 detection, there was no one better, more suitable model. The best approach for SVHN for model A was Mahalanobis (93.91% of AUC and 57.87% of TNR at TPR 95%), and for the model, B was LOF_{Cos} (93.00% of AUC and 58.80% of TNR at TPR 95%). For the CIFAR-100, the best was model A with Mahalanobis(91.21% of AUC and 58.80% of TNR at TPR 46.03%), the model B achieved (88.03% of AUC and 35.70% of TNR at TPR 95%) with $OSNN_{EUC}$. As we can see, the models strongly affect of OoD methods, especially for far problems.

Table 3.19: Comparison of the OoD based on two models with exactly the same architecture and comparable accuracy. See the significant differences.

Out-dist	Method	Model	DTACC	AUC	AUPR	TNR at TPR 95%
Noise	LOF_{Cos}	A	87.64	93.39	93.00	52.61
	LOF_{Cos}	B	99.30	99.90	99.89	100.00
	LOF_{Euc}	A	87.77	92.85	91.32	47.71
	LOF_{Euc}	B	98.51	99.46	99.33	100.00
	LOF_D_{Cos}	A	87.54	93.02	89.13	36.27
	LOF_D_{Cos}	B	99.31	99.70	99.23	100.00
	LOF_D_{Euc}	A	87.77	92.70	88.24	37.73
	LOF_D_{Euc}	B	98.56	99.22	98.70	100.00
	Mahalanobis	A	89.98	94.99	94.47	63.06
	Mahalanobis	B	100.00	100.00	100.00	100.00
	MaxSoftmax	A	86.63	91.41	90.08	37.94
	MaxSoftmax	B	95.14	97.55	97.07	87.14
	$MDistance_{Cos}$	A	82.91	88.14	86.15	29.18
	$MDistance_{Cos}$	B	93.23	97.17	96.53	84.11
	$MDistance_{Euc}$	A	82.69	87.94	85.92	30.31
	$MDistance_{Euc}$	B	92.42	96.30	95.47	75.43
	$OSNN_{Cos}$	A	86.00	91.17	90.21	35.89
	$OSNN_{Cos}$	B	95.33	97.87	97.49	91.33
	$OSNN_{Euc}$	A	86.05	91.03	89.87	35.33
	$OSNN_{Euc}$	B	95.95	97.73	97.22	94.06
SVHN	LOF_{Cos}	A	86.22	92.39	92.03	48.96
	LOF_{Cos}	B	86.43	93.00	92.73	57.80

Table 3.19: Comparison of the OoD based on two models with exactly the same architecture and comparable accuracy. See the significant differences.

Out-dist	Method	Model	DTACC	AUC	AUPR	TNR at TPR 95%
	LOF_{Euc}	A	86.08	91.81	90.71	45.64
	LOF_{Euc}	B	80.81	87.95	87.24	36.20
	LOF_D_{Cos}	A	86.24	92.05	88.28	34.79
	LOF_D_{Cos}	B	86.30	92.99	92.30	59.97
	LOF_D_{Euc}	A	86.10	91.60	87.28	36.98
	LOF_D_{Euc}	B	82.48	89.51	88.43	43.48
	Mahalanobis	A	87.66	93.91	93.73	57.87
	Mahalanobis	B	82.25	89.33	88.77	41.34
	MaxSoftmax	A	83.77	88.93	87.40	32.11
	MaxSoftmax	B	86.23	91.60	90.81	43.22
	$MDistance_{Cos}$	A	81.28	86.53	84.34	25.30
	$MDistance_{Cos}$	B	83.79	90.11	89.01	42.25
	$MDistance_{Euc}$	A	80.06	84.71	81.72	19.15
	$MDistance_{Euc}$	B	80.67	86.29	83.41	24.13
	$OSNN_{Cos}$	A	84.06	89.77	88.78	32.37
	$OSNN_{Cos}$	B	86.39	91.88	90.86	43.60
	$OSNN_{Euc}$	A	84.16	89.69	88.56	31.59
	$OSNN_{Euc}$	B	86.21	91.76	90.74	43.73
CIFAR-100	LOF_{Cos}	A	82.31	88.96	88.45	37.07
	LOF_{Cos}	B	79.48	87.22	86.74	47.84
	LOF_{Euc}	A	81.69	88.07	86.81	34.31
	LOF_{Euc}	B	76.59	84.06	83.38	40.78
	LOF_D_{Cos}	A	82.23	88.77	85.28	26.35
	LOF_D_{Cos}	B	79.78	87.34	86.42	47.36
	LOF_D_{Euc}	A	81.63	87.94	83.99	27.75
	LOF_D_{Euc}	B	77.55	84.88	83.93	41.41
	Mahalanobis	A	84.47	91.21	90.97	46.03
	Mahalanobis	B	78.55	85.91	85.16	37.59
	MaxSoftmax	A	79.40	84.73	82.58	25.46
	MaxSoftmax	B	81.69	87.83	86.82	36.98
	$MDistance_{Cos}$	A	77.66	82.87	80.10	22.05

Table 3.19: Comparison of the OoD based on two models with exactly the same architecture and comparable accuracy. See the significant differences.

Out-dist	Method	Model	DTACC	AUC	AUPR	TNR at TPR 95%
	$\text{MDistance}_{\text{Cos}}$	B	79.59	85.67	83.93	34.36
	$\text{MDistance}_{\text{Euc}}$	A	77.38	82.68	80.14	23.71
	$\text{MDistance}_{\text{Euc}}$	B	74.84	79.13	74.91	13.65
	OSNN_{Cos}	A	79.81	86.02	84.92	26.17
	OSNN_{Cos}	B	81.80	88.02	86.85	35.18
	OSNN_{Euc}	A	79.56	85.72	84.46	25.73
	OSNN_{Euc}	B	81.95	88.03	86.85	35.70

Based on the above case, we decided to check the results fluctuation during the learning process. The hypothesis question was that the long-fitting models are more stable in OoD detection problem. We trained another ResNet-101 model. We fitted it on the CIFAR-10 dataset, and we saved the model states over every 10 epochs during the training phase. See figure 3.11, where the accuracy over the validation subset is shown. The model nearly stopped improving its results from 95 epochs. However, the total loss on the training data was still minimizing so that the features space could be changing – on the other hand, the learning rate was low, so the changes should not be significant. The learning rate was changing in the following epochs 0 (lr=0.01), 88 (lr=0.001), 122 (lr=0.0001).

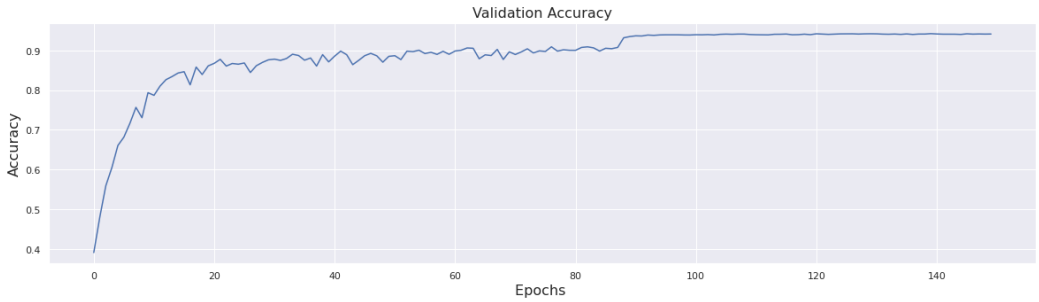


Figure 3.11: Changing the accuracy of the validation set over the epochs during the training process.

See figures 3.12, 3.13 and 3.14, where we presented the dynamic of changes AUC and TNR at TPR 95% over every 10 epochs. We can notice that the Noise and the SVHN were much less stable than the CIFAR-100.

For the Noise, there are many phenomena. First, the unfitted model (random weights) achieved remarkably good results for LOF_{Cos} and $\text{MDistance}_{\text{Euc}}$. They worked better than most methods even after long-term training. However, the $\text{MDistance}_{\text{Euc}}$ quickly dropped, then rose again (around 50 epoch) to drop again. Finally, it established around 50% for TNR at TPR 95%. The LOF_{Cos}, $\text{LOF\_D}_{\text{Euc}}$ and Mahalanobis worked as expected – the results were rising during training, and they were stable. The unexpected situations were between 50-70 epochs and 90-110 for OSNN_{Euc} and MaxLogits – they had the highest

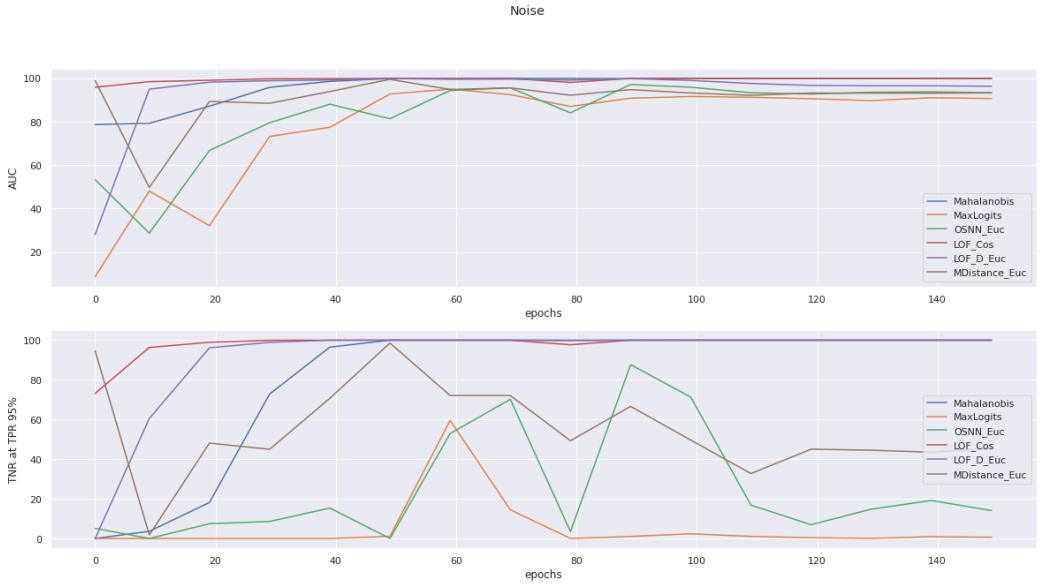


Figure 3.12: Comparison of the OoD approaches over the epochs on the Noise as OoD data.

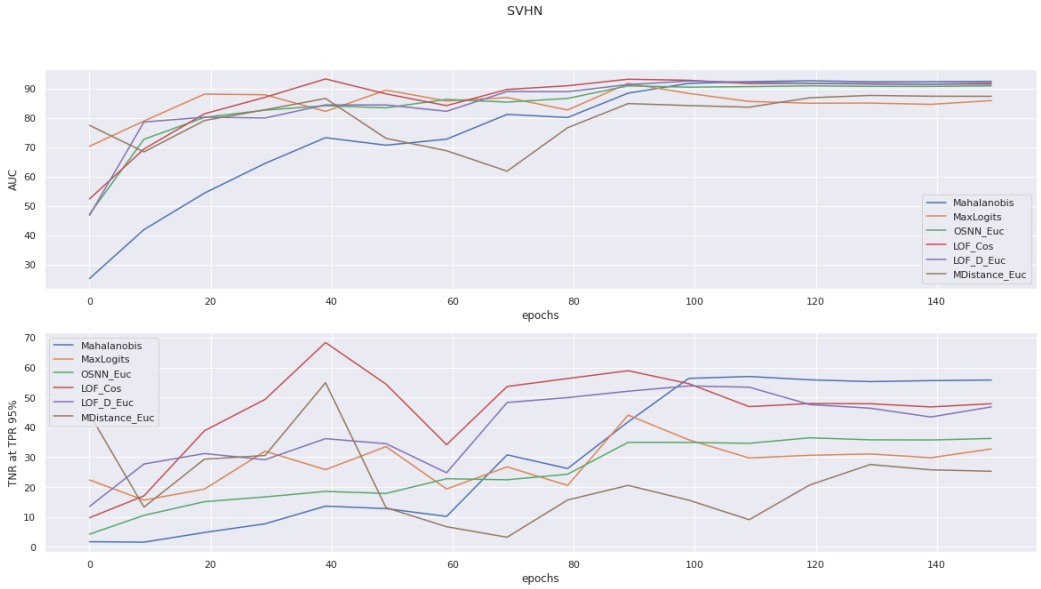


Figure 3.13: Comparison of the OoD approaches over the epochs on the SVHN as OoD data.

fluctuation. In the second window, the learning rate dropped – it could influence the network.

