ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

MSc in Data Science
Deep Learning

Fashion Item Recognition with Deep Learning

Kostantinos Papastamos
Student number: DS3617011

Ioannis Zikos
Student number: DS3617004

The files of the project are available here.
The code can also easily be accessed here.

Data

The given “mnist” fashion **training dataset** was split in (2) subsets:

- 80 % partition was used for training
- 20 % partition was used for validation

Therefore, the 80 % partition was used in order to fit the network’s parameters (adjusting weights) and the 20 % partition was used in order to tune the hyper parameters of the network and optimize its architecture by calculating validation accuracy and comparing it to the training accuracy.

The real accuracy of the trained network was finally calculated on “mnist” fashion **test dataset** that included unknown data to the model.

CNN Architecture

The Convolutional Neural Network (CNN) architecture was employed in order to solve the fashion items recognition problem which is a multiclass image classification task. More precisely the architecture employed in this project is similar to the inception network that uses parallel convolutions and then concatenates the results before feeding them to the next layer. Each one of these inception modules are used multiple times in order to build the network.

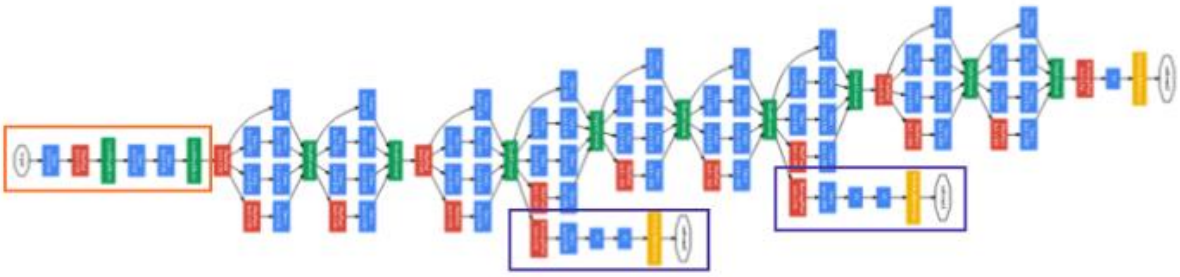

Figure 1: $Inception$ architecture.

The inception “Blocks” that we used in our network have the following architecture:

