Sprawozdanie

z projektu z przedmiotu Sieci Neuronowe i DL

Agnieszka Tracz
Piotr Lachowicz
Marcin Krzemiński

czerwiec 2024

W ramach naszego projektu przygotowaliśmy analizę zbioru danych dotyczących choroby Alzheimera Dane pochodzą ze strony https://www.kaggle.com/datasets/sachinkumar413/alzheimer-mri-dataset/data.

Opis zbioru danych:

Nasz zbiór danych jest złożony z 6400 obserwacji. Są to kolorowe zdjęcia głowy (w formacie .jpg) wykonane za pomocą rezonansu magnetycznego. Dane zostały zebrane z różnych szpitali oraz ogólnie dostępnych repozytoriów.

Mamy 4 klasy o następujących licznościach:

- Mild Demented (łagodna demencja) — 896 obrazków;
- Moderate Demented (umiarkowana demencja) — 64 obrazki;
- Non Demented (brak demencji) — 3200 obrazków ;
- Very Mild Demented (bardzo łagodna demencja) — 2240 obrazków.

Cel projektu:

Celem naszej analizy będzie stworzenie jak najdokładniejszych modeli predykcyjnych, które na podstawie zdjęcia z rezonansu magnetycznego, będą stwierdzać, czy pacjent ma demencję, a jeżeli tak, to jaki jest jej stopień.

Wstępna analiza danych:

Zestaw danych, który wykorzystaliśmy w niniejszej analizie, był już w wstępnie przygotowany do użycia, co oznacza, że nie zawierał żadnych wadliwych ani niekompletnych obrazów. Dzięki starannemu przygotowaniu i weryfikacji, które są standardem w przypadku danych medycznych, mogliśmy ograniczyć etap wstępnej analizy.

Przygotowanie danych do analizy:

Na początku zamieniamy zdjęcia na tensory wymiaru 3x128x128 (zdjęcie 128x128 pikseli, 3 kanały RGB). Dokonujemy także normalizacji, dzięki której każdy piksel jest reprezentowany przez wartość z przedziału [0,1].

Następnie dokonujemy podziału zbioru danych na trzy części: treningową, testową i walidacyjną, przy czym stosujemy proporcje 60%, 20% i 20%, dbając o zachowanie równowagi w proporcjach przynależności do poszczególnych klas.

Właściwa analiza danych:

Do predykcji wykorzystamy sieci CNN. Do budowania każdego z modeli wykorzystamy funkcję straty nn.CrossEntropyLoss() (entropia krzyżowa).

