Klasyfikacja obrazów z wykorzystaniem metod uczenia głębokiego

Szymon Suszek, Bartosz Lodek

Abstract

W ramach projektu analizowano problem klasyfikacji dla zbioru Fashion-MNIST [6]. Do rozwiązania zadania skorzystano z metod uczenia głębokiego. Zaimplementowano i sprawdzono skuteczność sieci opartych na modelu perceptrona wielowarstwowego oraz sieci splotowych. Łącznie zaproponowano pięć architektur.

1. Wstęp

Zbiór obrazów składa się z 60,000 przykładów treningowych oraz 10,000 przykładów testowych. Każdy obraz ma jeden kanał (skala szarości) o wymiary 28x28 i należy do jednej z 10 klas. Dodatkowo z zbioru treningowego wydzielono 5,000 przykładów do zbioru walidacyjnego – służy do obliczania metryk podczas treningu jako odniesie. Wszystkie wymienione zbiory są dobrze zbalansowane, każda z klas jest równoliczna.

Na początku badań sprawdzono skuteczność najprostszych technik – modelu wielowarstwowego perceptronu, następnie zbadano płytkie sieci splotowe i kolejno zaimplementowano bardziej złożone modele.

Częstą praktyką w uczeniu głębokim jest korzystanie z wcześniej wytrenowanej sieci, poprawia to skuteczność w przypadku uczenia na małych zbiorach obrazów. Wykorzystana sieć wcześniej zostaje wytrenowana na dużym zbiorze danych. Zbiór powinien być nie tylko duży ale i ogólny [2]. Przestrzenna hierarchia cech wyuczona przez wcześniej trenowany model może być przydatna dla innych zbiorów danych oraz innych zadań przetwarzania obrazów[3]. Podejście to przedstawiono w sekcji 1.4 oraz 1.5.

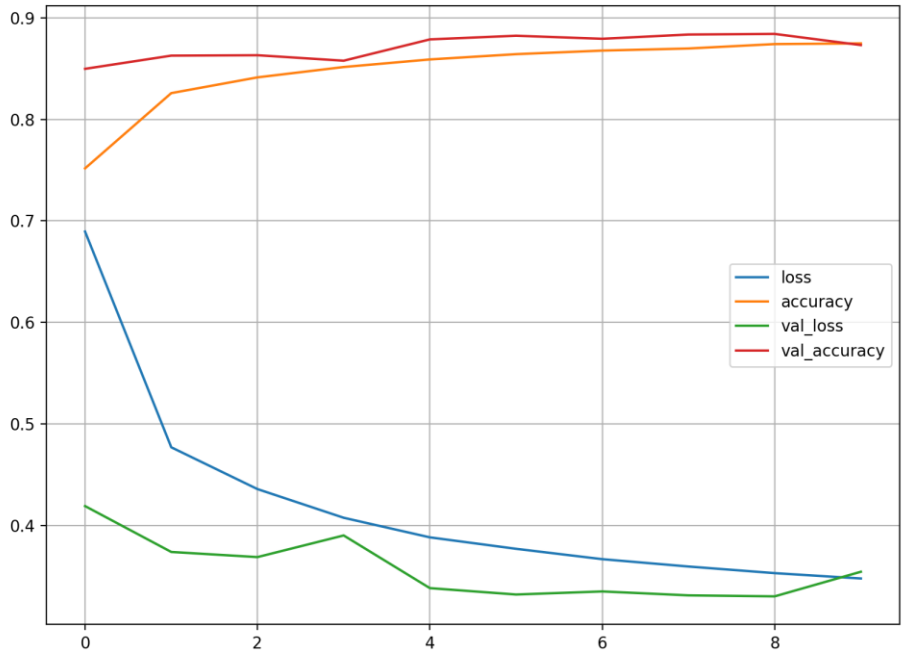
1.1 Gęsta sieć neurona – perceptron wielowartwowy

Badania rozpoczeto od zaimplementowania gęstej sieci neuronowej o dwóch warstwach ukrytych oraz jednej wyjściowej. Liczba neuronów w wartwach ukrytych wynosiła 512, w warstwie wyjściowej z kolei 10. Wejściem sieci jest jednowymiarowy wektor kolumny równy iloczynowi szerokości i wysokości obrazu. W celu zmniejszenia przeuczenia sieci (overfitting) wykorzystano z metody regularyzacji odrzucania (dropout)[4]. Dokładna struktura sieci:

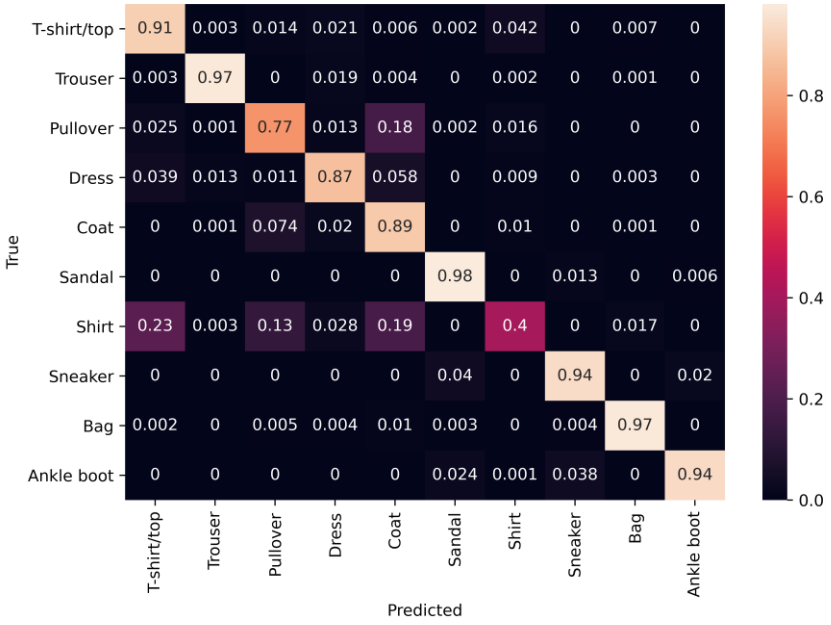
Layer (type)	Output Shape	Param #
dense (Dense)	(None, 512)	401920
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 512)	262656
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 10)	5130

Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0

Test accuracy: 0.8636999726295471



55 **Rysunek 1.** Wykres wartości straty oraz dokładności na zbiorze treningowym oraz walidacyjnym podczas uczenia, dla 10 epok.



Rysunek 2. Macierz pomyłek dla zbioru testowego.

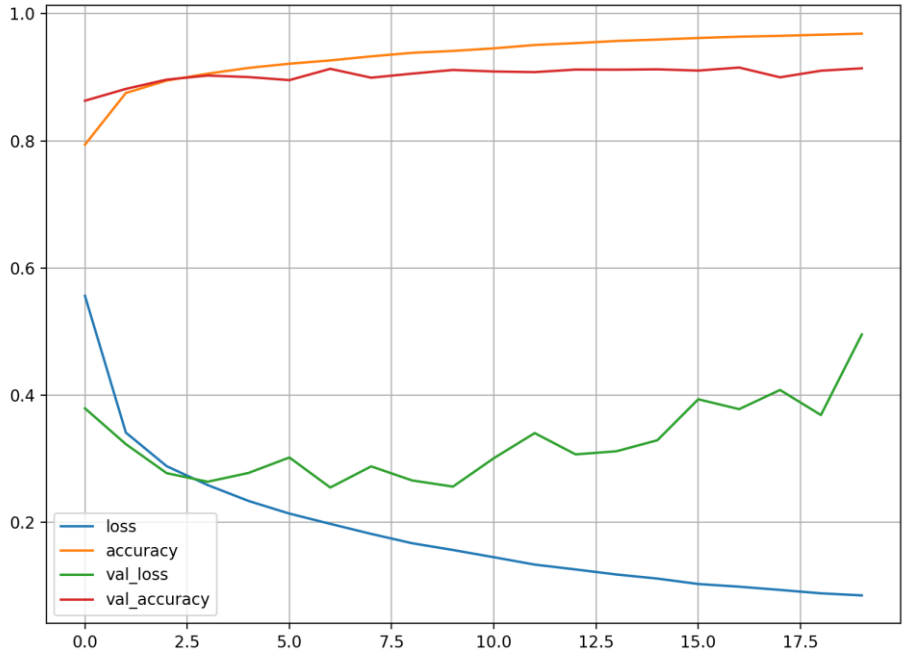
1.2 Płytka sieć splotowa

Następnie zaimplementowano sieć składającą się z trzech warstw splotowych, po dwóch z nich bezpośrednio zastosowano warstwy łączące (maxpooling) oraz jednej gęstej wraz z warstwą wyjściową. Warstwa łącząca ma na celu zmniejszenie wymiarowości obrazu oraz ekstrakcje cech. Dokładna struktura sieci:

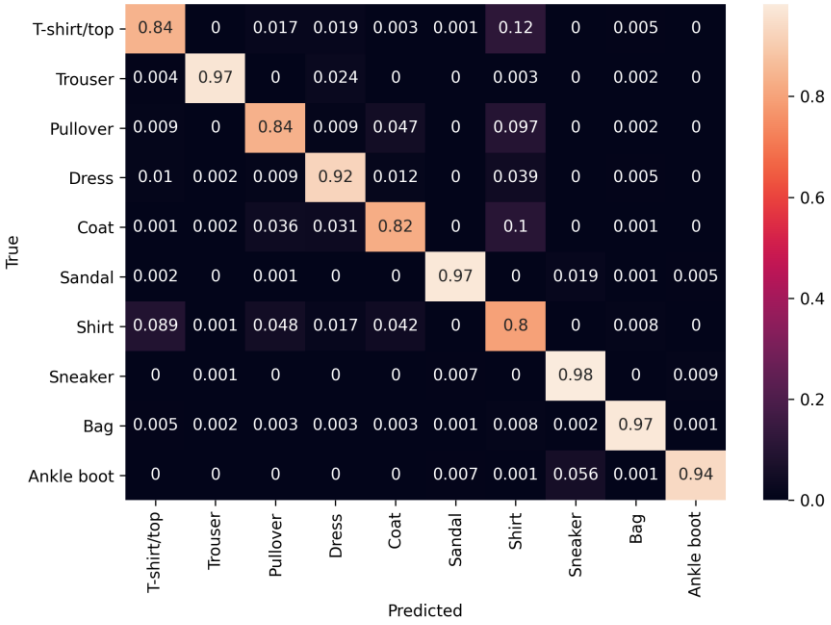
Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_2 (MaxPooling2	(None, 13, 13, 32)	0
conv2d_4 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_3 (MaxPooling2	(None, 5, 5, 64)	0
conv2d_5 (Conv2D)	(None, 3, 3, 64)	36928
flatten_1 (Flatten)	(None, 576)	0

```
dense_2 (Dense)              (None, 64)                36928
_________________________________________________________________
dense_3 (Dense)              (None, 10)                650
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0

Test accuracy: 0.9046000242233276
```



Rysunek 3. Wykres wartości straty oraz dokładności na zbiorze treningowym oraz walidacyjnym podczas uczenia, dla 20 epok.



Rysunek 4. Macierz pomyłek dla zbioru testowego.

1.3 Sieć splotowa

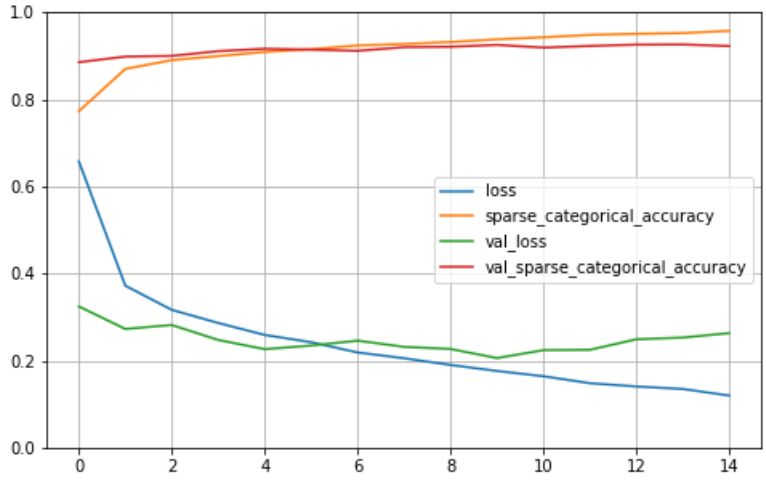
Na podstawie architektur osiągających najlepsze wyniki w klasyfikacji wybranych zbiorów danych np. ImageNet stworzono autorską implementację. Składa się z 7 warstw konwolucyjnych, po każdej warstwie dokonywana jest normalizacja [5] przed funkcją aktywacji. Z powodu małego rozmiaru obrazu wejściowego wykorzystano z uzupełniania obramowania zerami podczas wykonywania operacji konowlucji. Zgodnie z większością istniejących architektur [2][3] w raz z redukcją rozmiaru zdjęcia, rośnie głębokość sieci. Warstwy łączące (pooling) ożyte zostały w celu redukcji wymiarowości. Po 7 warstwach konwolucji i łączących na górze klasyfikatora występują dwie warstwy ukryte gęste o liczbie neuronów 256, 128 oraz warta wyjściowa 10 neuronów. Dokładna struktura sieci:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 64)	1664
batch_normalization (BatchNo	(None, 28, 28, 64)	256

