Sprawozdanie

Ćwiczenie 4
Prosta sieć konwolucyjna

Piotr Swirkaitis

246753

Badanie wpływu dodania warstw konwolucyjnych do sieci neuronowej

Wprowadzenie

W sprawozdaniu przedstawiono wyniki sprawozdania, badającego wpływ dodania warstw konwolucyjnych na skuteczność sieci neuronowej. W ramach badań zaimplementowano model sieci konwolucyjnej, którego strukturę przedstawiono na poniższym rysunku. Do implementacji użyto pakietu keras z biblioteki Tensorflow.

```
model = keras.Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    Conv2D(64, (3, 3), activation='relu'),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Rysunek 1 Konfiguracja modelu sieci konwolucyjnej

W ramach sieci konwolucyjnej dodano 2 warstwy konwolucyjne o następujących rozmiarach filtra: 32 oraz 64. Wielkość ramki w warstwie konwolucyjnej ustalono na 3x3, a jako funkcje aktywacji wybrano funkcję ReLU w obu warstwach. Na koniec dodano dwie warstwy w pełni połączone:

- 128 neuronów, aktywacja ReLU
- 10 neuronów, aktywacja softmax

Wyniki otrzymane podczas treningu sieci konwolucyjnej, porównano z wynikami sieci w pełni połączonej. Zaimplementowano sieć w pełni połączoną, składającą się z 3 warstw w pełni połączonych :

- 128 neuronów, aktywacja ReLU
- 64 neuronu, aktywacja ReLU
- 10 neuronów, aktywacja softmax

Zaimplementowany model przedstawiono na poniższym zrzucie ekranu.

```
model = keras.Sequential()
model.add(Flatten(input_shape=(IMG_HEIGHT,IMG_WIDTH)))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Rysunek 2 Konfiguracja modelu sieci w pełni połączonej

W zaimplementowanych modelach użyto następującej konfiguracji:

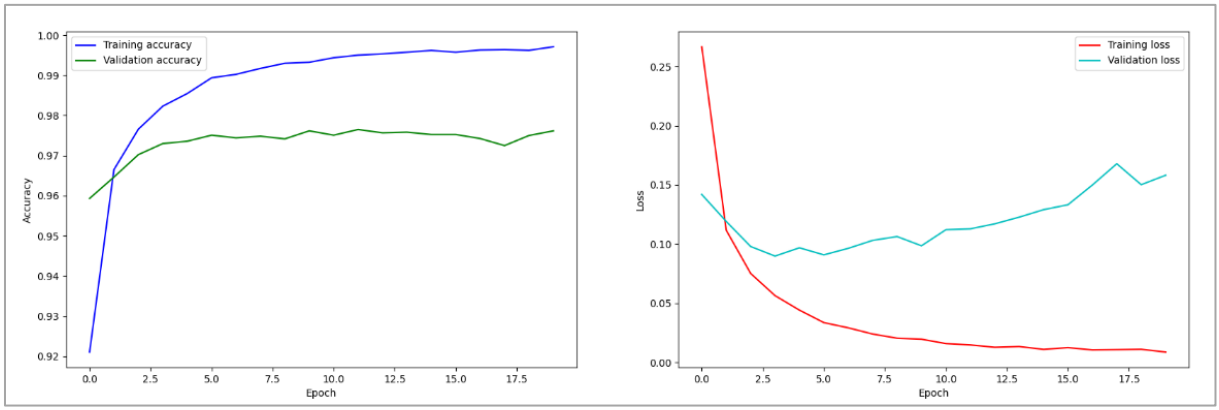
- Optymalizator: *Adam*
- Funkcja straty: *Categorical crossentropy*
- Metryka: *Accuracy*

Wszystkie badania zostały przeprowadzone, w zakresie 20 epok treningu.