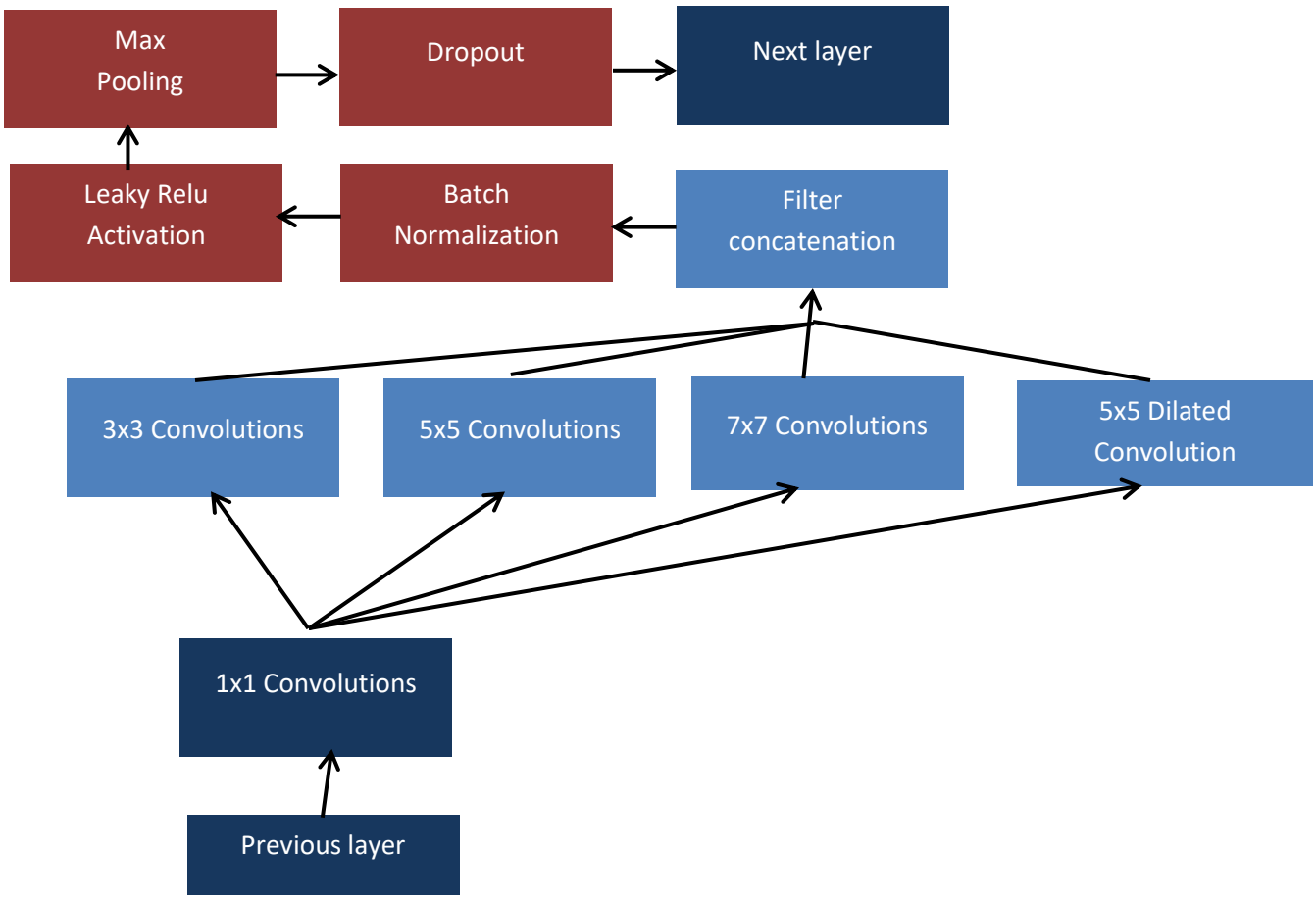

Figure 2: The Convolution Block

Three different convolution kernel sizes were used in order to understand patterns on both small and large sections of the image. Also, after evaluating the results, it was clear the models have a hard time differentiating 2 specific classes (1 and 7 as reported in the evaluation chapter) so we added a dilated convolution hoping to help the model better understand the differences of those classes.

The convolutional neural network architecture is as follows:

1. One Convolution Block applied on the input data with 32 filters.
2. Resulting filters are reduced to 64 with a 1x1 Convolution Layer of same padding.
3. One Convolution Block applied on the output of step 2 with 64 filters, same padding, dilation rate = 3 and dropout rate = 0,3 .
4. Resulting filters are reduced to 128 with a 1x1 Convolution Layer of same padding.
5. One Convolution Block applied on the output of step 4 with 128 filters, same padding, dilation rate = 3 and dropout rate = 0,3 .

6. Resulting filters are reduced to 256 with a 1x1 Convolution Layer of same padding.
7. One Convolution Block applied on the output of step 6 with 256 filters, same padding, dilation rate = 3 and dropout rate = 0,3 .
8. The output of step 7 is flattened and 2 Dense layers are applied with 256 and 128 neurons respectively. The dense layers use batch normalization, the leaky relu activation and a dropout rate of 0,3.
9. Finally a Dense layer is applied with 10 neurons, as the number of classes that it needs to predict, and the Softmax activation is applied since only one class can occur for each sample.

Model Training and Evaluation

The mini-batch gradient descent method is employed for the optimisation. The mini batch size parameter refers to the number of training examples used by optimizer in one forward / backward pass. This method reduces the variance of the parameters update and often leads to more stable convergence. The mini-batch size ranges often between 50 and 256 in the literature (Ruder, 2016). The selection of batch size is a often trade of between training time and accuracy (Thoma, 2017):

- **Training time:** The lower is the batch size, the longer is the training time needed for convergence. If the batch size is very big though, the training time is also higher than the minimum.
- **Model accuracy:** The lower is the batch size, the higher is the model's accuracy due to developing improved generalization ability.

