Michał Sumiński 249999, Jan Płoszaj 2249988

1. Opis architektury i sposobu trenowania modelu samodzielnie zaprojektowanego

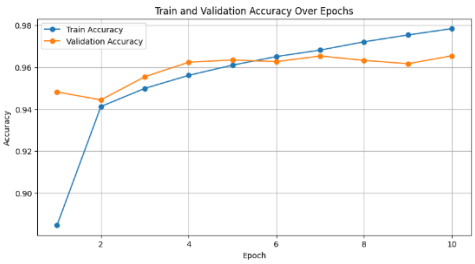
Model składa się z 2 części, czyli ekstraktora cech (na obrazku convolutional) oraz klasyfikatora (na obrazku linear).

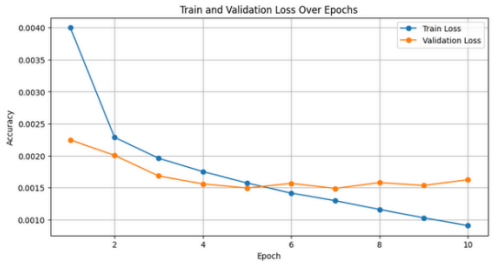
Ekstraktor, który na wejściu przyjmuje 3 kanałowe obrazki w rozmiarze 64x64, składa się z 3 warstw konwolucyjnych gdzie po każdej z nich stosowana jest funkcja aktywacji ReLu oraz MaxPooling – ekstraktor zwraca wektor 512 cech. Wektor ten jest następnie przekazywany do klasyfikatora, który 512 cech mapuje za pomocą 2 warstw liniowych i funkcji aktywacji ReLu po pierwszej z nich, na 2 klasy wyjściowe.

```
CNN(
  (convolutional): Sequential(
    (0): Conv2d(3, 64, kernel_size=(5, 5), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 256, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (linear): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=512, out_features=64, bias=True)
    (2): ReLU(inplace=True)
    (3): Linear(in_features=64, out_features=2, bias=True)
  )
)
```

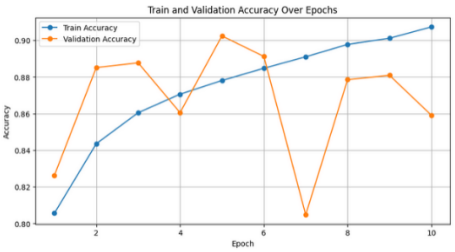
Podczas trenowania modelu zastosowano optymalizator Adam, funkcję straty CrossEntropyLoss oraz kryterium stopu EarlyStopping. Przebieg treningu można zaobserwować na poniższych wykresach:

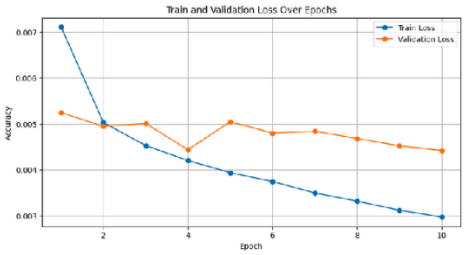
model ‘male’





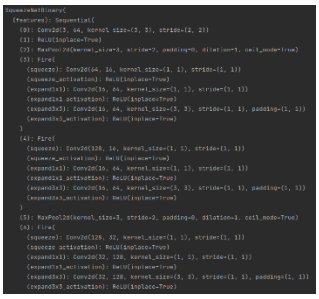
model 'eyeglasses’

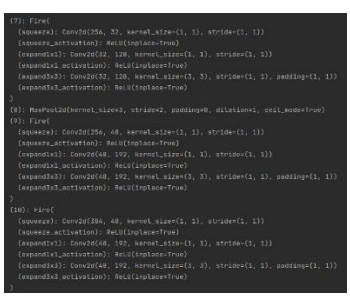




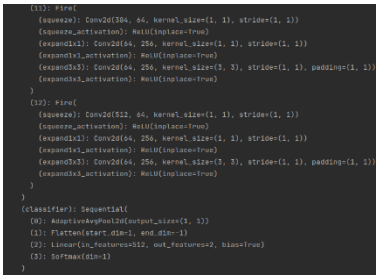
2. Opis wykorzystania istniejącego modelu

Do wytrenowania modelu wykorzystano wytrenowaną reprezentację modelu SqueezeNet i zmodyfikowano warstwę klasyfikatora, aby zwracał predykcje dla dwóch klas. Model SqueezeNet był trenowany na obrazach 227x227, jednakże obrazy Celeba poddawane były preprocesingowi: centercrop(178), resize(64,64). Następnie Model SqueezeNet dostosowywał rozmiar wejściowych obrazów do wielkości 256x256.Podczas trenowania modelu zastosowano optymalizator Adam, funkcję straty CrossEntropyLoss oraz kryterium stopu EarlyStopping.

