DEBIUTY STUDENCKIE

2023

INFORMATYKA W BIZNESIE

pod redakcją
Heleny Dudycz

Wydawnictwo Uniwersytetu Ekonomicznego we Wrocławiu
Wrocław 2023

Recenzja naukowa
Marcin Hernes

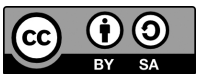
Redakcja wydawnicza
Małgorzata Tadrzak-Mazurek

Korekta
Aleksandra Śliwka

Skład i łamanie
Małgorzata Myszkowska

Projekt okładki
Beata Dębska

Na okładce wykorzystano zdjęcie z zasobów Adobe Stock



ISBN 978-83-67400-80-0 (wersja papierowa)
ISBN 978-83-67400-81-7 (wersja elektroniczna)

DOI: 10.15611/2023.81.7

Druk i oprawa: TOTEM

Bartosz Bicki

e-mail: Bartosz.bicki@gmail.com
ORCID: 0009-0003-6643-0249

Uniwersytet Ekonomiczny we Wrocławiu

Porównanie działania sieci neuronowych w Pythonie oraz C#

DOI: 10.15611/2023.81.7.02
JEL Classification: Y80

Streszczenie: Niniejszy artykuł zawiera wyniki przeprowadzonego porównania dwóch rozwiniętych języków programowania wysokiego poziomu, tj. Pythona oraz C#. Zostały one wykorzystane do stworzenia sieci neuronowej o określonej architekturze zastosowanej do problemu klasyfikacji cyfr. Celem badawczym było sprawdzenie, która z wymienionych technologii jest optymalna w całym procesie implementacji gotowego rozwiązania. W badaniu wykorzystano zbiór danych MNIST, a także zaprojektowano działającą wielowarstwową sieć neuronową. Kryteriami porównawczymi były statystyki tworzone podczas działania sieci. Na podstawie wyników przeprowadzonego badania stwierdzono, że optymalnym językiem programowania w obszarze uczenia maszynowego jest Python.

Słowa kluczowe: sieci neuronowe, klasyfikacja, Python, C#

1. Wstęp

Wykorzystanie algorytmów sieci neuronowych umożliwia sukcesywny rozwój sztucznej inteligencji. Jest to szczególnie widoczne w aktualnie dostępnych rozwiązaniach technologicznych, które oprócz zaprogramowanego działania są w stanie zrozumieć także sens przetwarzanego problemu badawczego. Ze względu na popularność tego obszaru uczenia maszynowego oraz rosnące możliwości wiedza na temat tworzenia algorytmów jest kluczową umiejętnością potrzebną przyszłym specjalistom IT.

Koncepcja sztucznych sieci neuronowych jest ideą, która miała swoje początki już w latach 40. ubiegłego wieku. Nie dostrzeżono wtedy jednak możliwości, które niosła ze sobą ta technologia. Ograniczone zasoby danych oraz niewydajne jednostki obliczeniowe wpłynęły na długi zastój rozwoju algorytmów sztucznych sieci neuronowych, co znacząco zmieniło się w ostatnich latach.

Celem przeprowadzonego badania było sprawdzenie, który z rozwiniętych języków programowania wysokiego poziomu jest optymalny w procesie implementacji sieci neuronowej wykorzystanej do klasyfikacji cyfr. Następnie porównano ze sobą statystyki tworzone podczas działania algorytmu oraz finalne wyniki obu zaimplementowanych sieci neuronowych. Zastosowaną metodą badawczą był eksperyment polegający na porównaniu działania tej samej architektury sieci zaimplemen-

towanej za pomocą dwóch odmiennych technologii. Wyniki badania mogą pomóc w wyborze języka programowania w projektach sieci neuronowych.

Wybrano dwa odmienne ze względu na charakterystykę języki programowania. Pierwszym był język Python, który jest technologią powszechnie stosowaną w zagadnieniach Data Science. Posiada on także głęboko rozwinięte narzędzia wykorzystywane w procesie tworzenia algorytmów. Został on porównany z językiem C# służącym do implementacji aplikacji w środowisku .NET. C# to język zorientowany obiektowo oraz silnie zorientowany na składnik. Obie technologie różnią się zarówno składnią kodu źródłowego, jak i sposobem jego przetwarzania przez maszynę.

Wybór technologii wykorzystanych do stworzenia rozwiązania jest istotną częścią cyklu życia oprogramowania. Określone czynniki wpływają na wszystkie etapy planowania, później wdrożenia oraz możliwość utrzymania i rozwoju. W celu porównania odmiennych języków programowania, zaimplementowano dwie sieci neuronowe, które posiadają taką samą architekturę. Aby umożliwić ten proces, wykorzystano cechy wspólne obu algorytmów powstałe podczas działania.

