Diagnostyka nawierzchni drogowej przy zastosowaniu metod sieci neuronowych – studium przypadku

ZUZANNA JÓŹWIAK

zuzanna.anna.jozwiak@gmail.com

ANDRZEJ POŻARYCKI

Politechnika Poznańska
andrzej.pozarycki@put.poznan.pl
ORCID: 0000-0002-1321-066X

PRZEMYSŁAW GÓRNAŚ

Politechnika Poznańska
przemyslaw.gornas@put.poznan.pl
ORCID: 0000-0002-0632-4191

W artykule porównane zostały dwa modele sieci neuronowych do wykrywania uszkodzeń, które można zaobserwować na nawierzchni jezdni. Zaprezentowano architekturę sieci neuronowej, którą opracowano głównie na potrzeby analizy obrazów cyfrowych. Modele testowano na zbiorze niezależnych od procesu uczenia ortogonalnych obrazach cyfrowych nawierzchni jezdni. Mając na uwadze różnice wynikające z różnej budowy wykorzystanych modeli, przedstawiono elementy analizy porównawczej związanej z oceną efektywności obu testowanych modeli na przykładzie zadania klasyfikacyjnego z obszaru obrazowej diagnostyki nawierzchni drogowych.

Podstawę motywacji wykonania diagnozy nawierzchni jezdni stanowi potrzeba pozyskiwania informacji, które umożliwiają dokonanie oceny jej stanu technicznego. Przy pomocy nowoczesnych, wydajnych i nieinwazyjnych systemów pomiarowych gromadzi się szczegółowe informacje o cechach eksploatacyjnych, a następnie bazując na analizie w obszarach wybranych strategii utrzymaniowych dokonuje się oceny, która jest podstawą podejmowania decyzji o zabiegach naprawczych. W diagnostyce nawierzchni jezdni powszechnie brane pod uwagę cechy eksploatacyjne wyrażane są przez:

- wskaźnik stanu spękań,
- wskaźnik ugięć nawierzchni,
- wskaźnik równości podłużnej,
- wskaźniki równości poprzecznej (nierówności poprzeczne, głębokość kolein, zaburzenia spadków),
- wskaźnik stanu powierzchni,
- wskaźnik właściwości przeciwpoślizgowych,
- wskaźniki makrotekstury czy chropowatości.

Każdy z tych parametrów może być kwalifikowany według klas i oceny jakościowej przy wykorzystaniu zmiennych lingwistycznych np. odcinek nawierzchni jezdni:

- Klasa A –w stanie dobrym,
- Klasa B – w stanie zadowalającym,
- Klasa C – w stanie niezadowalającym,
- Klasa D – w stanie złym [6].

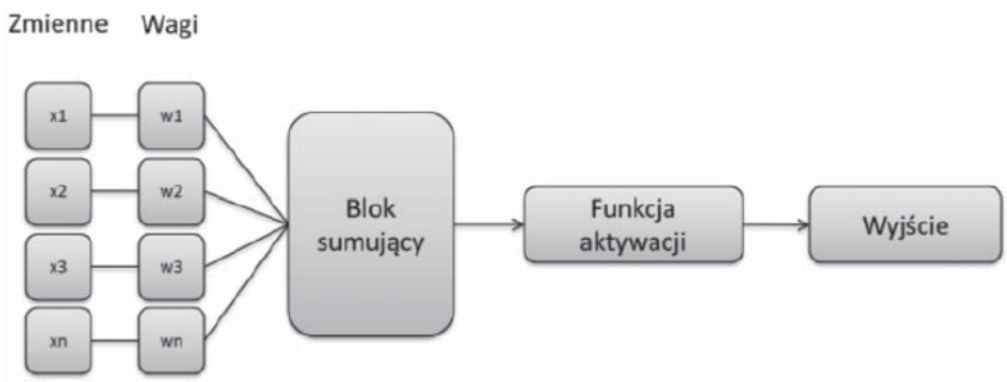
Procedury uczenia maszynowego

Podstawową jednostką sieci neuronowej jest prosty element obliczeniowy nazywany neuronem. Sieć neuronowa to połączenie sztucznych neuronów. Architektura sieci neuronowych składa się przynajmniej z trzech warstw:

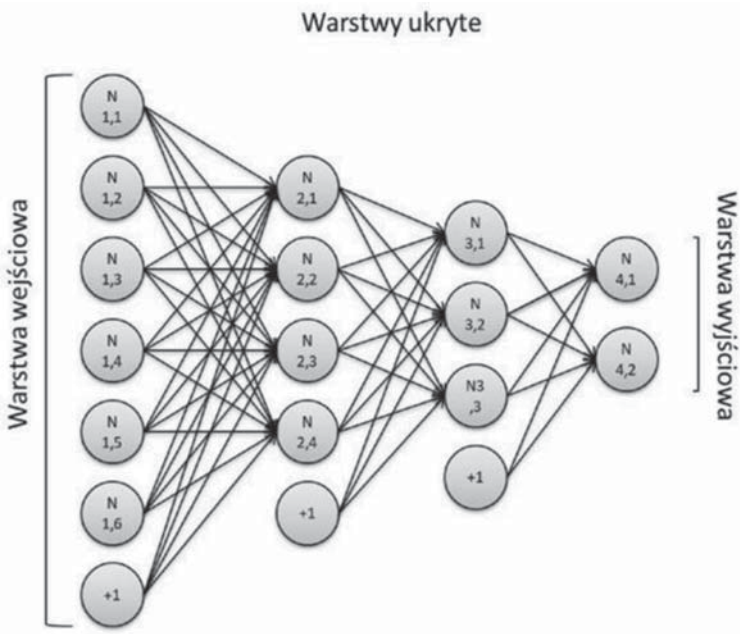
- **wejściowa** – jest odpowiedzialna za pobranie, a następnie przekazanie danych do pierwszej warstwy ukrytej,
- **ukryta** – nie można w sposób bezpośredni przekazywać informacji do warstwy ukrytej; jej zadaniem jest nauczenie i wykonanie obliczeń,
- **wyjściowa** – zwrócenie obliczonych wartości [2].

Sieć neuronowa przetwarza informacje za pomocą połączeń, które mają nadane odpowiednie wagi. Wartość wagi ulega zmianie przez cały proces uczenia i może przyjmować wartości ujemne, dodatnie lub równe zero. Suma cech wejściowych traktowana jest jako argument funkcji aktywacji, która jest wartością wyjściową danego neuronu. Przykładowy schemat został przedstawiony na rysunku 1.

Spośród funkcji aktywacji wykorzystywane są funkcje nieliniowe, liniowe oraz skokowe. Z praktycznych doświadczeń wynika, że największą skuteczność wykazują sieci wielowarstwowe [10]. Przykładowy schemat sieci wielowarstwowej jest pokazany na rysunku 2. Składa się ona co najmniej z jednej warstwy ukrytej.