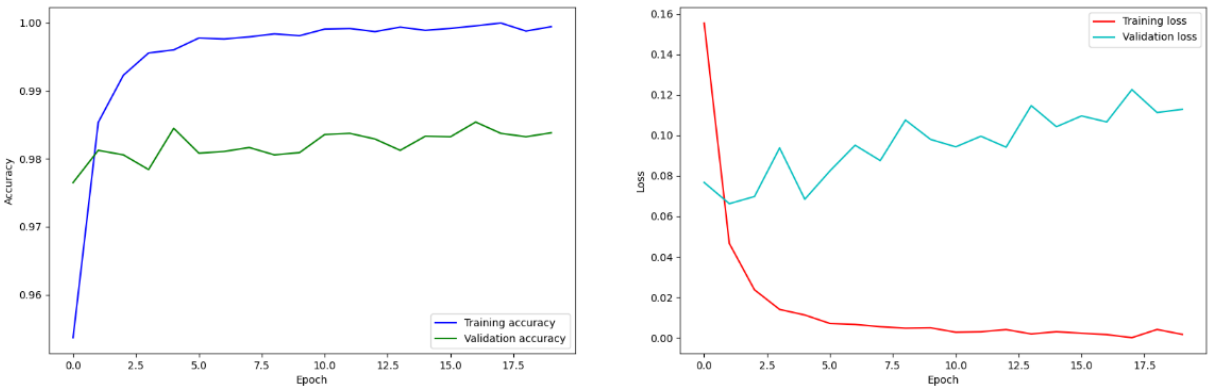
Badanie skuteczności sieci konwolucyjnej w porównaniu z siecią w pełni połączoną

W poniższym badaniu zbadano skuteczność sieci z warstwami konwolucyjnymi, porównując ją ze skutecznością sieci w pełni połączonej. W poniższej tabeli przedstawiono uzyskane wyniki badań

Wybrany model	Train accuracy	Train loss	Test accuracy	Test loss	Czas obliczeń jednej epoki
Conv	0.9995	0.0017	0.9838	0.0946	40s
Fully connected network	0.9971	0.0088	0.9753	0.141	4s



Rysunek 3 Skuteczność sieci w pełni połączonej



Rysunek 4 Skuteczność sieci konwolucyjnej

Wnioski

Analizując otrzymane wyniki, można stwierdzić, że obie sieci osiągnęły zbliżone wyniki. Na zbiorze treningowym obie sieci osiągnęły ponad 99% trafności predykcji. Wartość funkcji straty na zbiorze walidacyjnym wynosiła odpowiednio 0,0946 dla sieci konwolucyjnej i 0,141 dla sieci w pełni połączonej. Warto zwrócić uwagę na zachowanie funkcji straty w miarę postępowania epok treningu. Na zbiorze treningowym można zauważyć, że funkcja straty konsekwentnie maleje, natomiast na zbiorze walidacyjnym, w miarę postepowania epok zaczyna ona wzrastać. Takie zachowanie funkcji straty wskazuje na zjawisko przeuczenia modelu. Znaczącą różnice można zauważyć również w czasie obliczeń jednej epoki. W sieci w pełni połączonej czas obliczeń, wyniósł zaledwie 4s, podczas gdy w przypadku sieci konwolucyjnej było to średnio 40s.

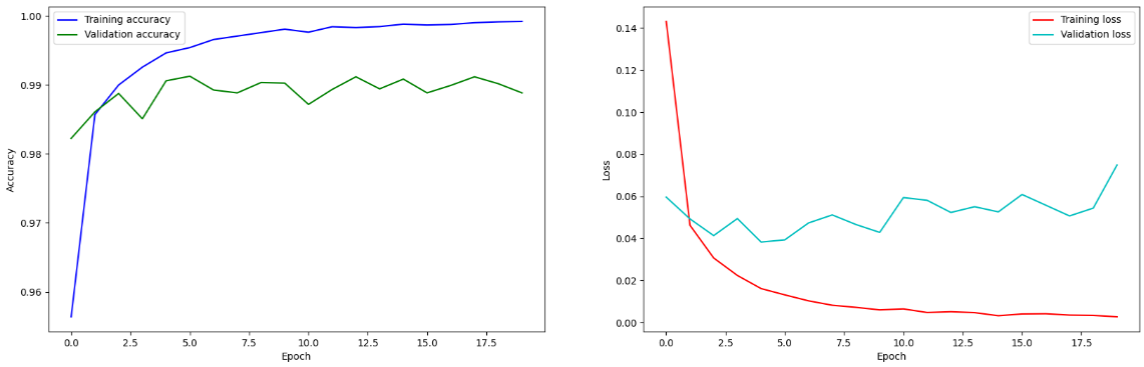
Badanie wpływu warstw MaxPooling na skuteczność sieci konwolucyjnej