For the SVHN the situation was similar – nearly all OoD approaches were not very stable during training. Again the $\text{MDistance}_{\text{Euc}}$ worked well on the untrained model. The LOF_{Cos} achieved the best results overall in around 40 epoch, but next, the result was dropped. Generally, most of the methods were dropping after 40 epochs. The $\text{LOF\_D}_{\text{Euc}}$ and the Mahalanobis again were the very stable method

The CIFAR-100 seemed to be the most stable dataset. First of all, the untrained model did not work. Most methods except $\text{MDistance}_{\text{Euc}}$ were stable – the results increased over time without significant drops. The MaxLogits was achieving the best result (around 90 epoch), but then the results slightly dropped.

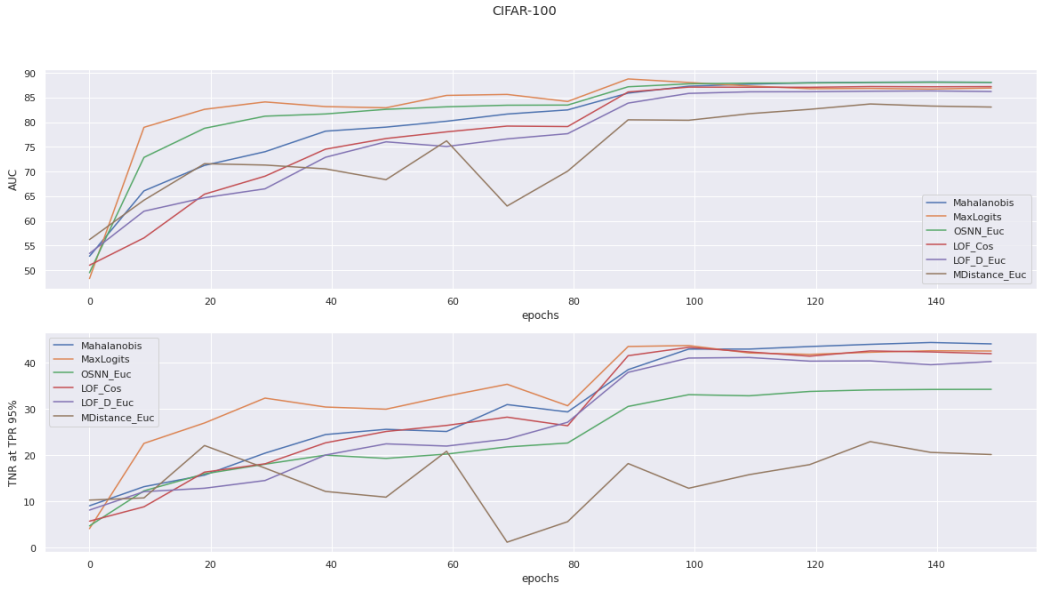


Figure 3.14: Comparison of the OoD approaches over the epochs on the CIFAR-100 as OoD data.

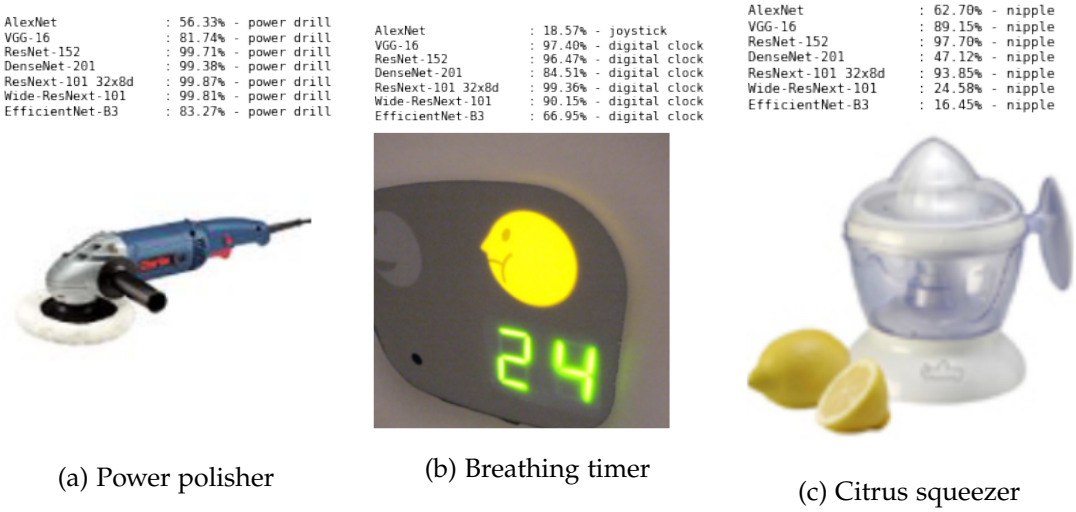
Section Summary The OoD approaches are very sensitive to network state - indeed, its weights. The models with the same CNN architecture but with slightly different hyperparameters can achieve completely different results - especially for more far OoD distribution problems. The problem is also visible for changing weights over the epochs during the training process. There were many hard-to-explain cases in our experiments - like good results on completely untrained models (see the Noise and $\text{MDistance}_{\text{Euc}}$) or unexpected significant drop (see SVHN and LOF_{Cos}). We suggest using Mahalanobis or $\text{LOF\_D}_{\text{Euc}}$ as the most stable methods. These results may be worrying in the context of benchmarks – where easily find the state of the model where one method overcomes another. For example, compare Mahalanobis and MaxLogits in 90 and 130 epoch for the SVHN and the CIFAR-100, the close-set accuracy was similar – in 90 epoch the MaxLogits overcome Mahalanobis, and in 130 was otherwise.

3.4.5 *Easy and Hard Subsets for OoD Detection*

Section Objectives The concept of near and far OoD examples is often used in literature[61][189]. However, there are no clear guidelines for that categorizing unknown images. Usually, we humans decide if the unknown classes are close enough (in image space) to in-distribution data to treat them as near or far. We argued with the above. The network generates features, and they determine if the specific image is close to others images with the same class or is not. Because the network can not be fully robust (i.e., by learning spurious correlations [272][240]), the interpretation of the image can be unstable. This subsection checked if there is always a connection between image and latent space. We propose a method to find images that can be far in the image space but close in the feature space or vice versa. Finally, we discuss if splitting into near and far images is right at all.

The root of our method was observing how different CNN classifiers the same images. There are many examples where models have misclassified the images in the exact same

wrong way with high probability. We performed experiments that confirmed the above. We based on the ImageNet datasets and the following CNNs models: the AlexNet, the VGG-16, the ResNet-152, the DenseNet-201, the ResNext-101 32x8d, the Wide-ResNext-101, and the EfficientNet-B3. See some examples in figure 3.15. The experiment provided that the problem lies in the training data, not the CNN architecture or how the model was fitted. We suggested that a similar effect exists in the OoD detection problem too. There should be images for which most CNNs generate the wrong features – i.e., in-distribution data are far (or OoD examples are near) from the other known images.



(a) Power polisher (b) Breathing timer (c) Citrus squeezer

Figure 3.15: The OoD examples for the ImageNet, where different models point out the same class.

We proposed a method for finding easy or hard examples. We assumed that the known-easy images are when any OoD method confidence/openness scores are low, and the known-easy ones when the score is high – and vice versa for unknown data. See the example in figure 3.16. If we had known and unknown datasets, we could easily find them extremely easy(low score for known, and high for unknown) and extremely hard subsets (high score for known, and low for unknown). Formally, for each $x \in X_{known}$ and $x \in X_{OoD}$ we need to calculate $y = d(x, X)$, where d is any OoD method which return confidence/openness score. Next, we need to sort y for X_{known} descending/ascending and for X_{OoD} ascending/descending. We can get extremely easy/hard datasets when we pick k first examples from each set based on sorted confidence scores. If we want to generate more robust results, we can calculate y as a mean of different OoD methods $y = \frac{1}{N_{d_{list}}} \sum_{d \in d_{list}} d(x, X)$, where d_{list} is list of OoD methods and $N_{d_{list}}$ number of these.

For instance, we generated easy and hard sets based on the ResNet-101 model and the CIFAR-10 as known and the SVHN and the CIFAR-100 as OoD data. We used LOF_{Euc} as the OoD method. Figure 3.17 presents picked examples. As we can see, the difference from humans perspective, was not significant, so we could easily recognize OoD examples – especially for the SVHN. In figure 3.18 we can see how the OoD method LOF_{Euc} for the model SplitNet-PyramidNet-272 (this model was not used for the process of picking the images) interpreted the classic the CIFAR-10 vs. the CIFAR-100 OoD example and for our extremely subsets.

We analyzed features for the easy and hard examples. Figure 3.19 presents PCA projection into 2-dimensional space. We can see that the images from the easy CIFAR-10 and the hard SVHN are inside the known clusters, and hard CIFAR-10 and easy SVHN are outside. The interesting thing is that specific classes were more willing to "attack".

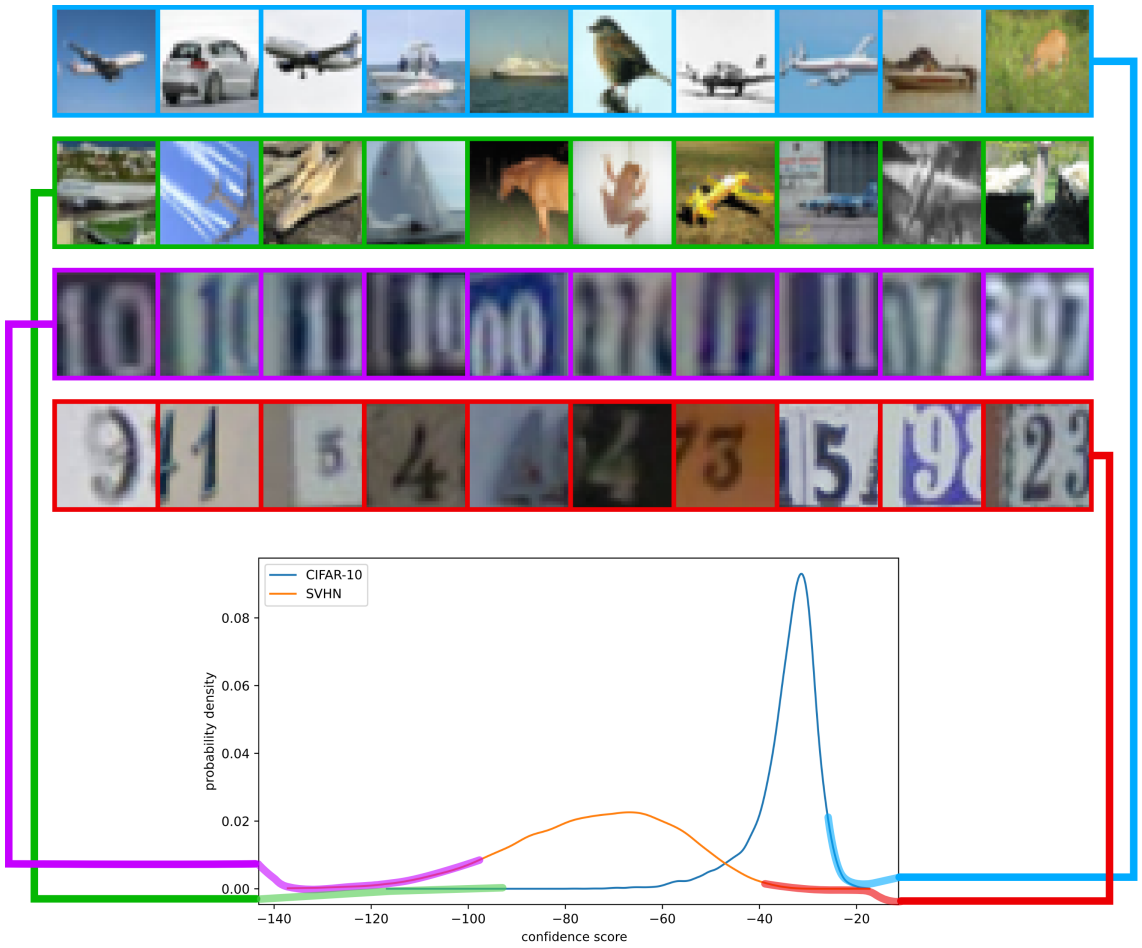


Figure 3.16: Choosing the easy or hard subsets is based on the confidence scores (here, we used Mahalanobis from DenseNet).

