Neuronowa diagnostyka COVID-19 na podstawie zdjęć rentgenowskich.

Krystian Kosek
Dariusz Lesiecki
Krzysztof Łakomy

1. Dataset

Na potrzeby trenowania i testowania modelu utworzono dwa zbiory danych:

Pierwszy zbiór:
- Dane trenujące: 3648
    - COVID 1216
    - NORMAL 1216
    - PNEUMONIA 1216
- Dane testujące: 409
    - COVID 144
    - NORMAL 136
    - PNEUMONIA 129

Drugi zbiór:
- Dane trenujące: 9964
    - COVID 3304
    - NORMAL 3312
    - PNEUMONIA 3348
- Dane testujące: 928
    - COVID 312
    - NORMAL 312
    - PNEUMONIA 304

Dane trenujące zostały przetasowane, a następnie podzielone w proporcji 80/20 na dane trenujące oraz walidujące, rozmiar zdjęć został przeskalowany do 180x180 pikseli. W celu skonfigurowania zbioru danych pod kątem wydajności skorzystano z metody **AUTOTUNE**.

2. Model

Założeniem projektu było opracowanie sieci neuronowej do diagnostyki COVID-19 na podstawie zdjęć rentgenowskich. W przedstawionym rozwiązaniu wykorzystano wstępnie wytrenowany model ***MobileNet V2*** opracowanego w Google. Model ten jest wstępnie przeszkolony na zestawie danych składającym się z 1,4 mln obrazów i 1000 klas.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Obraz 1. Architektura modelu MobileNet V2

Najpierw utworzono model bazowy MobileNet V2 bez górnej warstwy, co jest konieczne do wyodrębnienia cech.

```
# Create base model from pretrained MobileNet V2
base_model = tf.keras.applications.MobileNetV2(input_shape=input_shape,
                                               include_top=False,
                                               weights='imagenet')
```

Obraz 2. Tworzenie modelu bazowego MobileNetV2

Do modelu, który będzie tworzony, będą wprowadzane różnorodności do danych treningowych za pomocą **data_augmentation**, pomaga to zmniejszyć nadmierne dopasowanie oraz lepiej uogólniać model.

```
data_augmentation = tf.keras.Sequential([
    tf.keras.layers.experimental.preprocessing.RandomFlip('horizontal'),
    tf.keras.layers.experimental.preprocessing.RandomRotation(0.2),
    tf.keras.layers.experimental.preprocessing.RandomZoom(0.1),
])
```

Obraz 3. Modyfikacja zbioru danych

Następnie utworzono nowy model oparty na bazowym:

```
inputs = tf.keras.Input(shape=input_shape)
x = data_augmentation(inputs)
x = tf.keras.applications.mobilenet_v2.preprocess_input(x)
x = base_model(x, training=False)
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dropout(0.2)(x)
outputs = tf.keras.layers.Dense(3)(x)
model = tf.keras.Model(inputs, outputs)
```

Obraz 4. Tworzenie docelowego modelu

Przy pomocy utworzonej wcześniej metody **data_augmentation()** wprowadzono różnorodność do zbioru trenującego, a następnie przeskalowano wartości pikseli danych treningowych do wymaganego rozmiaru, za pomocą metody **preprocess_input()**. Następnie bazowy pretrenowany model ustawiono w tryb wnioskowania używając, co jest szczególnie ważne, ponieważ będziemy dostrajać ten model dla naszych danych. Kolejno przy pomocy **GlobalAveragePooling2D()** przekonwertowano funkcje na wektory. Następnie **tf.keras.layers.Dropout(0.2)** reguluje dane, losowo usuwając 20% jednostek wyjściowych z nałożonej warstwy podczas procesu uczenia. Ponieważ mamy 3 klasy ‘COVID’, ‘NORMAL’ oraz ‘PNEUMONIA’, oczekiwano na wyjściu predykcji tablicy z 3 wartościami określającymi dopasowanie do każdej z tych klas, w tym celu korzystano z **Dense(3)**.

