**Katedra**

**Podstaw**

**Konstrukcji**

**Maszyn**

**Wydział**

**Mechaniczny**

**Technologiczny**

**Politechnika**

**Śląska**

**Projektowanie układów i struktur inteligentnych**

**Raport końcowy**

**Rok akademicki:** 2023/24

**Temat zadania:** Detekcja i rozpoznanie znaku ustąp pierwszeństwa z wykorzystaniem sieci neuronowych

**Studenci w sekcji:** Kowalska Natalia, Suchta Paweł

**Kierunek:** Oprogramowanie w Automotive dla współczesnej mobilności - Cyfryzacja w Automotive

**Grupa:** Ustąp pierwszeństwa

**Data opracowania:** 2024-01-28

# Opis zadania projektowego

Celem projektu było stworzenie sieci neuronowej, która na przesłanym zdjęciu jest w stanie wykryć zadany znak drogowy. Znakiem, który został przydzielony do zespołu był znak ustąp pierwszeństwa (A-7).​

Działanie sieci zostanie sprawdzone poprzez użycie dowolnego zdjęcia ze znakami drogowymi. Na zdjęciu ze znakiem A-7 sieć zaznacza prostokąt wokół wykrytego znaku, natomiast pozostałe znaki nie zostają zaznaczone. W przypadku zdjęcia bez znaku A-7, na zdjęciu nie są zaznaczane żadne znaki.

Wspomniana sieć neuronowa powinna zostać utworzona w jednym z dwóch języków programowania – Matlab bądź Python.

# Podstawy teoretyczne

## Podstawy sieci neuronowych

Sieci neuronowe to rodzaj modeli uczenia maszynowego inspirowanych biologicznym mózgiem. Składają się z połączonych sztucznych neuronów, które przetwarzają dane, ucząc się na podstawie doświadczenia i poprawiając swoje działania w miarę otrzymywania nowych informacji. Oto kilka podstawowych pojęć związanych z sieciami neuronowymi:

**Neuron:** Podstawowa jednostka sieci neuronowej, która przyjmuje wejście, przetwarza je i generuje wyjście. W modelu matematycznym jest to funkcja aktywacji, która agreguje wagi wejściowe z danymi i stosuje funkcję aktywacji, aby wygenerować wyjście.

**Warstwy:** Neurony w sieci są zorganizowane w warstwy. Wiele sieci ma co najmniej trzy typowe warstwy: warstwę wejściową, warstwy ukryte (które przetwarzają dane) i warstwę wyjściową.

**Wagi:** Wagi określają siłę połączenia między neuronami. Są one modyfikowane podczas uczenia, aby sieć mogła lepiej dopasować się do danych treningowych.

**Funkcja aktywacji:** Jest to funkcja stosowana wewnątrz neuronów, która decyduje, czy neuron powinien zostać aktywowany czy nie. Popularne funkcje aktywacji to np. funkcja sigmoidalna, funkcja ReLU (Rectified Linear Unit) czy tangens hiperboliczny.

**Uczenie:** Proces, podczas którego sieć neuronowa dostosowuje swoje wagi w celu minimalizacji błędu predykcji na danych treningowych. Może to być nadzorowane (z nauczycielem) lub nienadzorowane (bez nauczyciela).

**Funkcja straty:** Określa ona różnicę między prognozowanymi wartościami a rzeczywistymi wartościami. Celem jest minimalizacja funkcji straty podczas procesu uczenia.

**Algorytmy optymalizacyjne:** Są to metody używane do dostosowywania wag sieci neuronowej podczas procesu uczenia. Popularne algorytmy to np. stochastyczny spadek gradientu (SGD), Adam, RMSprop.

**Sieci neuronowe konwolucyjne (CNN):** Specjalne rodzaje sieci neuronowych, które są często stosowane w zadaniach przetwarzania obrazów. Wykorzystują one warstwy konwolucyjne, które mogą efektywnie ekstrahować cechy z obrazów.

**Sieci neuronowe rekurencyjne (RNN):** Innego rodzaju sieci neuronowe, które są efektywne w przetwarzaniu danych sekwencyjnych, takich jak język naturalny czy sygnały czasowe. RNN mają pamięć wewnętrzną i mogą przechowywać informacje o poprzednich krokach czasowych.