Rys. 1. Model neuronu [2]



Rys. 2. Budowa jednokierunkowej sieci wielowarstwowej [2]

W kolejnych krokach, zgodnie z założoną strategią uczenia, algorytm zmienia wartości wag w taki sposób, aby minimalizować funkcję celu na wyjściu. Modyfikacja wag sieciowych połączeń między neuronami, szczególnie w początkowej fazie procesu uczenia, odbywa się na zasadzie prób i błędów. Proces uczenia jest wrażliwy na tzw. efekt przetrenowania lub niedouczenia sieci. W pierwszym przypadku sieć nie będzie rozpoznawać danych zbliżonych, nieidentycznych do danych wzorcowych, natomiast w drugim, rozpoznanie będzie obarczone nadmierną liczbą błędów. W obu przypadkach użyteczność sieci jest niewielka.

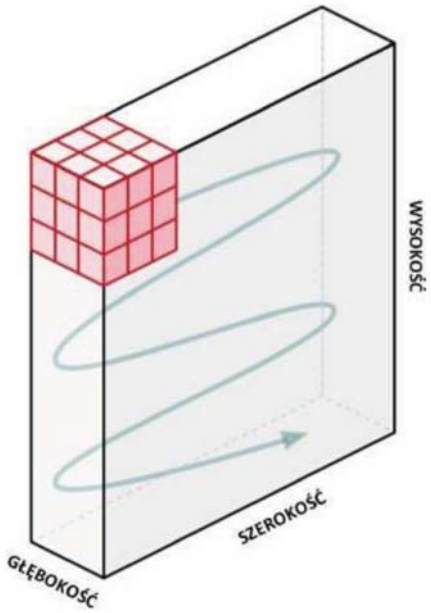
Idea procedury głębokiego uczenia na podstawie obrazów cyfrowych

W ogólnym przypadku można powiedzieć, że uczenie maszynowe, polega na komputerowej analizie powiązań między danymi. Im bardziej złożone są dane wejściowe, tym trudniej dokonuje się obiektywnej analizy relacji między nimi [5]. W pierwszym etapie procedury głębokiego uczenia opartego na zbiorach obrazów cyfrowych, dokonywana jest ekstrakcja cech (kontrast, jasność, lokalizacja krawędzi itp.). Z oryginalnych obrazów cyfrowych uzyskiwane są informacje o dominującym znaczeniu. W dalszej kolejności przestrzeń zmiennych redukowana jest do mniejszych wymiarów [11]. Jedną z dostępnych bibliotek do przetwarzania numerycznego obrazów cyfrowych w zakresie ekstrakcji cech z obrazu jest biblioteka Halcon [1]. Drugim etapem procedury głębokiego uczenia jest klasyfikacja. Na podstawie zebranych wartości liczbowych reprezentujących określone cechy obrazu wydzielane są grupy o zbliżonych cechach. Wyróżnia się trzy główne rodzaje uczenia maszynowego:

- **uczenie nadzorowane** – w tym sposobie uczenia maszynowego dysponuje się zbiorem przykładów uczących,
- **uczenie bez nadzoru** – w tym sposobie uczenia SSN nie wykorzystuje się przykładów uczących. Algorytm obliczeniowy jest ukierunkowany na wyszukiwanie zależności i wzorców w zbiorach analizowanych danych,
- **uczenie ze wzmocnieniem** – w tym przypadku algorytm komputerowy działa w oparciu o warunki brzegowe (ang. *constraints*).

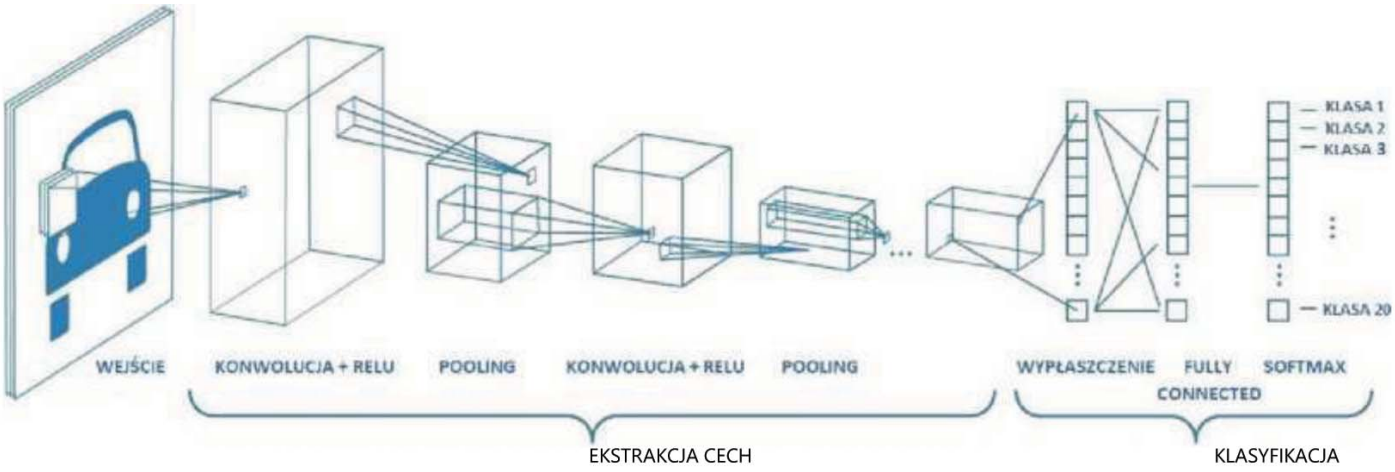
Zasada działania głębokiego uczenia maszynowego opiera się na podobieństwie budowy sztucznego neuronu do budowy neuronu biologicznego. Sieci neuronowe składają się z wielu komórek obliczeniowych połączonych miedzy sobą. Każda z nich wykonuje prostą operację obliczeniową [16]. Zaletą głębokiego uczenia, w porównaniu do klasycznych modeli matematycznych, jest zdolność do automatycznej ekstrakcji cech. Głębokie uczenie maszynowe wykorzystuje architekturę konwolucyjnych sieci neuronowych CNN (ang. *Convolutional Neural Network*), gdzie obraz cyfrowy rozkładany jest na czynniki pierwsze (cechy), a następnie wskazuje się jego przynależność do zdefiniowanej wcześniej klasy [15] (np. spękanie, wybój, studzienka, właz itd.).

Automatyczną ekstrakcję cech z obrazów cyfrowych często realizuje się za pośrednictwem operatora znanego jako konwolucja. Metoda konwolucji polega na rozwiązywaniu lokalnie skoncentrowanych zadań algebraicznych realizowanych w całej przestrzeni obrazu cyfrowego. Ilustracja graficzna reprezentująca filtrowanie przestrzeni obrazu tym sposobem pokazana jest rysunku 3.