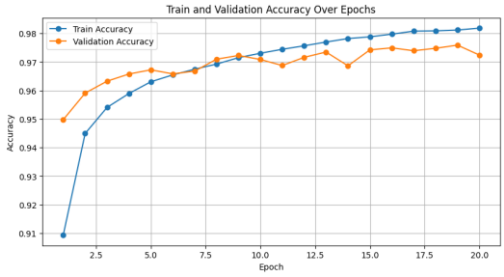


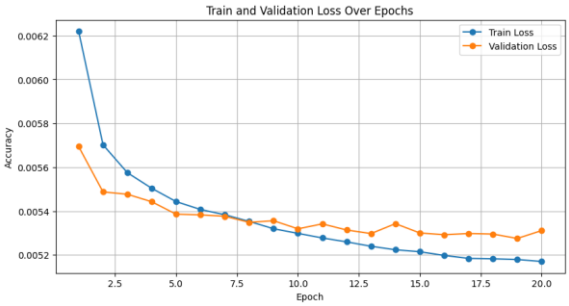


Podczas trenowania modelu zastosowano optymalizator Adam, funkcję straty CrossEntropyLoss oraz kryterium stopu EarlyStopping. Przebieg treningu można zaobserwować na poniższych wykresach:

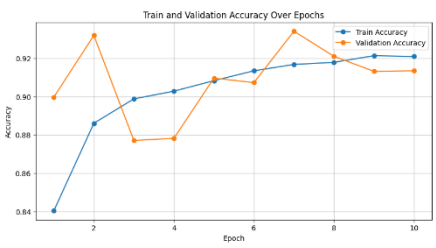


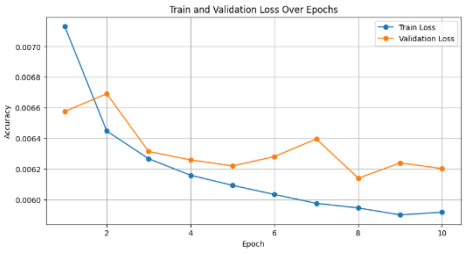
model ‘male’



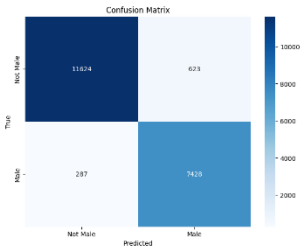


model 'eyeglasses’



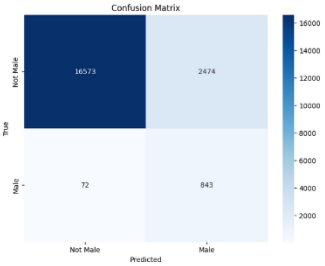


3. Wyniki testów obu modeli dla danych testowych dostępnych w zbiorze CelebA



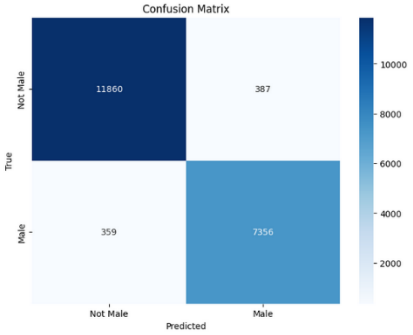
Accuracy: 0.9544
Weighted Precision: 0.9553
Weighted Recall: 0.9544
Weighted F1 Score: 0.9546
Macro Precision: 0.9493
Macro Recall: 0.9560
Macro F1 Score: 0.9523

Własny model ‘male’



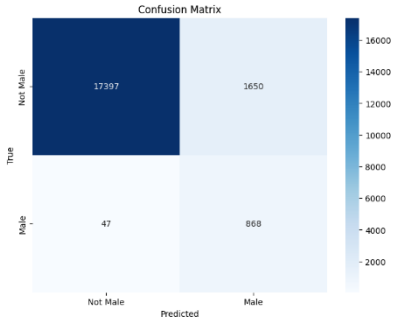
Accuracy: 0.8725
Weighted Precision: 0.9617
Weighted Recall: 0.8725
Weighted F1 Score: 0.9044
Macro Precision: 0.6249
Macro Recall: 0.8957
Macro F1 Score: 0.6635

