Analyzing Fashion-MNIST dataset

"Fashion Forward"

W200 – Project 2, Nov 2023

Trevor Lang, Ryan Powers, Carmen Liang

## 1 Introduction

The Fashion-MNIST dataset is used for benchmarking machine learning algorithms. The dataset contains 70,000 grayscale images of 10 different categories of fashion products, such as shirts, dresses, shoes, and bags. Each image is a 28x28 pixel labeled 0-9, corresponding to the type of fashion item it represents.

Fashion-MNIST has become a popular benchmark due to its complexity - there are subtle differences within categories. This tests a model's ability to learn discriminative features. Performance on this dataset indicates how well an algorithm can handle complex real-world image classification tasks.

Our project develops and compares Random Forest (RF) and Convolutional Neural Network (CNN) models for classifying Fashion-MNIST images. RF is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (for classification) or mean prediction (for regression) of the individual trees. CNN is a deep learning algorithm that is specifically designed for computer vision through hierarchical feature learning.

By comparing the performance of these algorithms on the Fashion-MNIST dataset, we aim to address the question below:

> ***How does the distribution of images across different classes in the Fashion-MNIST dataset affect the performance of Random Forest and CNN algorithms?***

Exploratory data analysis is used to investigate the dataset and identify issues that could influence the models.

## 2 Exploratory Data Analysis

### 2.1 Formatting Training, Test, and Validation Data Inputs

For both the convolution neural network and the random forest ensemble we started by loading the Fashion-MNIST dataset from the data repository. This dataset consists of fashion images (x) and their corresponding labels (y). We then split and shuffled the Fashion-MNIST dataset using a random state of 41 for reproducibility.

First, the images and labels were split and shuffled into training and test sets, allocating 20% of the data for testing (test size=0.2). Then, The training data and labels were then further split and shuffled, carving out 12.5% (test size=0.125) of it for validation. This division results in 70% of the data for training, 20% for testing, and 10% for validation.

We then prepared the image data and corresponding labels for training the models, ensuring the data was in the correct format and normalized for effective learning.

For the CNN, we reshaped the images to have rows, columns, and one color channel for grayscale input. This standardized the input shape required by the CNN architecture.

For the Random Forest, the images in the training, testing, and validation sets are also reshaped. We flattened the images into one-dimensional arrays as is commonly needed.

The pixel values of the images were then converted to 32-bit floating-point numbers and normalized for both algorithms. The normalization involved scaling the pixel values from a range between 0 and 255 to a range between 0 and 1 to facilitate the learning process of the neural network and the random forest ensemble.

For both algorithms, the class vector labels for the training, testing, and validation sets were then converted into binary class matrices using one-hot encoding. This process is used for categorical classification tasks, as it transforms the labels into a format that is suitable for training classification models.

## 2.2 CNN Model

A Convolutional Neural Network (CNN) was implemented using Keras/TensorFlow. The sequential model architecture consisted of:

- Two 2D convolutional layers with 32 and 64 filters of size 2x2, respectively, using LeakyReLU activations and 2x2 max pooling.
- Two dropout layers of 0.3 to reduce overfitting.
- A Flatten layer and two Dense Layers with ReLU activations and 256 and 10 units

The model was compiled with categorical cross-entropy loss, Nadam optimizer, and accuracy, precision, and recall metrics. It was trained for 30 epochs with a batch size of 256 on the training data (42,000 images), therefore the model weights were updated approximately 165 times for each of the 30 epochs.

The training data was used for model updates, test data to monitor performance during epochs, and validation data for post-training evaluation.

The CNN model achieved a final test accuracy of over 92.5% on unseen data, demonstrating its ability to effectively learn spatial features from images for classification.

### 2.3 Random Forest Model

The Random Forest Model was implemented using the scikit-learn library. The implementation involved the following steps:

- Initialize the classification with 100 decision trees and a random state of 41.
- Train the model on the training set using the fit function.
- Evaluate the model on the validation set using the predict function.
- Calculate performance metrics such as accuracy, precision, recall, and F1 score.

Random Forest trains decision trees on random subsets of the data and features and then combines their predictions to make a final prediction. This reduces correlation between trees and improves generalization.

During training, it learns patterns to make predictions on new, unseen data. Evaluation metrics such as accuracy, precision, recall, and F1 score measure how well the model is able to classify the input data.

Random Forest is particularly useful when data has many features or complex patterns. By combining tree predictions, it achieves high accuracy robustly.