Do zaimplementowanej sieci konwolucyjnej dodano dwie warstwy MaxPooling, o rozmiarze okna 2x2. Wyniki z przeprowadzonych badań porównano z wynikami sieci konwolucyjnej oraz sieci w pełni połączonej. Konfigurację modelu użytego do treningu z warstwami MaxPooling przedstawiono na poniższym rysunku.

```
model = keras.Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2), strides=2),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2), strides=2),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Rysunek 5 Konfiguracja modelu z warstwami MaxPooling

Wybrany model	Train accuracy	Train loss	Test accuracy	Test loss	Czas obliczeń jednej epoki
Conv	0.9995	0.0017	0.9838	0.0946	40s
Fully connected network	0.9971	0.0088	0.9753	0.141	4s
Conv+ MaxPooling	0.9992	0.0027	0.9901	0.0619	26s



Rysunek 6 Skuteczność sieci konwolucyjnej z warstwami MaxPooling

Wnioski

Z otrzymanych wyników można przed wszystkim wywnioskować, że użycie warstw MaxPooling znacząco zmniejsza czas obliczeń wymagany do przetworzenia jednej epoki. Czas obliczeń zmniejszył się do 26s, porównując z wcześniej otrzymaną liczbą 46s, potrzebnych na przetworzenie jednej epoki w modelu konwolucyjnym bez warstw MaxPooling. Analizując wyniki metryki accuracy, można zauważyć, że wyniki wszystkich 3 modeli oscylują w granicach 99% na zbiorze treningowym, natomiast na zbiorze walidacyjnym widać przewagę nowo zaimplementowanego modelu. Po zastosowaniu warstw MaxPooling, zauważono że wartość funkcji straty na danych walidacyjnych zmalała do 0,0619, jednakże zaobserwowana wcześniej sytuacja wzrastania wartości funkcji straty wraz z postępującymi epokami treningu, wystąpiła ponownie, co wskazuje że dodanie warstw MaxPooling nie eliminuje problemu przeuczenia sieci.

Badanie wpływu dodatkowych technik usprawniających uczenie na sieć konwolucyjną

Wprowadzenie

W drugiej części badań, zbadano wpływ stosowania kolejnych technik usprawniania procesu uczenia sieci neuronowej. Przeprowadzono następujące badania dotyczące dodawania kolejnych warstw.

- Badanie wpływu warstwy BatchNormalization na sieć konwolucyjną
- Badanie wpływu warstwy Dropout na sieć konwolucyjną

W celu usprawnienia procesu treningu sieci neuronowej zastosowano technikę augmentacji danych. Przeprowadzono następujące badania wpływu różnych rodzajów augmentacji danych

- Badanie wpływu rotacji danych wejściowych na skuteczność uczenia
- Badanie wpływu przesunięcia danych wejściowych na skuteczność uczenia
- Badanie wpływu przekrzywienia danych wejściowych na skuteczność uczenia
- Badanie augmentacji, pogarszających skuteczność uczenia

Wszystkie badania zostały przeprowadzone w zakresie 20 epok treningu.

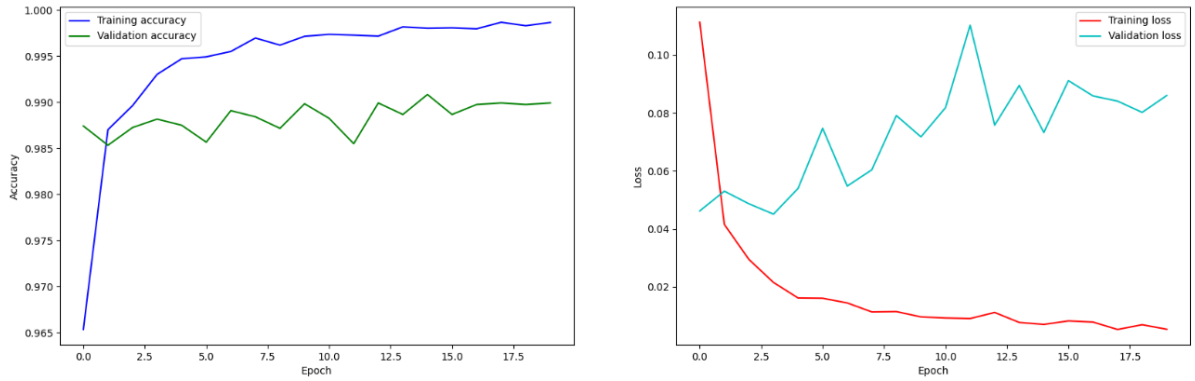
Badanie wpływu dodania warstwy BatchNormalization na skuteczność sieci konwolucyjnej

Do opracowanej w poprzednim badaniu sieci konwolucyjnej dodano warstwę BatchNormalization o domyślnych parametrach. Wyniki przeprowadzonych badań porównano z wynikami badania sieci konwolucyjnej z warstwami MaxPooling. Struktura modelu z dodaną warstwa BatchNormalization została przedstawiona na rysunku 7.