Własny model 'eyeglasses’




```
Accuracy: 0.9626
Weighted Precision: 0.9627
Weighted Recall: 0.9626
Weighted F1 Score: 0.9626
Macro Precision: 0.9603
Macro Recall: 0.9609
Macro F1 Score: 0.9606
```

SqueezeNet ‘male’

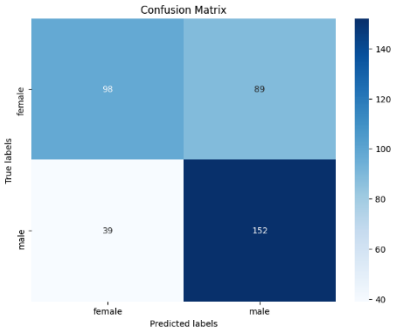



```
Accuracy: 0.9150
Weighted Precision: 0.9674
Weighted Recall: 0.9150
Weighted F1 Score: 0.9330
Macro Precision: 0.6710
Macro Recall: 0.9310
Macro F1 Score: 0.7296
```

SqueezeNet ‘eyeglasses’

4. Wyniki testów obu model dla danych testowych przygotowanych z wykorzystaniem zbioru WIDERFace

Zbiór testowy WIDERFace przygotowano w taki sposób, że zostało wybranych 100 zdjęć, ze zbioru treningowego, które zawierają od 3 do 5 twarzy, tą informację można było zdobyć na podstawie ilości bounding boxów, o których była zawarta informacja dla każdego zdjęcia. Następnie z oryginalnych zdjęć zostały wycięte same twarze, również na podstawie atrybuty bounding box oraz zostały ręcznie określone wartości cech ‘male’ oraz ‘glasses’. Ponieważ wycinki twarzy miały różne wymiary, każdy taki wycinek przed wpuszczeniem do modelu został poddany operacji resize(64, 64) co zapewniało kompatybilność z warstwą wejściową naszych modeli.




```
Accuracy: 0.6613756613756614
Precision(weighted): 0.6725691895452928
Precision(macro): 0.673016930672078
Recall(weighted): 0.6613756613756614
Recall(macro): 0.6599378447238009
F1(weighted): 0.654843556078124
F1(macro): 0.654320987654321
```

Własny model cecha ‘male’

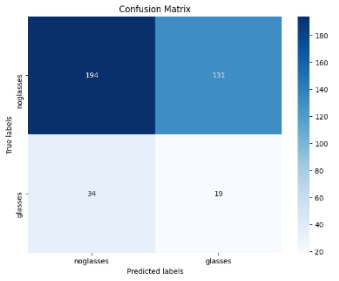
Można stwierdzić, że model ma tendencję do częstszego określania płci jako ‘male’.

Przykładowo, źle sklasyfikowane obrazy (po jednej z każdej klasy):

klasa = ‘female’, predicted = ‘male’

klasa = ‘male’, predicted = ‘female’

Własny model cecha ‘glasses’



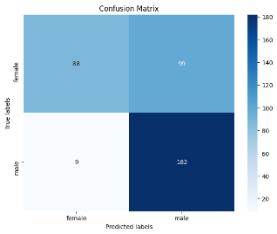

```
Accuracy: 0.5634920634920635
Precision(weighted): 0.749334447229184
Precision(macro): 0.4887719298245614
Recall(weighted): 0.5634920634920635
Recall(macro): 0.47770682148040633
F1(weighted): 0.6294976596778679
F1(macro): 0.444409802332107
```

Widać, że model całkiem dobrze określa klasę ‘noglasses’, natomiast kompletnie nie umie poprawnie rozpoznać klasy ‘glasses’, a za gdy już ją określa to jest to w dużej większości nietrafne.

Przykładowo, źle sklasyfikowane obrazy (po jednej z każdej klasy):

 klasa = ‘glasses’, predicted = ‘noglasses’  klasa = ‘noglasses’, predicted = ‘glasses’

- SqueezeNet cecha ‘male’



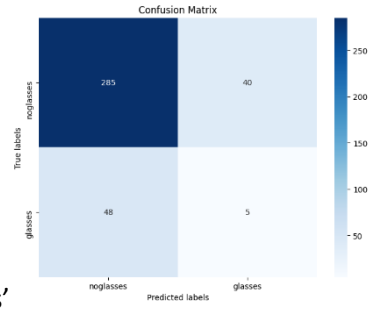

```
Accuracy: 0.7142857142857143
Precision(weighted): 0.7760784909774161
Precision(macro): 0.7774516637927872
Recall(weighted): 0.7142857142857143
Recall(macro): 0.711733908222975
F1(weighted): 0.6962537939501415
F1(macro): 0.6954523752685604
```

Na tej podstawie można wyciągnąć wnioski, że model o wiele lepiej klasyfikuje klasę ‘male’ niż ‘female’.