Rys. 3. Schemat konwolucji obrazu cyfrowego z filtrem [15]

W wyniku działania operatorów algebraicznych na poszczególnych poziomach sieci konwolucyjnej następuje redukcja liczby parametrów, które opisują obraz cyfrowy przy równoczesnym zachowaniu informacji o jego cechach. W kolejnych warstwach sieci ekstrahowane są poszczególne



Rys. 4. Schemat przedstawiający działanie metody CNN [15]

cechy obrazu cyfrowego. Mówi się tutaj o cechach „niskiego poziomu" np. ekstrakcja linii albo „wysokiego poziomu" np. ekstrakcja koloru czy kształtu danego obiektu widocznego na zdjęciu [15]. Schemat redukcji wymiarów w sieciach konwolucyjnych pokazany jest na rysunku 4.

Wyspecjalizowane filtry w modelach CNN powiązane są z:

- warstwami konwolucji – tworzenie mapy cech,
- warstwami typu *pooling* – redukcja wymiarów zdjęcia,
- warstwami *fully connected* – wskazanie przynależności.

Na podstawie otrzymanej mapy cech w warstwach *fully connected* wykonuje się analizy obecności cech o określonych atrybutach, a następnie określa się prawdopodobieństwo przynależności wejściowego zdjęcia do określonej klasy [14].

Chcąc wykonywać operacje na dużej ilości danych, metody głębokiego uczenia wymagają sporych mocy obliczeniowych. Operacje matematyczne wykorzystywane do uczenia maszynowego wyróżniają się dużą liczbą prostych instrukcji algebraicznych, dlatego efektywne uczenie modelu sieci warto jest realizować za pośrednictwem procesorów na kartach graficznych GPU, a nie CPU [9]. W sytuacji gdy zasoby obliczeniowe komputera osobistego nie są wystarczające, trenowanie sieci można wykonać w darmowym środowisku Google Colaboratory. Usługa Colab udostępnia możliwość pisania i uruchamiania kodu komputerowego (napisanego w języku programowania Python) za pomocą przeglądarki internetowej. Środowisko Google Colaboratory jest zintegrowane z narzędziami Google Drive i biblioteką PyTorch do uczenia maszynowego. Taki dostęp do zasobów GPU lub CPU powoduje, że powolny proces uczenia sieci nie obciąża zasobów indywidualnego użytkownika.

Wykorzystane modele

Do testów wzięto pod uwagę dwa modele: VGG16 oraz VGG19. Model VGG16 to model sieci neuronowej, autorstwa K. Simonyana i A. Zissermana z Uniwersytetu Oksfordzkiego. Model został zaproponowany w celu rozwiązania problemu klasyfikacji obrazów cyfrowych z zasobów bazy danych znanej jako „ImageNet" z dokładnością 92.7%. VGG16 był trenowany na karcie NVIDIA Titan Black [13]. Jedyna różnica między modelami sprowadza się do liczby warstw. Model VGG16 posiada szesnaście warstw, natomiast model VGG19 posiada ich dziewiętnaście.

Oryginalnie modele te były przeznaczone do klasyfikacji danych wejściowych dla 1000 klas, dlatego na potrzeby pracy [4] do warstwy *fully-connected* analizowanych modeli dodano dodatkową warstwę z dwoma klasami w warstwie wyjściowej, a mianowicie:

- dobry stan powierzchni – *good,*
- zły stan powierzchni – *bad*.

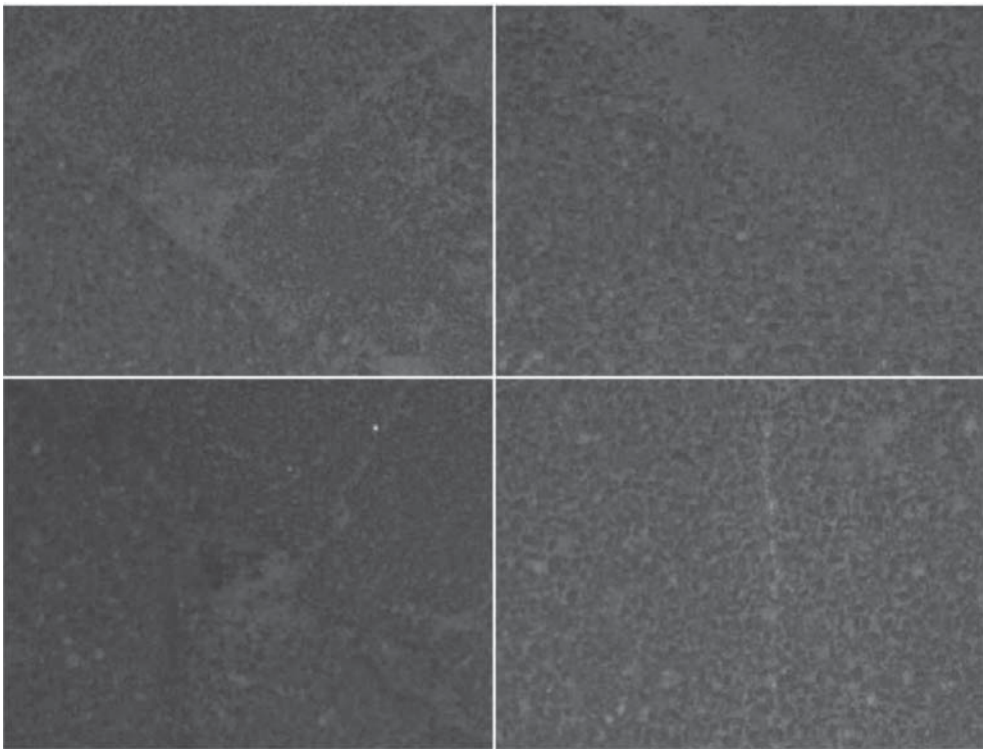
Pozostałe parametry kształtujące architekturę modeli VGG16 i VGG19 nie były modyfikowane w porównaniu do ustawień domyślnych.

Zastosowanie technik głębokiego uczenia maszynowego na przykładzie zdjęć nawierzchni jezdni

Zakłada się, że wskaźnik stanu powierzchni [7] w analizowanym przykładzie przyjmuje wartości od 0 do 1. Wartość 0 oznacza powierzchnię jezdni nieuszkodzonej, natomiast wartość 1 uszkodzonej. Na potrzeby studium, zadanie klasyfikacji obejmowało:

- uszkodzenia, które nie posiadają wymiaru (wyboje),
- wady posiadające wymiar podłużny i poprzeczny (spękania siatkowe, łaty, ubytki),
- wady posiadające wymiar podłużny i poprzeczny (pęknięcia pojedyncze).