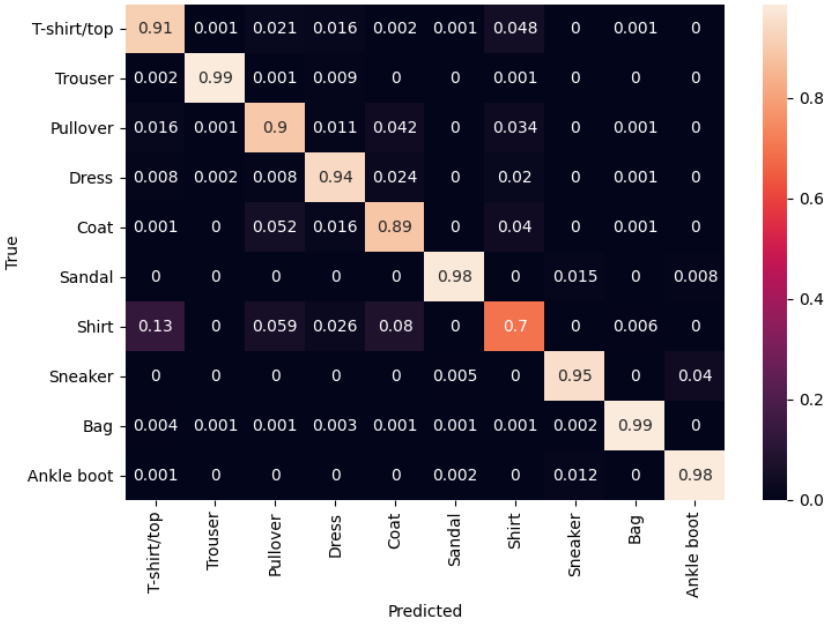
```
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 28, 28, 64)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 128)       73856
_________________________________________________________________
batch_normalization_1 (Batch (None, 14, 14, 128)       512
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 64)        8256
_________________________________________________________________
batch_normalization_2 (Batch (None, 14, 14, 64)        256
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 14, 14, 128)       73856
_________________________________________________________________
batch_normalization_3 (Batch (None, 14, 14, 128)       512
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 14, 14, 128)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 7, 7, 256)         295168
_________________________________________________________________
batch_normalization_4 (Batch (None, 7, 7, 256)         1024
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 7, 7, 256)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 7, 7, 128)         32896
_________________________________________________________________
batch_normalization_5 (Batch (None, 7, 7, 128)         512
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 7, 7, 256)         295168
_________________________________________________________________
batch_normalization_6 (Batch (None, 7, 7, 256)         1024
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 7, 7, 256)         0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 2304)              0
_________________________________________________________________
```

```
dense (Dense)                (None, 256)               590080
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 256)               0
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 128)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 1,409,226
Trainable params: 1,407,178
Non-trainable params: 2,048

Test accuracy: 0.92330002784729
```



Rysunek 5. Wykres wartości straty oraz dokładności na zbiorze treningowym oraz walidacyjnym podczas uczenia, dla 15 epok.



Rysunek 6 Macierz pomyłek dla zbioru testowego.

1.4 Ekstrakcja cech z sieci VGG16

Ekstrakcja cech polega na wykorzystaniu reprezentacji wyuczonej przez sieć wcześniej w celu dokonania ekstrakcji interesujących nas cech z nowych próbek [2]. W badaniu skorzystano z wytrenowanej wcześniej sieci VGG16 [1] na zbiorze ImageNet, który składa się z 1,4 miliona obrazów należących do 1000 klas. Większość konwolucyjnych sieci neuronowych stosowanych w zadaniach klasyfikacji składa się z serii warstw łączących (pooling) i warstw konwolucyjnych. Na końcu sieci znajduje się gęsto połączony klasyfikator. Pierwsza część sieci (złożona z warstw konwolucji i łączących) nazywana jest konowolucyjną bazą modelu.