3. Proces uczenia

Podczas uczenia utworzonego modelu wykorzystano typowy proces uczenia transferowego polegający na podzieleniu uczenia na dwa etapy. W pierwszym etapie wykonano 10 epok z warstwą konwolucyjną działającą w trybie wnioskowania, wówczas podczas uczenia stan zamrożonej warstwy nie będzie aktualizowany. Przed trenowaniem skompilowano model z optimizerem implementującym algorytm adama z wartością learning_rate ustawioną na 1e-4. Ponieważ istnieją trzy klasy użyto obliczanie strat metodą krzyżową między etykietami i przewidywaniami.

```
model.compile(optimizer=tf.keras.optimizers.Adam(lr=1e-4),
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Obraz 5. Kompilacja modelu

W tym etapie wykonaliśmy 10 epok dzięki którym model osiągnął zbieżność z danymi. Następnie celem dodatkowego precyzyjnego dostrojenia modelu, odblokowano górne warstwy modelu podstawowego oraz zablokowano dolne warstwy tak aby nie można było ich trenować, następnie na nowo skompilowano model.

```
base_model.trainable = True
fine_tune_at = 100
for layer in base_model.layers[:fine_tune_at]:
    layer.trainable = False

model.compile(optimizer=tf.keras.optimizers.RMSprop(lr=1e-5),
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Obraz 6: Wykonanie wyżej opisanych kroków

Na tym etapie użyto bardzo niskiego współczynnika uczenia się, w celu minimalizacji prawdopodobieństwa przetrenowania sieci oraz skorzystano z innego optymalizatora ***RMSprop***. W tym etapie wykonano 10 epok, dzięki którym udało się podnieść dokładność predykcji o kilka punktów procentowych.

4. Wyniki

Otrzymane wyniki na dwóch zbiorach danych prezentują się następująco:

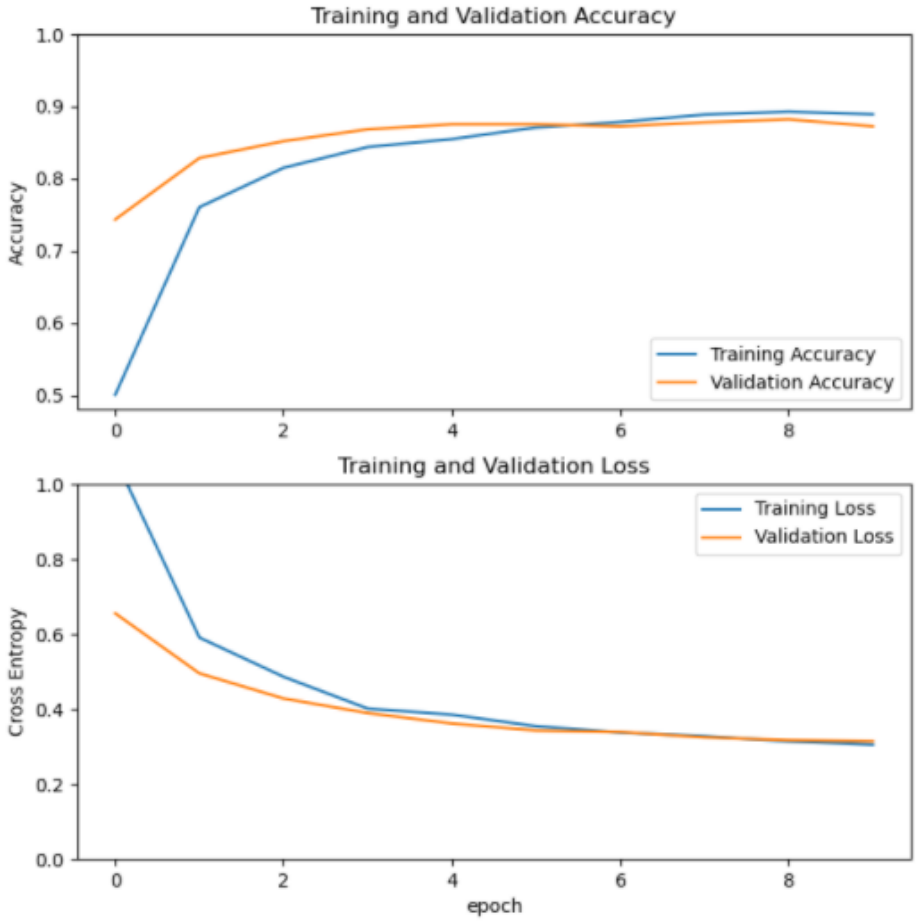
Zbiór 1:

Wstępne dopasowanie na zbiorze testowym dla przygotowanego modelu przed zastosowaniem uczenia wyniosło **34%**.