```
model = keras.Sequential([
    Conv2D(32, (3, 3), padding='same',
           activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2), strides=2),
    Conv2D(64, (3, 3), padding='same',
           activation='relu'),
    BatchNormalization(),
    MaxPooling2D((2, 2), strides=2),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Rysunek 7 Konfiguracja sieci konwolucyjnej z warstwą BatchNormalization

Wybrany model	Train accuracy	Train loss	Test accuracy	Test loss	Czas obliczeń jednej epoki
Conv+MaxPooling	0.9992	0.0027	0.9901	0.0619	26s
Conv+MaxPooling+ BatchNormalization	0.9986	0.0054	0.9896	0.0769	47s



Rysunek 8 Skuteczność sieci konwolucyjnej z warstwą BatchNorm

Wnioski

Analizując otrzymane wyniki, można zauważyć, że otrzymane rezultaty nie różnią się znacząco. Obie badane sieci osiągnęły wartość trafności predykcji na poziomie 99.9% na zbiorze testowym i około 99% trafności na zbiorze walidacyjnym. W przeprowadzonym badaniu nie stwierdzono faktu zmniejszenia się wartości funkcji straty na zbiorze walidacyjnym, natomiast zauważono, że zmienia się ona skokowo, wahając się w szerszych granicach wartości, niż funkcja straty sieci konwolucyjnej bez warstwy BatchNormalization. Spostrzeżono również, że czas przetworzenia jednej epoki się zwiększył.

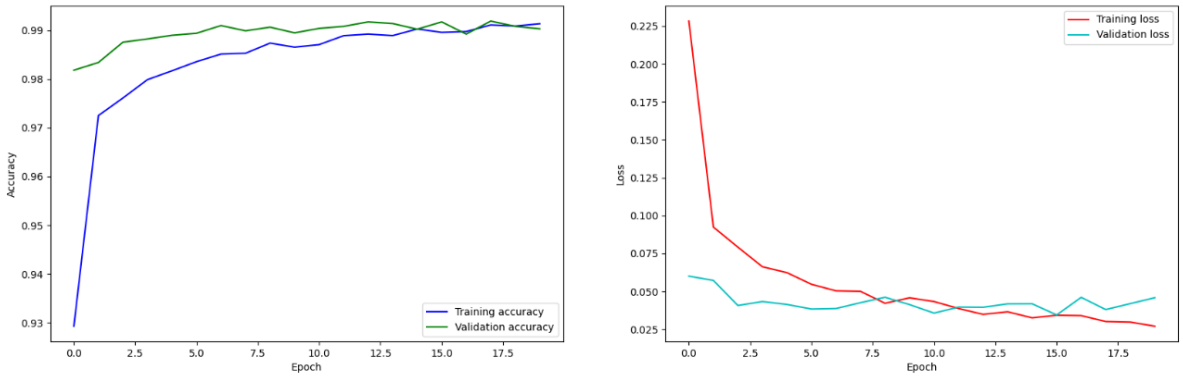
Badanie wpływu dodania warstwy Dropout na skuteczność sieci konwolucyjnej