To tylko kilka podstawowych pojęć związanych z sieciami neuronowymi. Istnieje wiele bardziej zaawansowanych technik i koncepcji, które mogą być stosowane w różnych rodzajach sieci neuronowych, w zależności od konkretnego zadania.

## Architektury sieci neuronowych stosowane w zadaniach przetwarzania obrazów

W zadaniach przetwarzania obrazów, różne architektury sieci neuronowych zostały opracowane i stosowane w zależności od konkretnej aplikacji i wymagań. Oto kilka najpopularniejszych architektur stosowanych w zadaniach przetwarzania obrazów:

• **Sieci neuronowe konwolucyjne (CNN)**

CNN są najczęściej stosowane w zadaniach związanych z przetwarzaniem obrazów.Zawierają warstwy konwolucyjne, które skanują obraz wejściowy za pomocą zestawu filtrów, aby wyodrębnić cechy lokalne. Te cechy są następnie łączone w warstwach grupujących, aby zwiększyć lokalność i zmniejszyć wymiarowość.

CNN są często stosowane w zadaniach klasyfikacji obrazów, detekcji obiektów, segmentacji semantycznej itp.

• **Residual Neural Networks (ResNet)**

ResNet to rodzaj CNN wprowadzający bloki resztkowe, które zawierają skrótowe połączenia między warstwami. Te skrótowe połączenia pomagają w zwalczaniu zjawiska zanikającego gradientu podczas trenowania bardzo głębokich sieci. ResNet jest wysoce skuteczny w klasyfikacji obrazów i innych zadaniach przetwarzania obrazów.

* **Inception Networks (GoogLeNet)**

GoogLeNet wprowadza moduły Inception, które wykonują różne operacje konwolucji na obrazie wejściowym i łączą wyniki. Dzięki temu możliwe jest wydobycie cech o różnych skalach i orientacjach bez konieczności wykorzystywania dużych filtrów konwolucyjnych. GoogLeNet jest znany ze swojej skuteczności i stosunkowo niskiego zużycia zasobów obliczeniowych.

## Metody uczenia maszynowego stosowane w rozpoznawaniu obiektów na obrazach

W rozpoznawaniu obiektów na obrazach stosuje się wiele metod uczenia maszynowego, z których wiele opiera się na głębokich sieciach neuronowych. Na przestrzeni lat metody te sukcesywnie się udoskonalały, przyspieszając szybkość swojego działania. Głównymi metodami uczenia w obszarze rozpoznawania obiektów na obrazach są:

* **R-CNN (Region-based Convolutional Neural Networks)**

R-CNN to metoda detekcji obiektów, która dzieli obraz na regiony zainteresowania (ROI), a następnie stosuje do nich CNN w celu wyodrębnienia cech. Następnie klasyfikuje każdy region, aby określić, czy zawiera on obiekt, oraz przeprowadza lokalizację obiektu za pomocą regresji. R-CNN był przełomem w detekcji obiektów, ale miał pewne wady, takie jak wolne przetwarzanie z powodu konieczności oddzielnej ekstrakcji cech dla każdego propozycji regionu.

* **Fast R-CNN i Faster R-CNN**

Fast R-CNN i Faster R-CNN to ulepszone wersje R-CNN, które poprawiają prędkość i wydajność poprzez integrację procesu generowania regionów zainteresowania z samym modelem CNN.

W Fast R-CNN regiony cechowe były mapowane na cechy uzyskane z jednej wspólnej konwolucyjnej warstwy. To rozwiązanie przyspieszyło proces uczenia i wnioskowania, ponieważ eliminuje konieczność wielokrotnego przetwarzania całego obrazu dla każdej propozycji regionu.

Faster R-CNN poszedł krok dalej i zastąpił ręcznie zaprojektowane metody generowania propozycji regionów, takie jak Selective Search, przez sieci neuronowe. Wprowadzono nową architekturę, zwana Region Proposal Network (RPN), która generuje propozycje regionów na podstawie cech obliczonych w trakcie przetwarzania obrazu. Dzięki RPN Faster R-CNN może generować propozycje regionów w sposób bardziej dokładny i efektywny, co prowadzi do lepszych wyników detekcji obiektów.