Zadaniem sieci neuronowej była klasyfikacja wszystkich obrazów cyfrowych, na których widać było znamiona wymienionych wad nawierzchni jezdni do grupy nawierzchni uszkodzonych. W tym celu należało spełnić warunek konieczny zbioru uczącego, który mówi, że w bazie danych obrazów cyfrowych wykorzystanych do nauki sieci neuronowej znajdują się wszystkie trzy typy wymienionych uszkodzeń.



Rys. 5. Przykłady obrazów cyfrowych zaliczonych do grupy – nawierzchnia uszkodzona

Przykład uszkodzonej nawierzchni został przedstawiony na rysunku 5.

Przed rozpoczęciem uczenia należy zapewnić odpowiednio liczną bazę danych zawierającą zdjęcia. W literaturze jest mowa, że minimalna liczba zdjęć do celów trenowania SSN wynosi 10000. W tym miejscu warto wspomnieć o technikach tzw. augmentacji. Augmentacja to technika wykorzystywana w uczeniu głębokich sieci neuronowych, która wprowadza dodatkową (można powiedzieć sztucznie wymuszoną) różnorodność zbiorów wejściowych. Techniki augmentacji pomagają zniwelować zjawisko nadmiernego dopasowania się uczonego modelu do zbioru treningowego, poprzez wprowadzenie różnych transformacji do zbioru [8]. Chodzi w nich o to, żeby np. z jednego rzeczywistego zdjęcia wygenerować kilka innych (podzielenie jednego zdjęcia na kilka mniejszych). Dodatkowo każde zdjęcie może np. obrócić o 30, 60, 90, 180 stopni lub każdą inną dowolną wartość kąta. Algorytmy komputerowe każde takie przekształcone wstępnie zdjęcie rozpoznają jako różne zdjęcia. Podejście okazuje się bardzo skuteczne, dlatego i w tym przypadku, zdjęcia z przygotowanej własnej bazy danych do procesu uczenia sieci neuronowej poddano następującym operacjom:

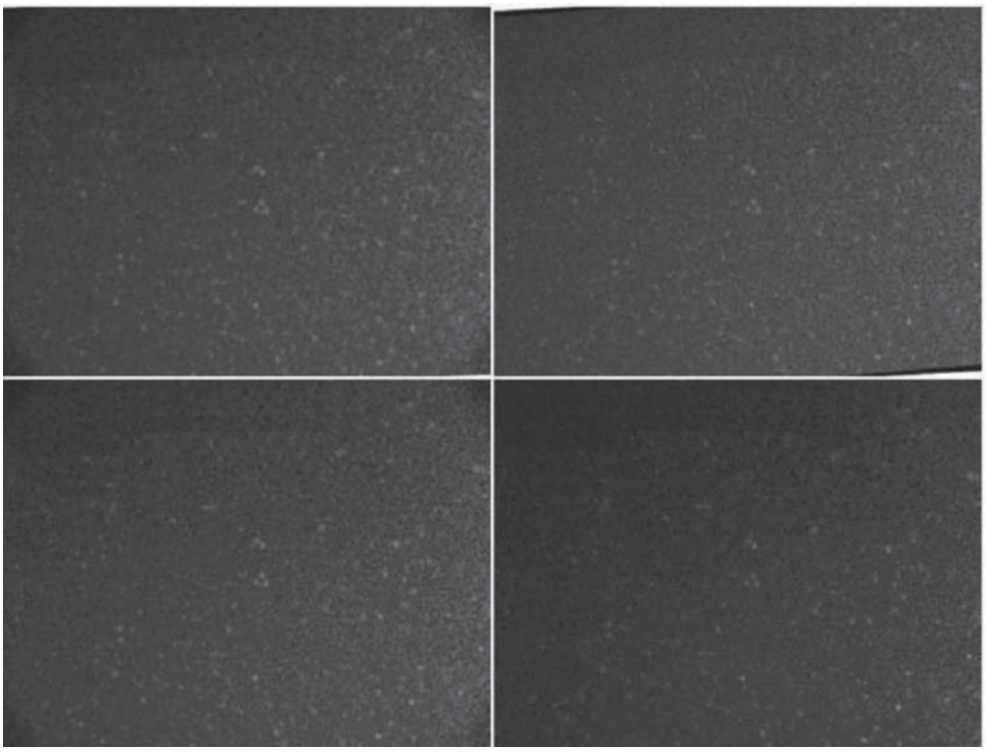
- skalowanie,
- obracanie,
- rozmycie,
- lustrzane odbicie,
- rozjaśnienie,
- przesunięcie.

Przykłady uzyskanych efektów zostały przedstawione na rysunku 6–7.

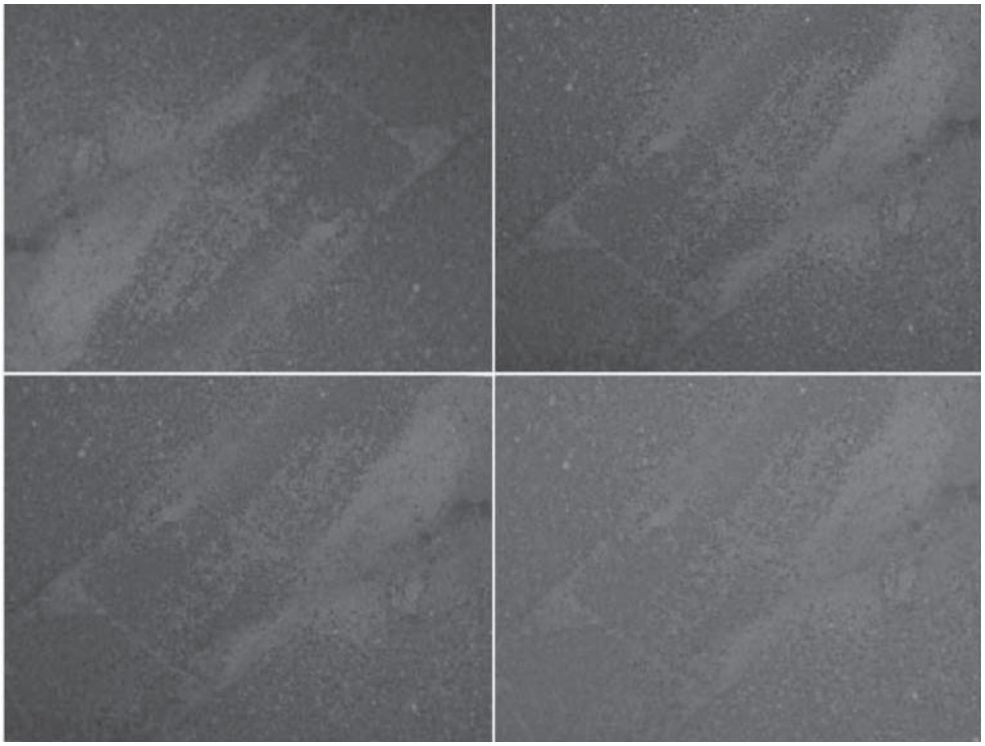
Na potrzeby wykorzystania środowiska Google-Colab w kolejnym kroku należało przygotować strukturę folderów. Korzystając z dostępu do Google Drive przygotowana struktura obejmowała trzy główne katalogi:

- „Zdjęcia” – zawierający zdjęcia nawierzchni uszkodzonej i nieuszkodzonej,
- „Programy” – zawierający skrypt do uczenia modelu oraz skrypt do sprawdzenia dokładności nauczonego modelu,
- „Modele” – do przechowywania modeli wyuczonej sieci.