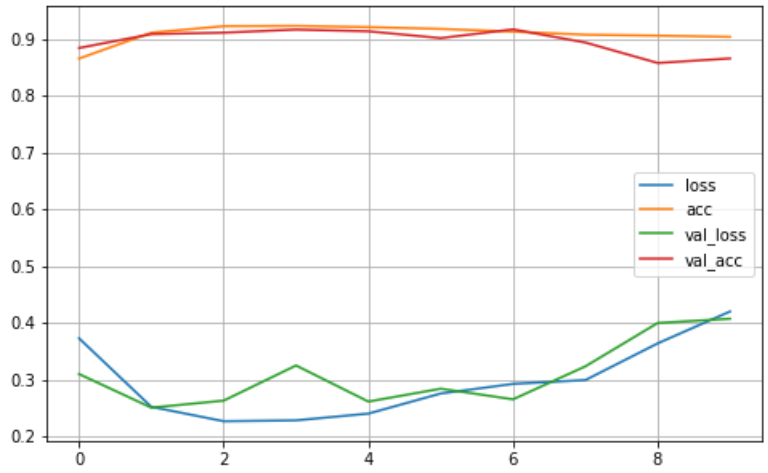
W sieciach konwolucyjnych ekstrakcja cech polega na skorzystaniu z wcześniej wyuczonej bazy konwolucyjnej, przepuszczeniu przez nią nowych danych i wytrenowaniu nowego klasyfikatora na bazie wyjścia tej sieci.

Najmniejszy dopuszczalny rozmiar obrazów wejściowych dla sieci VGG16 wynosi 32x32, w tym celu przeskalowano obrazy używających interpolacji dwuliniowej (Bilinear interpolation). Następnie skopiowano 3 krotnie każdy z obrazów, tak by obrazy wejściowe składały się z 3 kanałów. Tak przygotowane dane przepuszczone przez konwolucyjną bazę sieci VGG16 tworzą mapę cech (liczba próbek, 1, 1, 512). Mapy cech posłużą jako dane wejściowe podczas trenowania klasyfikatora.

Klasyfikator jako wejście przyjmuje zatem wyjście bazy konwolucyjnej sieci VGG16.

Składa się on z warstwy wejściowej o liczbie 256 neuronów oraz warstwy wyjściowej 10 neuronów.

Z Rysunek 7 widać szybkie przeuczenie modelu, dodanie warstw głębokich o większej liczbie neuronów oraz techniki odrzucania [4] mogłoby by wpłynąć pozytywnie na uczenie modelu.



Rysunek 7. Wykres wartości straty oraz dokładności na zbiorze treningowym oraz walidacyjnym podczas uczenia, dla 10 epok.

Dokładna budowa bazy konwolucyjnej:

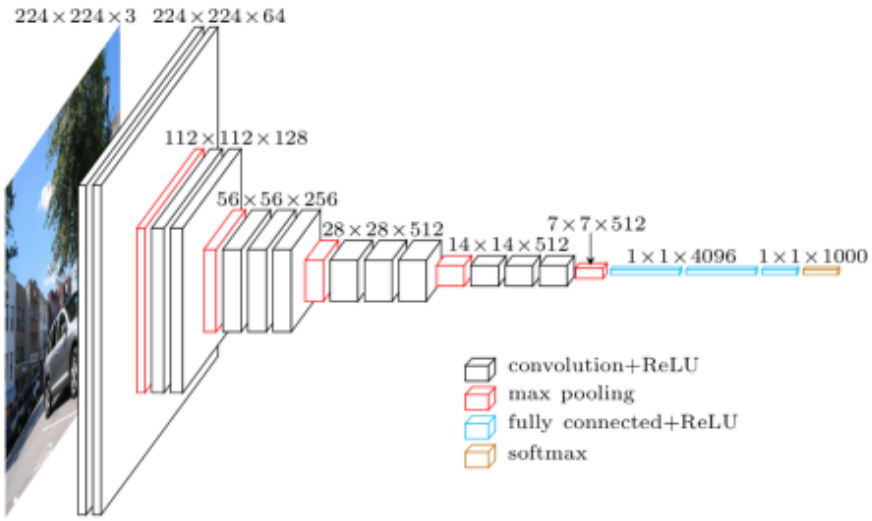
Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 32, 32, 3)]	0
block1_conv1 (Conv2D)	(None, 32, 32, 64)	1792
block1_conv2 (Conv2D)	(None, 32, 32, 64)	36928
block1_pool (MaxPooling2D)	(None, 16, 16, 64)	0
block2_conv1 (Conv2D)	(None, 16, 16, 128)	73856
block2_conv2 (Conv2D)	(None, 16, 16, 128)	147584
block2_pool (MaxPooling2D)	(None, 8, 8, 128)	0
block3_conv1 (Conv2D)	(None, 8, 8, 256)	295168

```
block3_conv2 (Conv2D)       (None, 8, 8, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)       (None, 8, 8, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)  (None, 4, 4, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)       (None, 4, 4, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)       (None, 4, 4, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)       (None, 4, 4, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)  (None, 2, 2, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)       (None, 2, 2, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)       (None, 2, 2, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)       (None, 2, 2, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)  (None, 1, 1, 512)       0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
```