Oba podejścia, Fast R-CNN i Faster R-CNN, stały się kluczowymi kamieniami milowymi w dziedzinie detekcji obiektów, przynosząc znaczące usprawnienia w wydajności i dokładności detekcji obiektów w obrazach.

* **YOLO (You Only Look Once)**

W przeciwieństwie do tradycyjnych metod, które wymagają wielu przebiegów sieci neuronowej dla każdego obszaru obrazu, YOLO wykonuje detekcję jednorazowo na całym obrazie. Dzięki temu osiąga wysoką prędkość przetwarzania. Obraz jest podzielony na siatkę o stałym rozmiarze. Każdy komórkowy segment siatki jest odpowiedzialny za przewidywanie obiektów, które się w nim znajdują. Każda komórka siatki przewiduje określoną liczbę ramek otaczających (bounding boxes) oraz odpowiednie prawdopodobieństwo istnienia obiektu wewnątrz tych ramek. Dodatkowo, dla każdej ramki YOLO przewiduje klasy obiektów.

Cały model YOLO jest uczony w sposób end-to-end, co oznacza, że jest trenowany na podstawie funkcji straty, która uwzględnia zarówno błędy lokalizacji obiektów, jak i błędy klasyfikacji. YOLO cieszy się popularnością ze względu na swoją szybkość i skuteczność, co sprawia, że jest często wykorzystywany w aplikacjach wymagających detekcji obiektów w czasie rzeczywistym, takich jak samochody autonomiczne, systemy monitorowania bezpieczeństwa i wiele innych.

# Rodzaje opracowanych modeli uczenia.

Do rozwiązania zadania wykrywania znaku pierwszeństwa na obrazie opracowane zostały dwa niezależne modele.

1. Model pierwszy, który wykrywa znak na zadanym obrazie, zaznacza go prostokątem i zwraca prawdopodobieństwo tego, że wykryty znak jest znakiem pierwszeństwa.
2. Model drugi, który wyłącznie klasyfikuje znak z przesłanego do sieci zdjęcia. Innymi słowy model zwraca informację czy obraz, na którym znajduje się wyłącznie jeden znak, jest znakiem ustąp pierwszeństwa.

Powodem istnienia dwóch niezależnych modeli była chęć sprawdzenia różnych podejść do rozwiązania problemu.

Pierwszym pomysłem było opracowanie modelu, który zaznaczałby wszystkie znaki na zadanym obrazie. Wykryty znak następnie byłby przesyłany do modelu drugiego, który poddawałby go klasyfikacji. Jak się później okazało, wytrenowanie sieci, której zadaniem byłaby detekcja dowolnego znaku na obrazie, była zbyt czasochłonna i niemal niemożliwa do wykonania z dostępną mocą obliczeniową posiadanych komputerów.

Ostatecznie pierwszy model wykrywa i klasyfikuje zadany znak na obrazie, który może posiadać wiele innych znaków bądź innych obiektów.

Drugi model klasyfikuje obiekt na obrazie, który posiada wyłączeni jeden, odpowiednio przycięty znak.

Wszystkie modele zostały utworzone w środowisku programowania MATLAB.

## Pierwszy model detekcji i klasyfikacji YOLO

Pierwszy model został utworzony za pomocą dostępnego w Matlabie algorytmu YOLOv2. Wytrenowany model jest w stanie rozpoznać wiele znaków na zdjęciu z szerokim kontekstem, z wieloma różnymi obiektami.

Podjęta została próba wykorzystania algorytmu YOLOv4, który jest nowszy i teoretycznie powinien być lepszy, jednakże wytrenowana sieć nie była w stanie wykryć znaku na żadnym obrazie.

### Opis zbioru danych

Pierwszą bardzo istotną czynnością podczas projektowania nowej sieci neuronowej jest opracowanie zbioru danych, który zostanie wykorzystany do procesu uczenia, walidacji i testowania modelu. Odpowiedni format danych, ich szczegółowość są niezwykle istotne dla uczenia sieci neuronowej. Nieodpowiednio dobrane dane mogą sprawić, że proces uczenia będzie niewydajny, bądź będzie powodował, że końcowe wyniki modelu będą niesatysfakcjonujące.