W zbiorze ImageNet, który został wykorzystany do nauki modeli VGG16 oraz VGG19, rozmiar zdjęć wynosi 224 × 224 piksele. W opisywanym tutaj zbiorze zdjęć ich rozmiar wynosił 2448 × 1224 pikseli. Chcąc budować modele SSN



Rys. 6. Przykładowe transformacje obrazów cyfrowych zaliczonych do grupy – nawierzchnia bez uszkodzeń



Rys. 7. Przykładowe transformacje obrazów cyfrowych zaliczonych do grupy – nawierzchnia uszkodzona

na własnym zbiorze zdjęć należało je znormalizować. Jest to proces, w którym należy doprowadzić rozmiar zdjęć, średnią wartość pikseli oraz średnią wartość odchylenia standardowego pikseli w poszczególnych kanałach RGB zdjęcia wejściowego do odpowiadających im wartości zbioru zdjęć z ImageNet. Dla zbioru ImageNet średnia wartość pikseli w kanałach RGB wynosi (0.485, 0.456, 0.406), natomiast średnia wartość odchylenia standardowego pikseli jest równa odpowiednio (0.229, 0.224, 0.225).

Do analizy wykorzystano bibliotekę PyTorch, w której normalizacja zdjęć wejściowych realizowana jest z wykorzystaniem operatora transforms.Compose(). Operator ten jest także odpowiedzialny za inne przekształcenia typu: rotacje, skalowanie, zmiany jasności, kontrastu oraz skali szarości [3]. Całą bazę zdjęć wykorzystaną w ramach pracy [4] podzielono na trzy grupy, dlatego katalog „zdjęcia" składał się z dodatkowych trzech podkatalogów. Mowa jest o grupach:

- **ucząca** – zbiór 1750 zdjęć reprezentujących stan powierzchni określony jako dobry stan techniczny (ang. *good*) i 1750 jako zły stan techniczny (ang. *bad*)
- **walidacyjna** – zbiór 750 zdjęć reprezentujących stan powierzchni określony jako dobry stan techniczny (ang. *good*) i 750 jako zły stan techniczny (ang. *bad*)
- **testowa** – zbiór 120 reprezentujących stan powierzchni określony jako dobry stan techniczny (ang. *good*) i 120 zdjęć jako zły stan techniczny (ang. *bad*).

Podczas procesu uczenia wykorzystywane są dwa pierwsze zbiory. Zbiór uczący i walidacyjny zawiera znormalizowane, posegregowane zdjęcia i dostosowane do wymogów środowiska Google-Colab.

Po przygotowaniu modułów uczącego i walidacyjnego wczytany został zestaw wag i parametrów inicjujących pierwszą iterację procesu uczenia. Po załadowaniu wybranych modeli, zgodnie z zasadami środowiska obliczeniowego wartości wag i parametrów inicjujących należało zablokować, by nie zostały one zaktualizowane w kolejnych iteracjach procesu uczenia modelu.

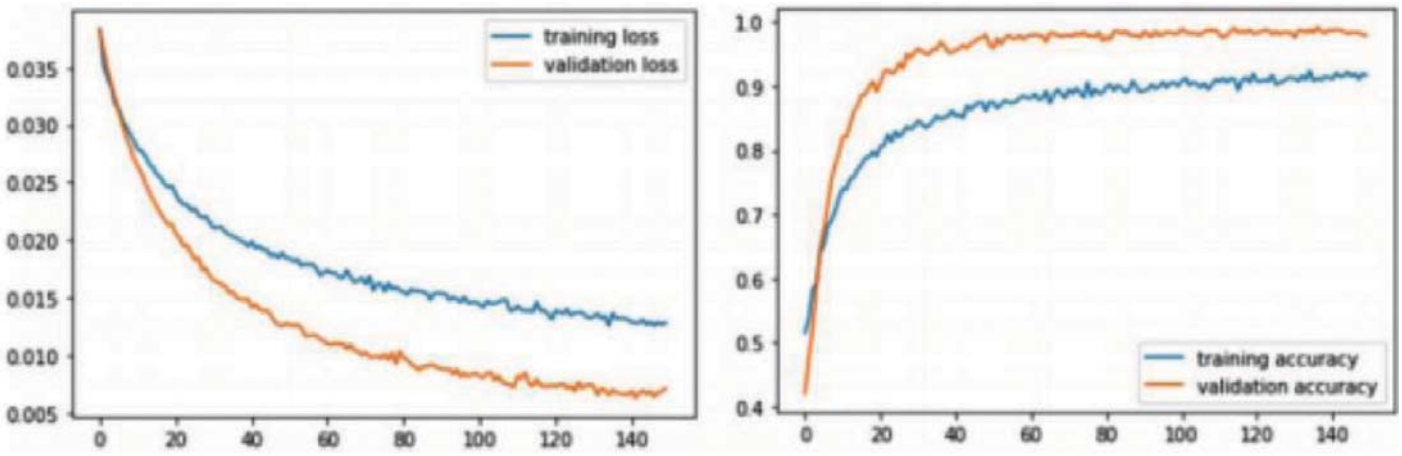
Proces uczenia i walidacji

Proces uczenia rozpoczęto od wybrania funkcji straty. Im mniejsza wartość funkcji straty, tym lepsze wyniki uczonego modelu. Funkcja straty odpowiedzialna jest za określenie różnicy pomiędzy wartością predykcji, a wartością oczekiwaną. Najbardziej istotnym parametrem, który należy określić podczas definiowania optymalizatora, jest prędkość uczenia. Ma on wpływ na wielkość zakresu zmian aktualizowanych wartości parametrów, biorących udział w procesie nauki. Dobór tego parametru należy do użytkownika i nie może być przypadkowy. Zbyt duża prędkość uczenia powoduje pominięcie lokalizacji minimum funkcji straty. To sprawia, że nie zostaną osiągnięte dostatecznie dobre wyniki.

