Deep Learning for Fashion: Performance Evaluation of Random Forests and CNNs for Fashion Image Classification

Anirith Pampati Medha Majumdar Fatemeh Zare Amulya Dama Sriveda Methuku

***Abstract*—This project classifies the Fashion-MNIST dataset using Random Forest and Convolutional Neural Network models. A random forest classifier was initially used as a first attempt model, which, after several tuning parameters, achieved a maximum accuracy of around 87.80%. However, because of the inefficiency in feature extraction, the attention was diverted to CNNs for better performance. Multiple CNN models were developed, which ranged in complexity from simple single layer CNN networks to complex models that used convolutional layers, dropout, and learning rate adjustment. By making use of the TensorFlow and Keras libraries, the CNN models were highly improved, with the final model reaching a high increase resulting in a test accuracy of 93.72%. Proper preprocessing of the data were conducted including normalization and EDA. Model performances were evaluated using accuracy, precision, recall, F1 score, and confusion matrices. This study has been informed by insights on architectural enhancement and optimization strategies highlighted in the tutorial TensorFlow Keras. It also highlights the effectiveness of CNNs in image classification tasks and underscores their capability to address the limitations of traditional machine learning approaches.**

***Index Terms*—CNN, RandomForestClassifier, F1 Score, Confusion Matrix and Accuracy**

I. Introduction

A. Dataset Overview

We test a few individual test cases and see the following: Fashion-MNIST is a common benchmark dataset for image

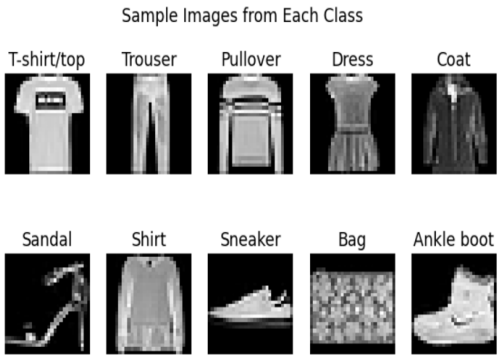

Fig. 1. Data samples from Fashion-MNIST

classification tasks. The total number of greyscale images in the dataset is 70,000, with 28×28 size, which is further divided into 60,000 training samples and 10,000 test samples. Each image is classified under one of the ten classes representing various types of apparel and accessories, such as T-shirts, trousers, sneakers, and handbags. This dataset is class-balanced and is considered a newer alternative to the MNIST dataset, presenting a more difficult task since intra-class variability is higher, while inter-class differences are lower.

B. Experiments

The project aims to classify images in the Fashion-MNIST dataset using both machine learning and deep learning techniques. The experimental process involves:

1) **Initial Baseline Model:** A Random Forest classifier was trained on flattened image data create a baseline performance to improve on. Different hyperparameters such as `n_estimators` and `max_depth`, were tuned to optimize accuracy.
2) **Transition to CNNs:** Due to the Random Forest's limitations in feature extraction and unable to achieve a high test accuracy, Convolutional Neural Networks (CNNs) were explored. Multiple CNN architectures were implemented:
   - A simple single CNN layer model with dense layers.
   - An improved CNN with convolutional and pooling layers for better feature extraction.
   - An advanced CNN model including dropout, batch normalization, and learning rate adjustment to manage overfitting and enhance generalization.
3) **Evaluation:** Models were evaluated using metrics such as accuracy, precision, recall, F1 score, and confusion matrices on validation datasets. The testing dataset were used on the final models selected. Graphs of training and validation metrics were also generated for visualization.

C. Literature Review

Fashion-MNIST has received considerable attention in the research area of image classification. This data set was for the first time introduced to the scientific community by Xiao et al. in their paper entitled "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms" [1], considering it an alternative data set to MNIST for benchmarking modern algorithms. Other works like Wang et al. [2] showed that CNNs can achieve high performance on this data set, which proves that their convolution layers are powerful in capturing spatial features.

The tutorial on TensorFlow Keras gave insight into the basic understanding of CNN-based methods: data preprocessing, design, and optimization strategy. Similarly, the techniques involving dropout and learning rate adjustment were extended to advanced CNNs in order to control overfitting and improve its generalization ability, as was discussed by Srivastava et al. [3].

