USING CNNS TO DETECT FASHION IMAGES

Ishan Venkatesh
Monte Vista High School

January 6, 2025

ABSTRACT

This project explores the application of convolutional neural networks (CNNs) to categorize various images in the fashion-mnist dataset. The aim is to develop an AI model capable of classifying and interpreting a range of clothing items, such as coat, shirt, pullover, bag, trouser, skirt etc. Using the power of deep learning, particularly CNN models known for their use in image recognition tasks, the project focuses on building a robust image classification system. The project was done in segments, firstly an initial model was built without any overfitting countermeasures in order to measure baseline accuracy and other metrics. After the testing of the initial model, the model was optimized with various countermeasures to prevent overfitting and it's metrics were taken. Finally, the optimized model was used to predict an image not in its dataset.

1 Introduction

CNN or Convolutional Neural Networks are widely used for image classification because of their efficiency in processing pixels of data. CNNs give the ability to create models that can efficiently categorize and predict images based on patterns recognized within the training images. CNN's are different from regular nueral networks since they use forwards propogation as opposed to backwards propogation. Our model is trained from the fashion mnist dataset, which gives it the ability to predict various grayscale images.

2 Methodology

2.1 Environment

The environment used to set up this model was fully web based. The model was built using a python notebook in google colab. The dataset was loaded from google API and downloaded through google drive. The machine learning library Keras was used because its ease of use and powerful computing. Keras was chosen opposed to Pytorch as it is much simpler and efficient for a small scale project such as this one.

2.2 Dataset

The fashion mnist dataset was readily available in a Keras module, which made it simple to load and use. The dataset is of medium size, and contained roughly 22,000 images. A training and validation split was chosen for this project, with 80 percent of the images used for training, and 20 percent used for validation. Each of the images were 28x28 pixels and could be classified into 10 separate categories.

2.3 Initial Model

The initial model contained four layers, including a rescaling layer and three convolution layers combined with max pooling layers. A learning rate of 0.001 was chosen for the model as well as the optimizer being Adam. The model used a concrete number of 32 filters, with the size of each filter being a standard 3x3. In each convolution layer, the model had a ReLu activation function which is the standard, widely used activation function for this type of model. The initial

model was tested without any overfitting countermeasures such as data augmentation, dropout, or early stopping. This was done to measure its initial performance with a relatively simple model structure. To measure accuracy and other metrics, PLT library was used. After running the model for 20 epochs, the results and model summary are presented in the next section.

2.4 Model Optimization

From our observation of the initial model, it was evident that the model had started to overfit. In general, some strategies to prevent overfitting are increasing dataset size, data augmentation, adding dropout layers, adding regularization parameters, adding batch normalization, incorporating an early stopping, and simplifying the model. Based on our model's performance, the strategies to add data augmentation, dropout layers, and early stopping were chosen. A single dropout layer was incorporated in the last layer. Dropout layer simply removes certain chunks of data from the output layer at random. Also, Keras library supported a data augmentation function which was used for the model. Data augmentation changes the data randomly in order to introduce variation. Finally, an early stopping function was implemented which pauses and saves the model when its validation accuracy has decreased a significant amount. Like the initial model, a learning rate of 0.001 was used to progress down the loss function, and the optimizer Adam was used. After running for 23 epochs (as dictated by early stopping), the results and model summary are presented in the next section.