Proces uczenia sieci neuronowej odbywa się w tak zwanych epokach. Liczba epok potrzebnych do wyuczenia sieci neuronowej to liczba iteracji obliczeniowych złożonych z pojedynczych obliczeń w poszczególnych warstwach. Jest to proces iteracyjny.

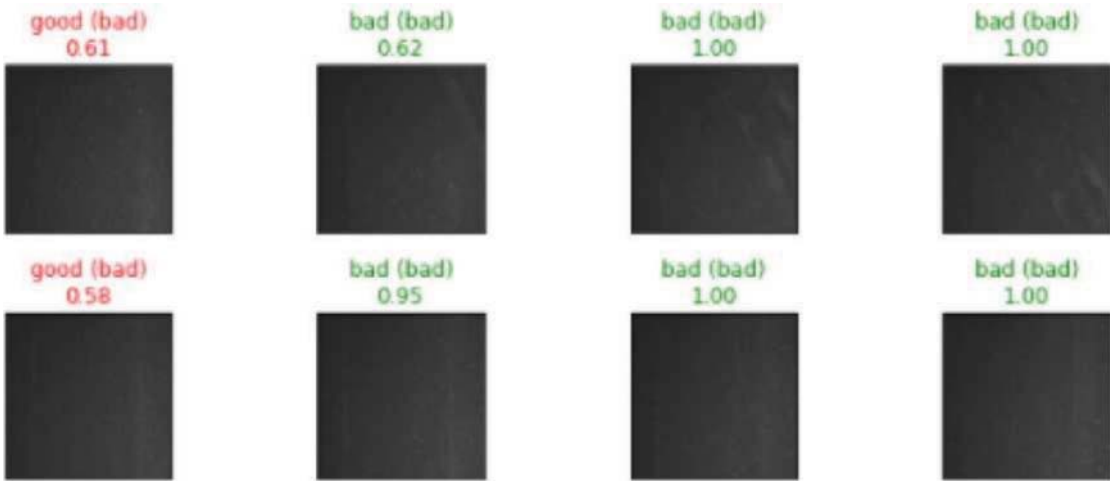
W każdej epoce zapamiętywana jest wartość funkcji straty, a także dokładność modelu dla zbiorów uczącego oraz walidacyjnego. Dokładność, z jaką dany model generuje wyniki, określana jest na podstawie stosunku ilości poprawnie i niepoprawnie zaklasyfikowanych zdjęć. Wraz z kolejnymi epokami można zauważyć zmniejszanie się wartości funkcji straty oraz coraz lepszą dokładność predykcji modelu. Na rysunku numer 8 pokazane są dwa wykresy obrazujące zmianę wartości parametrów, na podstawie których kontrolowany jest proces uczenia. Po lewej stronie pokazana jest funkcja straty, która maleje w kolejnych epokach, a po prawej stronie pokazano funkcje zmiany dokładności modelu w kolejnych epokach.

Po zakończonych obliczeniach we wszystkich epokach, następuje etap wyznaczenia parametrów modelu, dla którego algorytm osiągnął największą dokładność na zbiorze walidacyjnym. Jest to ostatni etap w całym procesie nauki. W środowisku Google Colab po zakończeniu procesu nauki stan parametrów modelu zostaje zapisany do osobnego pliku.

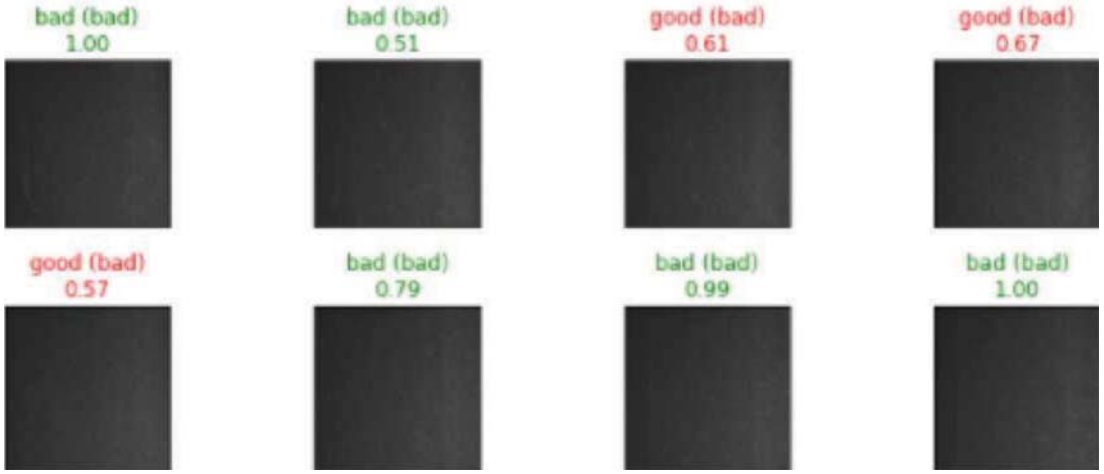


Rys. 8. Wykresy obrazujące zmianę wartości parametrów wykorzystywanych do kontroli procesu uczenia (od lewej: funkcje straty dla zbiorów treningowego i walidacyjnego – ang. training and validation loss oraz funkcja opisująca zmianę dokładności predykcji dla zbiorów treningowego i walidacyjnego – ang. training and validation accuracy)

Zgodnie z założeniami w opisywanym tu eksperymencie, wyuczony model powinien klasyfikować zbiór na dwie klasy: nawierzchnia uszkodzona i nawierzchnia nieuszkodzona. Trzeba też zauważyć, że w procesie nauki można wyróżnić przypadki zaklasyfikowania zdjęcia nawierzchni jako fałszywie uszkodzona lub fałszywie nieuszkodzona. Pierwszy przypadek wystąpi wtedy, gdy algorytm zaklasyfikuje zdjęcie nawierzchni nieuszkodzonej jako nawierzchnia uszkodzona. Drugi przypadek wystąpi, gdy algorytm zaklasyfikuje zdjęcie nawierzchni uszkodzonej do klasy nawierzchni nieuszkodzonej.



Rys. 9. Format wyników klasyfikacji wygenerowanych dla modelu VGG16



Rys. 10. Format wyników klasyfikacji wygenerowanych dla modelu VGG16

Do oceny ostatecznych wyników wykorzystano funkcję Softmax(), której zadaniem jest określanie wyników jako prawdopodobieństwo. Wartość funkcji Softmax() równa zero oznacza znikome prawdopodobieństwo przynależności do danej klasy, natomiast równa jeden (stu procentowa pewność przynależności zdjęcia do danej klasy).

Na rysunku numer 9 oraz 10 zostały zilustrowane przykłady klasyfikacji z przypisanymi wartościami zwracanymi przez funkcje Softmax() odpowiednio w modelach VGG16 i VGG19.