Niniejszy artykuł ma następującą strukturę: poniżej omówiono rozwój sieci neuronowych, następnie opisano założenia realizacji badania, tzn. zastosowaną procedurę badawczą, zaproponowane kryteria porównania języków – Pythona z C# – oraz użyte dane i opracowany model sieci neuronowej; w kolejnym punkcie omówiono uzyskane wyniki z przeprowadzonych eksperymentów oraz przedstawiono wnioski, materiał zakończono zaś podsumowaniem.

2. Rozwój sieci neuronowych

Pierwsza sieć neuronowa powstała w 1943 roku (McCulloch i Pitts, 1943). Opierała się na badaniach dotyczących komórek neuronowych połączonych synapsami. Powstały algorytm rozwiązywał tylko proste zadania funkcji logicznych. Zapoczątkowało to jednak nowy kierunek przeprowadzania obliczeń. Na podstawie tego modelu, piętnaście lat później Frank Rosenblatt zaproponował nowy algorytm, który został nazwany perceptronem (Rosenblatt, 1958). Ta koncepcja nie spotkała się z zainteresowaniem grona naukowców, a próba udowodnienia braku możliwości rozwiązania problemu nieliniowego XOR (Minsky i Papert, 1969) spowodowała długi przestój w rozwoju badań. Sytuacja uległa zmianie, gdy zaproponowano istotne zmiany w sposobie uczenia sieci (Rumelhart i in., 1986), którym był algorytm propagacji wstecznej.

Pomimo rewolucyjnych odkryć prace nad sieciami neuronowymi ustabilizowały się (Macukow, 2020). Spowodowane było to ograniczonym dostępem do danych oraz niewystarczającą mocą obliczeniową ówczesnych komputerów. Sytuacja zmieniała się jednak wraz z rozwojem technologii oraz opracowywaniem bardziej złożonych modeli. Obecnie można zauważyć duży wpływ obszaru sieci neuronowych na aktualny rozwój zagadnień dotyczących sztucznej inteligencji. Jest to temat zarówno kontrowersyjny, jak i budzący nadzieje na rozwiązanie nierozwiązywalnych dotych-

czas problemów. Wraz z przyrostem mocy obliczeniowych oraz rozwojem wszechobecnego Internetu dającego dostęp do nielimitowanych danych technologia ta rozwinęła się na tyle, że jest teraz w stanie nie tylko zrozumieć sens zadawanych pytań, ale również dawać odpowiedzi w czasie rzeczywistym (Ruby, 2023). Znaczenie tej koncepcji należy również zauważyć w dostępności narzędzi poświęconych dziedzinie uczenia maszynowego, a także samych sieci neuronowych. Dobrym przykładem jest wykorzystanie języka Python w bibliotece Keras (Chollet, 2017), która jest sukcesywnie rozwijana i optymalizowana. Warto rozważyć, jak jej dalszy rozwój wpłynie na ludzkość i technologie. Oprócz możliwych udoskonaleń w różnych dziedzinach nauki (Zhang i Lu, 2021), sztuczna inteligencja będzie prawdopodobnie w stanie zastąpić wiele stanowisk, pracując całą dobę z tą samą wydajnością. Należy zatem zadbać, aby proces ten nie był destabilizujący dla społeczeństwa.

3. Założenia realizacji badania

3.1. Zastosowana procedura badawcza

Motywem, który stał się inspiracją do przeprowadzenia niniejszego badania, jest aktualny rozwój technologii uczenia maszynowego, a także mnogość jej zastosowań. Dostępność przydatnych danych do nauczania algorytmów, a także moce obliczeniowe powodują obecnie ogromne możliwości poznawcze sieci neuronowych. W niedalekiej przyszłości algorytmy te będą wykonywać większość codziennych obowiązków obecnych specjalistów. Można zatem wnioskować, że znajomość rozwiązań uczenia maszynowego oraz ich implementacja stanie się wysoce pożądaną umiejętnością każdego eksperta.

Równie istotnym elementem badania był dobór języków programowania. Python i C# to dwie różne technologie wykorzystywane często do podejmowania problemów o znacząco innych charakterystykach. Niemniej jednak obie dają możliwość tworzenia i nauczania sieci neuronowych w sposób przystępny, wykorzystując do tego dedykowane biblioteki.

Docelowo badanie miało wykazać, który język programowania umożliwi optymalny sposób rozwiązania tego samego problemu, jednocześnie zachowując wysoki współczynnik statystyk działania. Wnioski opierają się na całym procesie tworzenia rozwiązania oraz finalnych wynikach zaimplementowanego algorytmu. Dla obu języków całość przebiegała w ten sam, ustalony sposób.

