Project Report On Deep Network with Fashion MNIST

Carolina Li, CS 5330 Computer Vision, Fall 2024

***Abstract*—This project presents a deep convolutional neural network (CNN) to classify images from the Fashion MNIST dataset which is a benchmark for evaluating computer vision models on greyscale fashion categories. The dataset consists of 70,000 images each having a size of 28 by 28. Each example in the dataset is associated with a label from 10 classes. Implemented the CNN architecture which comprises three convolution layers integrated with batch normalization, ReLU activations, max-pooling and drop-out layers for effective regularization. Then, I implemented a fully connected layer with a softmax activation function as this problem is related to multi-classification. The softmax activation function produces class probabilities for each of the 10 classes which are our target labels in this scenario. The model has been trained using Adam optimizer and early-stopping in order to mitigate overfitting of the data. I implemented systematic evaluation of my training dataset by making it split into validation that ensures rigorous evaluation. I have also delved into visualization of the metrics like training accuracy with respect to validation accuracy, training loss with respect to validation loss.**

***Index Terms*—CNN, Deep Learning, Training**

I. Introduction

Image recognition and classification are critical tasks in computer vision, with diverse applications across industries such as fashion, retail, and e-commerce. This project focuses on the Fashion MNIST dataset, a benchmark dataset comprising grayscale images of 28x28 pixels, representing 10 categories of clothing and accessories, such as shirts, dresses, sneakers, and bags. The goal is to classify these fashion items accurately using a deep learning approach. Convolutional Neural Networks (CNNs) are powerful tools for image-related tasks, capable of automatically extracting spatial hierarchies of features from input images. By utilizing convolutional, pooling, and activation layers, CNNs learn complex patterns and features that are key to accurate classification. This project implements a CNN architecture optimized for the Fashion MNIST dataset, incorporating techniques like data normalization, augmentation, and dropout for enhanced performance and generalization.

The CNN architecture is generally responsible for learning spatial hierarchies of features as described below: First Convolution Layer: It applies multiple filters to detect basic patterns such as edges or textures. Second and Third Convolution Layers: These layers are responsible for building on previous layer's features that would lead to detect more complex patterns like shapes or parts of clothing items. Each Convolution layer is followed by batch normalization to stabilize and accelerate training and ReLU functions to introduce non-linearity. Pooling Layers: After each convolution layer, a max pooling layer is used to reduce the spatial dimensions of feature maps while retaining the most prominent and important features. Pooling layers also help in lowering of computational cost due to reduction of dimensions of feature map. To prevent overfitting, dropout layers are applied at strategic points, randomly disabling a fraction of neurons during training so that we can enhance the model's generalization. Fully Connected Layer: This is the final feature map that is flattened and passed to a fully dense connected layer, which maps high-level features to the 10 output classes belonging to fashion categories. Output Layer: This output layer employs a softmax activation function, which is responsible for producing probability scores for each of the 10 classes. The class which is having high probability would be assigned as the corresponding class of that example image. DropOut: Dropout is used for regularization. Training utilizes the Adam optimizer, which adapts the learning rate during the training phase, which helps in ensuring faster convergence of the model. The architecture that I implemented balances both depth and efficiency, enabling the model to capture the intricate patterns in fashion items while maintaining computational cost. Before applying the CNN architecture, I augmented the training dataset in the form of applying rotation, shifting height and width of images and applying zoom in and zoom out. I implemented a data augmentation technique in order to enhance the diversity of the fashion training dataset. It helps to recognize the patterns and features that are invariant to these transformations, it will enhance the model model to better generalize the unseen data. The combination of Convolution layers followed by pooling layers and drop-out layers ensures that my model achieves high performance on both training and unseen test data.

II. Related Work

The paper Fashion-MNIST authored by Han Xiao et al., implemented preprocessing of images like transformation, resizing, sharpening and grayscale conversion to create a uniform dataset.

The author Xion, Han et al., [1] employed traditional machine-learning algorithms like random-forest, MLP Classifiers, Logistic Regression that achieved varying degrees of accuracy on Fashion-Dataset which is lower as compared to MNIST dataset. With MLP Classifier, they have achieved an accuracy of 87.1% on Fashion-MNIST dataset and on MNIST dataset they have achieved an accuracy of 97.2%. That showed that the Fashion-MNIST dataset is having complex patterns as compared to MNIST dataset. With Random-Forest Classifiers, with 100 estimators and max-depth=100 achieved an accuracy of 87.3% on Fashion-MNIST dataset however, 97% on MNIST dataset.

The paper, authored by Mohammed Kayed et al., [2] implemented different CNN architecture like LeNet that demonstrated remarkable classification accuracy in the Fashion-MNIST dataset. They achieved an accuracy of 98% using LeNet-5 as compared to other existing models. They compared this model's accuracy with respect to traditional machine-learning models like Decision-Trees, ExtraTreeClassifier, GaussianNB, Support Vector Classifier with Radial basis function (rbf) kernel, LogisticRegression, Support Vector Classifier (SVC) etc. Even though they used CNN with batch-normalization and residual skip connections they were able to manage to achieve an accuracy of 92.4%. They used precision, recall and accuracy metrics to compare their suggested model with state of the art.