The points focused on the center of these specific classes. We checked which classes were specific by drawing a histogram of how a close-set model classifies (here by using a different model than the one used for generating extreme examples - SplitNet-PyramidNet-272) – see the figure 3.20. We see that "problematic" classes were mainly "cat" and "dog". It suggested that these classes were likely to be more prone to an OoD attack. See how sure the CNN model was - figure more than 95% certainty for chosen examples. These images were natural adversarial images.

We tested our extremely easy and hard OoD subsets and presented results in the table 3.20. Again, we used a different model(SplitNet-PyramidNet-272) than the one used for generating extreme examples. See that nearly all OoD methods failed for Far problems and worked perfectly well on Near ones.

Table 3.20: Results for our easy and hard datasets using a different model (SplitNet-PyramidNet-272) than the one used for choosing examples.

Out-dist	Method	DTACC	AUC	AUPR	TNR at TPR 95%
Easy SVHN	LOF_{Cos}	100.00	100.00	99.50	100.00

Table 3.20: Results for our easy and hard datasets using a different model (SplitNet-PyramidNet-272) than the one used for choosing examples.

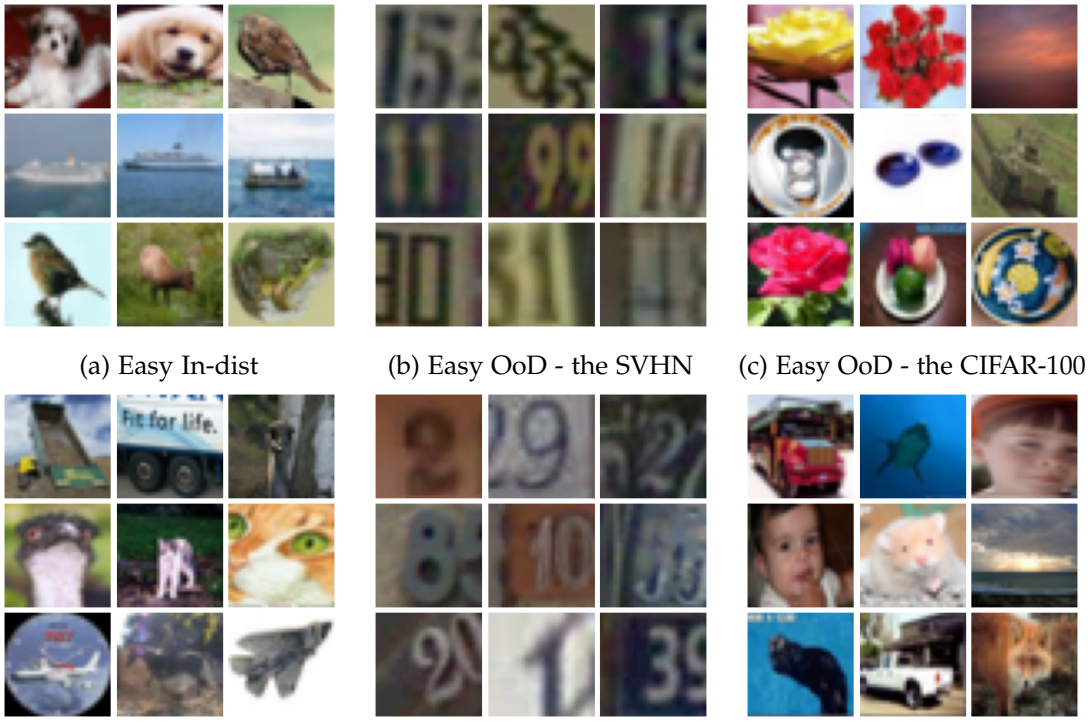
Out-dist	Method	DTACC	AUC	AUPR	TNR at TPR 95%
	LOF_{Euc}	100.00	100.00	99.50	100.00
	LOF_D_{Cos}	100.00	100.00	99.50	100.00
	LOF_D_{Euc}	100.00	100.00	99.50	100.00
	Mahalanobis	100.00	100.00	99.50	100.00
	MaxLogits	99.00	99.85	99.35	99.00
	MaxSoftmax	100.00	100.00	99.50	100.00
	$MDistance_{Cos}$	92.00	93.14	89.86	69.00
	$MDistance_{Euc}$	90.50	91.99	88.15	59.00
	$OSNN_{Cos}$	100.00	100.00	99.50	100.00
	$OSNN_{Euc}$	100.00	100.00	99.50	100.00
Hard SVHN	LOF_{Cos}	83.00	89.06	86.16	16.00
	LOF_{Euc}	83.50	88.56	86.69	21.00
	LOF_D_{Cos}	84.50	77.27	74.40	0.00
	LOF_D_{Euc}	84.00	82.09	78.00	0.00
	Mahalanobis	80.00	84.54	82.88	22.00
	MaxLogits	80.00	85.16	83.80	57.00
	MaxSoftmax	80.50	85.76	83.19	49.00
	$MDistance_{Cos}$	74.00	76.45	70.62	26.00
	$MDistance_{Euc}$	73.50	76.59	71.29	20.00
	$OSNN_{Cos}$	83.50	91.24	90.03	60.00
	$OSNN_{Euc}$	82.50	90.18	89.80	48.00
Easy CIFAR-100	LOF_{Cos}	99.50	99.97	99.47	100.00
	LOF_{Euc}	99.50	99.96	99.46	100.00
	LOF_D_{Cos}	99.00	99.96	99.46	100.00
	LOF_D_{Euc}	99.50	99.96	99.46	100.00
	Mahalanobis	99.50	99.97	99.47	100.00
	MaxLogits	89.00	89.15	85.31	79.00
	MaxSoftmax	94.00	95.61	93.67	90.00
	$MDistance_{Cos}$	90.00	91.97	88.51	70.00
	$MDistance_{Euc}$	88.00	90.20	86.28	54.00
	$OSNN_{Cos}$	98.50	99.86	99.36	99.00
	$OSNN_{Euc}$	99.00	99.88	99.38	99.00
Hard CIFAR-100	LOF_{Cos}	52.50	42.98	45.23	5.00
	LOF_{Euc}	52.00	41.57	44.36	4.00
	LOF_D_{Cos}	52.00	41.68	43.37	0.00

Table 3.20: Results for our easy and hard datasets using a different model (SplitNet-PyramidNet-272) than the one used for choosing examples.

Out-dist	Method	DTACC	AUC	AUPR	TNR at TPR 95%
	LOF_D_{Euc}	51.50	41.22	43.15	0.00
	Mahalanobis	50.00	35.97	41.20	2.00
	MaxLogits	56.00	54.06	56.47	16.00
	MaxSoftmax	55.00	51.55	54.17	14.00
	MDistance $_{Cos}$	52.50	40.53	44.15	7.00
	MDistance $_{Euc}$	53.50	46.07	47.45	7.00
	$OSNN_{Cos}$	54.00	47.23	48.95	11.00
	$OSNN_{Euc}$	54.50	45.39	48.23	10.00

Section Summary We showed that there was no clear connection between image and latent spaces. Based on it, we could propose the procedure of finding extremely hard or easy OoD subsets. The common usage of near and far definitions is inaccurate. We suggest that using the near and far should refer to feature space, not image space. For example, the SVHN set is often used as a far and easy problem, while as we showed that there are numerous near and hard images in this set too.

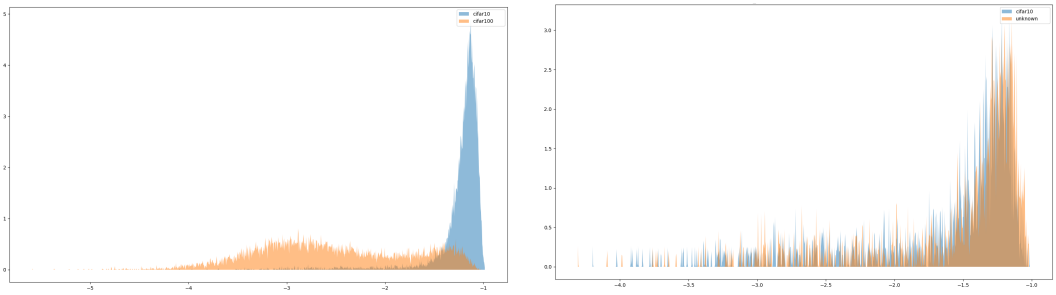
Moreover, our experiments confirmed that the OoD detection results depend on other factors (like training images that influence how the model generates its decisions boundaries) than the OoD methods themself. Finally, we showed that the hard OoD examples often focused on specific classes. We hypothesized that the problematic class should contain more robust image examples – however, it requires further research.



(a) Easy In-dist (b) Easy OoD - the SVHN (c) Easy OoD - the CIFAR-100

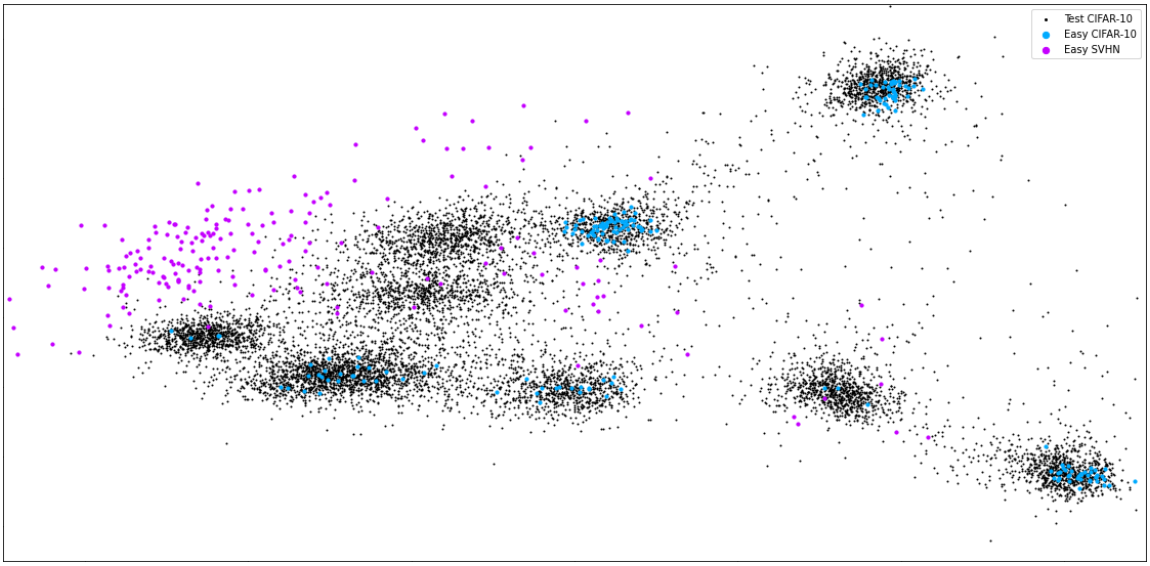
(d) Hard In-dist (e) Hard OoD - the SVHN (f) Hard OoD - the CIFAR-100

Figure 3.17: Examples of easy and hard exampels

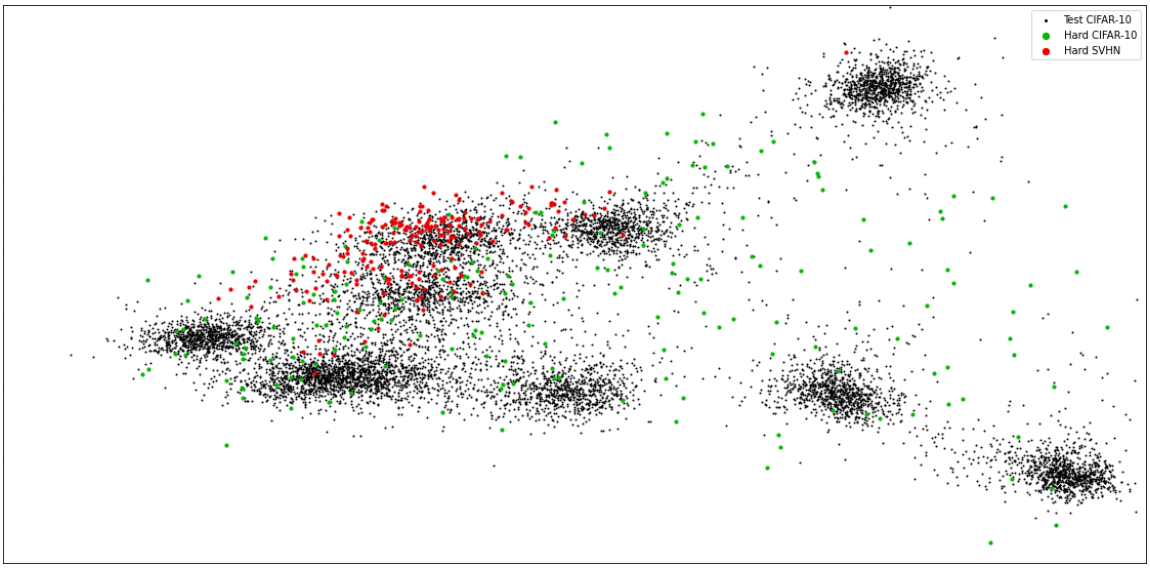


(a) v.s. CIFAR-100 (b) v.s. extremely far OoD

Figure 3.18: Confidence scores for the CIFAR-10 vs. the CIFAR-100(left) and our hard CIFAR-10 vs. hard CIFAR-100 subsets when used LOF_{Euc} for the model SplitNet-PyramidNet-272 (this model was not used for the process of picking the images).