Badanie podzielono na pięć następujących po sobie etapów, z których wyciągano osobne wnioski. Wyróżnione etapy to:

1. Poszukiwanie dokumentacji oraz informacji dotyczących implementacji
   Dostępność informacji na temat danej technologii oraz jej ilość wpływa na popularność przekładającą się na liczbę zastosowań. Wraz ze wzrostem tej liczby, rośnie również elastyczność rozwiązania. Wiodącą w świecie IT umiejętnością

jest odnajdowanie informacji potrzebnych do zrozumienia problemu oraz późniejszej implementacji. Istotną zaletą wielu technologii jest ich dokumentacja i zasoby informacyjne. Realizacja tego etapu wynika również z charakterystyki obu języków programowania: Python to język powszechnie stosowany w obszarze *Data Science* i *Machine Learning.* Jest to dostrzegalne w ilości materiałów dostępnych w literaturze oraz w Internecie. Język C# jest natomiast stosowany częściej do tworzenia aplikacji desktopowych i systemów w środowisku .NET. W celu porównania sprawdzono dostęp do dokumentacji poszczególnych bibliotek wykorzystanych następnie do implementacji rozwiązania.

2. Implementacja sieci neuronowych
   Należy zaznaczyć, że poprzedni etap wpływa bezpośrednio na cały przebieg tworzenia rozwiązania. Istotne było, aby wykorzystać maksymalnie rzetelne informacje i źródła pozwalające na stworzenie optymalnych rozwiązań. Założeniem tego etapu była implementacja takiej samej architektury sieci dla obu technologii, wykorzystując dostępne biblioteki. Pod uwagę wzięto również składnie języków. Implementacja metody, dającej ten sam rezultat, może być łatwiejsza do stworzenia i może lepiej spełniać swoją funkcję
3. Proces nauki sieci neuronowych
   Po zakończeniu procesu implementacji przystąpiono do nauczania sieci, które może przebiegać inaczej ze względu na zastosowany język. W tym etapie empirycznie dobierano hiperparametry sieci, w których wyróżniono: liczbę epok nauczania sieci, współczynnik nauczania oraz wielkość serii. Miało to na celu zbadanie wpływu tych parametrów na możliwości nauki sieci i ich istotność w przypadku zastosowania różnych języków programowania. Ukazuje to również, w jaki sposób algorytm wykorzystuje dostępne zasoby. W przypadku zaimplementowanego modelu wykorzystano zbiór danych zawierających 70 000 tysięcy obrazów w formacie 28x28 piksela. Dane podzielono na zbiór treningowy i testowy. Badanie przeprowadzono w takim samym środowisku sprzętowym dla obu języków. Moc obliczeniowa to bezpośredni czynnik wpływający na prędkość przetwarzania danych przez sieć neuronową, dlatego został on wyróżniony w założeniach.
4. Porównanie statystyk działania sieci
   Działająca sieć neuronowa podczas swojej pracy tworzy wiele istotnych statystyk, które opisują, w jaki sposób przebiegał proces nauczania oraz jak działa ona na nieznanym dotychczas zbiorze testowym. Zależnie od problemu różne modele mogą osiągać odmienne wyniki, tym samym tworząc lepsze wzorce oparte na dostępnych danych. Wyróżnione statystyki to cechy wspólne obu sieci. Umożliwiają one porównanie oraz ocenę stworzonych algorytmów. Otrzymane informacje to: czas uczenia się sieci, współczynnik błędu opisany w kolejnych epokach oraz dokładność sieci.
5. Podsumowanie badań i wnioski
   W ostatnim etapie podsumowano cały proces tworzenia algorytmów, biorąc pod uwagę wszystkie stadia. W podsumowaniu najważniejszym założeniem było

określenie, który z języków programowania był optymalny w implementacji sieci neuronowej. W poszukiwaniu informacji należało wyróżnić dostępność i ilość materiałów źródłowych, przy zachowaniu odpowiedniej jakości treści. Wpływa to bezpośrednio na tworzenie algorytmu. W procesie implementacji czynnikiem decydującym jest optymalność pisania kodu źródłowego, a także wykorzystanie dostępnych narzędzi danej technologii. Wzięto więc pod uwagę proces pisania kodu, wyróżniając to, w jaki sposób oba języki umożliwiły zaimplementowanie tych samych elementów sieci neuronowej. W procesie nauczania sieci sprawdzono wpływ hiperparametrów na uczenie się sieci oraz wykorzystanie zasobów przez algorytm. Przeanalizowano przetwarzanie zbioru danych przez oba zaimplementowane rozwiązania. Ostatni etap był porównaniem wyróżnionych statystyk wygenerowanych przez sieci – również w kontekście możliwego rozwoju sieci neuronowych.