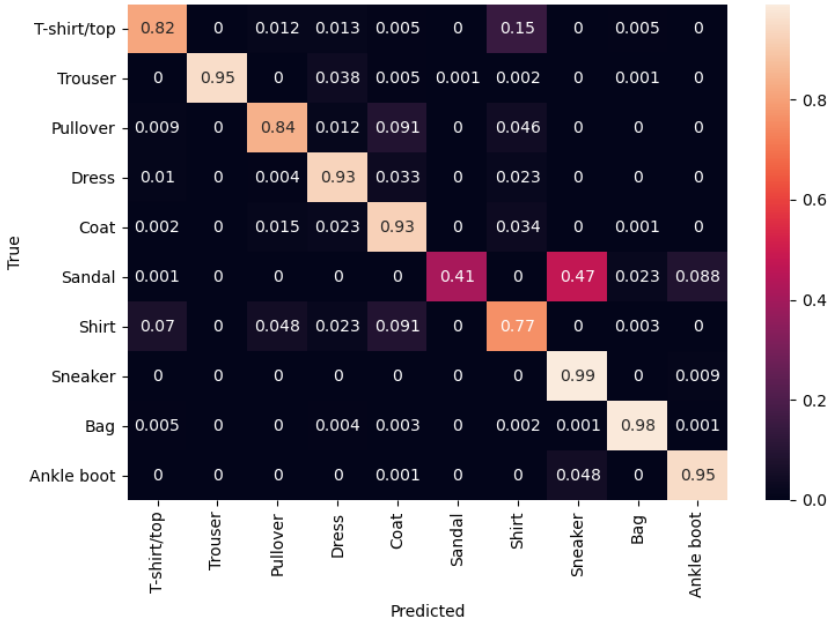
Dokładna struktura klasyfikatora:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 1, 1, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 256)               131328
_________________________________________________________________
dense_1 (Dense)              (None, 10)                2570
=================================================================
Total params: 14,848,586
Trainable params: 14,848,586
Non-trainable params: 0
```

Test accuracy: 0.8569999933242798



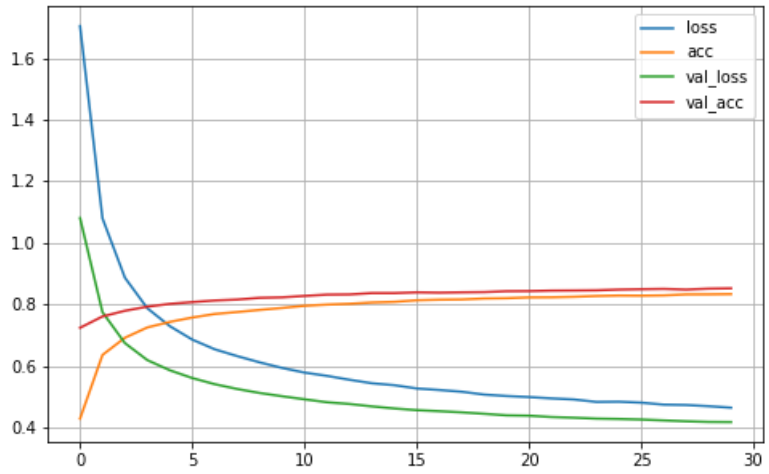
Rysunek 8. Architektura sieci VGG16 [7].



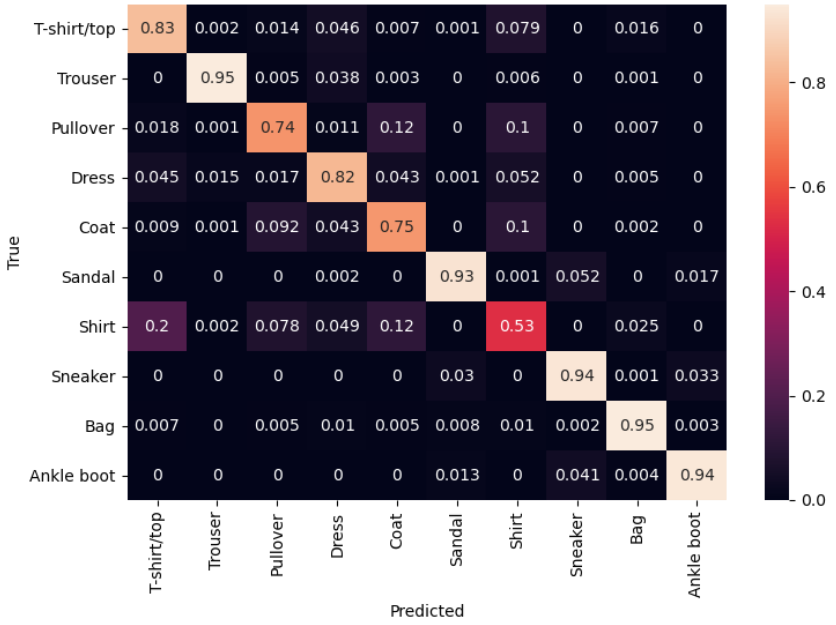
Rysunek 9. Macierz pomyłek dla zbioru testowego.

1.5 Trening sieci VGG16

Wykonano także trening wstępnie wytrenowanej sieci VGG16 na zbiorze ImageNet Obraz wejściowy tak jak w poprzedniej sekcji wymaga przetworzenia. Obrazy są interpolowane do rozmiaru 32x32 oraz liczba kanałów poprzez
285 skopiowanie wzrasta do 3 dla każdej próbki.



Rysunek 10. Wykres wartości straty oraz dokładności na zbiorze treningowym oraz walidacyjnym podczas uczenia, dla 30 epok.



Rysunek 11. Macierz pomyłek dla zbioru testowego.

2. Podsumowanie