Wszystkie obrazy do zbioru danych zostały uzyskane z platformy StreetView firmy Google. Zdjęcia znaków pierwszeństwa zostały pobrane z ulic z różnych części Polski, z różnych środowisk tak, aby model mógł lepiej generalizować wiedzę. Do zbioru danych dodane zostały zdjęcia, na których jednocześnie znajduje się wiele znaków ustąpienia pierwszeństwa. Ostatecznie stworzony zbiór danych składał się z 90 różnych zdjęć.

Obraz zawierający na wolnym powietrzu, znak, roślina, niebo

Opis wygenerowany automatycznie

Rys. 1 Przykład zdjęcia ze zbioru danych

Obraz zawierający tekst, na wolnym powietrzu, pojazd, Pojazd lądowy

Opis wygenerowany automatycznie

Rys. 2 Przykład zdjęcia ze zbioru danych

Po doborze odpowiednich zdjęć następnym etapem było utworzenie informacji, które zawierały współrzędne lokalizacji znaku pierwszeństwa na każdym ze zdjęć ze zbioru. Do tego celu użyte zostało narzędzie Image Labeler dostępne w środowisku Matlab.

Obraz zawierający na wolnym powietrzu, budynek, Pojazd lądowy, samochód

Opis wygenerowany automatycznie

Rys. 3 Zaznaczenie znaków pierwszeństwa

Na Rys.3 widoczny jest zrzut ekranu z aplikacji Image Labeler, które wykorzystywane jest do wygenerowania koordynat zaznaczonych prostokątów.

Obraz zawierający tekst, zrzut ekranu, Czcionka, biały

Opis wygenerowany automatycznie

Rys. 4 Koordynaty prostokątów z zaznaczonymi znakami

Rys. 4 przedstawia wygenerowaną tabelę, która posiada informacje o dokładnych pozycjach znaków pierwszeństwa na każdym z obrazów ze zbioru danych. Tabela ta przekazywana jest później do funkcji, która rozpoczyna proces trenowania sieci neuronowej.

### Obróbka zbioru danych

Gdy zbiór danych został wstępnie przygotowany, następnym krokiem było podzieleniu zbioru na trzy osobne podzbiory: zbiór treningowy, walidacyjny i testowy.

Podział danych wyglądał następująco: spośród 90 zdjęć, 60% zostało przypisanych do zbioru treningowego, 10% do zbioru walidacyjnego, a pozostałe 40% do zbioru testowego.

Zdjęcia ze zbiorów treningowych i walidacyjnych zostały dodatkowo poddane procesowi augmentacji. Proces augmentacji danych polega na zastosowaniu różnorodnych transformacji do istniejących danych w celu generowania nowych przykładów danych. Jest to powszechna praktyka w uczeniu maszynowym, szczególnie w dziedzinach przetwarzania obrazów i analizy danych. Augmentacja danych ma na celu zwiększenie różnorodności danych treningowych, co może poprawić zdolność generalizacji modelu oraz zmniejszyć ryzyko przeuczenia.

Przykładem zastosowanych technik augmentacji danych były:

* Obracanie obrazów, które może pomóc w uczeniu modelu na obrazach z różnych perspektyw.
* Regulacja jasności, kontrastu i innych parametrów kolorystycznych, która może pomóc w uczeniu modelu na obrazach w różnych warunkach oświetleniowych.
* Dodawanie losowych szumów do obrazów, które może pomóc w uczeniu modelu na obrazach o różnej czystości.

Obraz zawierający niebo, drzewo, chmura, na wolnym powietrzu

Opis wygenerowany automatycznie

Rys. 5 Cztery obrazy uzyskane po augmentacji jednego zdjęcia

### Zdefiniowane struktury sieci neuronowej

Struktura sieci neuronowej, która została użyta do opracowania modelu rozpoznającego znaki, nie została opracowana samodzielnie. Aby przyśpieszyć prace nad projektem użyta została gotowa, głęboka sieć neuronowa ResNet50. Jak sama nazwa wskazuje siec sumarycznie posiada 50 warstw. ResNet50 jest wstępnie nauczona rozpoznawania pewnych obiektów na obrazie, co niewątpliwie jest bardzo dużym atutem.

