Fashion MNIST Image Classification Project Report

Ivan Kulazhenkov
Ivan.Kulazhenkov@studenti.unipd.it

Andres Aranguren
andres.aranguren@studenti.unipd.it

1. Introduction

The number of images publicly available has skyrocketed during the 21st century. This has been caused by a number factors, including the internet boom as well as the rise of social media. This has been accompanied by parallel developments in image searching and analytics. The analytical side including but not limited to: object detection, labelling and scene identification. The wide availability of images has prompted the identification of the most powerful and accurate algorithms for the task. We would like to understand which algorithm out of a selected group(Logistic Regression, KNN, RandomForest, Neural Network(NN) and Convolutional Neural Network(CNN)) is the most effective in predicting items from our input dataset. For this task the fashion MNIST image database is adequate. We do not require complicated images initially as we would like our models to run quickly in order to better tune them and run comparisons between them. Once the best model is discovered we also want to understand what effect noisy/altered images have on the model's ability to predict correctly. This step is important as in providing the model imperfect examples of images we are hopefully improving its ability to generalize and learn unseen data. In real world applications we will often be processing images from sub-par/different data-set's hence it is important for a model performing image classification to cope with these imperfections. Ultimately our goal is to find the most robust model that is able to not only predict accurately but be flexible in image qualities that it encounters. Our initial hypothesis was that the CNN model would fit this description. After concluding our testing it turned out that we were correct. The CNN outperformed all linear models significantly. The closest to it was Random Forest and giving some serious tuning could potentially end up matching a CNN.

2. Dataset

Fashion MNIST data-set consists of a training set of 50000 images, validation set of 10000 and a test set of 10000. Each example is a 28x28 gray-scale image, associated with a label that ranges between 10 different classes consisting in types of clothing such as shoes, t-shirts, dresses, sandals and more. The data-set intended to serve as a replacement of the MNIST data-set used for benchmarking machine learning algorithms, since it is possible to routinely achieve error rates of 10 % or less [1].

The first step, to address this data-set is to develop a proper baseline, which involves developing the infrastructure for the test harness, so that any proposed model can be evaluated on the data-set. During this project test harness was divided in five key elements, loading of the dataset, preprocessing, definition of the model, evaluation of the model and presentation of results.

2.1. Preprocesssing

After loading the dataset, the first aspect to notice is that all the images are pre-segmented, meaning that every image contains a single piece of clothing, all have the same square size of 28*28 pixels.

Pixel value is a single value between 0 and 255 due to the images being gray scale ranging [0,1], which ensures that each pixel has a similar data distribution. Data normalization was carried out by subtracting the mean from each pixel and then dividing the result by the standard deviation. Data augmentation was also implemented which is done by altering images in the dataset.

3. Methods

3.1. Linear Model approach

First of all, we initially tested a combination of linear models employing various parameterization techniques to test their strength. By deciding to employ a set of almost default parameters we were able to identify the accuracy thresholds that each one of our tested models had.

1. Our first algorithm was Logistic Regression. We made changes to the default model to use the Stochastic Average Gradient as it produced the highest accuracy for us.