These collectively informed the design of the experiment and methodology chosen in this work to achieve robust and effective performance in classification.

II. Method

A. Classifier Implementations

Firstly, a inital baseline model was based on a Random Forest classifier, whose ensemble-based structure provided a strong starting point in terms of classification of multi-class data labels. In order to overcome its limitation in feature extraction, CNNs were implemented to explore various architectural improvements for optimal performance. CNN models were implemented using the TensorFlow and Keras frameworks to allow for scalability and ease of experimentation.

B. Implementation

1) **Data Preparation and Analysis:**
   - The Fashion-MNIST dataset was loaded and separated to training (48,000), validation (12,000) and testing (10,000) image sets.
   - Pixel values were normalized and the images were flattened for Random Forest.
   - An extra channel was added for CNN preprocessing of images, retaining their 2D structure.
   - Exploratory data analysis was conducted, such as:
     - Class Distribution:

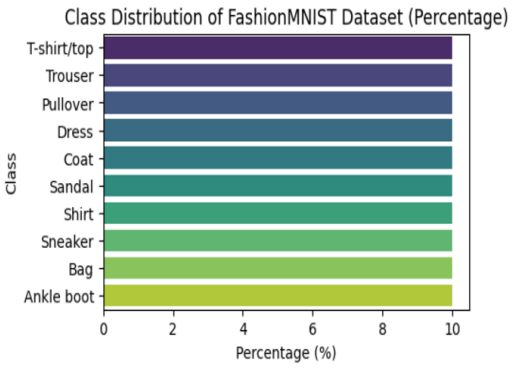

Fig. 2. Class Distribution of Fashion-MNIST Dataset

     - Statistical Analysis Box Plots:

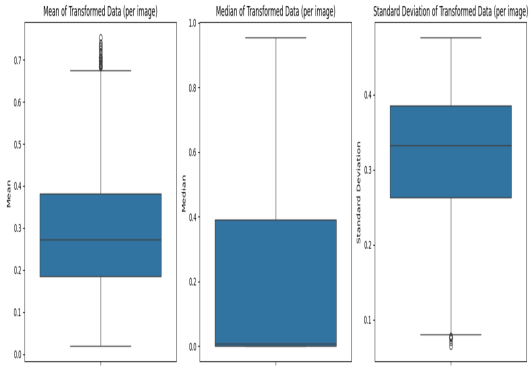

Fig. 3. Box Plots for Statistical Analysis of data per image

     - Correlation Matrix:

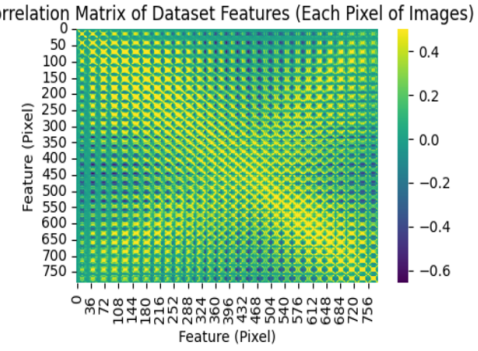

Fig. 4. Correlation Matrix for the Dataset

     - Scree Plot:
     - 3D PCA for the dataset:

More analysis reports can be found here.

2) **Random Forest Classifier:**
   - Basic RandomForestClassifier was initialized, attributes were checked and analyzed.
   - Hyperparameters were tuned like `n_estimators` (number of trees) and `max_depth` (maximum depth of trees).
   - Model was evaluated using metrics such as con-