Każdy z etapów miał wpływ na ocenę rozwiązań. Badanie przeprowadzono w takich samych warunkach dla obu technologii. Ze względu na ograniczone ramy niniejszego artykułu skoncentrowano się przedstawieniu i omówieniu uzyskanych statystyk działających sieci neuronowych w wybranych językach programowania.

3.2. Kryteria porównania

W badaniu wyróżniono trzy kryteria porównania, które na potrzeby badania uznano za cechy wspólne. Są to informacje wytworzone podczas nauczania sieci oraz przez działający algorytm.

Pierwszym kryterium jest czas, jakiego algorytm potrzebuje na naukę, a zatem to ogólny czas, w jakim stworzony system jest w stanie wykonać kod źródłowy, realizując dane zadanie. Pozwala to na przeanalizowanie optymalności działania sieci oraz szybkości, jaką niesie ze sobą dana technologia. Obecne złożone sieci neuronowe wymagają ogromnych zasobów obliczeniowych, które rosną wraz z rozwojem. Dlatego istotne jest zbadanie, jak język programowania wpływa na czas wnioskowania, który idealnie powinien być zbliżony do czasu rzeczywistego.

Kolejne kryterium to wartość funkcji straty algorytmu osiągnięta dla zbioru testowego. Określa ona różnicę między rzeczywistymi etykietami danych a rozkładem prawdopodobieństwa przynależności do klasy obliczonym przez algorytm. Służy ona do monitorowania postępów uczenia się sieci neuronowej i do ewentualnej korekty hiperparametrów lub modelu. Ukazuje również prędkość, z jaką algorytm dostosowuje swoje wagi w celu minimalizacji wartości funkcji błędu. Określa, kiedy sieć zaczyna tworzyć poprawne wzorce.

Ostatnim kryterium jest dokładność osiągnięta na zbiorze testowym. Stanowi o procencie poprawnie sklasyfikowanych próbek. Odnosi się do zdolności sieci do poprawnej klasyfikacji danych, które nie były wykorzystane w procesie uczenia. Porównanie dokładności wskazuje, który algorytm stworzył bardziej uniwersalne wzorce.

3.3. Dane oraz model sieci neuronowej

W badaniu wykorzystano znormalizowany zbiór danych MNIST, stosowany jako wzorzec do zagadnień klasyfikacji. Jest on popularny ze względu na swój rozmiar oraz stosunkową łatwość w zastosowaniu, a jednocześnie jest wystarczająco wymagający, aby przetestować możliwości algorytmów klasyfikujących.

Zaproponowany model to wielowarstwowa sieć neuronowa, posiadająca dwie warstwy ukryte, jedną wejściową oraz jedną wyjściową. Warstwa wejściowa posiada 784 neurony, każdy przyjmujący wartość pojedynczego piksela z wektora wejściowego przekształconych obrazów (28x28). W celu zapewnienia nieliniowości w modelach sieci neuronowej stosowane są warstwy ukryte, których liczba zależy od złożoności problemu. W badanym modelu zaprojektowano dwie warstwy ukryte zawierające odpowiednio 392 oraz 196 neuronów. W tych warstwach dokonywane są obliczenia pozwalające na odkrycie cech obrazów. Ostatnia warstwa modelu wynika z zakresu cyfr w zbiorze danych, a zatem odpowiada wszystkim możliwym klasom.

W procesie uczenia sieci wykorzystano entropie krzyżową jako funkcję straty oraz optymalizator Adam służący do jej minimalizacji.

4. Wyniki przeprowadzonego eksperymentu

4.1. Porównanie czasu działania algorytmów

4.1.1. Porównanie czasu działania sieci ze względu na liczbę epok

Pierwszym kryterium porównawczym w badaniu był czas działania algorytmu. To czas potrzebny na przejście przez zbiór treningowy we wszystkich określonych epokach uczenia. Wyróżnienie czasu ma swoją podstawę w istotności tego czynnika w zastosowaniu sieci neuronowych. Wymaga się, aby uczenie algorytmu zajmowało jak najmniej czasu, przy zachowaniu wysokiego współczynnika dokładności. Równie ważnym elementem jest ciągłe douczanie się sieci na nowych danych i przebudowywanie swojej struktury wzorców.