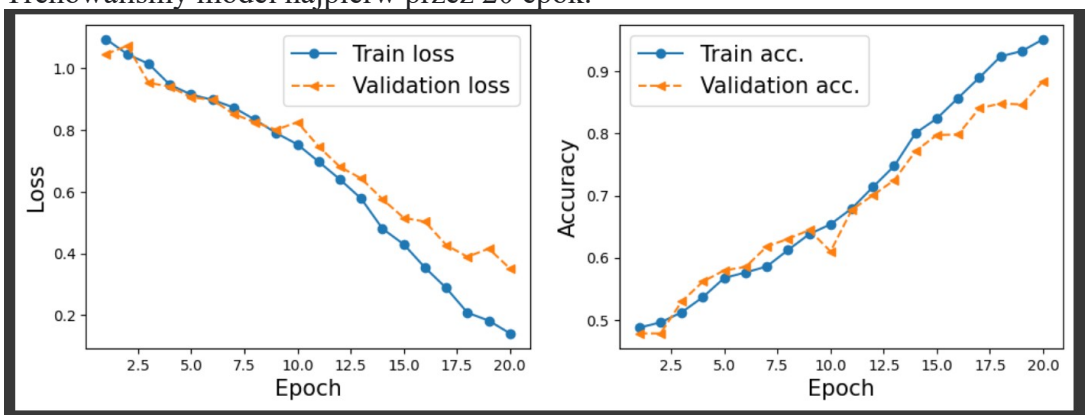
1. Model 1

```
print(model1)
```

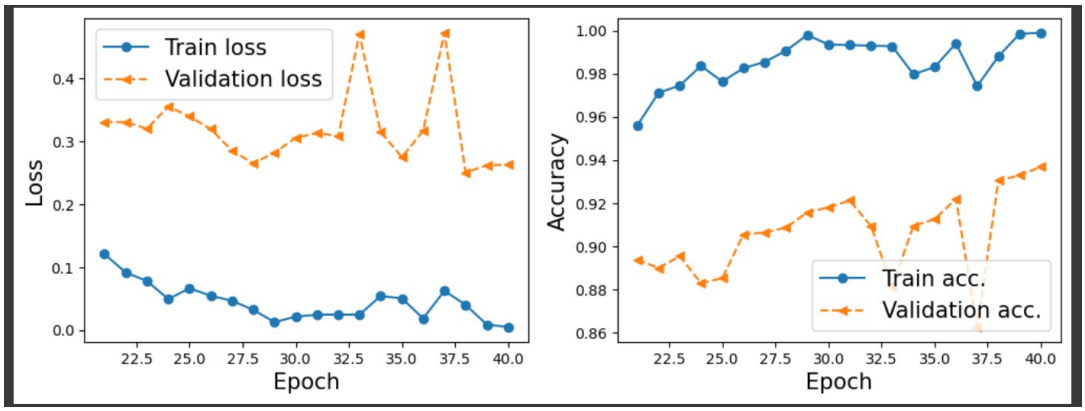
```
Sequential(
  (conv1): Conv2d(3, 32, kernel_size=(11, 11), stride=(1, 1), padding=(5, 5))
  (relu1): ReLU()
  (pool1): MaxPool2d(kernel_size=4, stride=4, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(32, 64, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3))
  (relu2): ReLU()
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv3): Conv2d(64, 80, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (relu3): ReLU()
  (pool3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc1): Linear(in_features=5120, out_features=1024, bias=True)
  (relu4): ReLU()
  (dropout): Dropout(p=0.5, inplace=False)
  (fc2): Linear(in_features=1024, out_features=4, bias=True)
)
```

Algorytm optymalizacyjny: Adam z learning rate = 0.001.

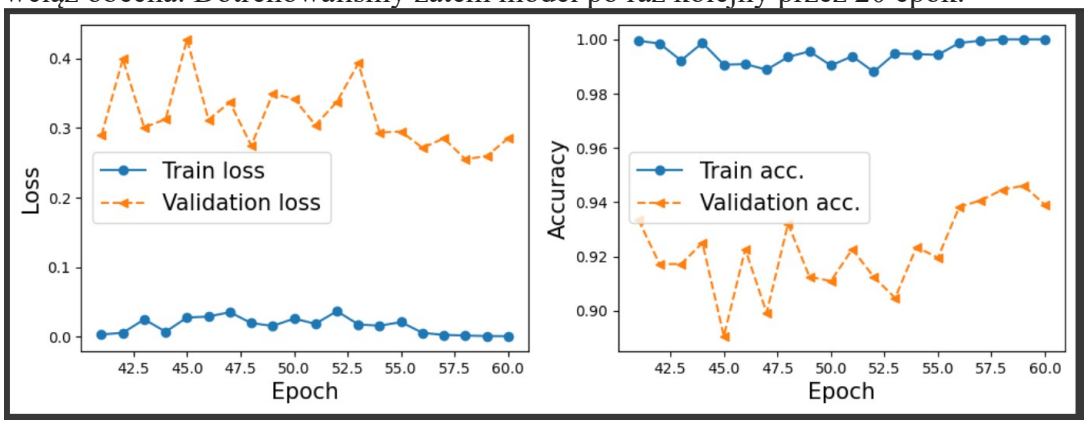
Trenowaliśmy model najpierw przez 20 epok.



Jak widać na powyższych wykresach, zaobserwowaliśmy mocną tendencję wzrostową dokładności na zbiorach train i valid. Postanowiliśmy zatem, że będziemy trenować model przez kolejne 20 epok.



Tendencja wzrostowa na wykresie dokładności była już nieco mniej wyraźna i zaburzona, ale wciąż obecna. Dotrenowaliśmy zatem model po raz kolejny przez 20 epok.



Model 1 podsumowanie:

- trening przez 60 epok
- po 60 epokach: train accuracy = 100%, valid accuracy = 94%.