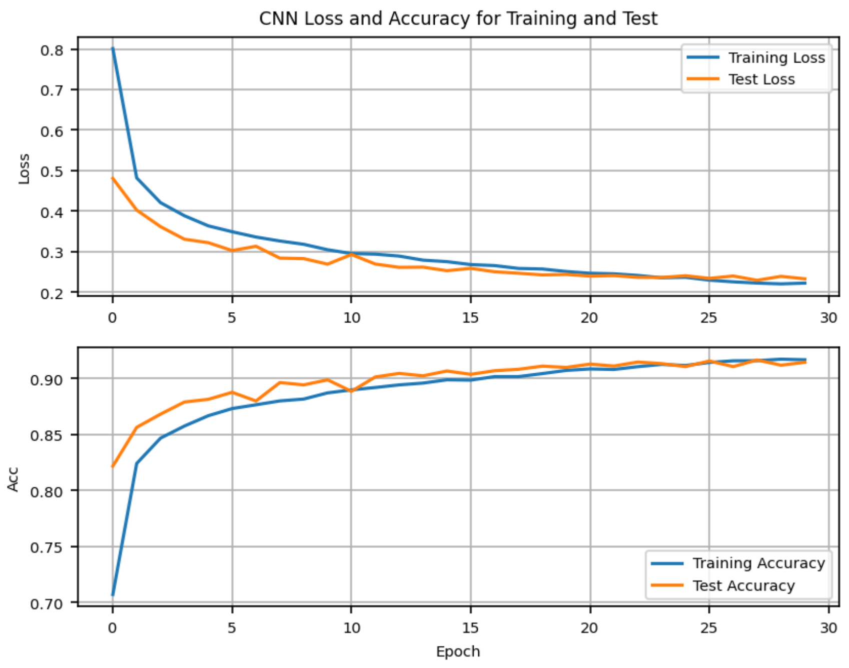
The Random Forest model achieved a final test accuracy of over 82.5% on unseen data, demonstrating its ability to effectively learn spatial features from images for classification.

## 3 Comparison

While the Random Forest classifier is a powerful machine learning algorithm, it may not be the best choice for image classification tasks. CNNs are a type of deep learning algorithm that are specifically designed for image classification tasks. CNNs are able to capture complex relationships between features in the dataset, which can improve their performance on image classification tasks. Additionally, CNNs are able to learn features directly from the raw pixel values, which can eliminate the need for manual feature engineering.

However, CNNs can be more difficult to train and require more computational resources than random forest classifiers. Additionally, CNNs may require more data preprocessing than random forest classifiers, such as data augmentation and normalization. Ultimately, the choice between a random forest classifier and a CNN depends on the specific requirements of the project and the characteristics of the dataset.

## 3.1 CNN Plots – during training

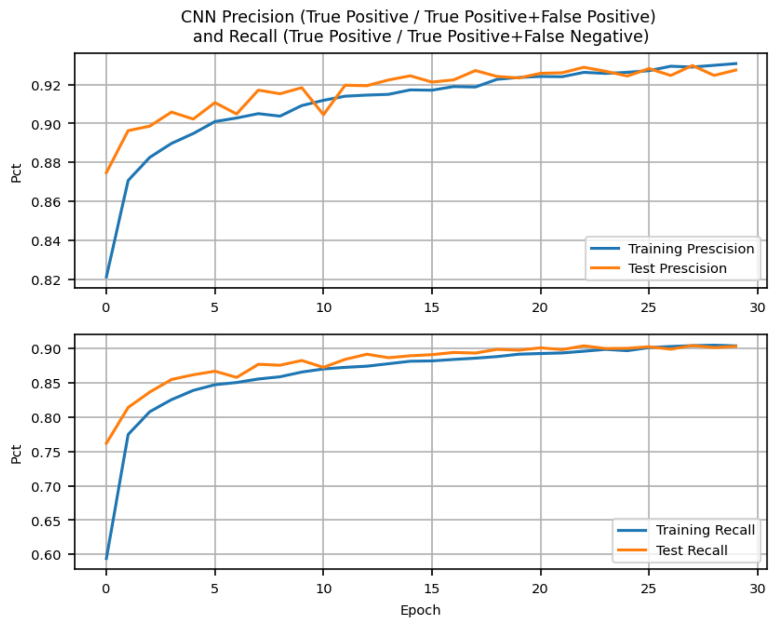


Categorical cross-entropy loss measures the difference between actual and predicted probability distributions by comparing predicted to true label distributions. Lower loss indicates better performance.