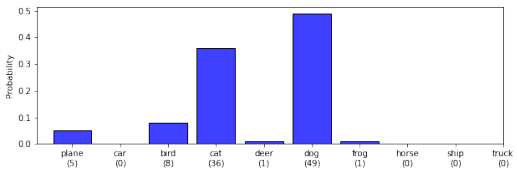


(a) Near

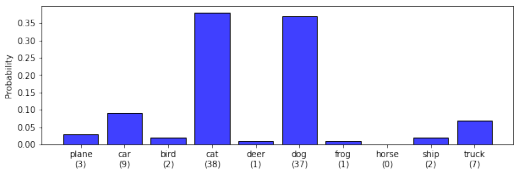


(b) Far

Figure 3.19: PCA projection into 2-dimensional space



(a) Histogram - Hard SVHN



(b) Histogram - Hard CIFAR-100

(c) Certainty - Hard SVHN

(d) Certainty - Hard CIFAR-100

Figure 3.20: Using the close-set classifier to test unknown hard images: histograms(top) and certainty (bottom).

3.4.6 *OoD Detection for Adversarial Attacks Protection*

Section Objectives The adversarial attacks are a severe threat and challenge to nowadays networks. This section focused on checking the research question if applying the OoD methods for close-set classifiers further increases the safety against the networks on these attacks. We suggested that the features generated by the network might not fit into the in-distribution data. The attacks focus on close-set classifiers (part of the networks), and these are based on the GAP features. Therefore, we hypothesized that the distinction between known data and attacks should be even more evident when changing the feature representation (by extract features from networks using different approaches). We performed our experiments using the CW, DeepFool, FGSM, OnePixel, PGD, and Square attacks on the ResNet-101 model fitted on the CIFAR-10 dataset.

The adversarial attacks are non easily defendable. Usually, the input image has to be modified, or the whole network retrained for a successful defense[278]. However, by examining the idea of standing behind these attacks, it was possible to propose another method. All attacks aim to cross the decision boundaries by a close-set classifier pointing to another class and minimizing input image changes. The infected images slightly move closer to another class in the feature space to finally fool the close-set classifier. However, the decision boundaries for the open-set approach are less vast. So, the shift in feature space could not be sufficient to cross the OoD's decision boundaries, thereby should increasing the safety against the networks on these attacks. In whole this work, we strongly propose using the OoD methods - so the adversarial attacks protection would be nearly cost-free. Using the OoD methods for the adversarial attacks is not a new idea, e.g., used in [132], but still not well studied.

We generated the strong attacks using the following methods: CW, DeepFool, FGSM, OnePixel, PGD, and Square (see section 2.3.2 for details). All of them were based on the ResNet-101 model fitted on the CIFAR-10 dataset. The attacks work on the different principles of operation, allowing for more generalized conclusions. In figure 3.21 was presented examples of attacks on the CIFAR-10 with showing the incorrect responses from the ResNet-101.

First, we performed the experiments by using the classic GAP features. See the results in table 3.21. We formed the rule that given protection against attack is acceptable when AUC >= 90% and TNR at TPR 95% >= 50%. For the assumed criterion all tests failed. Usually, the Mahalanobis achieved the best results. The PGD attacks were the most difficult for detection, and the OnePixel the easiest. We noticed that the close-set classifiers are usually based on the GAP features. It suggests that the above bad results are because all attacks target the GAP-generated features. The evaluation metrics should be increased by replacing the feature extraction method. We tested this hypothesis.

Table 3.21: Comparision of the OoD methods as the defense approach against the adversarial attacks based on the CIFAR-10 and the ResNet-101.

Attack	Method	DTACC	AUC	AUPR	TNR at TPR 95%
CW	LOF_{Cos}	81.25	85.78	84.19	22.40

Table 3.21: Comparision of the OoD methods as the defense approach against the adversarial attacks based on the CIFAR-10 and the ResNet-101.

Attack	Method	DTACC	AUC	AUPR	TNR at TPR 95%
	LOF_{Euc}	80.95	84.93	82.77	16.80
	LOF_D_{Cos}	81.35	85.38	82.33	17.40
	LOF_D_{Euc}	80.80	84.54	81.15	17.80
	Mahalanobis	83.85	88.80	87.71	32.50
	MaxLogits	80.65	83.86	81.29	13.10
	MaxSoftmax	82.00	86.20	84.15	22.40
	$MDistance_{Cos}$	80.80	84.42	81.18	21.50
	$MDistance_{Euc}$	79.90	83.35	80.22	19.30
	$OSNN_{Cos}$	81.30	85.77	84.08	18.80
	$OSNN_{Euc}$	80.40	85.39	83.62	15.80
DeepFool	LOF_{Cos}	78.35	83.32	81.20	17.90
	LOF_{Euc}	78.30	82.78	80.18	16.20
	LOF_D_{Cos}	78.65	82.92	79.61	13.30
	LOF_D_{Euc}	78.55	82.30	78.64	14.40
	Mahalanobis	80.95	86.36	85.18	29.10
	MaxLogits	77.35	80.96	77.85	10.50
	MaxSoftmax	79.10	83.08	80.50	18.50
	$MDistance_{Cos}$	78.65	81.89	77.91	17.00
	$MDistance_{Euc}$	78.20	82.02	78.91	18.90
	$OSNN_{Cos}$	78.95	83.51	81.43	16.20
	$OSNN_{Euc}$	78.40	83.13	80.99	14.10
FGSM	LOF_{Cos}	74.85	80.35	78.66	21.40
	LOF_{Euc}	74.40	79.77	77.80	19.70
	LOF_D_{Cos}	74.75	79.84	76.70	15.70
	LOF_D_{Euc}	74.50	79.16	75.71	16.50
	Mahalanobis	76.95	82.74	82.00	25.00
	MaxLogits	72.90	76.10	72.20	17.00
	MaxSoftmax	72.85	75.64	71.93	15.70
	$MDistance_{Cos}$	70.60	73.75	69.96	14.30
	$MDistance_{Euc}$	70.40	73.81	70.56	14.60
	$OSNN_{Cos}$	73.15	78.38	76.63	14.00
	$OSNN_{Euc}$	72.55	77.99	76.06	12.40
OnePixel	LOF_{Cos}	82.25	87.51	86.27	30.40
	LOF_{Euc}	81.35	86.60	84.83	26.20
	LOF_D_{Cos}	82.25	86.67	83.63	19.50

Table 3.21: Comparision of the OoD methods as the defense approach against the adversarial attacks based on the CIFAR-10 and the ResNet-101.

Attack	Method	DTACC	AUC	AUPR	TNR at TPR 95%
	LOF_D_{Euc}	81.40	85.80	82.40	20.70
	Mahalanobis	85.20	90.14	89.63	38.90
	MaxLogits	80.20	84.83	82.53	21.00
	MaxSoftmax	80.95	85.44	83.30	23.50
	$MDistance_{Cos}$	79.70	84.11	81.38	27.00
	$MDistance_{Euc}$	77.85	81.84	78.75	18.40
	$OSNN_{Cos}$	81.75	86.38	85.01	20.50
	$OSNN_{Euc}$	81.15	85.95	84.20	17.30
PGD	LOF_{Cos}	63.65	65.60	64.42	8.20
	LOF_{Euc}	64.60	67.29	65.91	9.40
	LOF_D_{Cos}	63.65	65.44	63.73	5.40
	LOF_D_{Euc}	64.40	66.91	64.57	7.70
	Mahalanobis	65.90	66.61	66.14	7.30
	MaxLogits	53.80	47.67	48.77	8.60
	MaxSoftmax	52.70	46.51	47.46	5.90
	$MDistance_{Cos}$	51.85	50.95	50.26	6.10
	$MDistance_{Euc}$	56.30	55.50	53.43	5.20
	$OSNN_{Cos}$	60.05	60.71	60.20	4.30
	$OSNN_{Euc}$	61.10	62.27	61.60	4.60
Square	LOF_{Cos}	79.70	84.51	82.65	20.10
	LOF_{Euc}	78.95	83.62	81.44	17.60
	LOF_D_{Cos}	79.70	84.10	81.05	14.50
	LOF_D_{Euc}	78.90	83.19	79.96	16.00
	Mahalanobis	83.80	88.12	86.90	29.30
	MaxLogits	79.10	82.85	80.36	14.80
	MaxSoftmax	80.45	84.32	82.13	20.20
	$MDistance_{Cos}$	78.80	83.02	79.99	19.90
	$MDistance_{Euc}$	76.30	79.39	75.37	8.20
	$OSNN_{Cos}$	79.55	84.61	82.94	17.50
	$OSNN_{Euc}$	78.75	84.13	82.06	15.20

In table 3.22 we presented the best results for the same problem, but with the best-fitted feature extraction strategy. We tested the following methods: GAP, GMP, GAP_All, CroW, and SCDA (see datils in section 2.2.8.1). Here, we do not perform MaxSoftmax and MaxLogits, because these do not depend on features but logits.