The mini batch size was selected to include 128 examples by empirical practice.

Moreover, three different optimizers were employed for model training. These are the following: SGD, $Adam$ and $Rmsprop$. The last two optimizers compute adaptive learning rates for each parameter. Rmsprop divides the learning rate by an exponentially decaying average of squared gradients. Adam stores an exponentially decaying average of past squared gradients (like rmsprop) and also keeps an exponentially decaying average of past gradients similar to momentum.

All optimizers were trained for the original mnist dataset and later the best optimizers (rmsprop and adam) were re-trained for the augmented data in order to enhance learning capability. Moreover, all the models were trained using Categorical Crossentropy as the cost function and they were evaluated based on the categorical accuracy metric. The training stage is described below:

Initial Models

1. The first model was trained using the Stochastic Gradient Descent optimizer which achieved a 88% validation and 87% test accuracy.
2. The second model was trained using the Root Mean Square Prop optimizer which achieved a 91,78% validation and 90,85% test accuracy.
3. The third model was trained using the Adaptive Moment Estimation optimizer which achieved a 91,58% validation and 91,06% test accuracy.

Data Augmentation Models

Here the two best optimizers were chosen, rmsprop and adam, and were trained from the beginning on augmented data. The data augmentation was implemented using the ImageDataGenerator function of keras and used the following augmentations:

- Rotation up to 45 Degrees
- Horizontal Flip
- Vertical Flip
- Zoom up to 20%

1. The fourth model was trained again using the Rmsprop optimizer but on augmented data. It achieved a 92,35% validation and 91,43% test accuracy.
2. The fifth model was trained again using the adam optimizer but on augmented data. It achieved a 91,88% validation and 90,88% test accuracy.

The Rmsprop model with augmented data was chosen since it seems to have the best performance over all networks trained. Its training history is depicted in figure 4 and its confusion matrix on the test data on Figure 3. It seems that the model has a hard time differentiating classes 7 and 1 (T-Shirt from shirt). It is also clear that the model fails in differentiating some coats and bags as seen on the 7th and 9th colums of the confusion matrix. The class labing is provided below:

1. T-shirt/top
2. Trouser
3. Pullover
4. Dress
5. Coat
6. Sandal
7. Shirt
8. Sneaker
9. Bag
10. Ankle boot

```
array([[436,   0,   1,   3,  15,   1, 457,   0,  87,   0],
       [  0, 949,   0,   1,   8,   0,  12,   0,  30,   0],
       [  4,   0, 400,   1, 156,   0, 385,   0,  54,   0],
       [  1,   2,   1, 543, 129,   0, 160,   0, 162,   2],
       [  0,   0,   5,   1, 878,   0,  92,   0,  24,   0],
       [  0,   0,   0,   0,   0, 976,   0,   0,  23,   1],
       [ 10,   0,   2,   2,  49,   0, 897,   0,  40,   0],
       [  0,   0,   0,   0,   1,  38,   1, 822, 133,   5],
       [  1,   0,   0,   0,   0,   0,   1,   0, 998,   0],
       [  0,   0,   0,   0,   0,  15,   2,   8, 117, 858]],
```