During training, loss value indicates how well predictions match labels: high loss means far from true classes, low loss means better performance.

Accuracy was also monitored, providing an intuitive measure of correctly classified instances but not capturing probability distribution nuances.

The plot shows loss and accuracy values for training and test sets during training. Loss decreases and accuracy increases with each epoch, indicating good model learning as training progresses. Both are good indicators of the model fitting the data distribution.

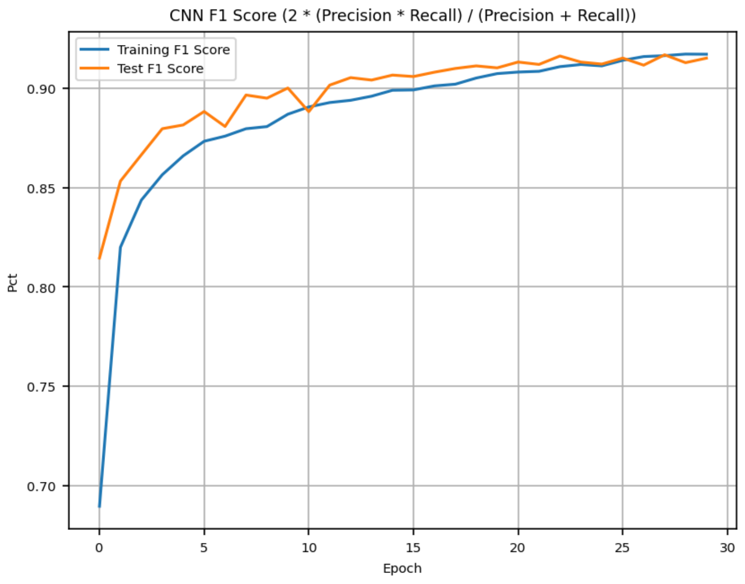


Precision measures the accuracy of positive predictions by calculating the ratio of true positives to true positives plus false positives. High precision means fewer false positives, important when false positives are costly.

Recall measures the ability to find all actual positive instances by calculating the ratio of true positives to true positives plus false negatives. High recall captures most positive instances, important when false negatives are detrimental.

In a CNN, precision and recall are key indicators of model performance. These metrics help in understanding how well the model is learning to classify the instances correctly.

The plot above shows precision and recall values for training and test sets during training. Both increase with each epoch, indicating the model is learning to better classify instances, as more true predictions are made and fewer false ones. This demonstrates good model performance over time.



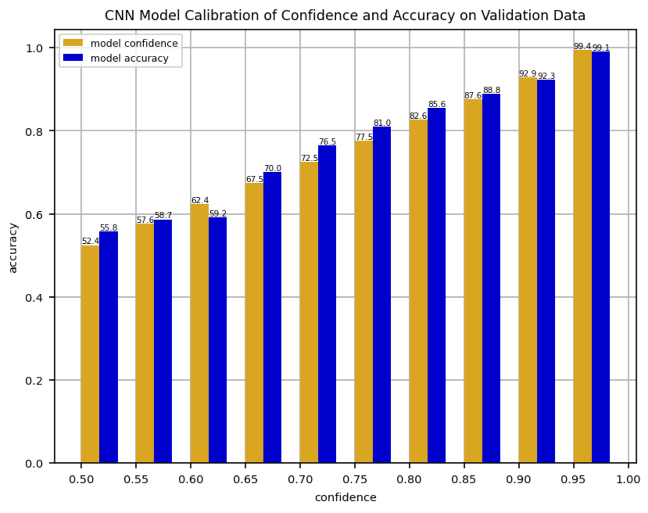
The F1 score evaluates classification models where balance between precision and recall is important.

It is the harmonic mean of precision and recall. Unlike an arithmetic mean, the harmonic mean gives more weight to lower values.

This means F1 will be low if either precision or recall is low, ensuring the model does not overly optimize one at the expense of the other. A high F1 score indicates a good balance between precision and recall, meaning the model performs well in both. During CNN training, monitoring F1 can guide adjustments to balance precision and recall, through architecture, hyperparameters, or threshold tuning.

The plot shows F1 increasing with each epoch, demonstrating growing balance between precision and recall as training progresses. This is a good indicator of model optimization.

