Advanced topics in Deep Learning - Mid Semester Project

Ori Malca (ID. 315150599),Kfir Sitalkil ID. 208722660)

Submitted as mid-semester project report for Advanced topics in Deep Learning course, Colman, 2023

1 Introduction

In our project, we asked to build a model that will be able to classify fashion MNIST data set which contains 70,000 images of 10 different classes. Each class represents a different item from different clothes.

1.1 Data

The data we are working on is fashion MNIST dataset, we got the data-set from keras library. The data-set contains 70,000 images (10 classes, 7,000 images for every class) , each image size is 28X28.

1.2 Problem

Our mission was to build a classifier that gets an input image from fashion MNIST data-set and outputs the class the item belongs to, with high accuracy rate. Also, in the project we been limited for building a fully connected network (without convolutional layers).

2 Solution

2.1 General approach

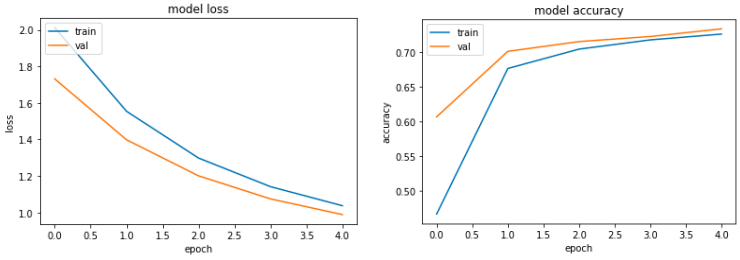
Our approach was to re scale the data set images, divide the training data to training set(80 percents) and validation set (20 percents).
We encoded the different labels to one hot encoding form.
We trained our model with 32 images per batch.

2.2 Design

Our model implemented with keras library.
The training time was not that long because there's no big amount of data, also we noticed that the model is not improving at some point when we use a big epoch number, so we were able to solve the problem with small amount of epochs which also contributed for the low training time.
The technical challenges we were facing was to implement the network without convolution layers, which prevented from us the option to make a good feature extraction based of the images. we believe that we could improve our performance even more with convolution layers.
the loss function we chose to use was categorical cross entropy, which is a convex loss function that can fix the model's weights in multi class problems.
Our final model's architecture contains 3 layers, the first layer is the input layer, contains 784 neurons as the number of pixels of each image. The second layer is a hidden fully connected layer with 512 neurons. The output layers is also a fully connected layer with 10 neurons, one for each class output.

2.3 Base Model

To build the base model, we used the same architecture that solved for us the binary classification problem with the same data set(fully connected input and output layers without hidden layers). to get some base architecture and try to improve it with the performance we get from the experiments.
Of course we could not use the architecture as is, the activation function at the output layer was sigmoid which is good for binary classification, we changed it to softmax so the network could handle multi class classifying.
Our intuition was that this model with our adjustments will be good enough as base model, because we're not dealing with a too complicated problem. The hyper parameters we chose are the following:
batch size: 32 (arribitary)
number of epochs: 5 (we're not facing a too complex problem, 5 should be enough)
optimizer: SGD
loss function: categorical cross entropy (convex loss function which can handle multi class problem)
The base model results and evaluation metrics:



Test set accuracy:

```
313/313 [==============================] - 1s 2ms/step
0.3648
```

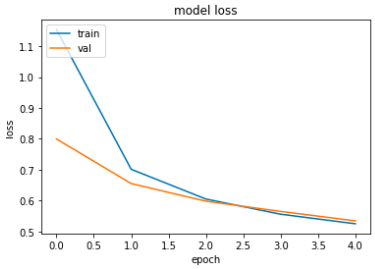
Base model evaluation metrics:

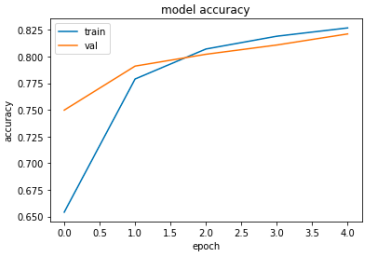
```
[[992   0   0   4   0   2   0   0   2   0]
 [241 755   0   4   0   0   0   0   0   0]
 [923   0  71   0   6   0   0   0   0   0]
 [673   0   0 327   0   0   0   0   0   0]
 [973   0   4   2  21   0   0   0   0   0]
 [618   0   0   0   0 330   0  33   0  19]
 [987   0   4   3   2   0   0   0   4   0]
 [346   0   0   0   0   7   0 623   0  24]
 [509   0   0   0   0   0   0   3 488   0]
 [237   0   0   0   0   0   0  21   0 742]]
```

	precision	recall	f1-score	support
0	0.94	0.29	0.44	1000
1	1.00	0.76	0.86	1000
2	0.90	0.07	0.13	1000
3	0.96	0.33	0.49	1000
4	0.72	0.02	0.04	1000
5	0.97	0.33	0.49	1000
6	0.00	0.00	0.00	1000
7	0.92	0.62	0.74	1000
8	0.99	0.49	0.65	1000
9	0.95	0.74	0.83	1000
micro avg	0.96	0.36	0.53	10000
macro avg	0.83	0.36	0.47	10000
weighted avg	0.83	0.36	0.47	10000
samples avg	0.36	0.36	0.36	10000

2.4 First Experiment

After testing the base model, we decided to add an hidden fully connected layer with 128 neurons to the model. At first we wanted our model to be light as possible, after the unsatisfying results, We decided to give our model more learn-able parameters to learn and maybe be able to solve our problem more efficiently. Also, we decided to change the activation function at the first layers from sigmoid to relu, to avoid vanishing gradients during backpropagation.
The changes we made increased our model accuracy while avoiding over fitting situation.
First experiment results and evaluation metrics:





Test set accuracy:

```
313/313 [==============================] - 1s 2ms/step
0.7469
```

First experiment evaluation metrics:

```
[[910   3   4  34   3   1  34   0  11   0]
 [ 40 924   7  24   4   0   0   0   1   0]
 [269   0 599   5  93   0  29   0   5   0]
 [198   9   1 760  16   0  14   0   2   0]
 [255   2  58  22 639   0  19   0   5   0]
 [ 85   0   0   0   0 837   0  51   2  25]
 [501   1  86  20  78   0 295   0  19   0]
 [ 47   0   0   0   0  24   0 877   0  52]
 [ 76   1   8   3   1   3   9   7 892   0]
 [ 20   0   0   1   0   9   0  46   1 923]]
```

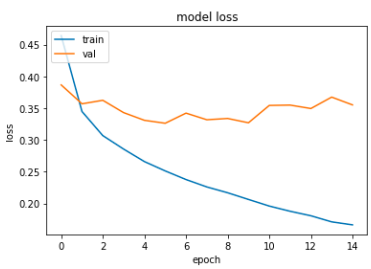
	precision	recall	f1-score	support
0	0.82	0.72	0.77	1000
1	0.98	0.92	0.95	1000
2	0.79	0.60	0.68	1000
3	0.87	0.76	0.81	1000
4	0.77	0.64	0.70	1000
5	0.96	0.84	0.89	1000
6	0.74	0.29	0.42	1000
7	0.89	0.88	0.89	1000
8	0.95	0.89	0.92	1000
9	0.92	0.92	0.92	1000
micro avg	0.88	0.75	0.81	10000
macro avg	0.87	0.75	0.80	10000
weighted avg	0.87	0.75	0.80	10000
samples avg	0.75	0.75	0.75	10000

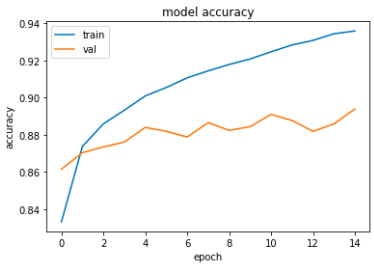
2.5 Second Experiment

At the second experiment, We chose to change the number of neurons at the hidden layer from 128 to 512 neurons. Also, we chose to change the optimizer from SGD optimizer to Adam optimizer and increased the number of epochs from 5 to 15 (we noticed that we can squeeze a bit more accuracy without getting to over fitting situation).
Adam is an extension of SGD that includes support for learning rates that are adaptive to the parameters. Adam uses a moving average of the parameters, which helps to eliminate oscillations and dampen noise in the gradients, making the optimization process more stable.
After we changed those parameters, we had significant improvement in our model, and we got our precision rates, also without getting to over fitting situation.
we chose this model to be our final model, and we will show the results in the next section.

2.6 Best model Results and Metrics

As mentioned at the above section, in our second experiment we got significant improvement which led us to our desired model performance. We chose the second experiment model to be our final model because his performance was satisfying enough.
Final model accuracy and loss:





Test set accuracy:

```
313/313 [==============================] - 1s 2ms/step
0.886
```

Final model evaluation metrics:

```
[[868   2  19  11   2   2  88   0   8   0]
 [  3 986   1   6   1   0   2   0   1   0]
 [ 40   2 856   4  54   0  43   0   1   0]
 [ 62   8   9 850  44   0  23   1   3   0]
 [ 37   0 115  14 791   0  39   0   4   0]
 [  2   0   0   0   0 945   0  44   1   8]
 [149   0  90  15  59   0 675   0  12   0]
 [  0   0   0   0   0   3   0 980   0  17]
 [ 10   0   0   1   2   1   3   4 979   0]
 [  0   0   0   0   0   6   1  39   0 954]]
```

	precision	recall	f1-score	support
0	0.86	0.84	0.85	1000
1	0.99	0.99	0.99	1000
2	0.79	0.86	0.82	1000
3	0.94	0.85	0.89	1000
4	0.83	0.79	0.81	1000
5	0.99	0.94	0.97	1000
6	0.77	0.68	0.72	1000
7	0.92	0.98	0.95	1000
8	0.97	0.98	0.97	1000
9	0.97	0.95	0.96	1000
micro avg	0.90	0.89	0.89	10000
macro avg	0.90	0.89	0.89	10000
weighted avg	0.90	0.89	0.89	10000
samples avg	0.89	0.89	0.89	10000

3 Discussion

In conclusion, after implementing our second neural network with keras with the limitation of using only fully connected layers, we really enjoined using keras frame work and see how easy it is to build a model if you understand the way that neural network works, the different meanings of the network architecture, hyper parameters, etc.
Also, we learned how important the convolution layers can be (at the previous, we were able to solve a more difficult problem with higher accuracy rate). Our insights from this project: Deep learning can be used to solve complex classification tasks, such as identifying different clothing items. Convolution neural networks (CNNs) are well-suited for image classification tasks, at this project we had the chance to learn and see how powerful they could be, but it was nice to see that we can solve the problem with fully connected layers only with satisfying performance. It is important to have a diverse and well-labeled dataset in order to train a successful deep learning model. Evaluating the model's performance on a holdout test set is crucial for getting a sense of how the model will generalize to unseen data.

4 Code

Link to colab notebook:
click here