Kolorem czerwonym zostały podpisane błędnie sklasyfikowane przez algorytm zdjęcia nawierzchni, natomiast kolorem zielonym zostały wyróżnione zdjęcia poprawnie sklasyfikowane.

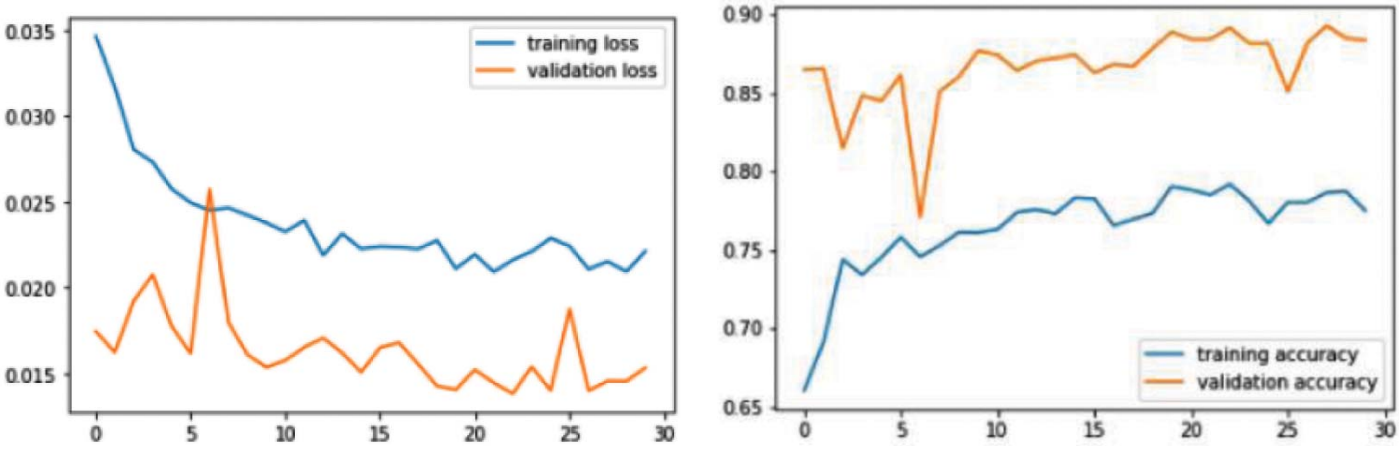
Porównanie efektywności

Model VGG16 przetestowano na 30 epokach, natomiast model VGG19 na 35 epokach. Czas przeuczania pierwszego modelu wynosi 314 minut i 51 sekund, natomiast drugiego modelu 546 minut i 23 sekundy. Na rysunkach 11 oraz 12 został przedstawiony proces przeuczania modeli. W przypadku obu wykorzystanych modeli wartość błędu na zbiorach treningowych i walidacyjnych maleje wraz z czasem, natomiast wartość dokładności wzrasta.

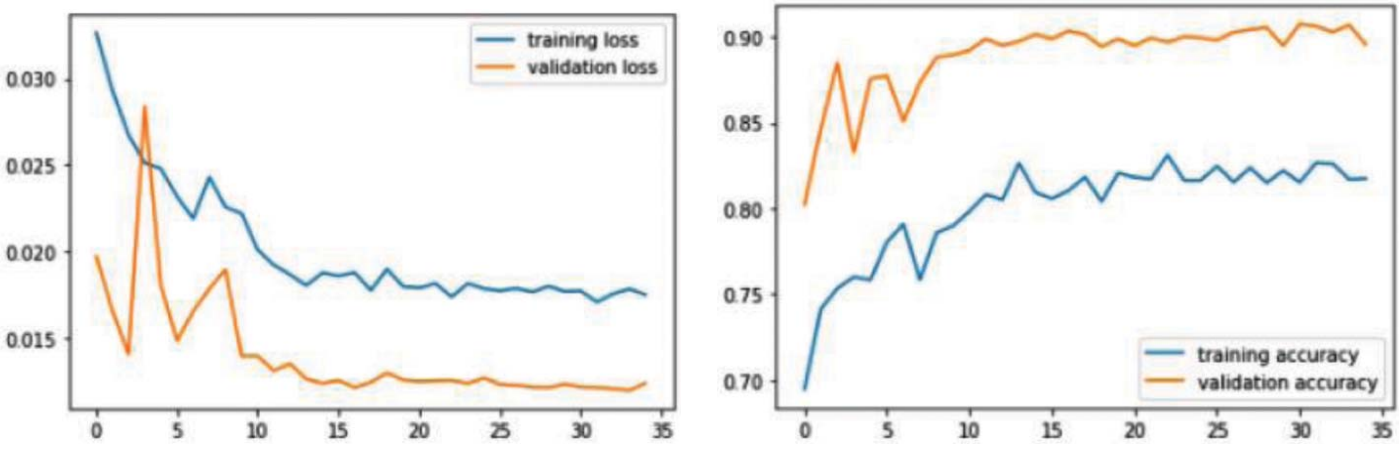
Dokładność modeli to procentowy stosunek liczby poprawnie skategoryzowanych przypadków zdjęć do liczby wszystkich zdjęć. Ostateczne testy dokładności sieci neuronowych zostały przeprowadzone na zbiorze zawierającym 240 zdjęć nawierzchni jezdni. W zbiorze tym znajdowało się 120 zdjęć przypisanych do klasy *nawierzchnia uszkodzona* oraz 120 do klasy *nawierzchnia nieuszkodzona*. Grupa zdjęć testowych nie była uwzględniona w procesie trenowania modelu.

Dla modelu VGG16 uzyskano zgodność predykcji z oczekiwaniami na poziomie 96,67%, a dla modelu VGG19 97,92%. Przypadków fałszywie dodatnich, czyli takich, w których model sklasyfikował dane zdjęcie do klasy *nawierzchnia uszkodzona*, a dokonał predykcji jako zdjęcie nawierzchni nieuszkodzonej, dla modelu VGG16 było 8, natomiast dla modelu VGG19 5 przypadków. Oznacza to, że zarówno w przypadku modelu VGG16, jak i VGG19 w kilku przypadkach nie udało się poprawnie sklasyfikować zdjęć nawierzchni bez uszkodzeń. Dla modelu VGG16 uzyskano 112, natomiast dla modelu VGG19 115 poprawnie sklasyfikowanych zdjęć reprezentujących klasę *nawierzchnia uszkodzona*. Wyniki dla obu modeli w oparciu o miary opisane w pracy [12] zostały zebrane i przedstawione w tabeli 1.

