Assignment 5: Classifying Clothes

Deep Learning Assignment

UMGC Data 640

David Hannon

Professor Ed Herranz

**Introduction:**

In this assignment, a convolutional neural network will be utilized to perform deep learning on a dataset to classify articles of clothing. This will be done using Python, specifically the Keras package, and the imported dataset labelled Fashion\_MNIST. The dataset was designed to be a direct replacement for the standard MNIST dataset of hand written digits.

**Dataset:**

A picture containing graphical user interface

Description automatically generatedThe Fashion\_MNIST dataset contains a training set with 60,000 images and a test set of 10,000 images. Each image is taken from Zalando, an online retailers that is branded as a fashion platform. An overview of what the data looks like can be seen in figure A1, and a snapshot of an individual image is seen in Figure 1. All images are in greyscale and broken down to 28x28 pixels. The goal of the convolutional neural network is to input the data and correctly label it as one of 10 labelled clothing types. The options are t-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot. Figure 1 shows us an image of data that would be correctly labelled as a sneaker. The training data has an even breakdown of 6,000 images of each label and the test set has an even breakdown of 1,000 images of each type. To model these images, a matrix is created for each with the saturation of the greyscale represented by a number (white=0, darker shading=higher number). The CNN will use this matrix to search for patterns that help classify the image.

Figure

**Models:**

All models were created by making edits and adjustements  
 to a CNN model created by John Cook. The initial setup utilized 2 sets of layers, each containing a convolutional layer, a pooling layer, a dropout layer, and a normalization layer. The initial convolutional layer utilized 236 filters and a kernel size of 2, the pooling layer used a pool size of 2, and the dropout layer used a dropout rate of .2. The only change in the second set of layers is the number of filters in the convolutional layer drops to 128. The model is then flattened and a dense layer is created for the output. The model is then trained using 10 epochs with a batch size of 128. Due to the batch size, each epoch has 469 steps as that is the 60,000 inputs divvied up into 128 batches. After running this, the evaluation reports an accuracy of 69.8% with a loss of .875. This will be the baseline for comparing models as various inputs are changed to search for improvement.

Text, application

Description automatically generated Model 1 was created by adding a third set of layers (in the same format with the same parameters). Figure 2 shows the evaluation output for Model 1 while figure A2 shows the improvements made with each iteration of training. There is a decrease in accuracy from 69.8% at the start to 61.9% by adding a third layer set. Thus, further models will maintain the initial two sets of layers.

Figure - Model 1

 Model 2 maintained all features of the initial model except for the dropout was changed to 30%. This means with each iteration, 30% of the neurons are left untouched to prevent overfitting (was previously 20%). The accuracy and loss can be seen in Figure 3, reporting once again a decrease in accuracy. Model 3 chose to experiment with decreasing the dropout rate. Since decreasing it to too low of a level would allow for overfitting, it was only decreased to 15%. Doing this led to the results seen in Figure 4. It shows a slight increase in accuracy and decrease in loss from the initial base model and thus further models will maintain a dropout rate of 15% to maintain the increased accuracy while not pushing the overfitting.

Figure - Model 3

Figure - Model 2

In Model 4, both the kernel size and number of epochs were increased. The kernel size was changed from 2 to 4 with the mentality that a CNN may need larger inputs to notice patterns from clothing images as opposed to text. The epochs were increased simply because it seems the more passes though of the training data, the higher the training accuracy. Increasing this number comes with some caution of overfitting. The results of this model are shown in Figure 5. This shows a significant increase in accuracy; almost a 10% rise from the base model to 79.11% accuracy. The final step of the training data shows an accuracy of 78% as well so there are not signs of overfitting. Looking through the training accuracies per epoch, at the 10th iteration, the accuracy was 68.75%. Comparing this to all the previous models which only utilized 10 epochs, this is a decrease in accuracy and thus the improvement from this specific model should be more attributed to the increase in epochs rather than the increase in kernel size.

Figure - Model 4

Model 5 takes the changes made in Model 4 a step further. The kernel size is returned to 2, but the epochs remain at 50 and the batch size is decreased from 128 to 50, meaning the network is updated much more frequently within each epoch. The results are shown in Figure 6, reporting an increase in accuracy to 81.33%: the highest achieved yet and also the lowest lost. The final training accuracy was 79.95% so there are no apparent issues with overfitting. For comparison, the training at the 10th epoch was 70.11% which shows improvement over the other models at that point which could imply a decreased batch size will increase accuracy.