Pierwszy etap uczenia polegał na wyłączeniu górnej warstwy.
Po dziesięciu epokach uzyskaliśmy **90.5%** skuteczności dopasowania na zbiorze testującym.

Na obrazie poniżej możemy zaobserwować ciągły, choć przy każdej kolejnej epoce mniej zauważalny wzrost dokładności trenowania. Dla pierwszych epok występują znaczne przyrosty, dla późniejszych mniejsze. Zarówno skuteczność dla treningu jak i walidacji dla końcowych epok zachowuje się prawie liniowo, występują nieznaczne wahania wartości.

Strata krzyżowa entropii zachowuje się podobnie lecz w przeciwnym kierunku, dla pierwszych epok znacznie spada, dla późniejszych spadki są coraz mniej zauważalne.



Obraz 7. Wykresy przedstawiające skuteczność oraz stratę krzyżową entropii po pierwszych 10 epokach

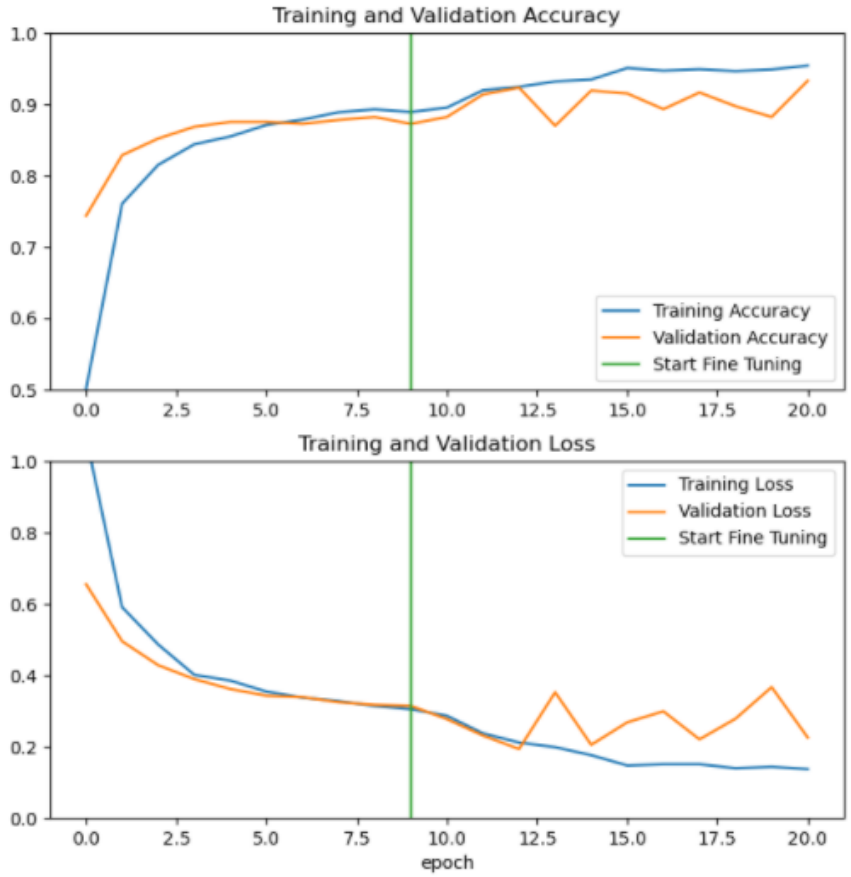
Drugi etap uczenie odbywał się już z włączoną górną warstwą oraz trwał kolejne dziesięć epok.