Zbadanie czasu nauczania sieci dla dwóch różnych języków programowania ukazuje, który algorytm szybciej wykonuje proces dopasowywania wartości wag. Osiągnięte w ten sposób wyniki również ukazują, który mógłby być wykorzystany w dalszym rozwoju i byłby w tym szybszy. Czas uczenia obrazuje, w jaki sposób algorytm wykorzystuje dostępne zasoby obliczeniowe.

Badanie przeprowadzono dla dwóch sieci neuronowych, w których odpowiednio zmieniono hiperparametry, posiadających bezpośredni wpływ na wykorzystanie zasobów samego algorytmu oraz sprzętowych.

Pierwszą zbadaną konfiguracją hieperparametrów była sieć o następujących współczynnikach: wielkość serii – 128, liczba epok – 10 oraz współczynnik uczenia – 0.001. Zgodnie z założeniami zmierzono czas, jakiego sieć neuronowa potrzebuje na przetworzenie całego zbioru uczącego zawierającego 60 000 tysięcy próbek

wraz z etykietami. Były to równocześnie czasy, w których wykonano kod źródłowy w obu językach programowania. Aby zwiększyć możliwości porównawcze, każdą konfigurację zbadano trzy razy. Tabela 1 przedstawia czas osiągnięty przez algorytmy w trzech niezależnych próbach dla liczby epok wynoszącej 10. W każdej kolejnej próbie wykorzystano to samo środowisko sprzętowe.

Tabela 1. Czas nauki sieci neuronowych dla liczby epok wynoszącej 10 (w sekundach)

Próba	Sieć neuronowa w języku Python	Sieć neuronowa w języku C#
1	36,31	32,47
2	35,89	32,77
3	36,20	32,12
Średnia	36,13	32,45

Źródło: opracowanie własne.

Przedstawione w tabeli 1 wyniki obrazują, w jakim czasie sieci przetworzyły cały zbiór danych treningowych. W przypadku sieci neuronowej napisanej w języku C# odnotowano niższą średnią, a także niższe czasy w każdej z prób. Średnia ta wyniosła 32,45 sekundy, czyli o prawie cztery sekundy szybciej niż algorytm w języku Python, którego średnia wyniosła 36,13 sekundy. Oznacza to, że kompilacja kodu źródłowego, a następnie nauczanie sieci zajęło mniej czasu przy wykorzystaniu języka C#. Istotny wpływ na to ma również interpretator języka Python, który cechuje się wolniejszym działaniem niż języki wykorzystujące preprocesor i kompilator.

Kolejną zbadaną konfiguracją była struktura posiadającą liczbę epok zwiększoną do 20, zachowująca jednocześnie wartość pozostałych hiperparametrów. Zwiększono zatem dwa razy liczbę iteracji przejścia przez cały zbiór danych uczących. Tabela 2 przedstawia czas nauki sieci neuronowych dla liczby epok wynoszącej 20.

Tabela 2. Czas nauki sieci neuronowych dla liczby epok wynoszącej 20 (w sekundach)

Próba	Sieć neuronowa w języku Python	Sieć neuronowa w języku C#
1	68,84	63,48
2	67,41	63,93
3	68,62	63,51
Średnia	68,29	64,64

Źródło: opracowanie własne.

W tabeli 2 można zauważyć, że w przypadku języka C# czasy wzrosły prawie dwukrotnie, analogicznie do podwojonej liczby epok. W języku Python czas został skrócony. Aby sprawdzić, czy zwiększenie liczby epok wpływa proporcjonalnie na

zwiększenie czasu potrzebnego algorytmowi na naukę, czasy zmierzono również dla 30 iteracji (tabela 3).

Tabela 3. Czas nauki sieci neuronowych dla liczby epok wynoszącej 30 (w sekundach)

Próba	Sieć neuronowa w języku Python	Sieć neuronowa w języku C#
1	100,92	95,10
2	102,01	95,74
3	99,56	93,62
Średnia	100,83	94,82

Źródło: opracowanie własne.

W przypadku zwiększenia liczby epok do 30 czas zwiększył się o kolejne 30 sekund. Oznacza to, że w przypadku tej konfiguracji sprzętowej granicą działania zaprojektowanej architektury sieci jest czas zbliżony do wartości 30 sekund potrzebny na wykonanie 10 epok. Warto zauważyć, że również w przypadku tej liczby iteracji szybciej wykonywał się kod źródłowy języka C#. Wraz ze zwiększeniem liczby iteracji uwidoczniła się różnica czasu potrzebnego na przetworzenie zbioru danych treningowych.

4.1.2. Porównanie czasu działania sieci ze względu na wielkość serii

Kolejnym użytym w badaniu czasu parametrem była wielkość serii. Argument ten wskazuje, na jakie partie dzielone mają być dane zbioru treningowego. Zwiększenie tej liczby oznacza zwiększenie liczby próbek przetwarzanych w tym samym czasie przez algorytm.