Do opracowanej w poprzednim badaniu sieci konwolucjnej z dodaną warstwą BatchNormalization o domyślnych parametrach, dodano 3 warstwy Dropout o wartości prawdopodobieństwa wyzerowania wag równej 0,3. Wyniki przeprowadzonych badań porównano z wynikami badania sieci konwolucyjnej z warstwami MaxPooling oraz z wynikami sieci z dodaną warstwą BatchNormalization. Struktura modelu z dodanymi warstwami Dropout została przedstawiona na rysunku 9.

```
model = keras.Sequential([
    Conv2D(32, (3, 3), padding='same',
           activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2), strides=2),
    Dropout(0.3),
    Conv2D(64, (3, 3), padding='same',
           activation='relu'),
    BatchNormalization(),
    MaxPooling2D((2, 2), strides=2),
    Dropout(0.3),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(10, activation='softmax')
])
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Rysunek 9 Konfiguracja sieci z dodanymi warstwami Dropout

Wybrany model	Train accuracy	Train loss	Test accuracy	Test loss	Czas obliczeń jednej epoki
Conv+MaxPooling	0.9992	0.0027	0.9901	0.0619	26s
Conv+MaxPooling+ BatchNormalization	0.9992	0.0027	0.9901	0.0619	26s
Conv+ MaxPooling+ BatchNormalization +Dropout	0.9914	0.0271	0.9915	0.0379	51s



Rysunek 10 Skuteczność sieci konwolucjnej z warstwami Dropout

Wnioski

Otrzymane podczas badań wskazują na znacząca poprawę skuteczności uczenia, po dodaniu warstw Dropout. Sieć osiągnęła podobną wartość trafności predykcji na danych treningowych, w każdej z 3 badanych konfiguracji. Oscylowała ona w granicach 99%. Wartość trafności na zbiorze walidacyjnym zrównała się z trafnością na zbiorze trenigowym, a w początkowych epokach można zauważyć, że była ona nawet nieznacznie wyższa, co wskazuje na przewagę nowej konfiguracji modelu. Analizując wartość funkcji straty na zbiorze walidacyjnym, można zauważyć, że nie rośnie ona już wraz z postępującymi epokami, tak jak miało to miejsce w poprzednich badaniach. Wartość funkcji straty na zbiorze walidacyjnym jest bardzo zbliżona do wartości funkcji straty na zbiorze treningowym, co wskazuje, że dodanie warstw Dropout pozwoliło na wyeliminowanie problemu przeuczenia sieci.

Badanie wpływu operacji rotacji obrazu w augmentacji danych na skuteczność sieci konwolucyjnej

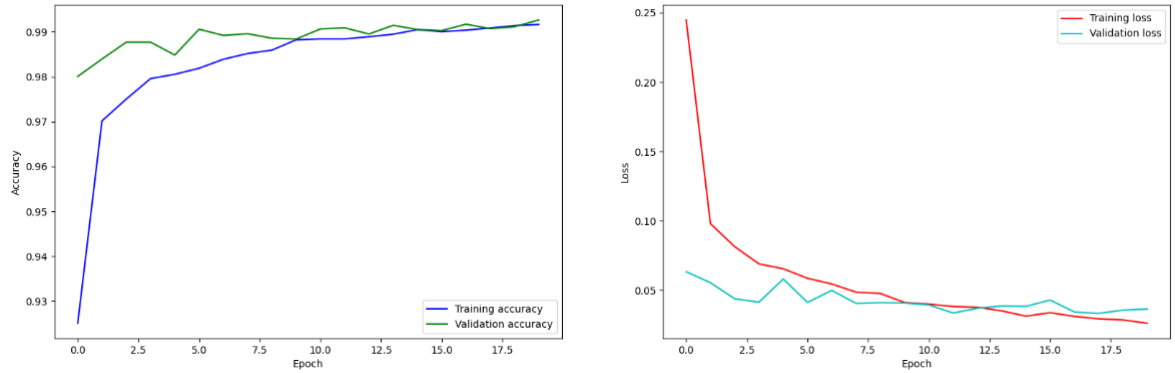
Wszelkie badania dotyczące dodania danych wejściowych poddanych augmentacji zostały przeprowadzone na sieci konwolucyjnej z warstwami MaxPooling, BatchNormalization oraz Dropout. Badania zostały przeprowadzone w zakresie 20 epok uczenia. Do zbioru danych wejściowych dodano dodatkowe 30 tys. nowych danych po transformacji, co oznacza, że wielkość zbioru danych wejściowych zwiększyła się do 90 tys. obrazków. Konfigurację sieci przedstawiono na rysunku 11.

```
model = keras.Sequential([
    Conv2D(32, (3, 3), padding='same',
           activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2), strides=2),
    Dropout(0.3),
    Conv2D(64, (3, 3), padding='same',
           activation='relu'),
    BatchNormalization(),
    MaxPooling2D((2, 2), strides=2),
    Dropout(0.3),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(10, activation='softmax')
])
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Rysunek 11 Konfiguracja sieci do badań wpływu augmentacji

W przeprowadzonym badaniu zastosowano transformację rotacji w zakresie 30 stopni w obie strony.