Podczas oceny stanu nawierzchni niedopuszczalne są przypadki skategoryzowania zdjęcia nawierzchni uszkodzonej jako poprawna – bez uszkodzeń (przypadek fałszywie dodatni). W celu eliminacji przypadków fałszywie dodatnich i fałszywie ujemnych możliwe jest wdrożenie algorytmu, któ-



Rys. 11. Wyniki testu dla modelu VGG16 (od lewej: funkcje straty dla zbiorów treningowego i walidacyjnego – ang. training and validation loss oraz funkcja opisująca zmianę dokładności predykcji dla zbiorów treningowego i walidacyjnego – ang. training and validation accuracy)



Rys. 12. Wyniki testu dla modelu VGG19 (od lewej: funkcje straty dla zbiorów treningowego i walidacyjnego – ang. training and validation loss oraz funkcja opisująca zmianę dokładności predykcji dla zbiorów treningowego i walidacyjnego – ang. training and validation accuracy)

Tabela 1.
Wyniki dokładności modeli VGG16 i VGG19 na zbiorze testowym

Model	Dokładność na zbiorze testowym	Prawdziwie dodatnie	Fałszywie dodatnie	Prawdziwie ujemne	Fałszywie ujemne	Suma prawdziwych	Suma fałszywych
VGG16	96,67%	120	8	112	0	232	8
VGG19	97,92%	120	5	115	0	235	5

ry kosztem odrzucenia przypadków prawdziwie dodatnich i prawdziwie ujemnych zmniejszy prawdopodobieństwo wystąpienia źle przyporządkowanych zdjęć nawierzchni. Algorytm opiera się na procentowej ocenie pewności dopasowania wejściowego zdjęcia do klasy, w tym przypadku do klasy *nawierzchnia uszkodzona* lub *nawierzchnia bez uszkodzeń*.

Podsumowanie

Opracowane modele z wykorzystaniem metod głębokiego uczenia maszynowego mogą być wykorzystane do diagnozowania nawierzchni drogowych, a funkcjonalność środowiska Google-colab znana jako *transfer learning* dostarcza narzędzi, które pozwalają korzystać z wyuczonych tym sposobem modeli w sposób niezależny od procesów trenowania i walidacji. Jest to o tyle istotne, że proces uczenia jest długi i musi się odbywać na licznych zasobach baz danych. W bibliotece PyTorch istnieją ogólnodostępne modele, których można użyć jako podstawy do realizacji własnych zadań. W artykule wykorzystano dwa z nich: VGG16 i VGG19. Zostały wytrenowane na podstawie własnej bazy danych ze zdjęciami sklasyfikowanymi jako nawierzchnie: uszkodzona i nieuszkodzona. Ostatecznie opracowane modele można wykorzystać do automatycznej kategoryzacji ortogonalnych zdjęć nawierzchni jezdni. Na podstawie przeprowadzonych badań, dla modeli VGG16 i VGG19 osiągnięto dokładność odpowiednio 96,67% oraz 97,92%. Można wnioskować, że w analizowanym przypadku większa liczba warstw w modelu wpływa pozytywnie na osiągniętą dokładność. W przypadku zdjęć nawierzchni, którym nadano status nawierzchni *nieuszkodzona*, modele VGG16 i VGG19 wykazały sto procent skuteczności, wszystkie przypadki nawierzchni nieuszkodzonej zostały sklasyfikowane poprawnie. W przypadku zdjęć nawierzchni określonej jako *uszkodzona*, przy użyciu modelu VGG16 uzyskano 8 sklasyfikowanych niepoprawnie, a przy użyciu modelu VGG19 takich sytuacji było 5.

Na niepoprawną ocenę zdjęć ma wpływ jakość danych wejściowych, na podstawie której wytrenowano i przetestowano algorytmy. Ponadto rozbieżności w zwracanych wynikach mogą wynikać z rodzaju i składu mieszanki, z której wykonano nawierzchnie. Panujących warunków atmosferycznych, które występowały podczas wykonywania materiału zdjęciowego (jakość oświetlenia, zamokniecie nawierzchni). Zdaniem autorów możliwości i potencjał programów opartych na metodach uczenia maszynowego przy zastosowaniu odpowiedniego podejścia są znacznie większe niż przedstawione w artykule. Możliwe jest opracowanie programu, który nie tylko rozróżniałby nawierzchnie na dwie klasy: uszkodzona i nieuszkodzona, ale również definiowałby rodzaj uszkodzenia i stopień szkodliwości.

Bibliografia

[1] W. Eckstein i C. Steger, The Halcon Vision System: An Example for Flexible Software Architecture, Technical Committe of Image Processing Applications, 2021, s. 1-6.
[2] K. Ciomcia, *Czym jest deep learning i sieci neuronowe*, https://bulldogjob.pl/articles/1136-czym-jest-deep-learning-i-sieci-neuronowe, 2020.
[3] Torch Contributors, PYTORCH DOCUMENTATION, TORCHVISION.TRANSFORMS, URL: https://pytorch.org/docs/stable/torchvision/transforms.html, (dostęp: czerwiec 2021).
[4] Zuzanna Jóźwiak-Ostach, *Diagnostyka nawierzchni drogowej przy zastosowaniu metod głębokiego uczenia maszynowego*, praca mgr pod kierunkiem A. Pożaryckiego, Poznań 2021
[5] A. Dey, Machine Learning Algorithms: A Review, (IJCSIT) International Journal of Computer Science and Information, 7(3), 2016, s. 1174-1179.
[6] M. Radzikowski, „Drogownictwo" 5/2020, *Stan techniczny nawierzchni dróg krajowych na koniec 2019 roku*.
[7] GDDKiA., Załącznik B, Instrukcja do części analitycznej — zasady oceny, klasyfikacje, Warszawa, Maj 2019.
[8] D. Ho i in., Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules, (ICML) International Conference on Machine Learning, 2019.
[9] A. Kayid, Y. Khaled i M. Elmahdy, Performance ofCPUs/GPUs-for Deep Learning workloads, The German University in Cairo 10.13140/RG.2.2.22603.54563, 2018, s. 1-2.
[10] Konkwistyka i Komunikacja, https://kognitywistyka.uwb.edu.pl/component/k2/item/406-sieci-neuronowe, 2019.
[11] G. Kumar i P. Bhatia, A Detailed Review of Feature Extraction in Image Processing Systems, Fourth international conference on advanced computing & communication technologies, 2014.
[12] B. Shmueli, Matthews Correlation Coefficient is The Best Classification Metric You've Never Heard Of,URL: https://towardsdatascience.com/the-best-classification-metric-youve-never-heard-of-the-matthews-correlation-coefficient-3bf50a2f3e9a, (dostęp: maj 2021).
[13] K. Simonyan i A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556, 2015.
[14] M. Stewart, Simple Introduction to Convolutional Neural Networks, URL: https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac, (dostęp: kwiecien 2021).
[15] S. Sumit, A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way,URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, (dostęp: maj 2020).
[16] J. Walsh i in., Deep Learning vs. Traditional Computer Vision, Computer Vision Conference at Las Vegas, 2019, s. 601-605.