Zachowując założenia badania, na tym etapie zwiększono tylko wielkości serii. Wykorzystana do zmierzenia czasów wartość wynosiła 256. Pozostałe parametry ustawiono na wartości początkowe. W tabeli 4 przedstawiono osiągnięte wyniki.

Tabela 4. Czasy osiągane przez sieci neuronowe dla serii wynoszącej 256 (w sekundach)

Próba	Sieć neuronowa w języku Python	Sieć neuronowa w języku C#
1	26,66	24,86
2	26,25	25,19
3	26,77	24,76
Średnia	26,56	24,93

Źródło: opracowanie własne.

Zwiększenie wielkości serii przetwarzanej w tym samym czasie wpłynęło na zmniejszenie czasu potrzebnego na wykonanie kodu źródłowego w obu przypadkach. Również w tym przypadku algorytm stworzony w języku C# szybciej przeszedł

przez proces nauczania, jednak różnica spadła do około dwóch sekund. Z tego wynika, że obie sieci wykorzystały zaprojektowaną architekturę w zbliżony sposób. Aby zweryfikować otrzymane wnioski, zmierzono czas potrzebny na przetworzenie serii o wielkości 512. Tabela 5 przedstawia otrzymane wyniki.

Tabela 5. Czasy osiągane przez sieci neuronowe dla serii wynoszącej 512 (w sekundach)

Próba	Sieć neuronowa w języku Python	Sieć neuronowa w języku C#
1	24,21	21,87
2	24,03	22,60
3	24,70	22,62
Średnia	24,31	22,36

Źródło: opracowanie własne.

Czasy osiągane przez obie sieci zmniejszyły się również w tym przypadku. Różnica wyniosła w przybliżeniu dwie sekundy dla czasów średnich, a także zmniejszyła się o zbliżoną wartość. Należy zauważyć, że w przypadku zwiększenia liczby serii sieć potrzebuje od około 2 do 3 sekund na przetworzenie jednej epoki. Zatem w przypadku tego modelu jest to granica, której nie jest w stanie przekroczyć.

Przeprowadzone badanie ukazuje, że sieć neuronowa zaimplementowana w języku C# osiągnęła niższe czasy w każdym z etapów badania. Ze względu na to, że przeprowadzenia pomiarów dokonano na tej samej konfiguracji sprzętowej, należy wykluczyć jej wpływ na przedstawione wyniki.

4.2. Porównanie dokładności i wartości funkcji straty

Następnym kryterium porównawczym sieci neuronowych są wartości funkcji straty. Funkcja ta jest miarą błędu pomiędzy rzeczywistymi etykietami danych a obliczonymi przez model. Wartość ta służy do oceny jakości modelu, a także uczestniczy w procesie optymalizacji wag. W zaprojektowanych modelach wykorzystano entropię krzyżową, która mierzy odległości pomiędzy rozkładem prawdopodobieństwa obliczonym przez algorytm a rzeczywistymi etykietami próbek. Zastosowany optymalizator dąży do minimalizacji tej funkcji.

W celu sprawdzenia, jak działający model radzi sobie z zagadnieniem klasyfikacji, osiągnięte na tym etapie wyniki otrzymano, wykorzystując zbiór testowy. Konfiguracja sieci wynika z wniosków wyciągniętych we wcześniejszych turach. Takie podejście pozwala na porównanie działania sieci jako spójnego i skończonego rozwiązania, które można wykorzystać w innych problemach badawczych.

Zauważono, że największy wpływ na działanie sieci na tym zbiorze danych mają dwa hiperparametry i ich odpowiednio dobrane wartości, które wpływają na optymalność pracy algorytmu. Liczbę epok zwiększono do 15, a wielkość serii do war-

tości 256. Model o tak przyjętych parametrach poddano procesowi uczenia, a następnie jego możliwości próbie klasyfikacji na danych zbioru testowego. Badanie przeprowadzono w trzech osobnych próbach oraz na różnym losowym rozłożeniu całego zbioru. Wartości funkcji straty przedstawiono w tabeli 6.

Tabela 6. Wartości funkcji straty sieci neuronowych na zbiorze testowym

Próba	Sieć neuronowa w języku Python	Sieć neuronowa w języku C#
1	0,0618	0,0715
2	0,0520	0,0655
3	0,0661	0,0805
Średnia	0,051	0,0725

Źródło: opracowanie własne.