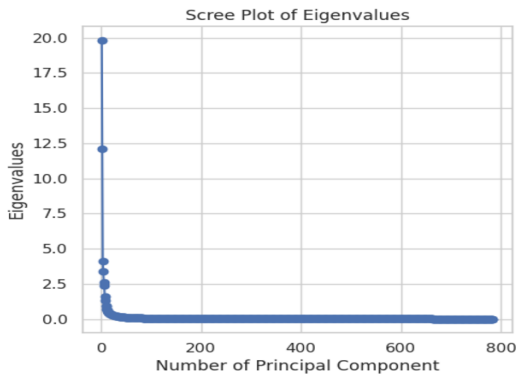

Fig. 5. Scree Plot of EigenValues

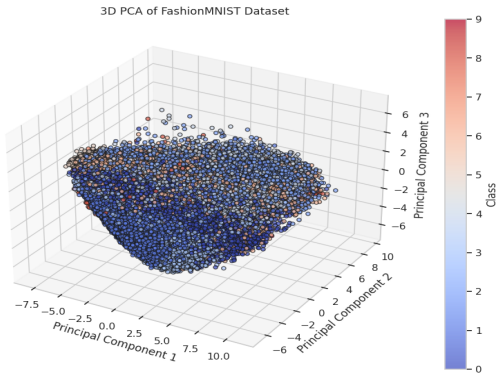

Fig. 6. 3D PCA for the Dataset

fusion matrix, accuracy, precision, recall, and F1 score.

3) **CNN Models:**
   - **Simple CNN:** A single CNN and pooling layer network.
   - **Improved CNN:** Incorporated multiple convolutional and max-pooling layers to extract spatial features.
   - **Advanced CNN:**
     - Multiple convolutional blocks with batch normalization and dropout were added to combat overfitting.
     - Adaptive learning rate using RMSprop optimizer was used.
     - Model was evaluated using metrics such as confusion matrix, accuracy, precision, recall, and F1 score.

C. Model Summary for the best model obtained

The model architecture starts with a 2D convolutional layer with 32 filters, followed by batch normalization and another convolutional layer with the same number of filters. Max pooling and dropout layers are applied at various stages to reduce spatial dimensions and prevent overfitting. As the network deepens, the number of filters in the convolutional layers increases, with 64 filters in the second block, 128 filters in the third, and 256 filters in the fourth. The final stages of the network involve flattening the output and passing it through two fully connected layers, the last of which produces the output with 10 classes.

```
Total params: 2,485,846 (9.48 MB)
Trainable params: 1,241,962 (4.74 MB)
Non-trainable params: 1,920 (7.50 KB)
Optimizer params: 1,241,964 (4.74 MB)
```

Fig. 7. Model Parameter details for the final CNN model

D. Equations

1) **Normalization**

$$x_{\text{normalized}} = \frac{x}{255} \tag{1}$$

where x is the original pixel value.

2) **Softmax Activation**

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \tag{2}$$

where z_i represents the logits for class i, and C is the total number of classes.

3) **Cross-Entropy Loss**

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log \hat{y}_{ij} \tag{3}$$

where y_{ij} is the true label, and $\hat{y}_{ij}$ is the predicted probability for class j of sample i.

4) **RMSProp**

$$v_t = \beta v_{t-1} + (1-\beta) g_t^2 \tag{4}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \epsilon}} g_t \tag{5}$$

In RMSprop, θ_t represents the model parameters at time step t, and g_t is the gradient of the loss with respect to θ_t. v_t is the moving average of the squared gradients, updated with the decay factor β (typically close to 1), while η is the learning rate, and ϵ is a small constant to prevent division by zero.

III. Experimental Results and Discussions

A. Random Forest Classifier and its Interpretability

A Random Forest Classifier is a type of ensemble learning that builds multiple decision trees during training. Each decision tree is trained on a random subset of the data via bootstrapping and random feature selection. In the case of classification, final predictions are made by aggregating the outputs of all constituent trees, usually by majority voting. In this way, overfitting is reduced, accuracy is enhanced, and the "wisdom of crowds" can be realized. Random forests are particularly effective in multi-class classification because they handle multiple categories by classifying samples into various classes and use majority voting for each. The robustness of random forests, their ability to handle high-dimensionality in data, and tolerance to noise and missing values ensure their ideal usability in complex classification problems, at the same time yielding very interpretable results.

We first use a default classifier. The goal is to have a baseline to improve our models on. We then move on to adjust various hyperparameters and obtain results corresponding to each of the changes, which we plot in the following graph:

After analyzing the graph, we select `n_estimators` = 500 and `max_depth` = 100 as our hyperparameters which are the best hyperparameters and obtain the following results on the test data: The goal of working through various hyperparam-