3 Results

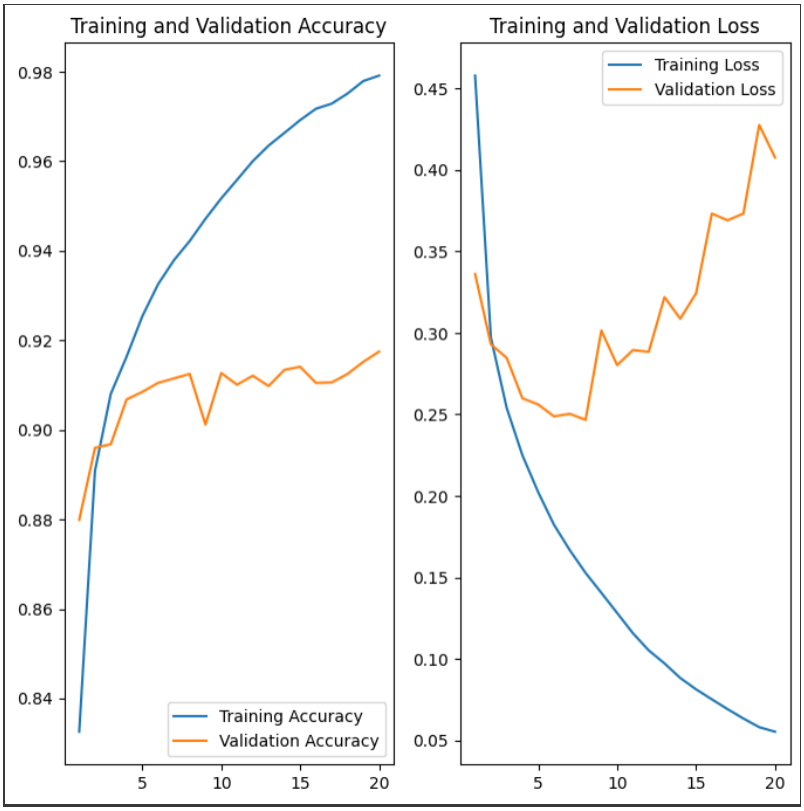
3.1 Initial Model

The initial model after training for 20 epochs was not a successful or efficient model. After a certain amount of epochs the model began to overfit at the 10 epoch mark. Final training accuracy was 0.98 and final validation accuracy was 0.91. This difference in values makes it evident the model was overfitting. On top of that, the difference between training loss and validation loss was significant as well. Around 10 epochs the validation accuracy started to decline while training accuracy continued increasing. This was the same case for the loss, with the validation loss increasing at around 10 epochs while the training loss continued decreasing. The final loss of the training set was 0.0500 and the final loss of the validation set was 0.4078. All in all, regardless of the simplicity of the dataset, the baseline model was clearly overfitting and required some changes in order to become effective in classifying the images.

Model: "sequential_11"

Layer (type)	Output Shape	Param #
sequential_10 (Sequential)	(None, 28, 28, 1)	0
rescaling_1 (Rescaling)	(None, 28, 28, 1)	0
conv2d_15 (Conv2D)	(None, 28, 28, 16)	160
max_pooling2d_15 (MaxPooling2D)	(None, 14, 14, 16)	0
conv2d_16 (Conv2D)	(None, 14, 14, 32)	4,640
max_pooling2d_16 (MaxPooling2D)	(None, 7, 7, 32)	0
conv2d_17 (Conv2D)	(None, 7, 7, 64)	18,496
max_pooling2d_17 (MaxPooling2D)	(None, 3, 3, 64)	0
flatten_5 (Flatten)	(None, 576)	0
dense_10 (Dense)	(None, 128)	73,856
dense_11 (Dense)	(None, 10)	1,290

Total params: 98,442 (384.54 KB)
Trainable params: 98,442 (384.54 KB)
Non-trainable params: 0 (0.00 B)