## 3.2 CNN PLOTS – POST TRAINING

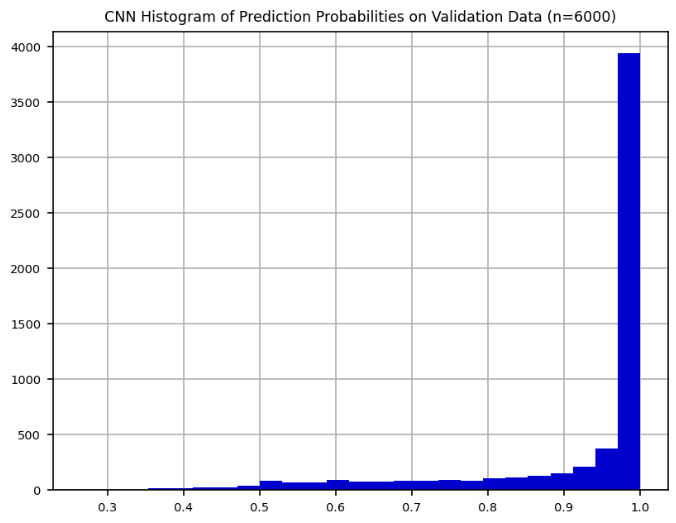


The plot on the left shows the average predicted probability (confidence) and average accuracy for a given confidence interval between 50 and 100 percent at 5 percent increments. The validation dataset was used to evaluate the calibration to ensure that only unseen data was used for evaluation.

The expectation is that at 50 percent confidence, you would expect a calibrated model to be close to

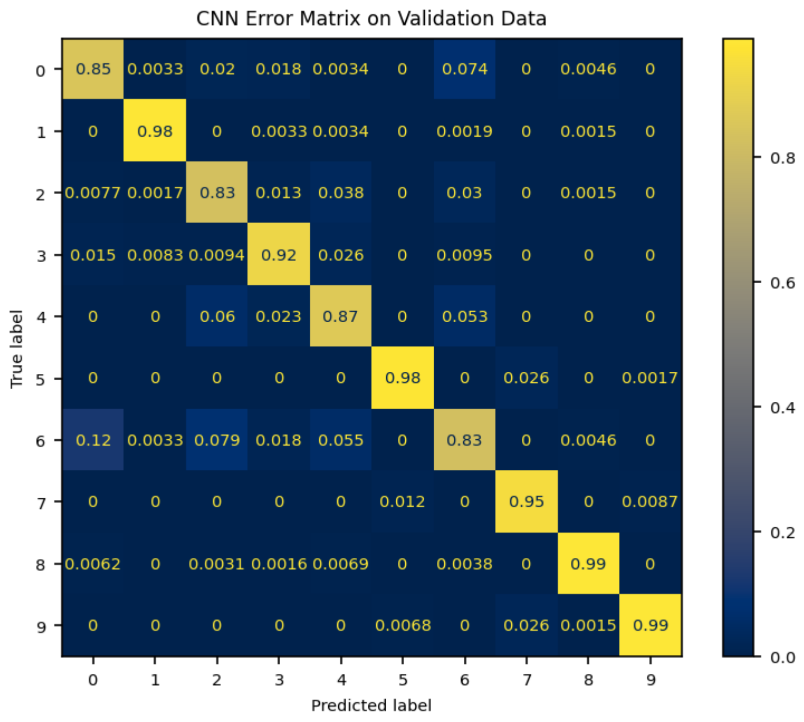
50 percent accurate. This assumption holds for all other confidence intervals as well.

The plot shows the average confidence and accuracy values are reasonably close for each confidence interval, indicating a well calibrated model.



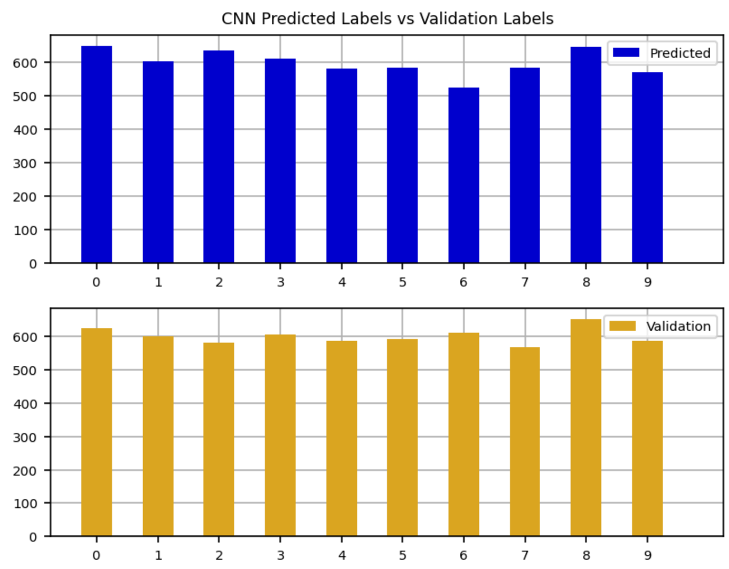
The plot on the left shows a histogram of predicted probability for all 6000 images of the validation dataset. Again, the validation dataset was used to ensure that only unseen data was used for evaluation.