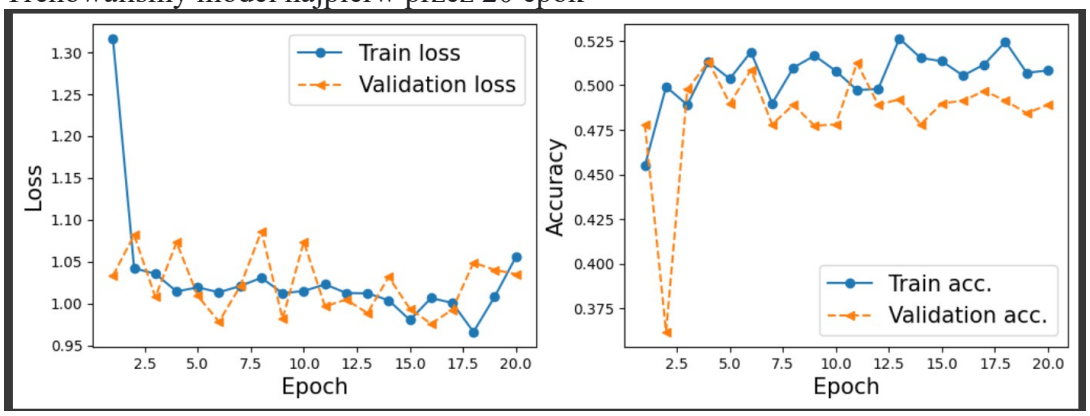
2. Model 2

```
print(model2)
```

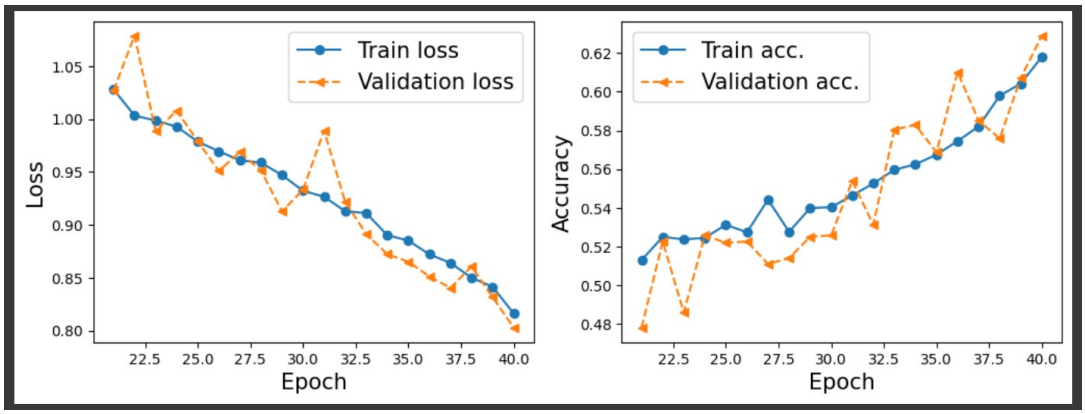
```
Sequential(
  (conv1): Conv2d(3, 50, kernel_size=(15, 15), stride=(1, 1), padding=(7, 7))
  (elu1): ELU(alpha=1.0)
  (pool1): AvgPool2d(kernel_size=2, stride=2, padding=0)
  (conv2): Conv2d(50, 50, kernel_size=(11, 11), stride=(1, 1), padding=(5, 5))
  (elu2): ELU(alpha=1.0)
  (pool2): AvgPool2d(kernel_size=2, stride=2, padding=0)
  (conv3): Conv2d(50, 50, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3))
  (elu3): ELU(alpha=1.0)
  (pool3): AvgPool2d(kernel_size=2, stride=2, padding=0)
  (conv4): Conv2d(50, 50, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (elu4): ELU(alpha=1.0)
  (pool4): AvgPool2d(kernel_size=2, stride=2, padding=0)
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc1): Linear(in_features=3200, out_features=1024, bias=True)
  (relu5): ReLU()
  (dropout): Dropout(p=0.25, inplace=False)
  (fc2): Linear(in_features=1024, out_features=1024, bias=True)
  (relu6): ReLU()
  (dropout2): Dropout(p=0.25, inplace=False)
  (fc3): Linear(in_features=1024, out_features=4, bias=True)
)
```