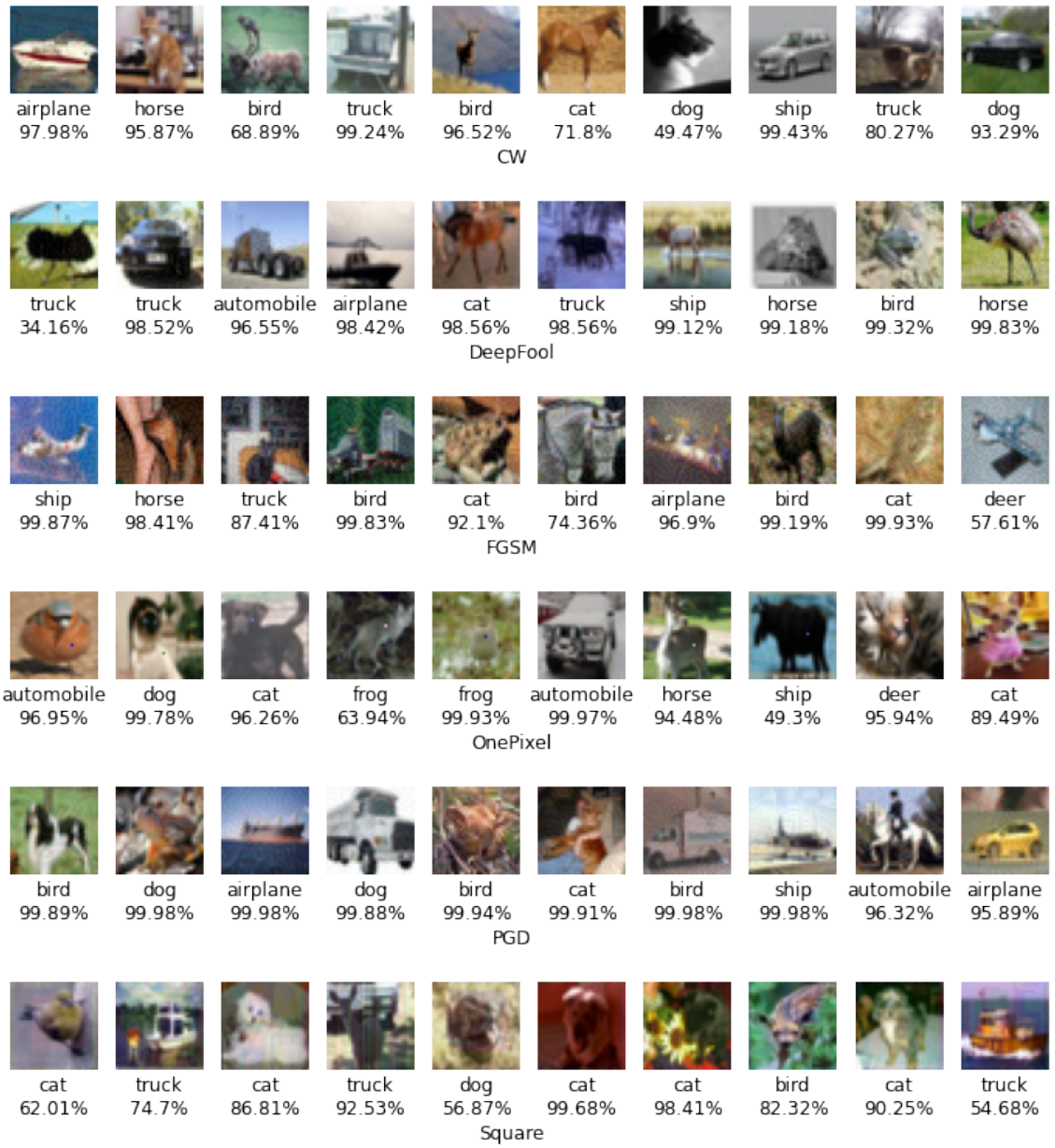


Figure 3.21: The examples of the attacks on the CIFAR-10. The attacked ResNet-101 model was fooled on these examples – see the incorrect response below each image.

Our hypothesis that changing the feature extraction strategy should improve performance on the attacks has been confirmed. The final results increased significantly by changing the feature extraction approach. The AUC increased compared to the best OoD method for each attack. Using the assumed criterion, the CW, the OnePixel, and the Square passed the test. The DeepFool and the PGD nearly passed (TNR at TPR 95% >= 45%), only the FGSM failed. The CW and the Square are well detected on nearly all OoD methods – only LOF_{Euc} performed good results – any others easily failed. In contrast, the PGD is still the most challenging task. There is no one leading the OoD method. There are some not apparent results here like the $\text{LOF\_D}_{\text{Euc}}$ which failed for most attacks. The CroW seems to be the best feature extraction for the tested task.

Table 3.22: Comparision of the OoD methods as the defense approach against the adversarial attacks with best picked feature extraction method.

Attack	Feature extractor	Method	DTACC	AUC	AUPR	TNR at TPR 95%
CW	CroW	LOF_{Cos}	89.75	94.44	93.68	73.40
	CroW	LOF_{Euc}	89.15	93.84	93.21	69.70
	CroW	LOF_D_{Cos}	89.60	92.86	89.12	71.80
	CroW	LOF_D_{Euc}	89.15	91.62	87.56	0.00
	CroW	Mahalanobis	89.30	93.70	92.69	52.90
	SCDA	MDistance $_{Cos}$	89.10	94.09	92.89	62.60
	CroW	MDistance $_{Cos}$	89.75	94.69	93.56	72.80
	CroW	$OSNN_{Cos}$	89.95	95.24	94.61	72.90
	CroW	$OSNN_{Euc}$	89.85	95.16	94.54	74.40
DeepFool	CroW	LOF_{Cos}	82.75	88.86	87.64	45.80
	CroW	LOF_{Euc}	82.30	88.14	86.90	40.70
	CroW	LOF_D_{Cos}	82.80	87.95	84.39	42.80
	CroW	LOF_D_{Euc}	82.50	86.91	82.96	0.00
	CroW	Mahalanobis	83.30	88.89	87.32	34.80
	CroW	MDistance $_{Cos}$	84.55	89.82	87.75	46.70
	SCDA	MDistance $_{Euc}$	82.60	88.16	86.42	40.40
	CroW	$OSNN_{Cos}$	84.70	90.52	89.40	44.00
	CroW	$OSNN_{Euc}$	84.85	90.38	89.20	45.30
FGSM	CroW	LOF_{Cos}	79.70	85.89	84.87	35.70
	CroW	LOF_{Euc}	79.50	86.20	85.32	35.00
	CroW	LOF_D_{Cos}	79.60	84.90	81.34	28.80
	CroW	LOF_D_{Euc}	79.55	84.90	81.04	0.00
	CroW	Mahalanobis	80.05	86.83	86.12	34.30
	CroW	MDistance $_{Cos}$	79.30	84.86	82.38	29.10
	CroW	MDistance $_{Euc}$	78.00	83.29	80.75	21.20
	CroW	$OSNN_{Cos}$	80.20	85.87	84.13	28.30
	CroW	$OSNN_{Euc}$	80.55	86.04	84.28	28.70
OnePixel	GMP	LOF_{Cos}	82.50	88.13	87.27	32.90
	GMP	LOF_{Euc}	82.15	87.77	86.09	28.90
	GMP	LOF_D_{Cos}	82.55	87.34	84.63	25.60
	GMP	LOF_D_{Euc}	81.95	86.97	83.84	23.50
	GMP	Mahalanobis	83.95	91.42	91.10	51.30
	CroW	MDistance $_{Cos}$	84.00	89.76	88.43	44.60
	CroW	MDistance $_{Euc}$	81.65	86.56	84.14	35.50
	CroW	$OSNN_{Cos}$	83.65	89.73	88.70	40.20

Table 3.22: Comparision of the OoD methods as the defense approach against the adversarial attacks with best picked feature extraction method.

Attack	Feature extractor	Method	DTACC	AUC	AUPR	TNR at TPR 95%
	CroW	$OSNN_{Euc}$	83.35	89.66	88.55	41.00
PGD	CroW	LOF_{Cos}	71.35	72.70	70.00	3.10
	CroW	LOF_{Euc}	87.60	92.65	92.12	45.30
	CroW	LOF_D_{Cos}	71.35	72.62	69.77	2.70
	CroW	LOF_D_{Euc}	87.55	90.82	86.92	0.00
	GAP_All	Mahalanobis	70.80	70.22	69.27	5.30
	CroW	$MDistance_{Cos}$	64.10	61.93	60.39	3.40
	CroW	$MDistance_{Euc}$	77.45	80.17	75.43	4.70
	GAP_All	$OSNN_{Cos}$	61.95	61.71	61.02	4.90
	CroW	$OSNN_{Euc}$	80.85	82.05	78.24	2.50
Square	CroW	LOF_{Cos}	91.85	95.66	94.74	78.50
	CroW	LOF_{Euc}	91.50	95.26	94.48	73.20
	CroW	LOF_D_{Cos}	92.00	94.11	90.39	79.40
	CroW	LOF_D_{Euc}	91.75	93.09	89.04	0.00
	CroW	Mahalanobis	92.05	95.10	94.14	58.10
	CroW	$MDistance_{Cos}$	91.90	96.40	95.54	82.40
	CroW	$MDistance_{Euc}$	90.85	94.43	92.98	64.30
	CroW	$OSNN_{Cos}$	92.10	96.49	95.89	80.00
	CroW	$OSNN_{Euc}$	91.90	96.46	95.92	81.60

Section Summary We tested the performance of adversarial attacks detection by using OoD methods. We tread the images attack as the unknown data. We generated the CW, DeepFool, FGSM, OnePixel, PGD, and Square attacks on the ResNet-101 model fitted on the CIFAR-10 dataset. The above attacks are based on different methodologies, which allowed for broad analysis. We confirmed that applying the OoD methods for close-set classifiers further increases the safety of the models against the attacks at no additional cost. However, to successfully improve it, we need to use other feature extraction methods than widespread GAP, often used in close-set classifier. Our experiments suggest using the CroW. There was no one leading OoD method. For most CW and the Square attacks, nearly all OoD methods worked well. The most problematic ones were the PGD and FGSM attacks.

Part III

Summary

4

Summary

4.1 Summary

One of our goals was to give practical recommendations on applying the Out-of-Distribution detection problem. Following the literature[4][158][217][59], the detection of OoD examples is an important component of trustworthy machine learning systems. However, there are no clear recommendations concerning adopting these approaches within many state-of-the-art methods.

We focused on analyzing the factors, which have the most significant impact on the performance of OoD. We showed the complexity of the problem for anyone who wanted to use it in practice. We showed the importance of decisions that are necessary to undertake and how they affect final results. First of all, there is no best OoD method, which was confirmed by our and others' research. For instance, a recent comprehensive study [243] showed that none of the SoTA methods consistently outperforms other methods on a set of 16 commonly used OoD benchmarks of In-distribution and Out-of-Distribution pairs of datasets. It follows that it is impossible to recommend one OoD approach as a universal solution. However, our research suggests three promising methods based on logits i.e., MaxLogits, parametric Mahalanobis, and density-based methods LOF or LOF_D. They all work based on different assumptions, suitable for different ID and OoD data pairs. We cannot suggest one of them - however, in our future work, we would like to test the ensemble approaches. Moreover, other decisions significantly influence OoD detection performance, among other: metrics (for example, the AUC does not reflect the results well because, in practice, we do not know the OoD data, so it is hard to set a final threshold based on it, or the TNR at TPR 95% which is much more unstable), the feature extraction method (see subsection 3.4.1), or the feature post-processing (see subsection 3.4.2).

Although the goal is to open some currently used close-set models, we postulated the research question that the better the close-set model, the better OoD detection efficiency. We achieved inconclusive results – see the details in subsection 3.2.4. The trend is visible for methods like Mahalanobis and OSNNs, but there is no direct dependency for other methods. We also noticed that the state of the model has a significant impact on OoD detection efficiency – see results presented in subsection 3.4.4. Therefore, looking only at the model's accuracy is insufficient. We showed that similar models with the same architecture and similar close-set accuracy could work entirely differently for the OoD benchmarks. Moreover, the evaluation metrics are very unstable during the training process, so it is hard to choose the final epoch for the best performance for open-set detection.

Many papers tried to make data more separable by using special techniques – for instance, ODIN (it uses the MaxSoftmax with temperature scaling and input pre-processing - see subsection 2.3.1.9) or Unified Framework (it uses the Mahalanobis with the input pre-processing and features ensemble - see subsection 2.3.1.10). We showed in subsection 3.3.2, that these techniques play a key role in the Unified Framework method. The results were the same or even slightly better when we changed the Mahalanobis into LOF. Many of these techniques do work well only for specific situations. For instance, the best results were usually achieved in our experiments, when input pre-processing was not applied. In section 3.3 we focused on Mahalanobis and EVM. The OoD problem generally, is the

extremely hard due the "curse of dimensionality". We showed the limitations of these methods, such as the instability of MVN used by Mahalanobis in high-dimensionality or margin distances in EVM, which often did not follow the Weibull distribution in many datasets working with CNNs.

We proposed to focus on building solid, robust feature representation with more attention. As we showed in section 3.4, the features quality significantly influences the execution of OoD methods. We showed and examined many approaches to affect the features – among others, changing the feature extraction methods, reducing the features, or choosing the proper augmentation strategy. Following papers [272] or [240](ours), the CNN networks can learn the spurious correlations. We suggest (although it requires further research) that this network property leads to some problems in OoD detection. For example, in subsection 3.4.5 we showed that there are globally easy or hard samples in popular datasets - independent of the CNN model or OoD methods. It suggests that the CNNs can not learn solid and robust features, so some OoD problems are nearly impossible to solve using some CNN models.