Obraz zawierający tekst, diagram, linia, zrzut ekranu

Opis wygenerowany automatycznie

Rys. 6 Początkowe warstwy sieci ResNet50

Do powyższej sieci zastosowana została technika Transfer Learning. Wykorzystanie transfer learningu pozwala na ponowne wykorzystanie wytrenowanych wag modelu ResNet50, które zostały nauczane na dużych zbiorach danych. Dzięki temu można uniknąć konieczności uczenia modelu od zera na niewielkim zbiorze danych, co może prowadzić do przeuczenia.

Dzięki transfer learningowi sieci ResNet50 można osiągnąć skuteczne rezultaty nawet przy niewielkiej ilości dostępnych danych treningowych. Model ResNet50 jest w stanie generalizować cechy na tyle dobrze, że nawet ograniczony zbiór danych treningowych może zapewnić satysfakcjonujące wyniki. Sieci ResNet są znane z ich zdolności do uczenia się bardziej ogólnych cech obrazów, co przekłada się na lepszą zdolność generalizacji. Dzięki temu model może efektywnie radzić sobie z różnorodnymi typami znaków na obrazach, niezależnie od ich kształtu, wielkości czy położenia.

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Rys. 7 Końcowe warstwy sieci ResNet50

Aby zastosować transfer learning do sieci ResNet50 dodana została dodatkowa warstwa wyjściowa, która po dobraniu odpowiednich wag w procesie uczenia, będzie w stanie rozpoznawać wybrane znaki. Rys.7 przedstawia strukturę warstw końcowych sieci.

### Proces uczenia

Do procesu uczenia zastosowane zostały domyśle parametry algorytmu Yolov2. Jednymi modyfikacjami było ustawienie następujących wartości:

MiniBatchSize – 16

InitialLearnRate - 1e-3,

MaxEpochs – 300

Liczba epok jest wysoka, ponieważ model potrzebuje wielu przejść po zbiorze danych, aby zmniejszyć funkcje straty poniżej 1 punktu. Po przejściu kilku epok rezultaty sieci są walidowane i oceniane.

Po około 4 godzinach treningu, model jest gotowy do testowania na zbiorze testowym.

### Proces testowania

Po wytrenowaniu modelu należało zweryfikować jakość jego predykcji. Testowanie modelu polega na iteracji zdjęć ze zbioru testowego i dla każdego takiego zdjęcia uruchomieniu modelu, który wykryje i sklasyfikuje znaki. Zbiór testowy posiada informacje na temat faktycznego położenia znaku A7 na każdym z obrazów. Faktycznie położenie znaku i położenie otrzymane z modelu są porównywane i ich różnice są wyświetlane na wykresie.

Obraz zawierający linia, diagram, Wykres, Równolegle

Opis wygenerowany automatycznie

Rys. 8 Wykreślona precyzja predykcji

Na Rys.8 można zauważyć, że średnia precyzja modelu została zdefiniowana na poziome 63%. Nie jest to wynik spektakularny, ale trzeba mieć świadomość tego, że model wykrywa znak na obrazie z szerokim kontekstem, zaznacza na nim wybrany znak w postaci Bounding Box-ów i tak otrzymane współrzędne są porównywane z oczekiwanymi współrzędnymi ze zbioru testowego. Nie możliwe jest, aby model w 100% zwracał takie same współrzędne rozpoznanych znaków jak te, zdefiniowane w zbiorze testowym. Wynik testów jest zadowalający i potwierdza fakt, że model jest w stanie wykrywać znaki na obrazach.

### Eksperymenty i wyniki

Do wytrenowanego modelu przesłane zostały zdjęcia otrzymane od prowadzącego przedmiotu, tak aby sprawdzić, jak rodzi sobie z obrazami, których nie widział wcześniej. Poniżej znajdują się wyniki eksperymentów.