Końcowa skuteczność dopasowania dla danych testowych wyniosła **93.4%.**

Daje to wzrost skuteczności o około **3 punkty procentowe**.

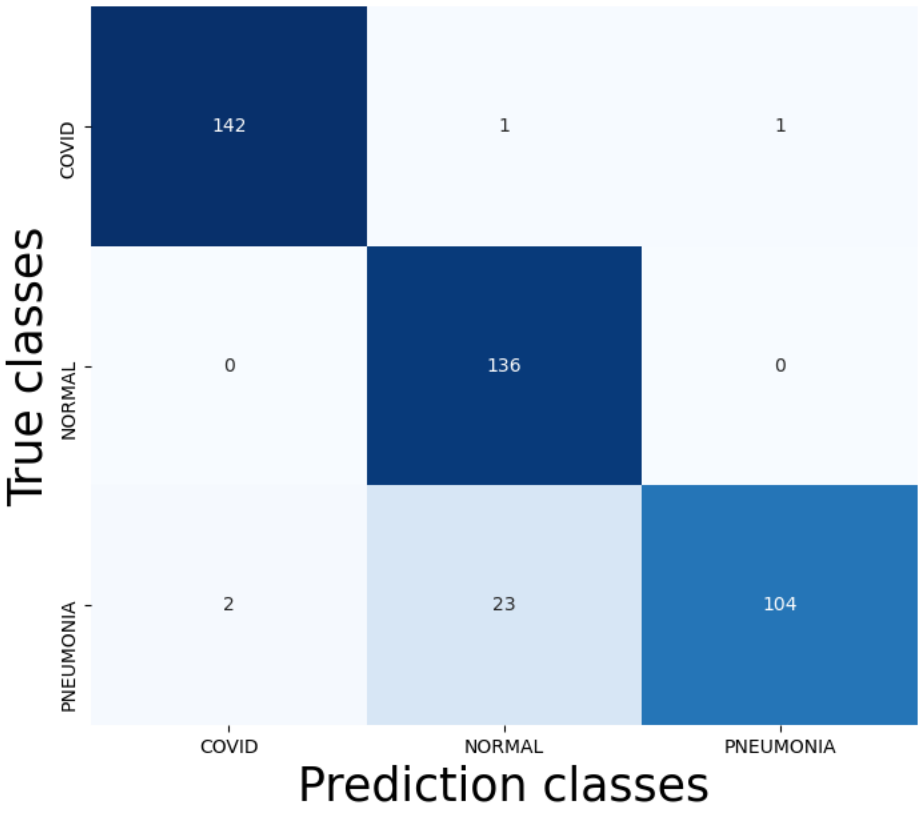
Na wykresie poniżej można zaobserwować lekki wzrost dokładności trenowania. Z kolei dokładność walidacji zachowuje się w bliżej nieokreślony sposób, z epoki na epokę wyraźnie zmieniają się jej wartości.

Strata krzyżowa entropii zachowuje się bardzo podobnie. Treningowa nieznacznie zmniejsza się co epokę, a walidacji znacząco zmienia wartość.



Obraz 8. Wykresy przedstawiające skuteczność oraz stratę krzyżową entropii po wszystkich 20 epokach

Na obrazie Obraz 9. znajduje się macierz pomyłek dla tej próby. Dla wykorzystanego tutaj zbioru możemy zauważyć, że przypadki dla COVID-19 zostały rozpoznane prawie wszystkie za wyjątkiem dwóch, dla człowieka zdrowego wszystkie zdjęcia zostały rozpoznane poprawnie, z kolei największy problem nasza sieć miała z przypadkami zapalenia płuc gdzie w sumie aż 25 nie zostało rozpoznanych odpowiednio.