Algorytm optymalizacyjny: Adam z learning rate = 0.001

Trenowaliśmy model najpierw przez 20 epok

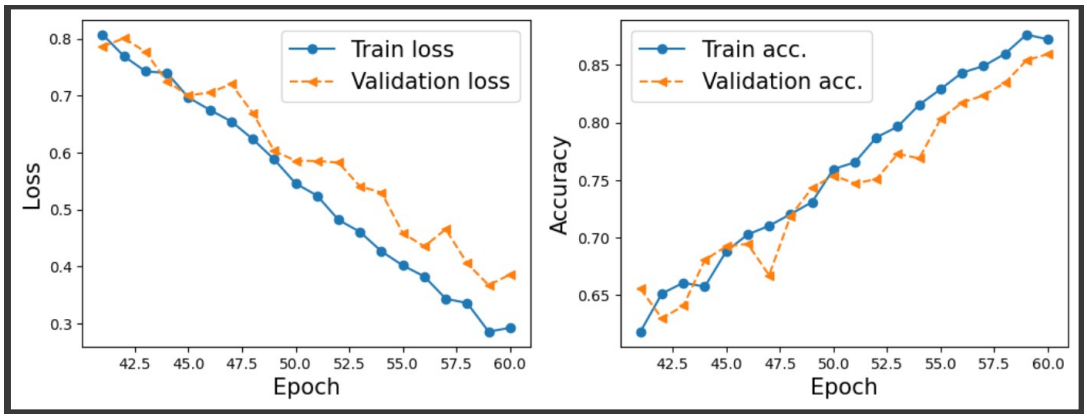


Dokładność na zbiorze testowym oraz walidacyjnym okazała się słaba. W związku z tym podjęliśmy decyzję o modyfikacji modelu poprzez zmniejszenie współczynnika uczenia (learning_rate) oraz zmniejszenie rozmiaru paczek danych (batch_size).

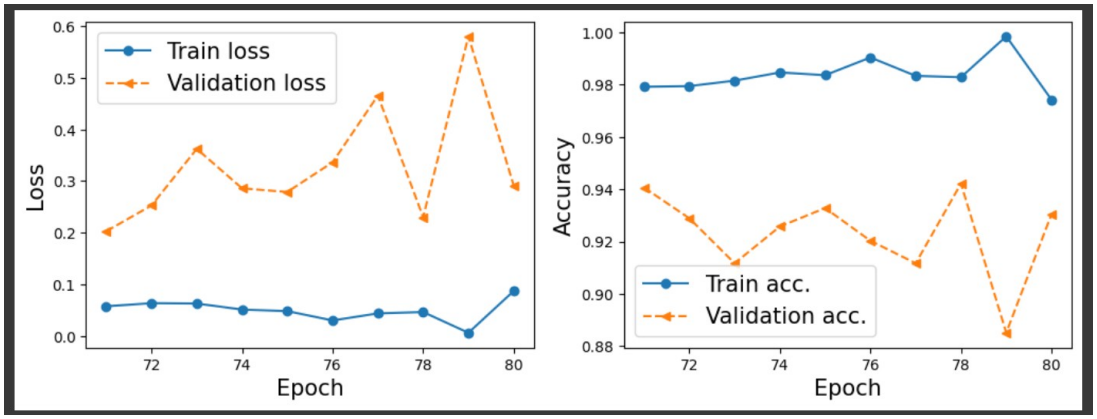


Przyniosło to pozytywne rezultaty – zaobserwowaliśmy wyraźną tendencję wzrostową zarówno w dokładności na zbiorze treningowym, jak i walidacyjnym. Dotrenowaliśmy zatem model po raz kolejny przez 20 epok.

Dokładność sieci nadal rosła, więc kontynuowaliśmy proces uczenia aż do momentu jej spadku. Aby zwiększyć stabilność rozwoju sieci, zdecydowaliśmy się na zmniejszenie wartości parametru dropout.



Dokładność nieco się poprawiła, ale nie wykazuje monotonicznego wzrostu. W związku z tym postanowiliśmy całkowicie wyłączyć dropout. Niestety, utknęliśmy na poziomie około 93%, a model staje się coraz mniej stabilny w trakcie uczenia.



Model 2 podsumowanie:

- trening przez 20 epok: train accuracy = 50%, valid accuracy = 49%.
- zmiana learning_rate i batch_size, trening po kolejnych 20 epokoach:
  train accuracy = 62%, valid accuracy = 63%.
- trening przez 40 epok: train accuracy = 87%, valid accuracy = 86%.
- brak stabilności – zmaina dropout i trening przez kolejne 5 epok:
  train accuracy = 98%, valid accuracy = 93%.
- Wyłączenie dropout i trening przez kolejne 5 epok – utykamy w miejscu gdzie valid accuracy = 93%.