Moreover, we noticed the mismatch between the image and feature space. The hard subsets presented in subsection 3.4.5 contain examples of images that are distant in the image space but close in the features space. The one-pixel attack (see subsection 2.3.2.2.7) is the method that generates opposite samples, which are very close in the images space and distant in the feature space. As we explained, it can be the property of CNNs, which cannot build fully robust features. Based on these observations, we propose to clarify the benchmarks. Usually, scholars (i.e., [61][189][3]) use the "near" or "far" terms to describe the "difficulty" of OoD detection. The definition is ambiguous, however it is based on semantic(image) space representation. We suggest focusing on feature space representation to better describe the complexity of the problem.

Most research focused on low-scale data in the context of OoD detection problems. Usually, the images are in low resolutions with a small number of known classes. We also tested the OoD problem on a large-scale – based on the ImageNet datasets as In-distribution data with 1000 known classes and much more resolution than the popular CIFARs. See the subsections 3.2.4 and 3.4.1. Our research led to, among others, the following conclusions: the popular Mahalanobis OoD approach seemed to achieve slightly worse results compared to other approaches (when working on a large-scale). The methods based on logits did not seem to work, probably due to the number of known classes. Non-parametric LOF method achieved stable results.

We also noticed the problems with reproducibility and benchmarking OoD methods. As we showed in subsection 3.4.4, the OoD results are sensitive to model state. New state-of-the-art methods often outperform other methods only by a single percentage point. Meanwhile, the retrained CNN models can significantly differentiate the results from the OoD methods. As we showed in subsection 3.4.5, there are hard and easy examples in popular datasets, so when we pick only part of them for evaluation, the results can change significantly. Moreover, many OoD methods work well only with specific assumptions (see section 3.3) or for specific OoD distribution (see subsections 3.2.2 and 3.2.3). The augmentation strategy, which often is overlooked in the papers, is an important factor too (see subsection 3.4.3). The OoD methods can also work differently in large-scale problems, which is crucial to usability in modern solutions (see subsection 3.2.4). Therefore, we encourage broder discussion about stability and reproducibility.

In section 3.4.6, we focused on one of the major issues in safety and trustworthiness in computer vision, which is the defense against adversarial attacks. We assumed that treating the adversarial attacks as OoD data is possible, which we confirmed. This approach can be

easily adapted and protects from adversarial attacks. The idea is not new – however, we proposed a new concept: changing the GAP to another extraction technique. In practice, the attacks focus on fooling the classifier, which is usually based on GAP features, so changing the extraction strategy should increase OoD efficiency and does not require retraining the models.

4.2 Conclusions

Conclusions and contributions from our work can be presented in the following points:

1. We showed the limitation of the popular EVM algorithm (see section 3.3.3) - precisely, the EVM's theoretical assumptions, in some cases, might not be entirely fulfilled in the context of CNNs and some popular datasets. The margin distances often do not follow the Weibull distribution.

2. We showed that the popular methods based on MultiVariate Normal(MVN) distribution could be unstable in high dimensional data and lead to limitations, where near OoD samples can not be distinguished from known data (see section 3.3.1). Moreover, we tested different variants of Mahalanobis distance in benchmark datasets and showed that none of them can be treated as the universal approach.

3. We showed that the nonparametric, density-based LOF approach performs better than the approaches based on MVN or logits (in many cases). However, this method is still not commonly used as a benchmark method in the literature. LOF requires the calculation of the nearest neighbors, which can be computationally intensive compared to Mahalanobis or MaxSoftmax. The complexity of building the LOF model is $O(d * N^2)$, and OoD detection is $O(d * N)$, where dimensions are denoted as d and train size as N. However, the performance of this method can be increased (for a certain distance metric) to $O(d * NlogN)$(for model building) and $O(d * logN)$(for detection) by using the k-d tree (as implemented in scikit-learn[1]) or R*-tree (as implemented in the ELKI framework[2]).

4. We showed the efficiency of the OoD method based on large-scale dataset benchmarks (see section 3.2.4). We conclude that the OoD methods based on logits work poorly, and Mahalanobis is slightly worse compared to other methods on a large-scale. The LOF seems to be the most stable method.

5. We showed that the correctly chosen feature extraction strategy can improve the efficiency of OoD detection (see section 3.4.1. The literature does not use any other method than standard Global Average Pooling (GAP). However, various approaches can focus on different components of images. Changing feature obtaining strategy into GMP, CroW, or SCADA can boost the OoD detection performance. There are no clear guidelines on which strategy should be used. We recommend using a new hyperparameter describing that strategy. It should be chosen for specific pairs of ID and OoD. One of our future works is the ensemble method using a different feature extraction strategy.

6. We showed that reducing the size of feature vectors leads to severe efficiency decrease for many methods (see section 3.4.2). We used the technique known from the image

[1] `https://scikit-learn.org`
[2] `https://elki-project.github.io`

retrieval problems: the L2 normalization, the PCA with whitening, and again L2 normalization. The LOF-based methods seem to be the only method we recommend using together with reduction dimensionally.

7. We confirmed that applying the OoD methods for close-set classifiers further increases the safety of the networks against the attacks at no additional cost (see section 3.4.6). The adversarial examples focus their attack on the classifier, which is based on the GAP features. Therefore, we recommend choosing a different feature extraction strategy than this one.

8. We showed the problem with the instability and repeatability of the OoD detection methods (see section 3.4.4). We showed that the slightly different model state can drastically change the OoD method efficiency. The above problems are visible for changing weights over the epochs during the training process. We conclude that results presented in the literature should be considered with caution.

9. We showed that there was no clear connection between image and latent spaces. 3.4.5). We conclude that the common usage of near and far OoD examples definitions is inaccurate. Moreover, we suggest that the problem lies in training data, so no model can be robust enough to solve the problem completely.

10. We showed there is no best OoD approach and it depends on the tested ID and OoD pair. We plan to use the ensemble of different OoD methods in our future works.

11. We showed the importance of proper data augmentation techniques in OoD detection problems (see section 3.4.3). We suggest using the idea of mixing images, such as MixUp, which improves robustness, but above all, it is beneficial for OoD detection.

4.3 Future Works

This work cannot deal with all issues connected to the vast problem of OoD detection. We focused only on chosen topics, and further work is needed. In this section, we wanted to show our ideas for future works.

First of all, we researched the problem of OoD detection in computer vision classification tasks. However, much of the proposed research should be repeated for other data types, such as texts, music, or biomedical data and other subproblems in CV, such as object detecting or image segmentation. Moreover, we focused only on CNNs, but the ViTs models (see subsection 2.1.1.5) are gaining more and more popularity. There are already papers focusing on those types of models in the context of OoD - e.g.[61]. However, more research is needed on the robustness of ViT's features such as [181].

We have made some assumptions (see subsection 3.1.1) such as not to retrain the models and not to use external data. However, many popular solutions allow for them. The retraining of the model allows changing the model's architecture and adding additional loss functions during the fitting phase. Contrastive learning (see subsection 2.2.6) is one of the most popular approaches concerning those changes. Retraining transforms the model to be dedicated to OoD problems, making the problem easier to solve. The problem of OoD detection can be further simplified by showing part of the unknown data (see subsection 2.3.1.11.1). It is called outlier exposure. By using retraining and outlier exposure, the OoD systems can move the decision boundaries making the data more distinguishable. On the other hand, the models cannot (or achieve worse results) be used in the close-set classification task, so it usually requires two systems: for detecting outliers and close-set

classifiers. Nevertheless, such solutions exist[91][50][145][267][214][55]. We plan to repeat some of the tests for the above methods suspecting that the results will be consistent with ours.

We noticed that generating robust features is the key to successful OoD detection. The more robust features are, the better the achieved results are. We noticed the tendency of the OoD methods to be unstable and sensitive to hyperparameters and model states. Therefore, we want to test which regularization technique (or propose the new one) can positively impact the above problems in the context of OoD detection tasks. With a few exceptions, currently used methods focus only on features from the network's last layer, i.e. GAP. Some OoD data should be better separable based on earlier network layers. We showed some convincing results on this topic, but further research is necessary. We consider using ensemble methods with different feature extraction strategy and/or different OoD methods.

Although we showed a positive influence on the safety and trustworthiness of AI models when applying the OoD detection mechanism, there are still many other methods worth further research. For instance, many strategies focus only on particular problems like defense against adversarial examples, increasing the robustness of the models, or allowing continual learning. One of our further goals is to propose a comprehensive method that merges the OoD detection with other methods and to see how it affects the final results.

Bibliography

[1] M. Abbasi and C. Gagné, "Robustness to adversarial examples through an ensemble of specialists," *arXiv preprint arXiv:1702.06856*, 2017.

[2] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive science*, vol. 9, no. 1, pp. 147–169, 1985.

[3] F. Ahmed and A. Courville, "Detecting semantic anomalies," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 3154–3162.

[4] D. Amodei *et al.*, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[5] P. Arcaini *et al.*, "Dealing with robustness of convolutional neural networks for image classification.," in *AITest*, 2020, pp. 7–14.

[6] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[7] A. B. Arrieta *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020.

[8] R. Ayachi *et al.*, "Traffic signs detection for real-world application of an advanced driving assisting system using deep learning," *Neural Processing Letters*, vol. 51, no. 1, pp. 837–851, 2020.

[9] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[10] A. Babenko and V. Lempitsky, "Aggregating deep convolutional features for image retrieval," *arXiv preprint arXiv:1510.07493*, 2015.

[11] A. Babenko *et al.*, "Neural codes for image retrieval," in *European conference on computer vision*, Springer, 2014, pp. 584–599.

[12] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[13] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML workshop on unsupervised and transfer learning*, JMLR Workshop and Conference Proceedings, 2012, pp. 37–49.

[14] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv preprint arXiv:2003.05991*, 2020.

[15] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International Conference on Machine Learning*, PMLR, 2017, pp. 449–458.

[16] A. Bendale and T. E. Boult, "Towards open set deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1563–1572.

[17] L. Beyer *et al.*, "Are we done with imagenet?" *arXiv preprint arXiv:2006.07159*, 2020.

[18] D. Blalock *et al.*, "What is the state of neural network pruning?" *arXiv preprint arXiv:2003.03033*, 2020.

[19] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[20] F. Bodria *et al.*, "Benchmarking and survey of explanation methods for black box models," *arXiv preprint arXiv:2102.13076*, 2021.

[21] D. Bolya *et al.*, "Yolact: Real-time instance segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9157–9166.

[22] M. M. Breunig *et al.*, "Lof: Identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104.

[23] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[24] T. B. Brown *et al.*, "Adversarial patch," *arXiv preprint arXiv:1712.09665*, 2017.

[25] S. Bubeck, "Convex optimization: Algorithms and complexity," *arXiv preprint arXiv:1405.4980*, 2014.

[26] J. Buolamwini and T. Gebru, "Gender shades: Intersectional accuracy disparities in commercial gender classification," in *Conference on fairness, accountability and transparency*, PMLR, 2018, pp. 77–91.

[27] Y. Burda *et al.*, "Exploration by random network distillation," *arXiv preprint arXiv:1810.12894*, 2018.

[28] A. Buslaev *et al.*, "Albumentations: Fast and flexible image augmentations," *Information*, vol. 11, no. 2, p. 125, 2020.

[29] R. M. Byrne, "Counterfactuals in explainable artificial intelligence (xai): Evidence from human reasoning.," in *IJCAI*, 2019, pp. 6276–6282.

[30] Z. Cao *et al.*, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 1, pp. 172–186, 2019.

[31] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*, IEEE, 2017, pp. 39–57.

[32] A. Chattopadhay *et al.*, "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks," in *2018 IEEE winter conference on applications of computer vision (WACV)*, IEEE, 2018, pp. 839–847.

[33] S. Chaudhari *et al.*, "An attentive survey of attention models," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 12, no. 5, pp. 1–32, 2021.

[34] C. Chen *et al.*, "This looks like that: Deep learning for interpretable image recognition," *arXiv preprint arXiv:1806.10574*, 2018.

[35] T. Chen *et al.*, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*, PMLR, 2020, pp. 1597–1607.

[36] W. Chen *et al.*, "Deep image retrieval: A survey," *arXiv preprint arXiv:2101.11282*, 2021.