Obraz zawierający tekst, na wolnym powietrzu, niebo, drzewo

Opis wygenerowany automatycznie

Rys. 9 Eksperyment, gdy na zdjęciu jest znak A7

Obraz zawierający na wolnym powietrzu, niebo, Znak drogowy, chmura

Opis wygenerowany automatycznie

Rys. 10 Eksperyment, gdy na zdjęciu nie ma znaku A7

Obraz zawierający na wolnym powietrzu, Znak drogowy, drzewo, znak

Opis wygenerowany automatycznie

Rys. 11 Eksperyment, gdy na zdjęciu jest znak podobny do znaku ostrzegawczego

Obraz zawierający pomarańcza/pomarańczowy, żółty

Opis wygenerowany automatycznie

Rys. 12 Eksperyment bliskiego przybliżenia znaku A7

Można zauważyć, że model nie był w stanie wykryć znaku z bliższej perspektywy. W zbiorze danych nie było zdjęć z takim przybliżeniem, być może była to przyczyna nierozpoznania znaku.

Obraz zawierający na wolnym powietrzu, niebo, chmura, samochód

Opis wygenerowany automatycznie

Rys. 13 Eksperyment, gdy na zdjęciu jest wiele znaków

## Drugi model klasyfikujący

W modelu drugim sieć neuronowa to typowa architektura CNN, która została dostosowana do zadania klasyfikacji polskich znaków drogowych poprzez odpowiednie dostosowanie liczby warstw, ich rozmiaru i hiperparametrów treningowych.

### Zdefiniowane struktury sieci neuronowej

Stworzono sieć neuronową składającą się z dwunastu warstw oraz zoptymalizowano zbiór danych poprzez ujednolicenie ich rozmiaru i losowe przetasowanie zdjęć, co miało na celu uzyskanie jak najlepszej ogólnej zdolności uczenia się modelu.

Obraz zawierający tekst, paragon

Opis wygenerowany automatycznie

Rys. 14 Struktura sieci neuronowej

### Opis zbioru danych

Zbiór obejmował wyłącznie zdjęcia znaków drogowych o niewielkiej rozdzielczości, w których 90% obrazu stanowił znak drogowy. Na każdym obrazie znajdował się wyłącznie jeden znak drogowy. Każdy znak drogowy był umieszczony w folderze, którego nazwa oznaczała znak drogowy. Zbiór pochodził ze strony Kaggle (https://www.kaggle.com/code/kattat/polish-traffic-signs-classification). ​

Obraz zawierający trójkąt

Opis wygenerowany automatycznie

Rys. 15 Przykład zdjęć ze zbioru danych

Zbiór składał się ze zdjęć znaków drogowych o niskiej rozdzielczości. Opracowano sieć o dwunastu warstwach. Dane wejściowe (zdjęcia) posiadały różne rozmiary, dlatego aby możliwe było ich użycie, zbiór danych zmodyfikowano tak, że ujednolicono ich rozmiar. Dodatkowo zdjęcia były umieszczone w osobnych folderach, dlatego też konieczne było wymieszanie zdjęć przed uczeniem, aby sieć neuronowa uczyła się na wszystkich rodzajach znaków.

### Obróbka zbioru danych

Zdecydowano się na zastosowanie augmentacji danych. W kontekście rodzaju obiektów, jakie należało rozpoznać, skupiono się głównie na przesunięciu i skosie obrazu. Uniknięto modyfikacji obrotu lub odbicia, ponieważ mogłoby to prowadzić do powstania zupełnie innych znaków drogowych, co mogłoby być mylące dla modelu.

Augmentacja została zastosowana zarówno do zbioru treningowego, jak i walidacyjnego, przy zachowaniu niezmienionych parametrach uczenia. Mimo to, po przeprowadzeniu procesu uczenia na nowych danych, zaobserwowano pogorszenie wyników zarówno dla danych treningowych, jak i walidacyjnych, w porównaniu do poprzedniego procesu uczenia przy tej samej liczbie epok. W odpowiedzi na to zwiększono liczbę epok do 40, co doprowadziło do osiągnięcia podobnych rezultatów zarówno pod względem dokładności, jak i funkcji straty, jak w przypadku wcześniejszego procesu uczenia.