W przeprowadzonych trzech niezależnych próbach mniejsze wartości funkcji straty osiągnęła sieć w języku Python. Oznacza to, że w oparciu o ten sam zbiór danych różnica w rozkładzie prawdopodobieństwa była większa dla drugiego w kolejności algorytmu. Pomimo niskich wartości, należy zauważyć, że zbiór testowy jest częścią zbioru danych MNIST zawierającego dużą liczbę różnych próbek, które jednak posiadają wspólne cechy. Dlatego ważne jest, aby na tak wczesnym etapie pracy optymalizator jak najbardziej zminimalizował funkcje straty. Wpływa to na późniejsze możliwości klasyfikacji algorytmu.

Ostatnim kryterium porównawczym jest dokładność klasyfikacji zaimplementowanych sieci neuronowych. Wartość ta przedstawia, ile poprawnych klasyfikacji dokonała sieć na zbiorze testowym. Służy to do oceny skuteczności modelu. Im ta wartość jest wyższa, tym lepiej jest on w stanie przyporządkować wcześniej nieznane dane do odpowiedniej klasy. Ze względu na charakterystykę zbioru danych, który posiada odpowiednio zróżnicowane próbki, dokładność jest skutecznym wskaźnikiem. W przypadku analizowania niezbalansowanego zbioru, w którym jedna z klas dominuje nad pozostałymi, należy użyć innych metryk oceniających skuteczność.

Zgodnie z założeniami badania przeprowadzono trzy niezależne próby modelu, o określonych wyżej, empirycznie dobranych parametrach. W tabeli 7 przedstawiono wyniki tego etapu.

Tabela 7. Dokładność działania sieci na zbiorze testowym

Próba	Sieć neuronowa w języku Python	Sieć neuronowa w języku C#
1	0,9839	0,9813
2	0,9874	0,9833
3	0,9863	0,9790
Średnia	0,9859	0,9812

Źródło: opracowanie własne.

Można zauważyć, że sieć neuronowa napisana w języku C# charakteryzuje się nieznacznie niższą dokładnością niż sieć neuronowa napisana w języku Python. Wynika to z zastosowania tej samej architektury dla obu przypadków, której strukturę optymalnie wykorzystał algorytm stworzony w języku Python. Warto zauważyć, że pomimo wysokiego wyniku dokładność nie przekroczyła poziomu 99% dla zbioru testowego liczącego 10 000 próbek. Oznacza to, że ponad 100 próbek błędnie sklasyfikowano. Biorąc pod uwagę problematykę badania, należy stwierdzić, że osiągnięty został dobry poziom. Należy jednak przeanalizować sytuację zastosowania modelu w bardziej złożonych problemach, które wymagają dokładności na poziomie powyżej 99,99%. Adekwatnym przykładem może być klasyfikacja cyfr znaków drogowych przez komputer pokładowy w samochodzie".

Przeprowadzone badania działania sieci uwidaczniają różnice między dwoma technologiami. Każda z analizowanych posiada swojej atuty, które należy w odpowiedni sposób wykorzystać.

4.3. Wnioski z przeprowadzonego porównania działania sieci neuronowych

W empirycznej części badania zbadano wpływ trzech hiperparametrów na działanie sieci. Wyróżniono: współczynnik uczenia optymalizatora, liczbę epok uczenia oraz wielkość serii. Po sprawdzeniu, jak zmiana ich wartości wpływa na działanie sieci, zauważono, które pozwalają na optymalne wykorzystanie modelu. Zostały one również użyte w dalszej części badania. W ostatnim etapie sprawdzono statystyki działania sieci neuronowych. Zgodnie z założeniami były to: czas uczenia algorytmu, wartości funkcji straty i dokładność.

Czas osiągany przez algorytmy został zbadany ze względu na możliwości rozwoju sieci neuronowych. Działanie dzisiejszych sieci opiera się na ciągłym douczaniu na podstawie nowych partii wiedzy. Zastosowanie technologii może mieć kluczowe znaczenie dla dostarczenia działającego oprogramowania. W każdej z prób, sieć napisana w języku C# osiągała lepsze czasy, niezależnie od konfiguracji. Jest to kluczowa informacja, jeśli chodzi o zastosowanie tego języka w systemach dedykowanych.

W kolejnej części badania wykorzystano empirycznie dobrane wartości parametrów oraz sprawdzono działanie sieci na zbiorze testowym. Następnym kryterium porównawczym była wartość funkcji straty, czyli różnica między wartością obliczonego rozkładu prawdopodobieństwa przynależności do klasy a rzeczywistą etykietą. W trzech niezależnych próbach lepszy wynik osiągnęła sieć w języku Python. Oznacza to, że pomimo tej samej architektury, narzędzia dostępne w tej technologii lepiej poradziły sobie z zadaniem optymalizacji.