Wybrany model	Train accuracy	Train loss	Test accuracy	Test loss	Czas obliczeń jednej epoki
Optimal Conv	0.9914	0.0271	0.9915	0.0379	51s
Optimal Conv + Rotate Augmentation (0.3)	0.9928	0.0263	0.9927	0.0260	67s



Rysunek 12 Skuteczność sieci konwolucjnej uczonej na danych poddanych augmentacji Rotation

Wnioski

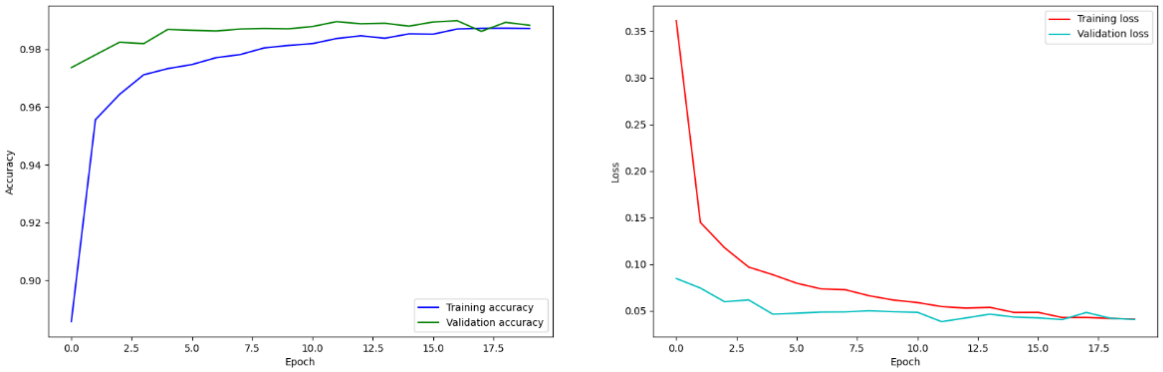
Otrzymane w badaniach wyniki są zbliżone do wyników osiągniętych przy zastosowaniu warstw Dropout w sieci konwolucyjnej. Wartości trafności predykcji na obu zbiorach osiągnęły podobne wartości, wahające się w granicach 99% trafności. Zauważono, że wartość funkcji straty na zbiorze walidacyjnym rozkładała się bardziej równomiernie wraz z postępującymi epokami. Zanotowano również spadek wartości funkcji straty z poziomu 0.0379 na 0.026. Czas przetwarzania jednej epoki zwiększył się o 16s, jednak jest to związane z większą ilością danych w zbiorze danych wejściowych

Badanie wpływu operacji przesunięcia obrazu w augmentacji danych na skuteczność sieci konwolucyjnej

Badania zostały przeprowadzone na skonfigurowanej wcześniej sieci konwolucyjnej, której konfigurację przedstawiono na rysunku 11. Do zbioru danych wejściowych dodano dodatkowe 30 tys. nowych danych po transformacji, co oznacza, że wielkość zbioru danych wejściowych zwiększyła się do 90 tys. obrazków.

W przeprowadzonym badaniu zastosowano transformację przesunięcia obrazu względem oryginału Obraz został przesunięty w zakresie 20% całkowitej wysokości oraz 20% całkowitej szerokości.

Wybrany model	Train accuracy	Train loss	Test accuracy	Test loss	Czas obliczeń jednej epoki
Optimal Conv	0.9914	0.0271	0.9915	0.0379	51s
Optimal Conv + Shift Augmentation(0.2 ,02)	0.9928	0.0263	0.9927	0.0260	63s



Rysunek 13 Skuteczność sieci konwolucjnej uczonej na danych poddanych augmentacji Shift