Przykładowo, źle sklasyfikowane obrazy (po jednej z każdej klasy):

 klasa = ‘female’, predicted = ‘male’  klasa = ‘male’, predicted = ‘female’

- SqueezeNet cecha ‘glasses’




```
Accuracy: 0.7671957671957672
Precision(weighted): 0.7514339736561959
Precision(macro): 0.48348348348348347
Recall(weighted): 0.7671957671957672
Recall(macro): 0.48563134978229316
F1(weighted): 0.7591087769116427
F1(macro): 0.48415110725141114
```

Ponieważ w zbiorze testowym znajdowało się o wiele więcej elementów ‘noglasses’ niż ‘glasses’ (w stosunku około 6:1), to ciężko uznać te wyniki za wiarygodne, ponieważ model może w większości przypadków przewidywać ‘noglasses’, a ponieważ w zbiorze testowym głównie takie przypadki się znajdowały to osiągnął dość wysoką wartość accuracy.

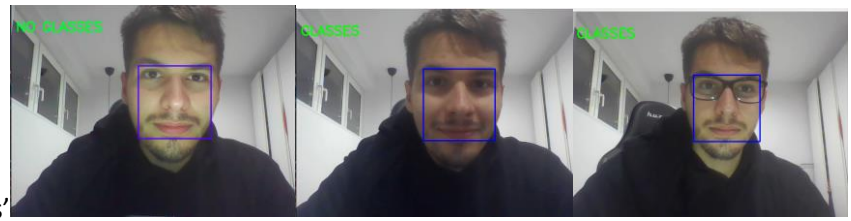
Przykładowo, źle sklasyfikowane obrazy (po jednej z każdej klasy):

 klasa = ‘glasses’, predicted = ‘noglasses’ (ten obrazek, choć na oko oczywisty i łatwy do określenia to widać, że oba modele źle go sklasyfikowały)

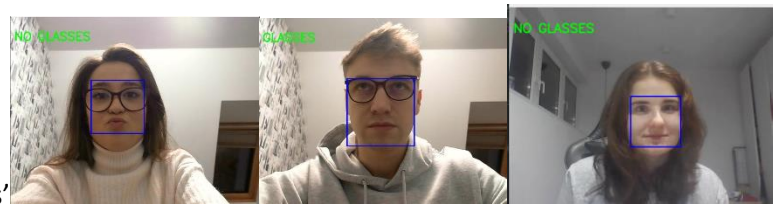
 klasa = ‘noglasses’, predicted = ‘glasses’

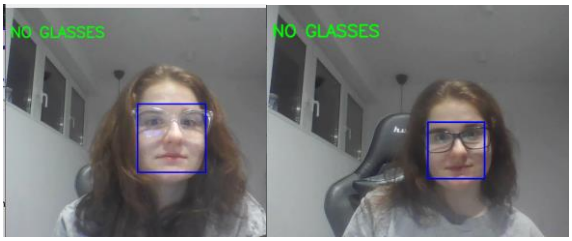
5. Wyniki testów obu model w udostępnionym programie testowym

Własny model cecha ‘male’ 

Własny model cecha ‘glasses’ 

SqueezeNet cecha ‘male’ 

SqueezeNet cecha ‘glasses’ 



6. Wnioski

Ogólny wniosek jaki nam się nasuwa jest taki, że testowanie na zbiorze WIDERFace oraz na kamerce nie jest do końca odpowiednie przy zbiorze treningowym CELEBa, ponieważ zdjęcia ze zbioru treningowego CELEBa zawierają całą twarz, wraz z włosami, czasami szyją i otoczeniem dookoła twarzy. Natomiast bounding boxy ze zbioru WIDERFace lub ramki, które znajduje detektor kaskadowy w skrypcie camera.py łapią tylko wycinki twarzy, więc pewne cenne informacje, na które zwraca uwagę model podczas treningu na zbiorze CELEBa mogą być poza danym wycinkiem, co osłabia skuteczność predykcji modelu.