This plot was selected to compare with the previous plot to show that while the model is calibrated across confidence intervals, the raw counts show that the model tends to be confident in its predictions.



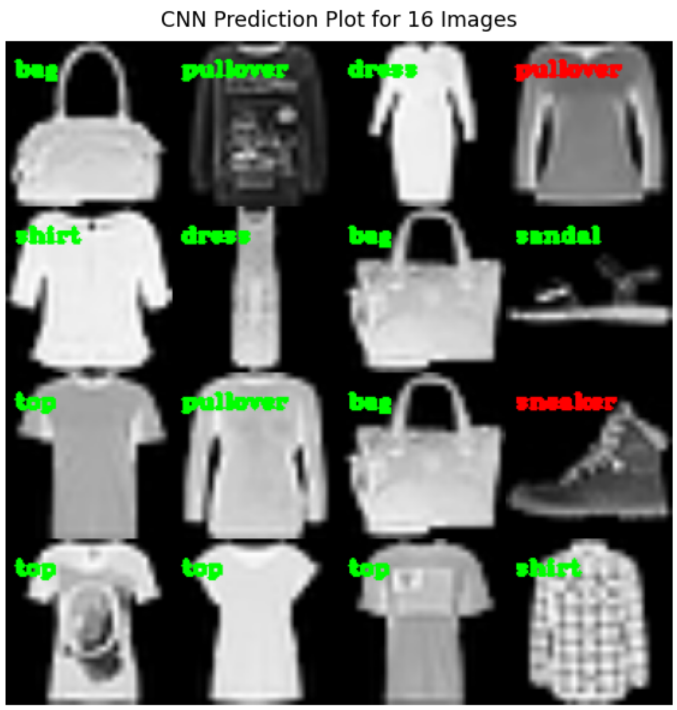
The plot on the left is an error matrix, also known as a confusion matrix, which represents the percent of prediction across all classes when compared to the validation labels. The Y axis represents the true class values and the X axis represents the predicted class values. Again, the validation dataset was used to ensure that only unseen data was used for evaluation.

The line where X = Y represents the correct prediction, with the values inside each box representing the percent of predictions for a given class (X axis) compared to the actual class (Y axis). The information above indicates the model has a more difficult time identifying classes 0 (“top”), 2 (“pullover”), 4 (“coat”), and 6 (“shirt”). Further follow on analysis will need to be conducted to see why the features in these classes are easily misidentified.



The plots on the left show histograms of predicted classes and actual classes for all 6000 images of the validation dataset. Again, the validation dataset was used to ensure that only unseen data was used for evaluation.

This plot was selected to compare with the previous plot to show that the data set itself has a reasonably distributed set of images across all classes.



The left plot is a mosaic of 16 random images taken from the validation data set. Each image was passed into the model to predict the class the image belongs with. Green text indicates a correct classification and red text represents an incorrect classification. This plot was selected to show a sample of what the image data looks like and how the model performed.