Mimo zastosowania augmentacji danych, nie zaobserwowano poprawy działania sieci. Znaki drogowe to bardzo specyficzna kategoria obrazów, które wymagają precyzyjnego rozróżniania, jednak augmentacja danych nie tylko nie przyniosła oczekiwanych efektów, ale również wpłynęła negatywnie na proces uczenia. W rezultacie uczenie stało się bardziej czasochłonne, nie przynosząc jednocześnie zauważalnego wzrostu dokładności. Z tego względu postanowiono, że optymalną siecią do rozpoznawania obrazów będzie sieć korzystająca z oryginalnego zbioru danych bez zastosowania augmentacji.

### Proces uczenia

Jako metodę trenującą wybrano metodę stochastyczną - Adaptive moment estimation (Adam). Liczbę epok ustalono na 10. Wynik walidacji sprawdzano co 130 iteracji. Całkowita liczba iteracji dla uczenia wynosiła 1300. Po skończonym uczeniu funkcja została zaprojektowana do zwrócenia modelu o najlepszej funkcji straty dla zbioru walidacyjnego. Pozostałe parametry zostały ustalone na domyślne.

Obraz zawierający tekst, Czcionka, zrzut ekranu, algebra

Opis wygenerowany automatycznie

Rys. 16 Parametry uczenia

Podczas treningu sieci obserwowano postęp poprzez analizę wyników modelu w odpowiedzi na zbiór walidacyjny. Zauważono, że początkowy etap uczenia, czyli pierwsza epoka, przyniósł widoczną poprawę wyników, jednak kolejne epoki nie przyniosły istotnej zmiany. Analiza porównawcza wyników na zbiorze treningowym i walidacyjnym sugeruje, że model doświadczał niewielkiego nadmiernego dopasowania (overfittingu). Jedną z potencjalnych przyczyn tego zjawiska mogła być ograniczona wielkość zbioru danych, co wpłynęło na zdolność modelu do ogólnego generalizowania wzorców.