Obraz 9. Macierz pomyłek

Covid precision: 98,6%	Covid recall: 98,6%
Normal precision: 85%	Normal recall: 100%
Pneumonia precision: 99%	Pneumonia recall: 80,6%

Zbiór 2:

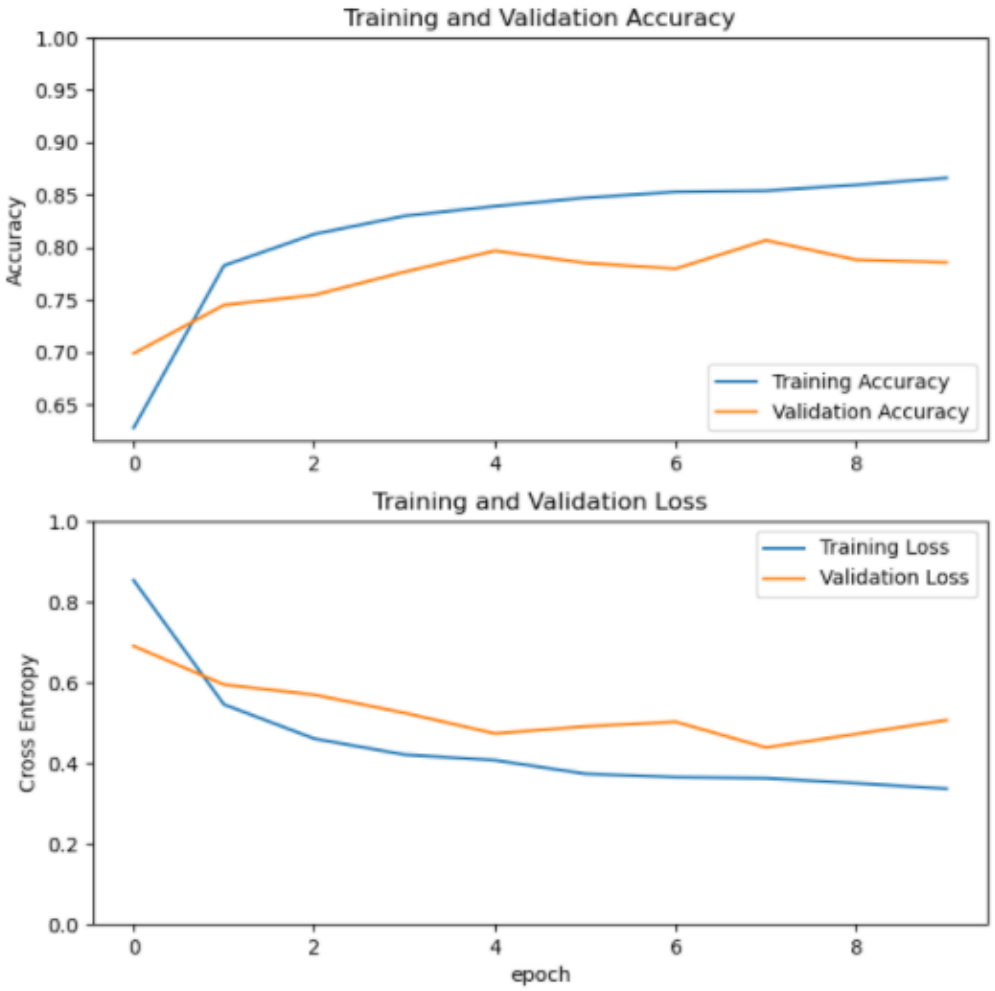
Wstępne dopasowanie na zbiorze testowym dla przygotowanego modelu przed zastosowaniem uczenia wyniosło **31%**.

Po pierwszych dziesięciu epokach z wyłączoną górną warstwą uzyskaliśmy **76.7%** skuteczności dopasowania na zbiorze testującym.

Jak możemy zaobserwować na rysunku poniżej.

Zarówno dokładność treningu jak i walidacji miało tendencję wzrostową, jednak dokładność walidacji była trochę niższa od treningowej. Dla treningu wzrastała spokojniej, a dla walidacji pojawiały się dosyć rozbieżne wyniki.

Strata krzyżowa entropii zachowywała się bardzo podobnie. Dla treningu konsekwentnie malała, a dla walidacji malała choć jej wartości były bardziej porozrzucane.