### 3.3 RF PLOTS - POST TRAINING

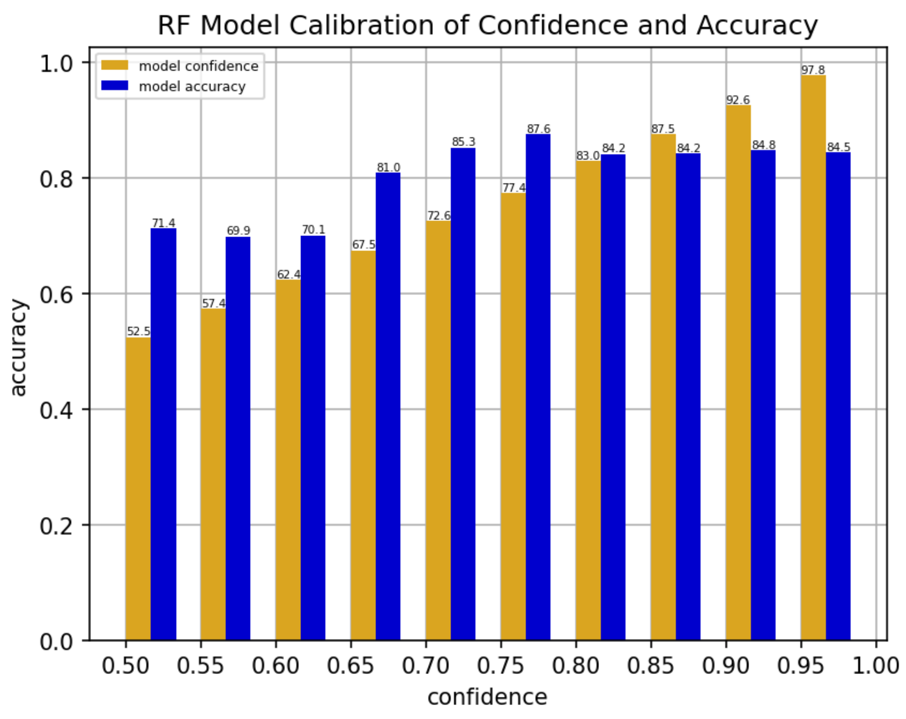
Unlike CNNs, random forests do not use epochs, batches or loss during training. Instead, they train using multiple decision trees. The scikit-learn implementation did not have the functionality to generate metrics during training. Instead we computed the overall values on the validation set after the training cycle. Below are the cumulative results for accuracy, precision, recall and F1 score.

RF Accuracy: 0.82

RF Precision (True Positive / True Positive+False Positive): 0.9309413547755834

RF Recall (True Positive / True Positive+False Negative): 0.8003333333333333

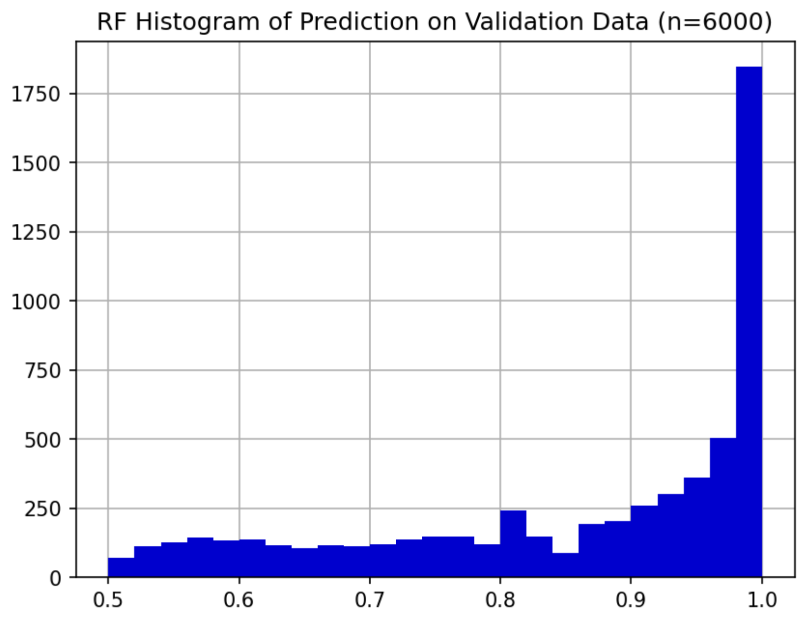
RF F1 Score (2 * Precision*Recall / Precision+Recall): 0.8607107846291333



The plot above shows the average predicted probability (confidence) and average accuracy for a given confidence interval between 50 and 100 percent at 5 percent increments. The validation dataset was used to evaluate the calibration to ensure that only unseen data was used for evaluation.