Figure - Model 5

A picture containing text

Description automatically generated Model 6 maintains the improvements that were found in Model 5 and tests out a different optimizer from the Keras package. The initial base model used the Adadelta optimizer which uses gradient descent to address the decreasing learning rates per epoch. For Model 6, the optimizer was switched to Adamax, which seems to be a bit less intensive than Adadelta. The output of the model returned Figure 7, showing 88.1% accuracy, a significant increase from the previous model. However, the final training accuracy was 90.97%, and while a 2-3% decrease from training to testing accuracy is not huge, it does leave the possibility of overfitting.

Figure - Model 6

 Lastly, Model 7 was created to see if changing the pooling size could have an effect on the outcome. It was increased from 2 to 4. This should improve the networks ability to notice key features and potential patterns among them. All other parameters remained the same as Model 6. The results are shown in Figure 8 with an accuracy of 86.9%, slightly below the results of Model 6. Once again, the results of the training data were a bit higher, at 89.9% accuracy but a small decrease does not necessarily imply overfitting.

Figure - Model 7

**Results:**

Of all the created models, the best results come from Model 6 which reported 88.1% accuracy in correctly labelling the images as 1 of the 10 different clothing articles. This is a significant improvement over the initial base model which reported 69.8% accuracy. These improvements are a result of increasing the number of epochs, decreasing the batch size, decreasing the dropout rate, and using Adamax optimization, while maintaining the same filters/units, kernel size, order of layers, number of layer sets, and pool size. The specific set up of layers for this CNN can be seen in Figure A3. It seems that the greatest improvements came when the number of epochs was increased, when the batch size was decreased, and when other optimizers were used.

**Conclusions:**

Given the 10,000 images in the Fashion\_MNIST dataset to label into 10 groups by article, one would expect an unlearned model to get 10% from just randomly assigning them as there were 1,000 of each type. Using deep learning, specifically a Convolutional Neural Network, this accuracy was increased to 88.1%. While this is obviously a major improvement, clothing stores such as Zalando may not find this high enough since this implies essentially 12 out of every 100 images would be mislabeled. On Zalando’ss website, they advertise that they are a meeting point for over 3,000 brands, so that would add up to be a lot of errors. While the current modelling does not have a clean was to look into the nature of these errors, it is not far fetched to assume that these errors would be caused by similarities shared between some of the labels. The most obvious pairings would be that the model could mistake sneakers, sandals, and ankle boots for one another, or make errors between the classification between a shirt and a t-shirt. While the distinction between these is important for customers to be able to filter search results the way they want, from a modelling standpoint, it may be best to streamline the classifications by grouping the clothing items that look alike, especially in shape, to improve the results of the CNN.

There are certainly other ways to improve the accuracy of the predictions as well. Specifically, increasing the number of epochs and decreasing the batch size further would likely improve accuracy with a bit of a risk of overfitting. For this model though, the concept of overfitting a bit may not be the worst thing since, as previously mentioned, many of these clothing categories are very similar and the very minute details that separate a sneaker from and ankle boot become very important. A faster processing speed may be needed to achieve optimum levels of epochs and batch size as these as the current levels took about 20 minutes to complete the selected model. Also, ideally there is someway to visualize the optimization of all the adjustable parameters, similar to that of the cutoff node in SAS such that optimum levels for all could be selected within the same model.

**References**:

Pranjal SrivastavaI am a Senior Undergraduate at IIT (BHU). (2020, May 18). Long Short Term Memory: Architecture Of LSTM. Retrieved November 03, 2020, from https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/

Research, Z. (2017, December 07). Fashion MNIST. Retrieved November 10, 2020, from https://www.kaggle.com/zalando-research/fashionmnist?select=fashion-mnist\_train.csv

Team, K. (n.d.). Keras documentation: Adamax. Retrieved November 10, 2020, from https://keras.io/api/optimizers/adamax/

Zalandoresearch. (2020, January 13). Zalandoresearch/fashion-mnist. Retrieved November 10, 2020, from https://github.com/zalandoresearch/fashion-mnist/blob/master/README.md

**Appendix**:

Background pattern

Description automatically generated

Figure A1

A screenshot of a cell phone

Description automatically generated

Figure A2

Text

Description automatically generated

Figure A3