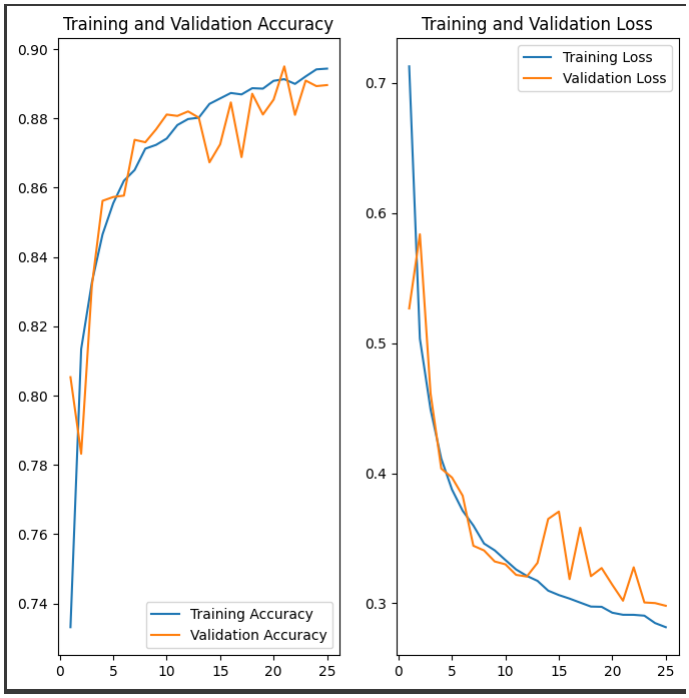
3.2 Final Model

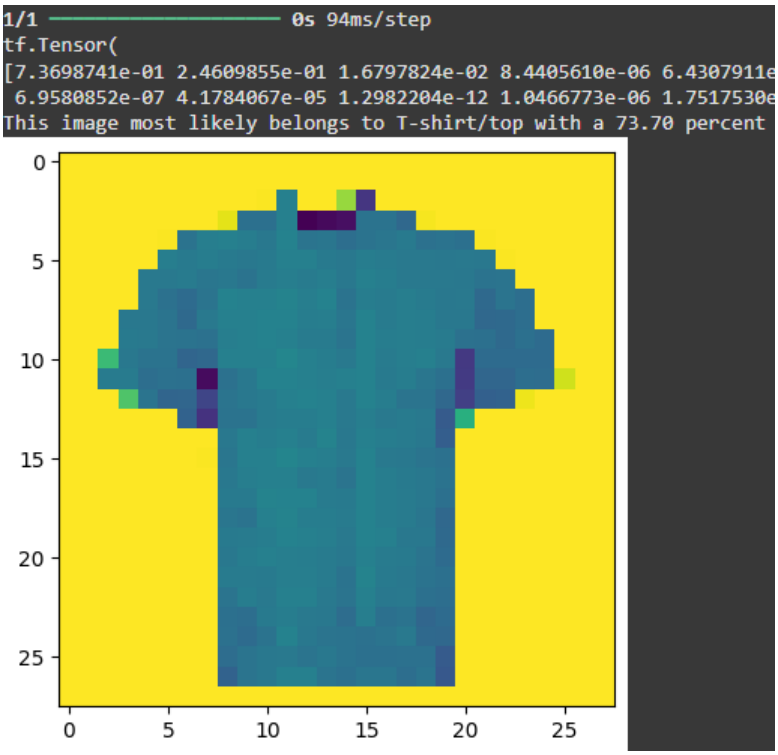
The improved model after training for 20 epochs was a successful and efficient model. The model did not overfit at all for the first 20 epochs. The validation and training accuracy had a small difference, as well as the training loss and validation loss. The final training accuracy was 0.895 and the final validation accuracy was 0.8896 This is a relatively small difference and can be interpreted as good results. The final training loss was 0.2814 and the final validation loss was 0.2981. The validation accuracy did start to decline after 25 epochs but the early stopping was able to find and save the best possible model. Our model was able to successfully classify and categorize images with a 90 percent accuracy and a 0.3 loss. When used to predict a new image not included in the dataset and never before seen by the model, it was able to correctly classify it with a 71 percent confidence score. All in all, the model was able to successfully avoid overfitting using the countermeasure implemented, and the validation accuracy rose consistently along with the training accuracy with little to no difference.

Model: "sequential"

Layer (type)	Output Shape	Param #
rescaling (Rescaling)	(None, 28, 28, 1)	0
conv2d (Conv2D)	(None, 28, 28, 16)	160
max_pooling2d (MaxPooling2D)	(None, 14, 14, 16)	0
conv2d_1 (Conv2D)	(None, 14, 14, 32)	4,640
max_pooling2d_1 (MaxPooling2D)	(None, 7, 7, 32)	0
conv2d_2 (Conv2D)	(None, 7, 7, 64)	18,496
max_pooling2d_2 (MaxPooling2D)	(None, 3, 3, 64)	0
flatten (Flatten)	(None, 576)	0
dense (Dense)	(None, 128)	73,856
dense_1 (Dense)	(None, 10)	1,290

Total params: 98,442 (384.54 KB)
Trainable params: 98,442 (384.54 KB)
Non-trainable params: 0 (0.00 B)





4 Conclusion

Overall, the results of our model demonstrate the concept of using Convolutional Nueral Nets in categorizing images from the fashion-mnist dataset. Using a relatively short time period of 20 epochs, which is roughly 45 mins of training, the CNN Model was able to improve its validation accuracy substantially. As forseen, the optimized model's performance was much greater than the initial model. The techniques used in this simple project applies to future

projects and gives insights on how to prevent overfitting on more complex machine learning models through the use of techniques applied on this model.

5 Future Work

In the future, continual improvement of the existing model in trying to increase the accuracy of the model. Future work might also include applications of this model with non-grayscale, complex images of various clothing items. While our Convolutional Neural Network (CNN) model achieved promising results on the Fashion MNIST dataset, several avenues remain open for further improvement and exploration. One potential direction is to experiment with more advanced CNN architectures. Currently, our model is relatively simple, and integrating more sophisticated techniques such as residual networks (ResNets) or DenseNet could lead to improved accuracy and model robustness. Exploring other techniques like hyperparameter optimization and regularization to prevent overfitting could also be another line of future work. In conclusion, there are many possibilities and areas of improvement for our current model, and working towards these would be extremely rewarding.