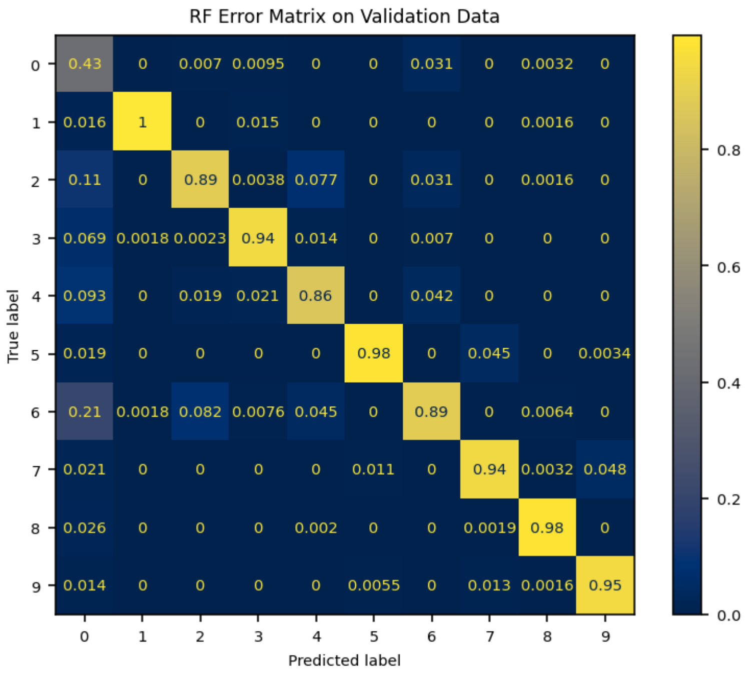
The expectation is that at 50 percent confidence, you would expect a calibrated model to be close to 50 percent accurate. This assumption holds for all other confidence intervals as well.

The plot shows the average confidence and accuracy values, which are not as close for each confidence interval as I would expect. The greatest positive difference was between the 50 to 55 percent confidence interval with an average confidence of 52.5 and an average accuracy of 71.4. The biggest negative difference was between the 95 to 100 percent confidence interval with an average confidence of 97.8 and an average accuracy of 84.5.



The plot on the left shows a histogram of predicted probability for all 6000 images of the validation dataset. Again, the validation dataset was used to ensure that only unseen data was used for evaluation.

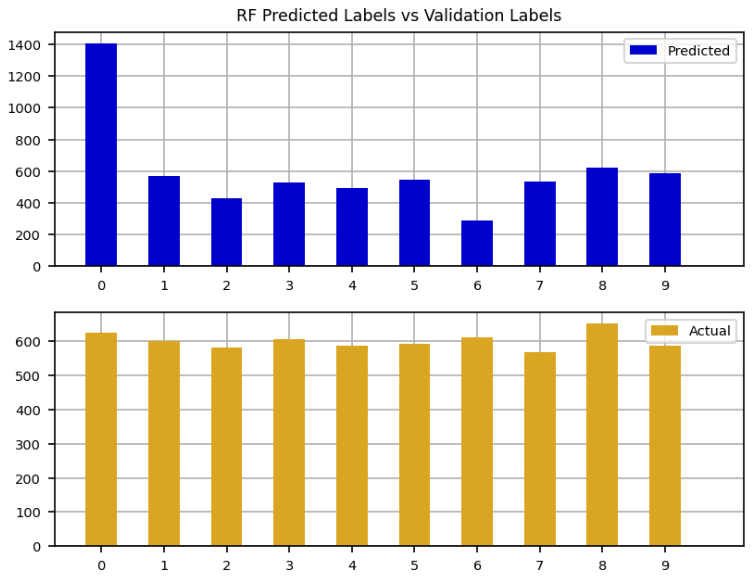
This plot was selected to compare with the previous plot to show that while the model is calibrated across confidence intervals, the raw counts show that the RF model tends to be confident in its predictions, but is also slightly less accurate than what it predicts in that confidence interval.



|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.43 | 0 | 0.007 | 0.0095 | 0 | 0 | 0.031 | 0 | 0.0032 | 0 |
| 1 | 0.016 | 1 | 0 | 0.015 | 0 | 0 | 0 | 0 | 0.0016 | 0 |
| 2 | 0.11 | 0 | 0.89 | 0.0038 | 0.077 | 0 | 0.031 | 0 | 0.0016 | 0 |
| 3 | 0.069 | 0.0018 | 0.0023 | 0.94 | 0.014 | 0 | 0.007 | 0 | 0 | 0 |
| 4 | 0.093 | 0 | 0.019 | 0.021 | 0.86 | 0 | 0.042 | 0 | 0 | 0 |
| 5 | 0.019 | 0 | 0 | 0 | 0 | 0.98 | 0 | 0.045 | 0 | 0.0034 |
| 6 | 0.21 | 0.0018 | 0.082 | 0.0076 | 0.045 | 0 | 0.89 | 0 | 0.0064 | 0 |
| 7 | 0.021 | 0 | 0 | 0 | 0 | 0.011 | 0 | 0.94 | 0.0032 | 0.048 |
| 8 | 0.026 | 0 | 0 | 0 | 0.002 | 0 | 0 | 0.0019 | 0.98 | 0 |
| 9 | 0.014 | 0 | 0 | 0 | 0 | 0.0055 | 0 | 0.013 | 0.0016 | 0.95 |