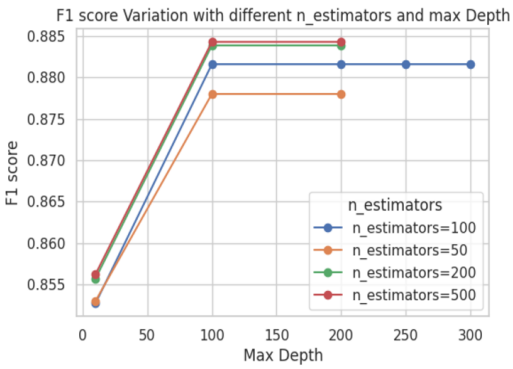

Fig. 8. Graph corresponding to various hyperparameter changes in RandomForestClassifier

Metric	Value obtained (%)
Accuracy	87.80
Precision	87.69
Recall	87.80
F1 Score	87.66

TABLE I
RANDOM FOREST CLASSIFIER FINAL CLASSIFICATION REPORT

eters and creating a graph was to visually compare each F1 score and check which one will be a best fit for the model. More detailed analysis of the Random Forest Classifiers can be found in the code. We were not satisfied with the test results obtained, so we tried using Convolutional Neural Networks for better accuracy results.

B. Convolutional Neural Network and its Interpretability

CNNs automatically learn hierarchical features such as edges, textures, and shapes through convolutional layers for multi-class image classification problems, which extracts the spatial relationships and sophisticated patterns from images. Unlike Random Forests, which do much better with structured data but have difficulties in working with spatial dependencies between pixels, CNNs are really good at object recognition because they learn from pixel-level features of images, so they could be the best for image-based classification. We chose the

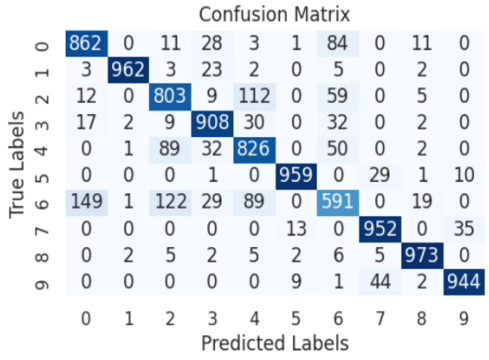

Fig. 9. Confusion Matrix

simplest CNN model, with one convolutional layer followed by a pooling layer, and then fully connected layers, using the softmax activation function for multi-class classification problems. The Adam optimizer was used-a variant of stochastic gradient descent that is efficient-and SparseCategorical-Crossentropy() as a loss function for integer-targeted multi-class classification. We ran the model to train up to 50 epochs. We assessed the performance of the model by plotting the loss: The goal is to see what a simple CNN model loss and accuracy

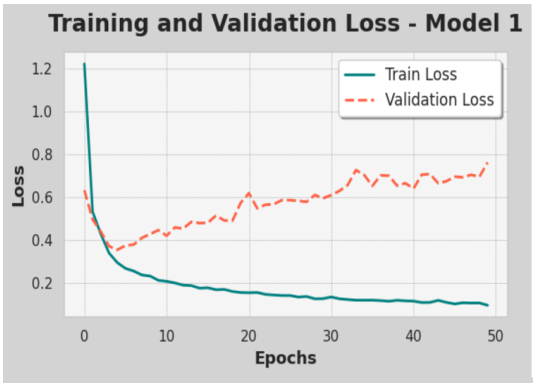

Fig. 10. Graph for training and validation loss: Model 1

top reaches are so that we can modify our models. We now create another model, with more CNN and Pooling layers so that we can extract more features. The graph we obtain after 50 epochs are: We observe that we are still overfitting our

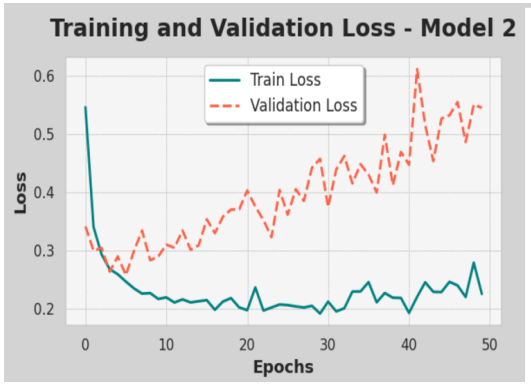

Fig. 11. Graph for training and validation loss: Model 2

model, so we try to reduce the learning rate and use batch normalization with an advanced CNN model so that we can get a better result. The goal of this model is to make sure our model learns generalized features, without overfitting. On running 50 epochs we find (detailed analysis in code): After we observe that our model does reach an accuracy above 90%, we use this model to train and analyze our test dataset and obtain the following metrics: We test a few individual test cases and see the following:

IV. CONCLUSION

These experiments show that for image classification tasks, CNNs outperform traditional machine learning models such

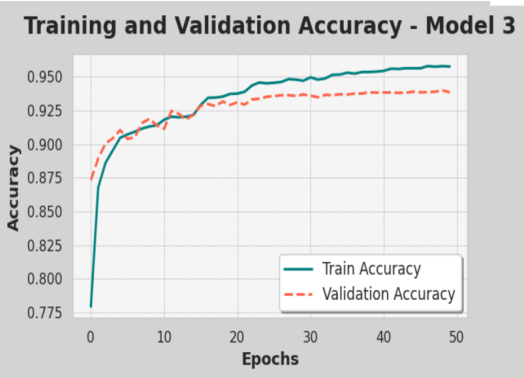

Fig. 12. Graph for training and validation loss: Model 3

Metric	Value obtained (%)
Accuracy	93.72
Precision	93.74
Recall	93.72
F1 Score	93.72

TABLE II
CNN FINAL CLASSIFICATION METRICS

as Random Forest. This is because CNNs make effective use of spatial relationships, which results in significant performance gains whereas Random Forest has its limitations. The training techniques are guided by Srivastava et al. [3] and Wang et al. [2], focusing on architectural considerations and regularization techniques. SpecificTensorFlow Keras tutorial guide was used to employ adaptive optimizers and dropout layers to guarantee efficient convergence and minimize overfitting.

A. Business Insights

The Random Forest Classifier is a very strong and accurate method applied to almost all industries, such as finance for fraud detection, e-commerce for product recommendations, and energy for predictive maintenance. With its ease of handling large, complex datasets containing missing values and preventing overfitting, it will be ideal for customer segmentation, market analysis, and sentiment analysis.

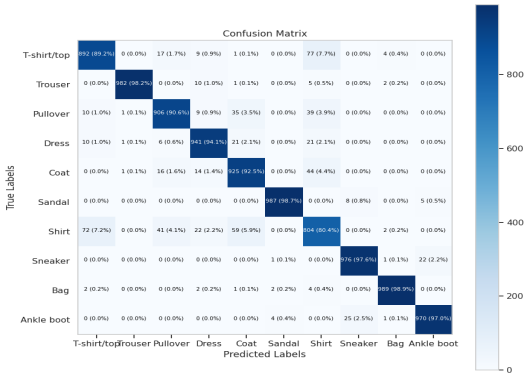

Fig. 13. CNN Final Model Confusion Matrix

Fig. 14. Test Results

CNNs help with image and video data analysis-such jobs as medical image analysis in healthcare, obstacle detection in autonomous driving, product classification in retail, and face identification problems in security. They efficiently accomplish complex tasks visually, thereby enhancing decision-making and user experiences across multiple industries.

B. Future Works

Several avenues for future exploration emerged from this study:

1) **Cross-Domain Fashion Classification:** Extending the Fashion-MNIST dataset to include real-world fashion images such as those from e-commerce platforms and applying domain adaptation techniques may help the CNN generalize better for more complex and diverse datasets.
2) **Multimodal Approaches:** Combining textual descriptions of fashion items with images (e.g., using multimodal models like CLIP [4]) may enhance classification accuracy and enable zero-shot learning.
3) **Optimization Techniques:** Incorporating techniques like Bayesian optimization for hyperparameter tuning or exploring advanced optimizers like Nadam could further improve performance [5].

REFERENCES

[1] X. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1708.07747
[2] Y. Wang et al., "Convolutional Neural Networks for Fashion Image Classification," International Journal of Computer Vision, 2019.
[3] N. Srivastava et al., "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," Journal of Machine Learning Research, vol. 15, pp. 1929-1958, 2014.
[4] A. Radford et al., "Learning Transferable Visual Models From Natural Language Supervision," in Proc. Int. Conf. Mach. Learn. (ICML), 2021.
[5] T. Dozat, "Incorporating Nesterov Momentum into Adam," ICLR Workshop Track, 2016.

Link to Github page: code