2. Our second algorithm was a KNN implemented with a nearest neighbour parameter of 7.

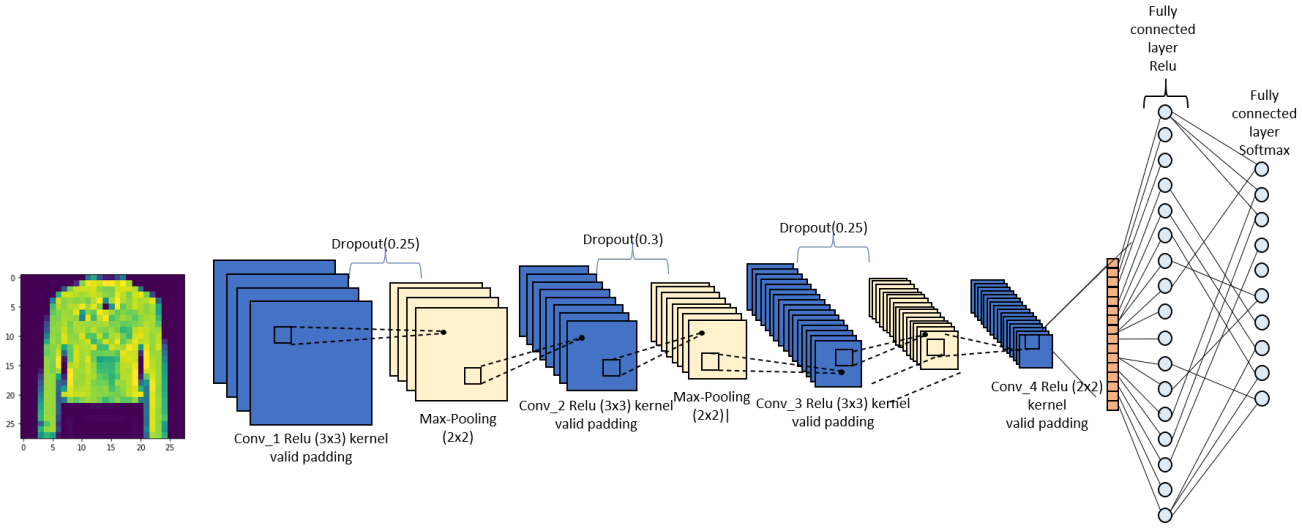

Figure 1. CNN Process Diagram

3. Our third model was a Random Forest classifier was running with gini impurity. Between the gini impurity and information gain entropy the accuracy was almost identical. Therefore, we used the less computationally expensive gini for our final random forest model. The max depth for our model was maximized at 80.

3.2. Neural networks approach

The first NN model that we implemented was a linear one layer model. This was a benchmark for more advanced models that we would go on to implement. The NN was also a good opportunity for us to test the effectiveness of different activation functions. We tested relu, tanh and sigmoid functions with our results demonstrating that relu was the strongest one. The best model was comprised of a single relu activated 128 node dense layer forwarding the predictions directly to the output neurons. The accuracy lacked yet we left this NN as a base layer to which we could compare our hypothetically more powerful CNN's.

After these initial linear models, we went on to begin our main testing. We predicted that even a single layer CNN's would outperform our baseline NN. What we wanted to observe was whether essentially out of the box it could also outperform our other linear models. Consequently, we left the parameters for our simple CNN almost identical to our linear one. We used a single layer convolutional layer followed by a max pooling 2d. This gave us a good starting point as the CNN already rivalled our strongest linear model (Random Forest). Our next question was how do we proceed to determine what could improve this basic model and take our accuracy upwards. After vigorous testing of four different CNN models we obtained our best. This final model can be seen in Figure (1).

In order to create the model we looked to incorporate a variety of different ideas. Most critically we increased the number of convolutional and pooling layers within the model. In total for the final mode we used 3 Conv2D and MaxPool2D layers with a final Conv2D layer and Flatten before output classification. This was the optimal model which we used as our skeleton.
From this point we focused on reducing model over-fitting as well as adding noise augmentation. As a final step we added batch normalization between our Conv2D and MaxPool2D layers.

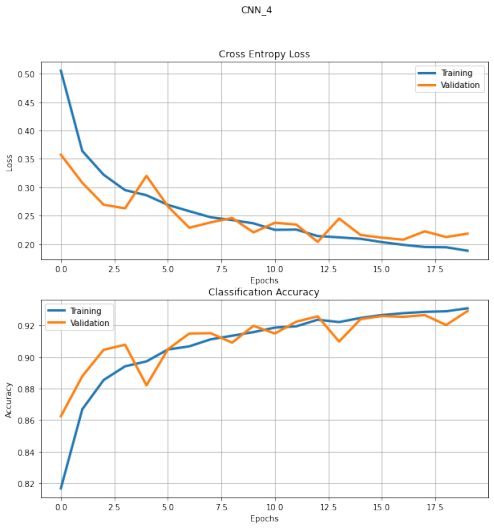

Figure 2. Final CNN Model train-val loss and accuracy graph

4. Experiments

4.1. Neural Networks results

The first parameters we studied were the activation functions. We compiled three models and applied the rectified activation function (ReLu), hyperbolic tangent function (Tanh), and sigmoid function. These models were then tested and validated using the training and validation data-set. Models' performance were evaluated based on the cross-entropy loss and classification accuracy over 20 iterations as seen in figure (2).