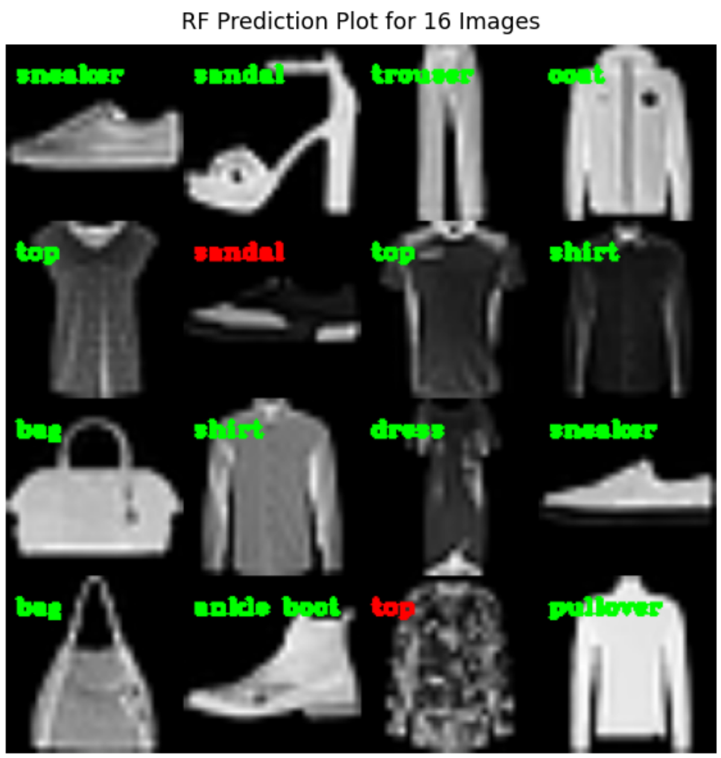
The plot on the left is an error matrix, also known as a confusion matrix, which represents the percent of prediction cross all classes when compared to the validation labels. The Y axis represents the true class values and the X axis represents the predicted class values. Again, the validation dataset was used to ensure that only unseen data was used for evaluation.

The line where X = Y represents the correct prediction, with the values inside each box representing the percent of predictions for a given class (X axis) compared to the actual class (Y axis). The information above indicates the model has a more difficult time identifying classes 0 ("top"), 2 ("pullover"), 4 ("coat"), and 6 ("shirt"). This is similar to the CNN model results, but in the case of class 0, the RF model performed significantly worse. Further follow on analysis will need to be conducted to see why the features in these classes are easily misidentified.



The plots on the left show histograms of predicted classes and actual classes for all 6000 images of the validation dataset. Again, the validation dataset was used to ensure that only unseen data was used for evaluation.

This plot was selected to compare with the previous plot to show that the data set itself has a reasonably distributed set of images across all classes and also highlights the significant incorrect predictions for class 0.



The plot above is a mosaic of 16 random images taken from the validation data set. Each image was passed into the model to predict the class the image belongs with. Green text represents a correct classification and red text represents an incorrect classification.

This plot was selected to show a sample of what the image data looks like and how the model performed.

## 4 Conclusion

The random forest classifier achieves an accuracy of approximately 82% on the validation dataset. The confusion matrix shows that the model performs well for most classes but needs help distinguishing between shirts and T-shirts/tops and between trousers and shorts. The CNN achieved a final test accuracy of over 91.8% on unseen data, demonstrating its ability to effectively learn spatial features from images for classification.

## 5 Appendix

http://yann.lecun.com/exdb/mnist/

https://github.com/zalandoresearch/fashion-mnist

https://developers.google.com/machine-learning/crash-course/ml-intro

https://developers.google.com/machine-learning/glossary

https://www.tensorflow.org/api_docs/python/tf/

https://scikit-learn.org/stable/modules/classes.html