Figure 3: Augmented Rmsprop Model Confusion Matrix

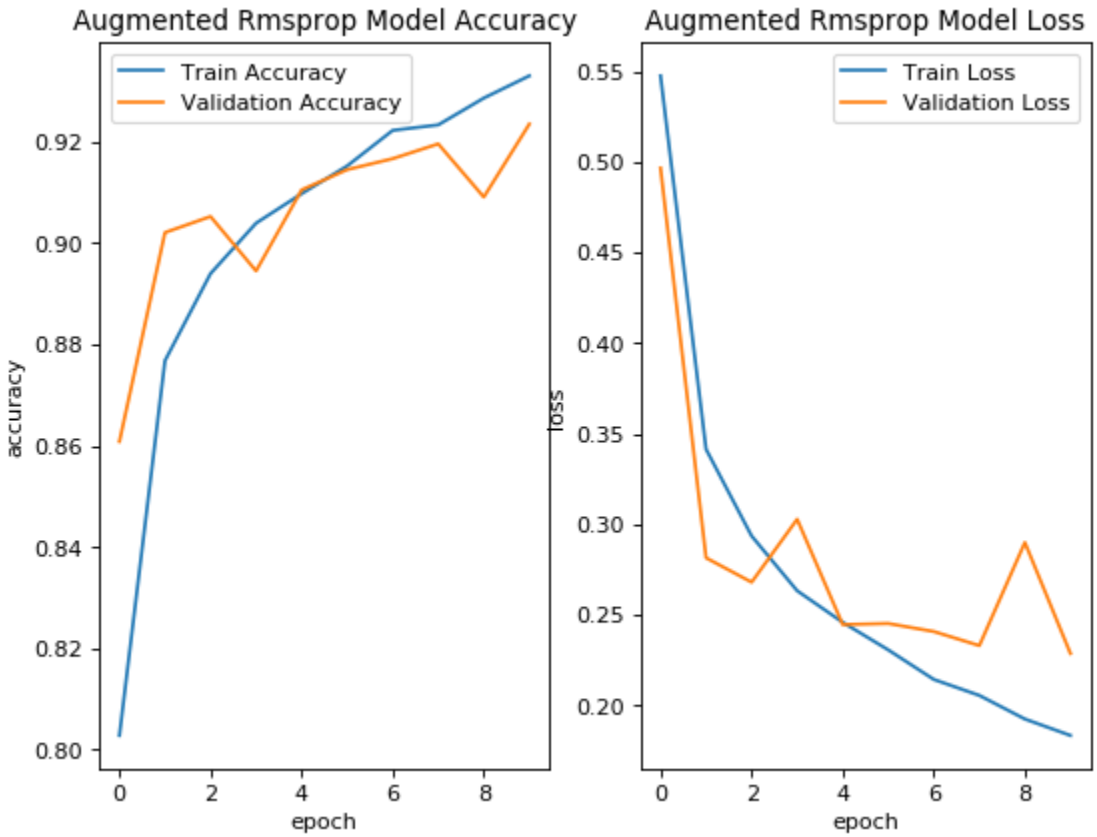

Figure 4: Augmented Rmsprop Model Training History

Challenges Tackled

1) Model complexity

The initial architecture employed for training included (5) convolution layers. However, it was observed that the accuracy of the model could not exceed 90% without overfitting the training data. Therefore, we reduced the number of convolution layers by one and re-trained the model with finally (4) convolution layers.

2) Model generalization

The drop out technique (30 %) was employed in order to deal with the overfitting problem that was initially noticed by the training vs validation accuracy plot per epoch.

3) Model accuracy

Augmentation of data was implemented in order to increase validation accuracy. However, the increase was not significant, and one possible reason could be that the evaluation images do not differ greatly from the training data, so the network couldn't leverage what it had learned from the augmentation.

Conclusions

All above demonstrate that both Adam and Rmsprop models trained with augmented data have good performance in general although each model has specific weaknesses to predict certain classes from the dataset. A proposal for improvement would be to use an ensemble prediction method using the strongest characteristics of each model in order to enhance predictive capability for specific classes and improve the accuracy in total.

Bibliography

Malakasiotis, P., 2018. *Deep Learning.* Athens: Athens University of Economics and Business.

Ruder, S., 2016. *An overview of gradient descent optimization algorithms.* [Online]
Available at: http://ruder.io/optimizing-gradient-descent/index.html#gradientdescentvariants
[Accessed 25 Dec 2018].

Thoma, M., 2017. *Analysis and Optimization of Convolutional Neural Network Architectures.* Karlsruhe: Karlsruhe Institute of Technology.