[37] W. Chen *et al.*, "Beyond triplet loss: A deep quadruplet network for person re-identification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 403–412.

[38] Y. Chen *et al.*, "Cascaded pyramid network for multi-person pose estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7103–7112.

[39] Y. Chen *et al.*, "Dual path networks," in *Advances in neural information processing systems*, 2017, pp. 4467–4475.

[40] Z. Chen *et al.*, "Gated square-root pooling for image instance retrieval," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, IEEE, 2018, pp. 1982–1986.

[41] H.-F. Cheng *et al.*, "Explaining decision-making algorithms through ui: Strategies to help non-expert stakeholders," in *Proceedings of the 2019 chi conference on human factors in computing systems*, 2019, pp. 1–12.

[42] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[43] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, IEEE, vol. 1, 2005, pp. 539–546.

[44] F. Croce *et al.*, "Robustbench: A standardized adversarial robustness benchmark," *arXiv preprint:2010.09670*, 2020.

[45] E. D. Cubuk *et al.*, "Autoaugment: Learning augmentation strategies from data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 113–123.

[46] W. Dabney *et al.*, "Implicit quantile networks for distributional reinforcement learning," in *International conference on machine learning*, PMLR, 2018, pp. 1096–1105.

[47] Z. Dai *et al.*, "Coatnet: Marrying convolution and attention for all data sizes," *arXiv preprint arXiv:2106.04803*, 2021.

[48] J. Deng *et al.*, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[49] J. Devlin *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[50] T. DeVries and G. W. Taylor, "Learning confidence for out-of-distribution detection in neural networks," *arXiv preprint arXiv:1802.04865*, 2018.

[51] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," *arXiv preprint arXiv:1605.09782*, 2016.

[52] J. Donahue and K. Simonyan, "Large scale adversarial representation learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 10 542–10 552.

[53] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[54] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[55] X. Du *et al.*, "Vos: Learning what you don't know by virtual outlier synthesis," *arXiv preprint arXiv:2202.01197*, 2022.

[56] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization.," *Journal of machine learning research*, vol. 12, no. 7, 2011.

[57] V. Dumoulin *et al.*, "Adversarially learned inference," *arXiv preprint arXiv:1606.00704*, 2016.

[58] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, "A study of the effect of jpg compression on adversarial images," *arXiv preprint arXiv:1608.00853*, 2016.

[59] K. Eykholt *et al.*, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1625–1634.

[60] W. Feng *et al.*, "Audio visual speech recognition with multimodal recurrent neural networks," in *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2017, pp. 681–688.

[61] S. Fort, J. Ren, and B. Lakshminarayanan, "Exploring the limits of out-of-distribution detection," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[62] J. Frankle, D. J. Schwab, and A. S. Morcos, "Training batchnorm and only batchnorm: On the expressive power of random features in cnns," *arXiv preprint arXiv:2003.00152*, 2020.

[63] M. Frid-Adar *et al.*, "Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification," *Neurocomputing*, vol. 321, pp. 321–331, 2018.

[64] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*, PMLR, 2018, pp. 1587–1596.

[65] A. Galdran *et al.*, "Data-driven color augmentation techniques for deep skin image analysis," *arXiv preprint:1703.03702*, 2017.

[66] D. Gaur, J. Folz, and A. Dengel, "Training deep neural networks without batch normalization," *arXiv preprint arXiv:2008.07970*, 2020.

[67] C. Geng, S.-j. Huang, and S. Chen, "Recent advances in open set recognition: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[68] Z. Geng *et al.*, "Bottom-up human pose estimation via disentangled keypoint regression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 676–14 686.

[69] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[70] R. Girshick *et al.*, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[71] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[72] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[73] I. Goodfellow *et al.*, *Deep learning*, 2. MIT press Cambridge, 2016, vol. 1.

[74] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[75] A. Gordo *et al.*, "Deep image retrieval: Learning global representations for image search," in *European conference on computer vision*, Springer, 2016, pp. 241–257.

[76] S. Gowal *et al.*, "Uncovering the limits of adversarial training against norm-bounded adversarial examples," *arXiv preprint:2010.03593*, 2020.

[77] S. Grigorescu *et al.*, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[78] J.-B. Grill *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 271–21 284, 2020.

[79] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *arXiv preprint arXiv:1412.5068*, 2014.

[80] Y. Gu, C. Li, and Y.-G. Jiang, "Towards optimal cnn descriptors for large-scale image retrieval," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 1768–1776.

[81] K. S. Gurumoorthy *et al.*, "Efficient data representation by selecting prototypes with importance weights," in *2019 IEEE International Conference on Data Mining (ICDM)*, IEEE, 2019, pp. 260–269.

[82] T. Haarnoja *et al.*, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, PMLR, 2018, pp. 1861–1870.

[83] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 1025–1035.

[84] D. Han, J. Kim, and J. Kim, "Deep pyramidal residual networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5927–5935.

[85] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," in *2015 aaai fall symposium series*, 2015.

[86] K. He *et al.*, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[87] ——, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[88] K. He *et al.*, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[89] W. He *et al.*, "Adversarial example defense: Ensembles of weak defenses are not strong," in *11th {USENIX} workshop on offensive technologies ({WOOT} 17)*, 2017.

[90] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *arXiv preprint arXiv:1610.02136*, 2016.

[91] D. Hendrycks, M. Mazeika, and T. Dietterich, "Deep anomaly detection with outlier exposure," *arXiv preprint arXiv:1812.04606*, 2018.

[92] D. Hendrycks *et al.*, "Natural adversarial examples," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 262–15 271.

[93] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[94] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[95] R. Houthooft *et al.*, "Vime: Variational information maximizing exploration," *arXiv preprint arXiv:1605.09674*, 2016.

[96] A. Howard *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1314–1324.

[97] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[98] Y.-C. Hsu *et al.*, "Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 951–10 960.

[99] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[100] G. Huang *et al.*, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[101] X. Huang *et al.*, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Computer Science Review*, vol. 37, p. 100 270, 2020.

[102] Y. Huang *et al.*, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," in *Advances in neural information processing systems*, 2019, pp. 103–112.

[103] R. Imbriaco, C. Sebastian, E. Bondarev, *et al.*, "Aggregated deep local features for remote sensing image retrieval," *Remote Sensing*, vol. 11, no. 5, p. 493, 2019.

[104] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, PMLR, 2015, pp. 448–456.

[105] P. Isola *et al.*, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[106] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[107] M. Jaderberg *et al.*, "Reinforcement learning with unsupervised auxiliary tasks," *arXiv preprint arXiv:1611.05397*, 2016.

[108] H. Jégou and O. Chum, "Negative evidences and co-occurences in image retrieval: The benefit of pca and whitening," in *European conference on computer vision*, Springer, 2012, pp. 774–787.

[109] S. Jégou *et al.*, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 11–19.

[110] D. Jha *et al.*, "Doubleu-net: A deep convolutional neural network for medical image segmentation," in *2020 IEEE 33rd International symposium on computer-based medical systems (CBMS)*, IEEE, 2020, pp. 558–564.

[111] H. Jun *et al.*, "Combination of multiple global descriptors for image retrieval," *arXiv preprint arXiv:1903.10663*, 2019.

[112] Y. Kalantidis, C. Mellina, and S. Osindero, "Cross-dimensional weighting for aggregated deep convolutional features," in *European conference on computer vision*, Springer, 2016, pp. 685–701.

[113] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.

[114] T. Karras *et al.*, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[115] S. Khan *et al.*, "Transformers in vision: A survey," *arXiv preprint arXiv:2101.01169*, 2021.

[116] B. Kim, R. Khanna, and O. O. Koyejo, "Examples are not enough, learn to criticize! criticism for interpretability," *Advances in neural information processing systems*, vol. 29, 2016.

[117] B. Kim *et al.*, "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)," in *International conference on machine learning*, PMLR, 2018, pp. 2668–2677.

[118] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[119] G. Koch, R. Zemel, R. Salakhutdinov, *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, Lille, vol. 2, 2015.

[120] A. Köchling and M. C. Wehner, "Discriminated by an algorithm: A systematic review of discrimination and fairness by algorithmic decision-making in the context of hr recruitment and hr development," *Business Research*, pp. 1–54, 2020.

[121] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[122] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE journal*, vol. 37, no. 2, pp. 233–243, 1991.

[123] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[124] A. Kurakin, I. Goodfellow, S. Bengio, *et al.*, *Adversarial examples in the physical world*, 2016.

[125] T. Kurbiel and S. Khaleghian, "Training of deep neural networks based on distance measures using rmsprop," *arXiv preprint arXiv:1708.01911*, 2017.

[126] T. Kurutach *et al.*, "Model-ensemble trust-region policy optimization," *arXiv preprint arXiv:1802.10592*, 2018.

[127] Z. Lan *et al.*, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[128] A. Latif *et al.*, "Content-based image retrieval and feature extraction: A comprehensive review," *Mathematical Problems in Engineering*, vol. 2019, 2019.

[129] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[130] Y. LeCun *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[131] C.-Y. Lee, P. W. Gallagher, and Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree," in *Artificial intelligence and statistics*, 2016, pp. 464–472.

[132] K. Lee *et al.*, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," *Advances in neural information processing systems*, vol. 31, 2018.

[133] K. Lenc and A. Vedaldi, "Understanding image representations by measuring their equivariance and equivalence," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 991–999.

[134] M. Lewis *et al.*, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[135] Y. Li *et al.*, "Ms-rmac: Multiscale regional maximum activation of convolutions for image retrieval," *IEEE Signal Processing Letters*, vol. 24, no. 5, pp. 609–613, 2017.

[136] Y. Li *et al.*, "Data-driven neuron allocation for scale aggregation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 526–11 534.

[137] Z. Li and S. Arora, "An exponential learning rate schedule for deep learning," *arXiv preprint arXiv:1910.07454*, 2019.

[138] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," *arXiv preprint arXiv:1706.02690*, 2017.

[139] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[140] T.-Y. Lin *et al.*, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[141] Z. C. Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.," *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[142] C. Liu *et al.*, "Progressive neural architecture search," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 19–34.

[143] Q. Liu, M. J. Kusner, and P. Blunsom, "A survey on contextual embeddings," *arXiv preprint arXiv:2003.07278*, 2020.

[144] S. Liu *et al.*, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8759–8768.

[145] W. Liu *et al.*, "Energy-based out-of-distribution detection," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 464–21 475, 2020.

[146] Y. Liu *et al.*, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[147] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," *arXiv preprint arXiv:2103.14030*, 2021.

[148] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[149] J. Lu, T. Issaranon, and D. Forsyth, "Safetynet: Detecting and rejecting adversarial examples robustly," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 446–454.

[150] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, Citeseer, vol. 30, 2013, p. 3.

[151] H. Maciejewski, T. Walkowiak, and K. Szyc, "Out-of-distribution detection in high-dimensional data using mahalanobis distance - critical analysis," in *International Conference on Computational Science*, https://www.iccs-meeting.org/iccs2022/ https://link.springer.com/conference/iccs-computsci/, Springer, 2022.

[152] A. Madry *et al.*, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[153] D. Mahajan *et al.*, "Exploring the limits of weakly supervised pretraining," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 181–196.

[154] P. C. Mahalanobis, "On the generalized distance in statistics," National Institute of Science of India, 1936.

[155] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[156] P. R. Mendes Júnior *et al.*, "Nearest neighbors distance ratio open-set classifier," *Machine Learning*, vol. 106, no. 3, pp. 359–386, 2017.

[157] J. H. Metzen *et al.*, "On detecting adversarial perturbations," *arXiv preprint arXiv:1702.04267*, 2017.

[158] D. Miller *et al.*, "Dropout sampling for robust object detection in open-set conditions," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 3243–3249.

[159] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[160] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, PMLR, 2016, pp. 1928–1937.

[161] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[162] A. Nagabandi *et al.*, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 7559–7566.

[163] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.

[164] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European conference on computer vision*, Springer, 2016, pp. 483–499.

[165] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 427–436.

[166] H. Noh *et al.*, "Large-scale image retrieval with attentive deep local features," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3456–3465.

[167] N. Papernot *et al.*, "The limitations of deep learning in adversarial settings," in *2016 IEEE European symposium on security and privacy (EuroS&P)*, IEEE, 2016, pp. 372–387.