Tabela 1 przedstawia dokładności badanych sieci na zbiorze testowym. Najwyższą dokładność uzyskała sieć o architekturze zaproponowanej przez autorów raportu. Model ten najgorzej radził sobie z odróżnieniem kategorii koszulki na krótki rękaw i bez rękawów od koszuli. Po spojrzeniu na obie kategorie widać w niektórych przypadkach bardzo subtelne różnice, np. występują koszule na krótki rękaw. Warto zauważyć, że (Rysunek 6) kategorie Trousers, Sandal, Bag, Ankle boot uzyskują dokładność powyżej lub równą 98%. Trudność w poprawnej klasyfikacji może powodować mały rozmiar w skali szarości obrazów.

Tabela 1

Nazwa	Dokładność
1.1 Perceptron wielowarstwowy	0.8636999726295471
1.2 Płytka sieć splotowa	0.9046000242233276
1.3 Sieć splotowa	0.92330002784729
1.4 Ekstrakcja cech z sieci VGG16	0.8569999933242798
1.5 VGG16	0.8382999897003174

Odniesienia

[1] K. Simon, A. Zisserman Very Deep Convolutional Networks for Large-Scale Image Recogniction

[2] F. Chollet Deep Learning with Python

[3] A. Geron Hands-on Machine Learning with Scikit-Learn, Keras and Tensorflow: Concept, Tools, and techniques to Build Intelligent System, 2nd Edition

[4] G. Hinton, A. Krizkevsky, N. Srivastava, I. Sutskever, R. Salakhutdinov Dropout: A simple Way to Prevent Neural Networks from Overfitting

[5] S. Ioffe, Ch. Szegedy Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariance Shift

[6] H. Xiao, K. Rasul, R. Vollgraf Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms

[7] https://neurohive.io/en/popular-networks/vgg16/