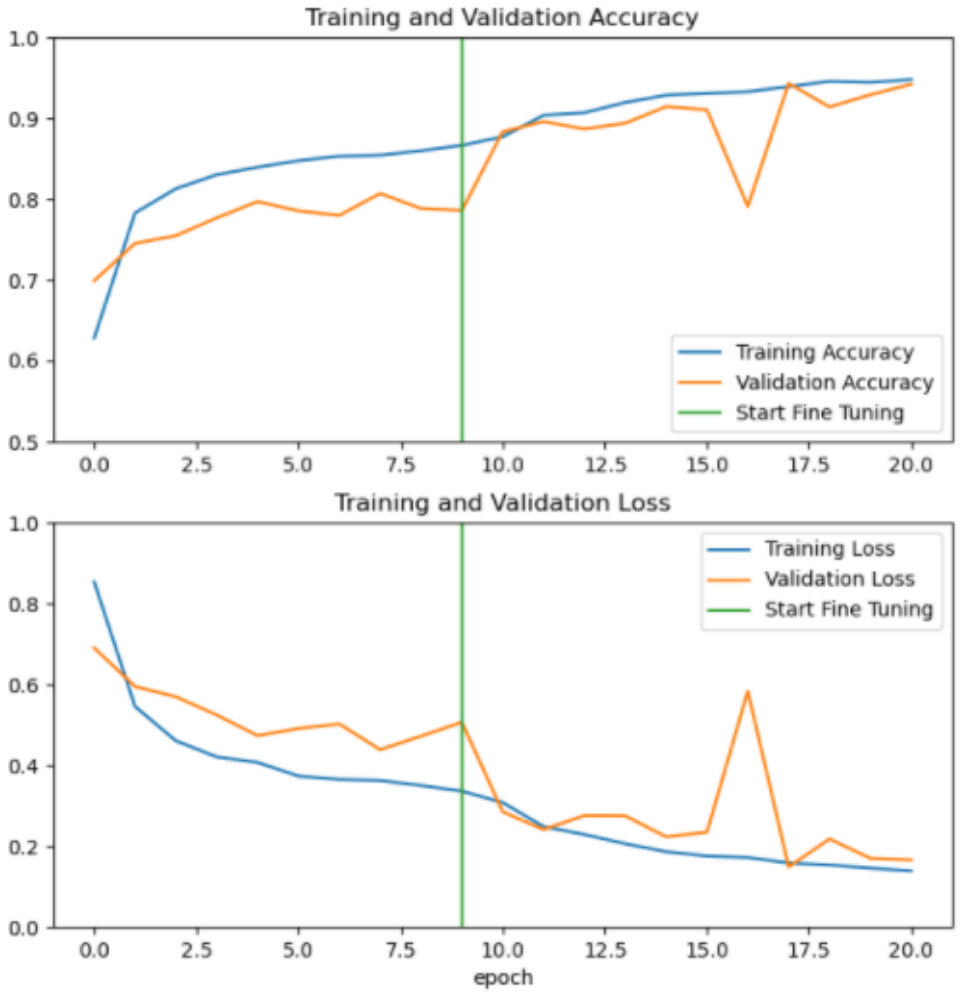
Obraz 10. Wykresy przedstawiające skuteczność oraz stratę krzyżową entropii po pierwszych 10 epokach

Końcowa skuteczność dopasowania dla danych testowych po sumarycznie 20 epokach z włączoną górną warstwą wyniosła **95%.**

Daje to kolosalny wzrost skuteczności o około **20 punktów procentowych.**.