[168] N. Papernot *et al.*, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 2017, pp. 506–519.

[169] T. Park *et al.*, "Semantic image synthesis with spatially-adaptive normalization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2337–2346.

[170] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017.

[171] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[172] M. Peters *et al.*, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 2227–2237.

[173] H. Pham *et al.*, "Meta pseudo labels," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 557–11 568.

[174] L. Pishchulin *et al.*, "Deepcut: Joint subset partition and labeling for multi person pose estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4929–4937.

[175] O. Poursaeed *et al.*, "Generative adversarial perturbations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4422–4431.

[176] T. M. Quan, D. G. Hildebrand, and W.-K. Jeong, "Fusionnet: A deep fully residual convolutional neural network for image segmentation in connectomics," *arXiv preprint arXiv:1612.05360*, 2016.

[177] F. Radenović, G. Tolias, and O. Chum, "Fine-tuning cnn image retrieval with no human annotation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 7, pp. 1655–1668, 2018.

[178] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[179] A. Radford *et al.*, "Improving language understanding by generative pre-training," 2018.

[180] A. Radford *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[181] M. Raghu *et al.*, "Do vision transformers see like convolutional neural networks?" *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[182] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[183] P. Ramachandran *et al.*, "Stand-alone self-attention in vision models," *arXiv preprint arXiv:1906.05909*, 2019.

[184] E. Real *et al.*, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, 2019, pp. 4780–4789.

[185] S.-A. Rebuffi *et al.*, "Data augmentation can improve robustness," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[186] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[187] ——, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[188] J. Redmon *et al.*, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[189] J. Ren *et al.*, "A simple fix to mahalanobis distance for improving near-ood detection," *arXiv preprint arXiv:2106.09022*, 2021.

[190] S. Ren *et al.*, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, pp. 91–99, 2015.

[191] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[192] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241.

[193] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[194] R. Rothe, M. Guillaumin, and L. Van Gool, "Non-maximum suppression for object detection by passing messages between windows," in *Asian conference on computer vision*, Springer, 2014, pp. 290–306.

[195] E. M. Rudd *et al.*, "The extreme value machine," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 762–768, 2017.

[196] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[197] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[198] C. Rudin, C. Wang, and B. Coker, "The age of secrecy and unfairness in recidivism prediction," *arXiv preprint arXiv:1811.00731*, 2018.

[199] C. Rudin *et al.*, "Interpretable machine learning: Fundamental principles and 10 grand challenges," *arXiv preprint arXiv:2103.11251*, 2021.

[200] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[201] A. A. Rusu *et al.*, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[202] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *arXiv preprint arXiv:1710.09829*, 2017.

[203] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," *Advances in neural information processing systems*, vol. 29, pp. 901–909, 2016.

[204] W. Samek and K.-R. Müller, "Towards explainable artificial intelligence," in *Explainable AI: interpreting, explaining and visualizing deep learning*, Springer, 2019, pp. 5–22.

[205] J. Sánchez and F. Perronnin, "High-dimensional signature compression for large-scale image classification," in *CVPR 2011*, IEEE, 2011, pp. 1665–1672.

[206] M. Sandler *et al.*, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[207] A. Santoro *et al.*, "Relational recurrent neural networks," *arXiv preprint arXiv:1806.01822*, 2018.

[208] F. Scarselli *et al.*, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[209] W. J. Scheirer *et al.*, "Meta-recognition: The theory and practice of recognition score analysis," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 8, pp. 1689–1695, 2011.

[210] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[211] J. Schulman *et al.*, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[212] J. Schulman *et al.*, "Trust region policy optimization," in *International conference on machine learning*, PMLR, 2015, pp. 1889–1897.

[213] J. Schulman *et al.*, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[214] V. Sehwag, M. Chiang, and P. Mittal, "Ssd: A unified framework for self-supervised outlier detection," *arXiv preprint arXiv:2103.12051*, 2021.

[215] R. R. Selvaraju *et al.*, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[216] A. Serban, E. Poll, and J. Visser, "Adversarial examples on object recognition: A comprehensive survey," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–38, 2020.

[217] M. Sharif *et al.*, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the 2016 acm sigsac conference on computer and communications security*, 2016, pp. 1528–1540.

[218] J. Shijie *et al.*, "Research on data augmentation for image classification based on convolution neural networks," in *2017 Chinese automation congress (CAC)*, IEEE, 2017, pp. 4165–4170.

[219] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[220] D. Silver *et al.*, "Deterministic policy gradient algorithms," in *International conference on machine learning*, PMLR, 2014, pp. 387–395.

[221] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[222] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[223] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," *Advances in neural information processing systems*, vol. 29, 2016.

[224] S. Son, S. Nah, and K. M. Lee, "Clustering convolutional kernels to compress deep neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 216–232.

[225] Y. Song *et al.*, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," *arXiv preprint arXiv:1710.10766*, 2017.

[226] D. Soydaner, "A comparison of optimization algorithms for deep learning," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 34, no. 13, p. 2 052 013, 2020.

[227] J. T. Springenberg *et al.*, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[228] A. Srinivas *et al.*, "Bottleneck transformers for visual recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 519–16 529.

[229] N. Srivastava *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[230] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[231] C. Sun *et al.*, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 843–852.

[232] T. Szandała, "Review and comparison of commonly used activation functions for deep neural networks," in *Bio-inspired Neurocomputing*, Springer, 2020, pp. 203–224.

[233] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[234] C. Szegedy *et al.*, "Inception-v4, inception-resnet and the impact of residual connections on learning," *arXiv preprint arXiv:1602.07261*, 2016.

[235] C. Szegedy *et al.*, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[236] K. Szyc, "Comparison of different deep-learning methods for image classification," in *2018 IEEE 22nd International Conference on Intelligent Engineering Systems (INES)*, IEEE, 2018, pp. 000 341–000 346.

[237] ——, "An impact of different images color spaces on the efficiency of convolutional neural networks," in *International Conference on Dependability and Complex Systems*, Springer, 2019, pp. 506–514.

[238] ——, "Generalized convolution: Replacing the classic convolution operation with the sub-network," in *International Conference on Dependability and Complex Systems*, Springer, 2021, pp. 437–446.

[239] ——, "An impact of data augmentation techniques on the robustness of cnns," in *International Conference on Dependability and Complex Systems*, http://depcos.pwr.edu.pl/ https://link.springer.com/conference/depcos/, Springer, 2022.

[240] K. Szyc, T. Walkowiak, and H. Maciejewski, "Checking robustness of representations learned by deep neural networks," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2021, pp. 399–414.

[241] ——, "Why out-of-distribution detection in cnns does not like mahalanobis–and what to use instead," *arXiv preprint arXiv:2110.07043*, 2021.

[242] J. Tack *et al.*, "Csi: Novelty detection via contrastive learning on distributionally shifted instances," *Advances in neural information processing systems*, vol. 33, pp. 11 839–11 852, 2020.

[243] F. Tajwar *et al.*, "No true state-of-the-art? ood detection methods are inconsistent across datasets," *arXiv preprint arXiv:2109.05554*, 2021.

[244] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[245] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 781–10 790.

[246] S. Thys, W. Van Ranst, and T. Goedemé, "Fooling automated surveillance cameras: Adversarial patches to attack person detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[247] G. Tolias, R. Sicre, and H. Jégou, "Particular object retrieval with integral max-pooling of cnn activations," *arXiv preprint arXiv:1511.05879*, 2015.

[248] I. Tolstikhin *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *arXiv preprint arXiv:2105.01601*, 2021.

[249] H. Touvron *et al.*, "Fixing the train-test resolution discrepancy," in *Advances in Neural Information Processing Systems*, 2019, pp. 8252–8262.

[250] ——, "Fixing the train-test resolution discrepancy: Fixefficientnet," *arXiv preprint arXiv:2003.08237*, 2020.

[251] H. Touvron *et al.*, "Training data-efficient image transformers & distillation through attention," in *International Conference on Machine Learning*, PMLR, 2021, pp. 10 347–10 357.

[252] O. Tursun *et al.*, "Enhancing feature invariance with learned image transformations for image retrieval," *arXiv e-prints*, arXiv–2002, 2020.

[253] J. R. Uijlings *et al.*, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[254] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[255] A. Vaswani *et al.*, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[256] A. S. Vezhnevets *et al.*, "Feudal networks for hierarchical reinforcement learning," in *International Conference on Machine Learning*, PMLR, 2017, pp. 3540–3549.

[257] O. Vinyals *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[258] T. Walkowiak, K. Szyc, and H. Maciejewski, "On validity of extreme value theory-based parametric models for out-of-distribution detection," in *International Conference on Computational Science*, Springer, 2021, pp. 142–155.

[259] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Scaled-yolov4: Scaling cross stage partial network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13 029–13 038.

[260] H. Wang *et al.*, "Score-cam: Score-weighted visual explanations for convolutional neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 24–25.

[261] J. Wang *et al.*, "Deep metric learning with angular loss," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2593–2601.

[262] J. Wang *et al.*, "Learning fine-grained image similarity with deep ranking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1386–1393.

[263] Y. Wang *et al.*, "Improving adversarial robustness requires revisiting misclassified examples," in *International Conference on Learning Representations*, 2019.

[264] Z. Wang *et al.*, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*, PMLR, 2016, pp. 1995–2003.

[265] X.-S. Wei *et al.*, "Selective convolutional descriptor aggregation for fine-grained image retrieval," *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2868–2881, 2017.

[266] A. Weller, "Transparency: Motivations and challenges," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, Springer, 2019, pp. 23–40.

[267] J. Winkens *et al.*, "Contrastive training for improved out-of-distribution detection," *arXiv preprint arXiv:2007.05566*, 2020.

[268] H. Wu *et al.*, "Cvt: Introducing convolutions to vision transformers," *arXiv preprint arXiv:2103.15808*, 2021.

[269] Y. Wu and K. He, "Group normalization," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[270] Z. Wu *et al.*, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[271] C. Xiao *et al.*, "Generating adversarial examples with adversarial networks," *arXiv preprint arXiv:1801.02610*, 2018.

[272] K. Xiao *et al.*, "Noise or signal: The role of image backgrounds in object recognition," *arXiv preprint arXiv:2006.09994*, 2020.

[273] Q. Xie *et al.*, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 687–10 698.

[274] S. Xie *et al.*, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[275] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *arXiv preprint arXiv:1704.01155*, 2017.

[276] Y.-Y. Yang *et al.*, "A closer look at accuracy vs. robustness," *Advances in neural information processing systems*, vol. 33, pp. 8588–8601, 2020.

[277] Z. Yang *et al.*, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[278] X. Yuan *et al.*, "Adversarial examples: Attacks and defenses for deep learning," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 9, pp. 2805–2824, 2019.

[279] S. Yun *et al.*, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6023–6032.

[280] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[281] J. Zbontar *et al.*, "Barlow twins: Self-supervised learning via redundancy reduction," in *International Conference on Machine Learning*, PMLR, 2021, pp. 12 310–12 320.

[282] M. D. Zeiler, "Adadelta: An adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[283] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, Springer, 2014, pp. 818–833.

[284] H. Zhang *et al.*, "Mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[285] X. Zhang *et al.*, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848–6856.

[286] X. Zhang *et al.*, "Polynet: A pursuit of structural diversity in very deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 718–726.

[287] S. Zhao *et al.*, "Towards better accuracy-efficiency trade-offs: Divide and co-training," *arXiv preprint arXiv:2011.14660*, 2020.

[288] S. Zheng *et al.*, "Improving the robustness of deep neural networks via stability training," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 4480–4488.

[289] B. Zhou *et al.*, "Places: A 10 million image database for scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1452–1464, 2017.

[290] P. Zhou *et al.*, "Towards theoretically understanding why sgd generalizes better than adam in deep learning," *arXiv preprint arXiv:2010.05627*, 2020.

[291] J.-Y. Zhu *et al.*, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[292] Y. Zhu and S. Newsam, "Densenet for dense flow," in *2017 IEEE international conference on image processing (ICIP)*, IEEE, 2017, pp. 790–794.

[293] B. Zoph *et al.*, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

List of Figures

List of Tables