Obraz zawierający tekst, zrzut ekranu, linia, diagram

Opis wygenerowany automatycznie

Rys. 17 Wykres uczenia modelu

# Wnioski

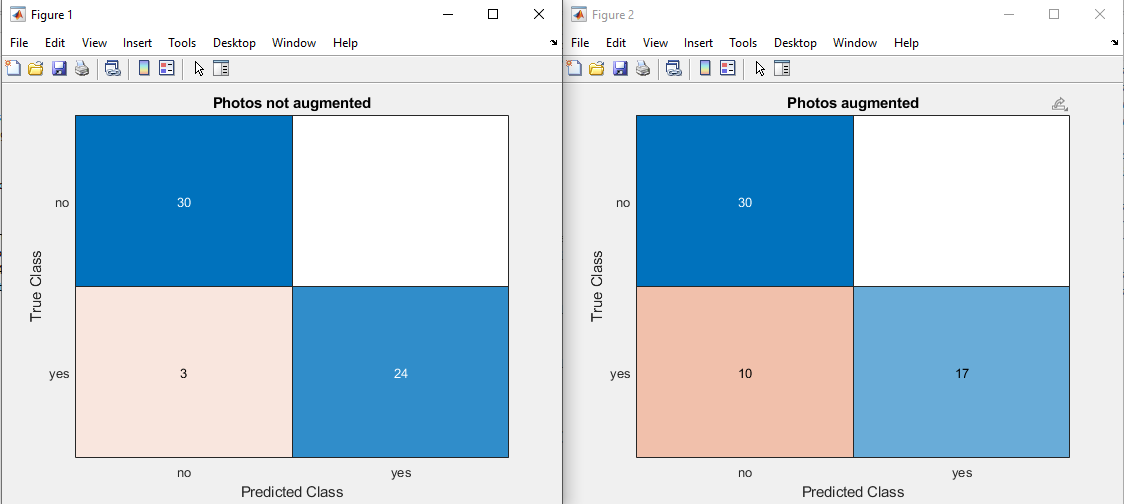
## Model Yolo

Wykorzystanie transfer learningu do sieci ResNet50 w kontekście rozpoznawania znaków na obrazie pozwoliło na skuteczne wykorzystanie potencjału tej zaawansowanej architektury do zadania rozpoznawania obiektów, jednocześnie minimalizując potrzebę dużej ilości danych treningowych i redukując ryzyko przeuczenia. Model pierwszy był w stanie zidentyfikować i sklasyfikować znaki pierwszeństwa na obrazach dostarczonych przez prowadzącego zajęcia. Algorytm YOLO zastosowany w tym modelu jest w stanie zaznaczać wiele znaków na jednym obrazie i podawać pewność jego wyników. Gdy na obrazie jest wiele obiektów mogących być rozpoznanych, algorytm ten jest wysoce wydajny i skuteczny.

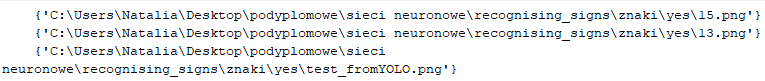
## Model drugi

Pomimo swojej bardzo prostej budowy (tj. 12 warstw), opracowana sieć sprawdziła się do rozpoznawania znaków. Ponieważ klasyfikacja obejmowała wszystkie rodzaje znaków, a nie tylko klasyfikację zero-jedynkową (znak A-7 oraz znak różny od A-7) można przyjąć, że potencjał sieci został niewykorzystany. Warto byłoby zastanowić się nad stworzeniem drugiego zbioru danych, dla którego występowałyby tylko dwie grupy obrazów – znaki A-7 oraz znaki inne niż A-7 i obie grupy byłyby jednakowo liczne. Jednakże w opisywanej sieci dobre efekty uczenia były osiągnięte głównie przez duże zbiory danych, co uniemożliwia ręczne przygotowanie własnego zbioru uczącego dla dwóch klasyfikatorów.

Aby udowodnić poprawność uczenia się i generalizacji sieci przygotowano zestaw testowy, dla którego została wykonana predykcja w oparciu o wyuczony model sieci. Ze względu na przygotowanie modelu w oparciu o augmentację zdjęć oraz teoretycznie zbieżne wyniki uczenia po 40 epokach dla sieci z danymi po augmentacji konieczne było wykonanie macierzy pomyłek dla obu modeli – bez augmentacji oraz po augmentacji. Oprócz tego dla sieci bez augmentacji (dla której oczekiwano lepszych rezultatów) przygotowano kod pokazujący, które zdjęcia (wraz z pełnymi ścieżkami) zostały nieprawidłowo sklasyfikowane.



Rys. 18 Macierze pomyłek modeli bez augmentacji danych (po lewej) oraz po augmentacji danych (po prawej)



Rys. 19 Ścieżki dostępu (nazwy) zdjęć nieprawidłowo sklasyfikowanych przez model bez augmentacji

Na podstawie ścieżek widać, że błędy dotyczyły trzech zdjęć. Dwa pierwsze zdjęcia zostały w pełni nieprawidłowo sklasyfikowane, ponieważ sprawdzane zdjęcia zostały odpowiednio przygotowane, tj. znak obejmował około 90% zdjęcia, a samo zdjęcie było niewielkich rozmiarów. W przypadku zdjęcia ostatniego trudno określić błędną klasyfikację jako błąd uczenia sieci per se. Ostatnie zdjęcie bowiem było zdjęciem użytym w sieci YOLO, a więc nie było to zdjęcie właściwe dla tego modelu. Oprócz tego, po zmniejszeniu zdjęcia znak osiągnął bardzo niewielki rozmiar, rzędu kilkunastu pikseli, a więc nie mógł być realnie traktowany jako obiekt do klasyfikacji.

Podsumowując, sieć dwunastowarstwowa poradziła sobie bardzo dobrze z klasyfikacją otrzymanego znaku drogowego. Należy również nadmienić, że jeden z obrazów testowych (otrzymany od Prowadzącego), przedstawiał znak nieistniejący, tzn. znak niebędący prawnie uznanym za znak drogowy. Mimo tego sieć prawidłowo sklasyfikowała go jako znak inny niż A-7. W związku z tym można jasno przyznać, że model sieci jest skuteczny, a jego uczenie jest bardzo szybkie.