Relu activation function demonstrated higher accuracy compared to tanh and sigmoid functions. This can be explained by a general problem when non linear functions are applied to deep neural networks. It is their saturation which means that during back propagation steps large inputs snaps to 1.0 and small values to -1 or 0 for tanh and sigmoid respectively. The neural networks losses a lot of useful information to update the weights, a problem called vanishing gradient. Since ReLu acts like a linear function this is no longer a problem. The gradient will not reach a local minimum nor a non representative value that will block the algorithm's learning rate [2].

As expected the simple NN exhibits metrics similar to the linear method. This is caused by the loss of relevant image information when converting to 1D arrays and the number of neurons in the single layer.

The chosen hyper parameters with which we found our best model were, alpha = 0.001, batch size 32 and number of epochs=20.

Class	Class label	Precision	Recall	F1-score
T-shirt	0	0.88	0.90	0.89
Trouser	1	0.99	1.00	0.99
Pullover	2	0.86	0.91	0.89
Dress	3	0.95	0.91	0.93
Coat	4	0.87	0.88	0.88
Sandals	5	0.98	0.99	0.99
Shirt	6	0.82	0.78	0.80
Sneaker	7	0.98	0.96	0.97
Bag	8	0.99	0.99	0.99
Ankle Boots	9	0.97	0.98	0.98
Overall		0.93	0.93	0.93

Figure 3. Per Class Classification Accuracy

The final model architecture is shown in figure (1). Composed of four convolutional layers, three followed by max pooling operations tasked with extracting the relevant features of the input images.[5] Lastly one fully connected layer composed of 128 neurons leading to an output layer of 10 neurons. This layer was softmax activated in order to predict the final belonging class of the input image.

Two regularization approaches were appliedas well as four dropout operations which defined the fraction of neurons activated during forward propagation. We also used batch normalization between the convolutional layers and max pooling. This operation have been heavily demonstrated as capable of increasing model performance [3]. It is however, important to consider their proper placing in the networks' architecture in order to prevent optimization instabilities during weights update. A final precaution to avoid over-fitting and consequently decreased performance in the test data-set was the addition of Gaussian noise. This is added to the input variables, with a hyper-parameter setting the standard deviation to 0.1 [4]. Figure (2) describes the model performance during the fitting of the training and validation data sets. The figure shows accuracy peaking across 20 epochs at 93 %. Figure (3) shows the accuracy across the 10 labels within the validation dataset.

The last phase for model testing was applying k-fold cross validation which allowed us to perform re-sampling of the training and validation dataset. A k fold value of 5 was specified, computing the final precision as the average between the five data resamples, results are demonstrated in figure (5).

k-fold	Precision
1	93.06
2	95.13
3	95.21
4	94.96
5	96.43
Average	94.96

Figure 4. 5-Fold Cross Validation Precision Scores

4.2. Performance Comparison for all Models

Model	Regularization	Precision	Recall	F1-score
Logistic Regression n_iters=1000		0.84	0.84	0.84
Logistic Regresion n_iters = 5000		0.84	0.84	0.84
KNN n_neighbours=5		0.86	0.85	0.85
KNN n_neighbours=7		0.86	0.85	0.85
Random Forest max_depth = 70		0.88	0.88	0.88
Random Forest max_depth = 80		0.88	0.88	0.88
Linear Neural Network		0.82	0.84	0.82
CNN (single conv2d layer)		0.89	0.89	0.89
CNN (three layers)	Dropout (0.3)	0.89	0.89	0.89
CNN (three layers)	Dropout (0.3) Gaussian noise (0.1) Batch normalization	0.92	0.92	0.92
CNN (four layers)	Dropout (0.3) Gaussian noise (0.1) Batch normalization	0.93	0.93	0.93

Figure 5. Precision/Recall chart for all Models

References

[1] Ahmed Anter. Classification of garments from fashion mnist dataset using cnn lenet-5 architecture, 02 2020.

[2] Yujun Shi Chang-Yu Hsieh Benben Liao Shengyu Zhang Guangyong Chen, Pengfei Chen. Deep learning using rectified linear units (relu), 02 2019.

[3] Yujun Shi Chang-Yu Hsieh Benben Liao Shengyu Zhang Guangyong Chen, Pengfei Chen. Rethinking the usage of batch normalization and dropout in the training of deep neural networks, 05 2019.

[4] Agnieszka Mikołajczyk and Michał Grochowski. Data augmentation for improving deep learning in image classification problem, 05 2018.

[5] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 11 2015.