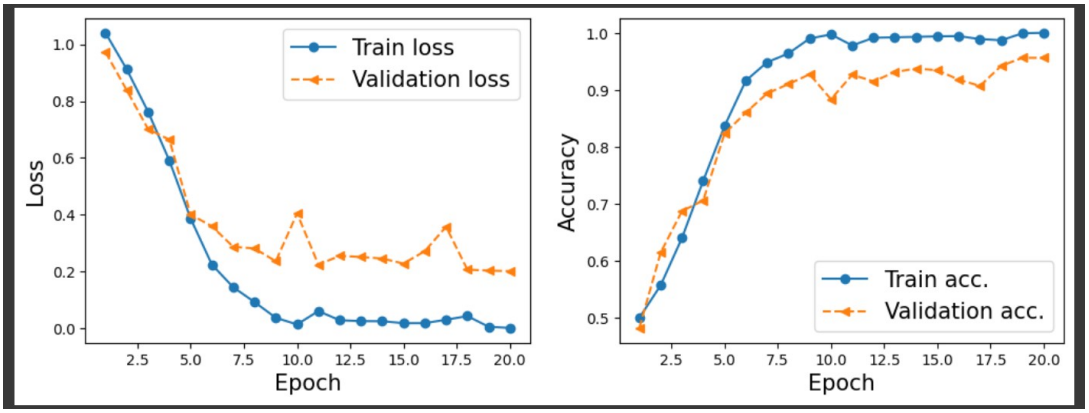
3. Model 3

```
print(model3)
```

```
Sequential(
  (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu1): ReLU()
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu2): ReLU()
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu3): ReLU()
  (pool3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc1): Linear(in_features=16384, out_features=128, bias=True)
  (relu4): ReLU()
  (fc2): Linear(in_features=128, out_features=4, bias=True)
)
```

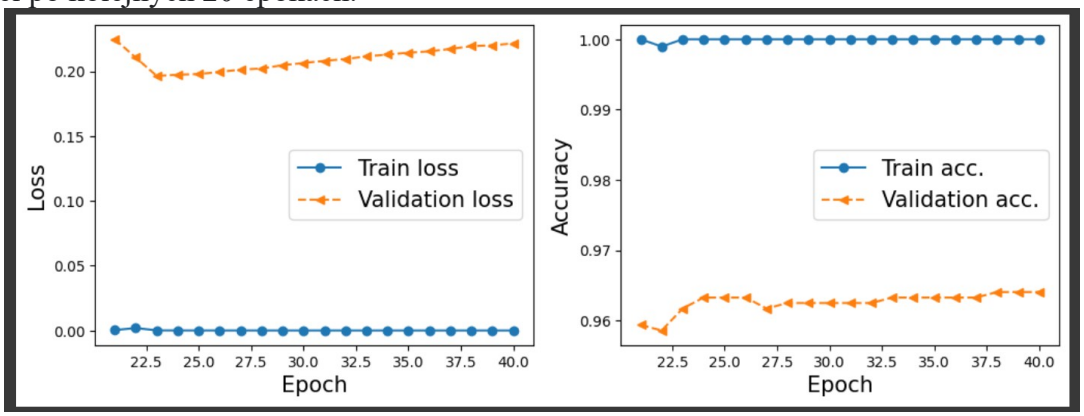
Algorytm optymalizacyjny: Adam z learning rate = 0.001

Model po 20 epokoach.



Model powoli się stabilizuje, więc zmniejszmy tempo uczenia i kontynuujmy naukę.

Model po kolejnych 20 epokach.



Ostatecznie osiągnęliśmy satysfakcjonujący wynik, co pozwoliło nam zakończyć etap nauki modelu.

Model 3 podsumowanie:

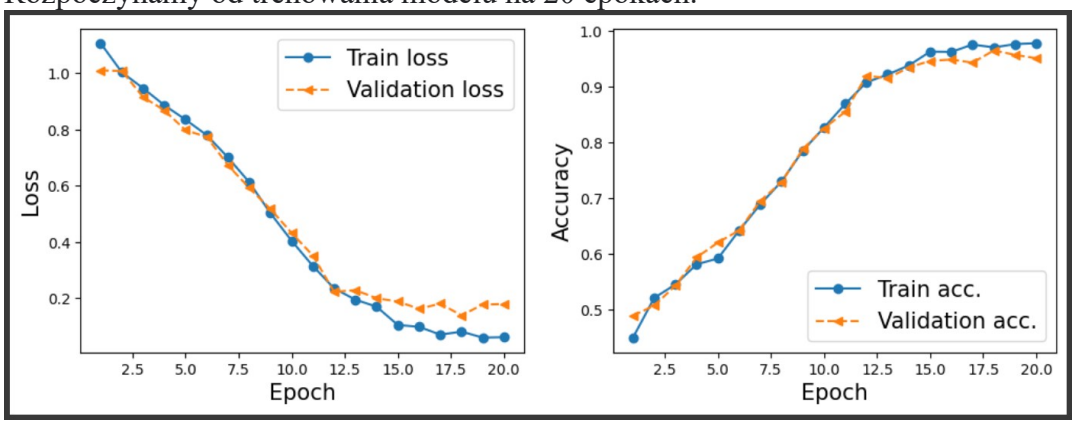
- trening przez 20 epok: train accuracy = 100%, valid accuracy = 95%.
- zmniejszenie learning_rate i nauka przez kolejne 20 epok:
  train accuracy = 100%, valid accuracy = 96%.

Zauważamy, że model ten wypada najlepiej, zatem spróbujemy go jeszcze udoskonalić. Zachowamy jego strukturę, a zmodyfikujemy filtry, typ poolingu i dodamy dropout w sieci FC. Budujemy model 3.5.

```
print(model35)
```

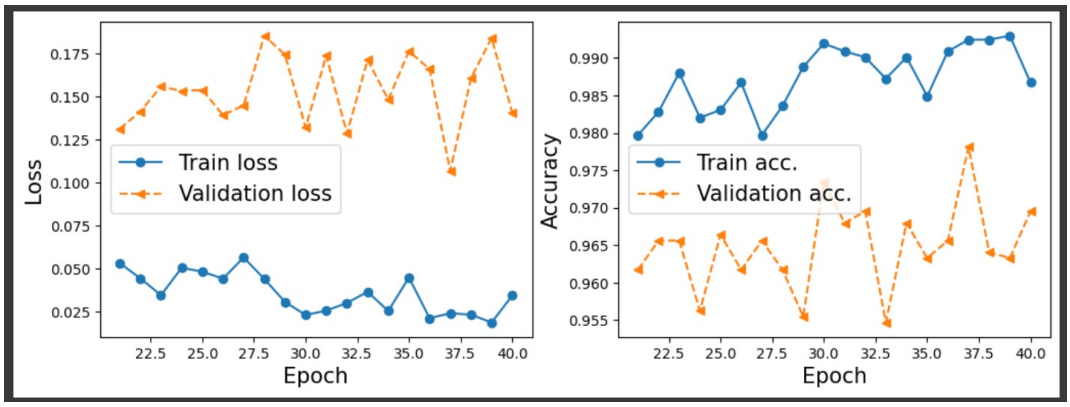
```
Sequential(
  (conv1): Conv2d(3, 20, kernel_size=(7, 7), stride=(1, 1), padding=(2, 2))
  (elu1): ELU(alpha=1.0)
  (pool1): AvgPool2d(kernel_size=2, stride=2, padding=0)
  (conv2): Conv2d(20, 40, kernel_size=(6, 6), stride=(1, 1), padding=(2, 2))
  (elu2): ELU(alpha=1.0)
  (pool2): AvgPool2d(kernel_size=2, stride=2, padding=0)
  (conv3): Conv2d(40, 30, kernel_size=(6, 6), stride=(1, 1), padding=(2, 2))
  (elu3): ELU(alpha=1.0)
  (pool4): AvgPool2d(kernel_size=2, stride=2, padding=0)
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc1): Linear(in_features=6750, out_features=1024, bias=True)
  (relu4): ReLU()
  (dropout): Dropout(p=0.5, inplace=False)
  (fc2): Linear(in_features=1024, out_features=4, bias=True)
)
```

Rozpoczynamy od trenowania modelu na 20 epokach.