Wnioski

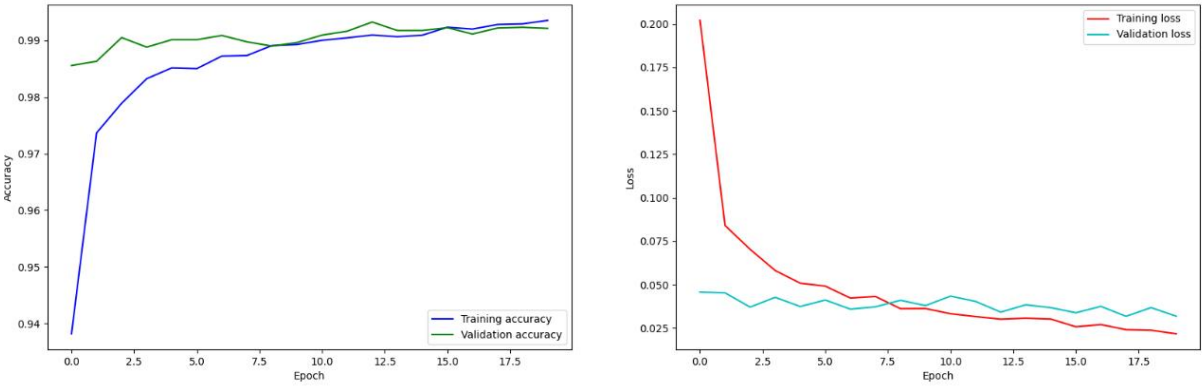
Analizując otrzymane wyniki zauważono, że sieć uczona na danych poddanych transformacji przesunięcia, osiąga znakomite rezultaty. Przedstawiona na wykresach historia uczenia pokazuje, że trafność predykcji na zbiorze walidacyjnym w początkowych epokach była wyższa niż na zbiorze treningowym, a następnie obie wartości się zrównały. Analizując wykres wartości funkcji straty można zauważyć, że funkcja straty na zbiorze walidacyjnym podlegała o wiele mniejszym wahaniom w porównaniu do wcześniej przeprowadzonych badań. W okolicach 16 epoki uczenia wartość funkcji straty na zbiorze walidacyjnym, zrównała się z wartością funkcji straty na zbiorze treningowym.

Badanie wpływu operacji przekrzywiania obrazu w augmentacji danych na skuteczność sieci konwolucyjnej

Badania zostały przeprowadzone na skonfigurowanej wcześniej sieci konwolucyjnej, której konfigurację przedstawiono na rysunku 11. Do zbioru danych wejściowych dodano dodatkowe 30 tys. nowych danych po transformacji, co oznacza, że wielkość zbioru danych wejściowych zwiększyła się do 90 tys. obrazków.

W przeprowadzonym badaniu zastosowano transformację przekrzywienia obrazu względem oryginału Obraz został przekrzywiony w zakresie intensywności równym 30 stopniom.

Wybrany model	Train accuracy	Train loss	Test accuracy	Test loss	Czas obliczeń jednej epoki
Optimal Conv	0.9914	0.0271	0.9915	0.0379	51s
Optimal Conv + Shear Augmentation(0,3)	0.9935	0.0218	0.9924	0.0241	52s



Rysunek 14 Skuteczność sieci konwolucyjnej uczonej na danych poddanych augmentacji Shear

Wnioski

Z przeprowadzonych badań można wywnioskować, że uczenie sieci konwolucyjnej na danych poddanych transformacji skrzywienia, osiąga wyniki zbliżone do wyników otrzymanych w trakcie badania wpływu transformacji przesunięcia. Można zauważyć, że wartość metryki accuracy prezentowała większą rozbieżność pomiędzy zbiorem walidacyjnym a testowym. Dodanie obrazków, zmienionych transformacją przekrzywienia wpłynęło na wartość funkcji straty na zbiorze walidacyjnym. Widoczne są większe wahania wartości w postępujących epokach, w porównaniu z badaniem danych transformowanych za pomocą przesunięcia.

Badanie parametrów augmentacji danych pogarszających skuteczność sieci konwolucyjnej

Badania zostały przeprowadzone na skonfigurowanej wcześniej sieci konwolucyjnej, której konfigurację przedstawiono na rysunku 11. Do zbioru danych wejściowych dodano dodatkowe 30 tys. nowych danych po transformacji, co oznacza, że wielkość zbioru danych wejściowych zwiększyła się do 90 tys. obrazków.

W poniższym badaniu starano się znaleźć taki rodzaj augmentacji danych, który będzie pogarszał otrzymane wyniki uczenia sieci konwolucyjnej. Przeprowadzono eksperyment, używając różnych rodzajów augmentacji i przedstawiono wyniki otrzymane z zastosowaniem kombinacji:

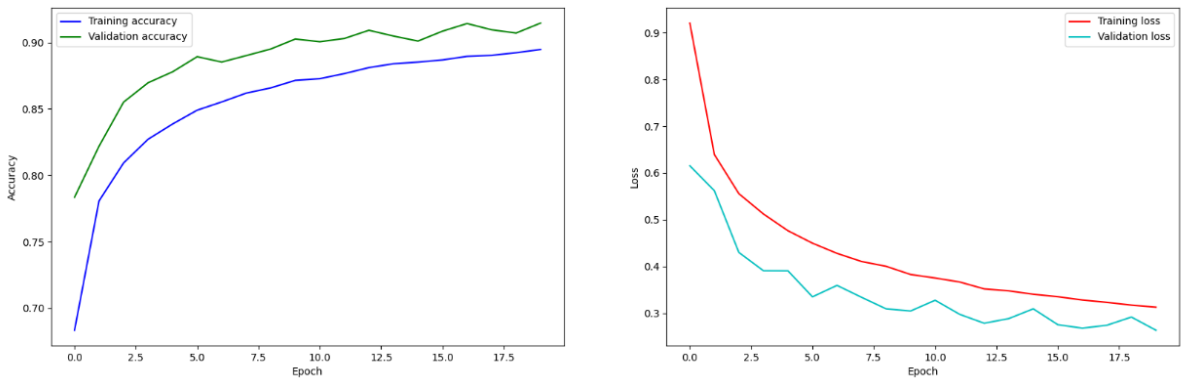
- Przybliżenie obrazu o 40%
- Odbicie obrazu w pionie i poziomie

Konfigurację generatora augmentowanych danych przedstawiono na rysunku 15.Otrzymane wyniki porównano z wynikami badań sieci konwolucyjnej z warstwami MaxPooling, Dropout oraz BatchNormalization.

```
data_generator = ImageDataGenerator(vertical_flip=True,
                                    horizontal_flip=True,
                                    zoom_range=0.4,
                                    fill_mode='nearest')
```

Rysunek 15 Konfiguracja generatora augmentowanych danych

Wybrany model	Train accuracy	Train loss	Test accuracy	Test loss	Czas obliczeń jednej epoki
Optimal Conv	0.9914	0.0271	0.9915	0.0379	51s
Optimal Conv + Worsening Augmentation	0.894	0.288	0.9874	0.0341	88s



Rysunek 16 Skuteczność sieci konwolucyjnej uczonej na danych poddanych pogarszającej augmentacji

Wnioski

Analizując otrzymane wyniki można zauważyć znaczący wpływ dodania danych poddanych pogarszającym transformacjom. Na zbiorze treningowym sieć osiągnęła wartość trafności predykcji na poziomie 89%, co wskazuje na pogorszenie trafności o 10% względem wyników sieci konwolucyjnej operującej na oryginalnym zbiorze danych wejściowych. Należy również zwrócić uwagę na wysoką wartość funkcji straty, która po 20 epokach osiągnęła ponad 30% straty. W przypadku ewaluacji modelu na zbiorze walidacyjnym, wyniki nie odbiegały znacząco od wyników sieci konwolucyjnej bez transformowanych danych. Wynika to z faktu, że sieć zawierała proporcje 1:2 danych transformowanych do danych oryginalnych, przez co nadal możliwe było osiągnięcie wysokich wyników trafności predykcji analizując zbiór walidacyjny. Również wartość funkcji straty osiągnęła niski wynik na poziomie 3%, co jest zadowalającym rezultatem. Analizując historię uczenia, można zauważyć, że wartości metryki accuracy oraz wartości funkcji straty prezentowały dosyć wysoką rozbieżność pomiędzy zbiorem treningowym, a walidacyjnym. W miarę postępowania kolejnych epok, obie wartości utrzymywały się na podobnym poziomie, nie rejestrując znaczących wahań, pomijając początkowe 5 epok. Obie linie zostały stosunkowo szybko wypłaszczone, co wskazuje, że model nie potrafił zwiększyć trafności predykcji w kolejnych epokach.

Obrazki poddane pogarszającej augmentacji znacząco odbiegały od oryginalnych danych. Analizując dane wejściowe które przedstawiają cyfry, można zauważyć że operacja obrócenia według osi pionowej i poziomej znacząco zmienia obraz, który sieć ma przeanalizować. Również operacja przybliżenia może znacząco zaciemniać faktyczny obraz cyfry. Na podstawie przeprowadzonych badań można wysnuć wniosek, że dobranie pewnych technik augmentacji z niewłaściwymi parametrami może generować zupełnie rozbieżny rezultat w odniesieniu do oczekiwanego. Poprzez niewłaściwe dobranie parametrów augmentacji, dane wejściowe mogą pogarszać proces uczenia modelu zamiast go usprawniać.