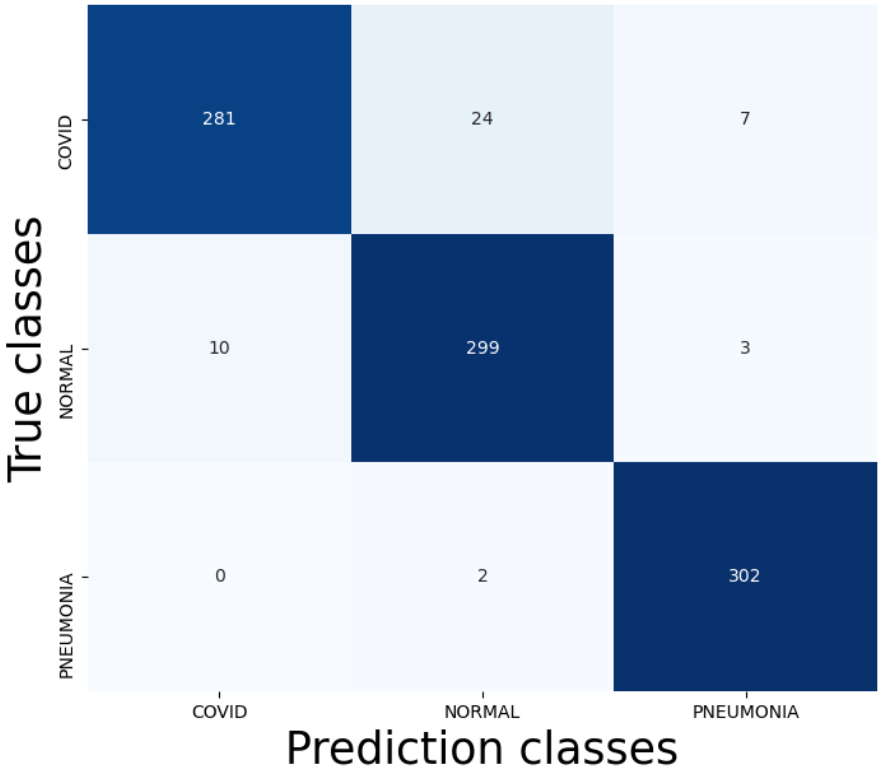
Na wykresach poniżej widać, że dokładność treningu rosła cały czas dosyć znacząco, z kolei dokładność walidacji rosła chociaż jej wartości znacznie się wahały.

Strata krzyżowa entropii zachowywała się podobnie. Konsekwentnie malała z epoki na epokę dla treningu, a dla walidacji malała chociaż zdarzały się epoki, w których występowały wahania.



Obraz 11. Wykresy przedstawiające skuteczność oraz stratę krzyżową entropii po wszystkich 20 epokach

Na obrazie Obraz 12. znajduje się macierz pomyłek dla tej próby. Dla wykorzystanego tutaj zbioru danych możemy zauważyć, że najlepiej wykrywane jest zapalenie płuc, dla którego tylko 2 przypadki nie zostały rozpoznane odpowiednio, dla osoby zdrowej trochę więcej nie trafiło tam gdzie trzeba, jednak jest to nadal bardzo dobry wynik.



Obraz 12. Macierz pomyłek

Covid precision: 96,6% Covid recall: 90,1%
Normal precision: 92% Normal recall: 95,8%
Pneumonia precision: 96.8% Pneumonia recall: 99.3%

5. Porównanie wyników

Dla obu naszych zbiorów danych uzyskaliśmy bardzo dobre wyniki odpowiednio **93.4%** oraz **95%**. Dla mniejszego zbioru 1 tak jak się można było spodziewać uzyskaliśmy mniejszą dokładność niż dla bardziej licznego zbioru 2. Co ciekawe jednak na pierwszych etapach uczenia oraz dla mniejszej liczby epok dominował zbiór 1 dla którego szybko uzyskaliśmy wysoką skuteczność, z kolei dla zbioru 2 dokładność wraz z kolejnymi epokami treningu rosła. Dobrze obrazują nam to wyniki testowe po połowie epok, w których dla zbioru 1 uzyskaliśmy aż **90.5%** co jest imponującym wynikiem przy **76.7%** dla zbioru 2.

Może to wynikać z większej różnorodności wśród liczniejszego zbioru zdjęć jakim jest zbiór 2 co na początku stanowiło przeszkodę w szybkiej nauce ale koniec końców doprowadziło do lepszego wyniku testowego.

Na tle wyników z publikacji otrzymane przez nas wyniki wypadają bardzo dobrze, ponieważ oba nieznacznie przewyższają otrzymane tam **93.3%**.

6. Literatura

- https://www.tensorflow.org/api_docs
- https://keras.io/api/