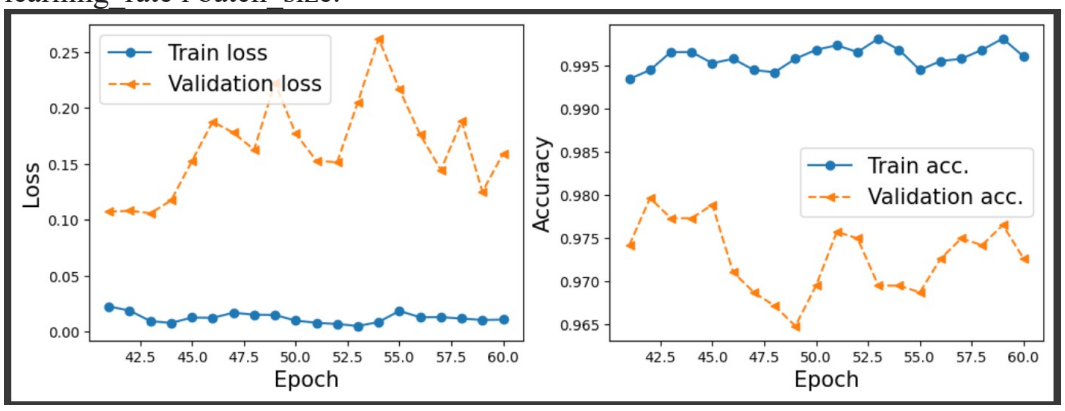


Zauważamy, że sieć już w 18-stej epoce osiągnęła wyniki lepsze niż w modelu3 po 40 epokach.

Kontynuujemy uczenie przez kolejne 20 epok.



Wyniki wypadają naprawdę dobrze, ale spróbujemy je jeszcze poprawić. Modyfikujemy learning_rate i batch_size.



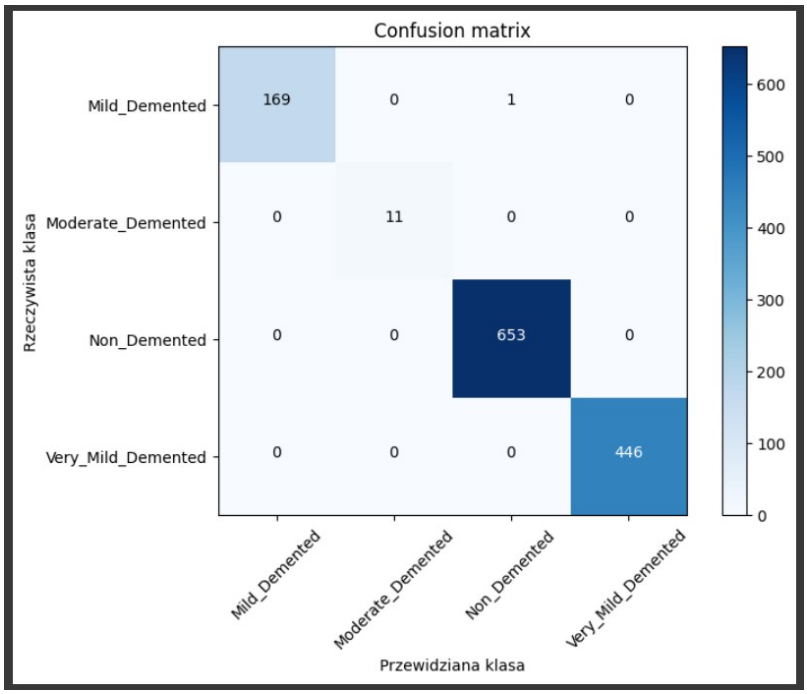
Dostajemy najlepszy z wyników.

Model 3.5 podsumowanie:

- trening przez 20 epok: train accuracy = 97%, valid accuracy = 95%.
- nauka przez kolejne 20 epok: train accuracy = 98%, valid accuracy = 96%.
- zmiana learning_rate i batch_size : train accuracy = 99%, valid accuracy = 97%.

Podsumowanie

Model 3.5 osiągnął najlepsze wyniki na zbiorze walidacyjnym. Na koniec nauczyliśmy ten model na zbiorze utworzonym przez połączenie zbioru walidacyjnego i zbioru treningowego, a następnie oceniliśmy jego dokładność — po 40 epokach treningu była ona na imponującym poziomie 99,9%.


Confusion matrix
Mild_Demented
Moderate_Demented
Non_Demented
Very_Mild_Demented
Rzeczywista klasa
169
0
1
0
0
11
0
0
0
0
653
0
0
0
0
446
600
500
400
300
200
100
0
Mild_Demented
Moderate_Demented
Non_Demented
Very_Mild_Demented
Przewidziana klasa