Ostatnie kryterium to dokładność na zbiorze testowym. Biorąc pod uwagę złożoność problemu badawczego, należy stwierdzić, że obie sieci osiągnęły dobre wyniki, jednak również tu bardziej optymalny był język Python.

Wszystkie etapy badania przebiegły pomyślnie, a z każdej kolejnej części wyciągnięto kluczowe informacje, wymagane do porównania.

Biorąc pod uwagę wszystkie dane zebrane podczas badania, sformułowano następujące wnioski:

1. Optymalną technologią do zastosowania w dziedzinie sieci neuronowych jest język Python. Posiada on rozbudowane narzędzia, które przyspieszają tworzenie rozwiązań. Mnogość oraz dostępność źródeł umożliwia optymalną implementację, nawet dla początkującego użytkownika. W języku C# proces ten jest bardziej złożony.
2. Składnia języka Python znacznie wpływa na tworzenie rozwiązania. W porównaniu z językiem C# nie występuje tu wiele czynników, które mogą powodować ewentualne błędy.
3. Język C# jest technologią szybszą w działaniu, co jest kluczowe w tworzeniu systemów dedykowanych. Pomimo mniejszej liczby skupionych na sieciach neuronowych narzędzi, algorytm nadal osiągał porównywalnie dobre wyniki.
4. Należy zauważyć, że technologia C# może być wykorzystywana z sukcesem w obszarze sieci neuronowych. Jest to informacja kluczowa dla systemów stworzonych w środowisku .NET. Specjaliści mogą zatem tworzyć rozwiązania w języku C#, bez konieczności integracji z kolejną technologią.

5. Zakończenie

W niniejszym artykule przeanalizowano wyniki przeprowadzonego eksperymentu dotyczącego porównania dwóch języków programowania – tj. Python oraz C# – mającego na celu sprawdzenie, który z nich jest optymalny w procesie implementacji sieci neuronowej. Opierając się na dostępnym zbiorze danych, dla obu technologii stworzono działające algorytmy, które osiągały zadowalające wyniki. Tak zaprojektowana sieć neuronowa może być z sukcesem stosowana również jako podstawa działania sieci, które zajmują się bardziej złożonymi problemami badawczymi.

Na podstawie wyników należy zauważyć korzyści płynące z zastosowania obu technologii w tej dziedzinie uczenia maszynowego: szybkość przetwarzania kodu przez język C# oraz rozwinięte narzędzia dostępne w języku Python umożliwiające implementację sieci neuronowych. Otrzymane wyniki mogą posłużyć do następnych badań dotyczących wykorzystania sieci w rozbudowanych systemach dedykowanych.

Badania należałoby również uzupełnić o kolejne zbiory danych, które zwiększyłyby ich zakres, a także wykazałyby następne różnice pomiędzy algorytmami, które mogły pozostać niezauważone podczas przebiegu tego eksperymentu.

Literatura

Chollet, F. (2017). *Deep Learning with Python*. O'Reily.

Macukow, B. (2020). Sieci neuronowe, historia badań i podstawowe modele. *Prace Naukowe Wydziału Elektroniki i Technik Informacyjnych Politechniki Warszawskiej*, *1*, 90-108. https://pages.mini.pw.edu.pl/~macukowb/wspolne/PNEiTI.pdf

McCulloch, W. i Pitts, W. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5, 115-116.

Minsky, M. i Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry.* MIT Press.

Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review, 65*(6), 386-408. https://doi.org/10.1037/h0042519

Ruby, M. (2023). *How ChatGPT Works: The Model Behind The Bot.* https://towardsdatascience.com/how-chatgpt-works-the-models-behind-the-bot-1ce5fca96286

Rumelhart, D., Hinton G. i Williams, R. (1986). Learning Representations by Back-Propagating Errors. *Nature, 323*. Nature.com. https://www.nature.com/articles/323533a0

Zhang, C. i Lu, Y. (2021). Study on Artificial Intelligence: The State of the Art and Future Prospects. *Journal of Industrial Information Integration*, 23.

Comparison of the Performance of Neural Networks in Python and C#

Abstract: This article provides a process of comparison between two deeply developed high-level programming languages. The languages chosen are Python and C#. They were used to create a neural network with a specific architecture, used for the problem of digit classification. The research objective was to see which of the mentioned technologies is optimal throughout the implementation of the finished solution. In the study, the MNIST dataset was used and an operational multi-layer neural network was designed. The comparison criteria were the statistics produced during the operation of the network. Based on the results of the study, the optimal programming language, in this field of machine learning, was found to be Python.

Keywords: neural network, classification, Python, C#