The paper, authored by Edmira Xhaferra et al., [3] demonstrated the comparison of two CNN-C1 and CNN-C2 models for the classification of the Fashion MNIST dataset. In their proposed solution, CNN-C1 was the conventional CNN model, however for CNN-C2 it was an enhanced CNN model tackling overfitting with hyperparameters tuning, batch normalization and dropout layers. The baseline model showed signs of overfitting however CNN-C2 model demonstrated superior performance with higher accuracy 93.11

III. Methodology

The primary goal of this project is to classify images from the Fashion MNIST dataset into one of ten categories using a Convolutional Neural Network (CNN). My approach incorporates a series of preprocessing, model design, training, and evaluation steps to ensure robust performance and generalization. The key methods are outlined below:

A. Data Preprocessing

The dataset was split into training (80%), validation (20

B. Data Augmentation

To artificially expand the training dataset and enhance generalization, random transformations were applied to the images, including: Rotation (up to 10 degrees), Shifting (up to 10% horizontally and vertically), Zooming (up to 10%). These augmentations were implemented using the ImageDataGenerator class in TensorFlow.

C. Batch Processing

Training, validation, and test datasets were processed in batches of 32 to efficiently manage memory and speed up training.

D. CNN Architecture

The CNN model was constructed using TensorFlow's Sequential API with the following layers: Input layer: Accepts 28x28x1 grayscale images. Three convolutional layers with 64, 128, and 256 filters respectively, using 3x3 kernels and ReLU activation. Each convolutional layer is followed by: Batch Normalization: To normalize activations and reduce internal covariate shift. Max Pooling: To reduce spatial dimensions while preserving features. Dropout (0.2, 0.3, and 0.4): To prevent overfitting by randomly deactivating neurons. A fully connected layer with 128 neurons and ReLU activation, followed by Batch Normalization and a Dropout of 0.5. Output layer: A dense layer with 10 neurons and a softmax activation function to predict class probabilities.

E. Model Training

The model was compiled using the Adam optimizer, sparse categorical cross-entropy loss, and sparse categorical accuracy as the evaluation metric. Early stopping was employed to halt training when validation loss did not improve for five consecutive epochs, restoring the best model weights.

F. Evaluation and Visualization

The trained model was evaluated on the test set to measure accuracy and loss. Training and validation metrics, including accuracy and loss, were visualized across epochs to assess model performance. A confusion matrix was generated to highlight classification performance for individual categories.

G. Reproducibility

All code for data preprocessing, model architecture, and training was implemented in TensorFlow and Python, ensuring reproducibility. The implementation follows principles outlined in the paper Classification of Standard Fashion MNIST Dataset Using Deep Learning Based CNN Algorithms by Edmira Xhaferra et al., [3] The source code for this project will be made available.

IV. Experimental Results

The experiments were conducted on the Fashion MNIST dataset, consisting of 60,000 training images and 10,000 test images, categorized into 10 classes of clothing and accessories. The dataset was split into training (80%), validation (20%), and testing sets. Images were normalized to pixel values between 0 and 1 and augmented using random transformations such as rotation, shifting, and zooming to enhance generalization. The CNN was trained for up to 20 epochs using the Adam optimizer, and managed via early stopping based on validation loss. Batch processing (batch size = 32) was employed to ensure efficient computation.

A. Loss Function: Sparse Categorical Cross Entropy

The Cross Entropy Loss function was used in tracking the validation and training loss. This loss is suitable for multi-class classification tasks where the labels are integers rather than one-hot encoded vectors. It measures the difference between the predicted probability distribution and the true class labels, encouraging the model to assign high probabilities to the correct classes.

B. *Optimizer: Adam*

The Adam optimizer combines the benefits of Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp), making it robust and efficient for most deep learning tasks. It adjusts learning rates dynamically for each parameter, facilitating faster convergence with reduced manual training.

C. *Performance Metrics: Sparse Categorical Accuracy*

Sparse Categorical Accuracy evaluates the fraction of correctly classified samples, providing an intuitive measure of model performance.

D. *Early Stopping:*

The use of Early Stopping ensures that training halts when validation loss stops improving for 5 consecutive epochs:

$$\text{patience} = 5 \tag{1}$$

This prevents overfitting and reduces computational overhead by stopping the training once the model reaches its optimal performance. The 'restore_best_weights = True' parameter ensures that the best model state is retained, even if subsequent epochs degrade performance.

E. *Validation Loss:*

The validation loss parameter incorporates a validation dataset to monitor generalization performance. The 'val_loss' metric is used as a benchmark for stopping criteria, ensuring the model performs well on unseen data.

F. *Epochs:*

Training is capped at 20 epochs, striking a balance between sufficient training and computational efficiency. Early Stopping typically reduces the number of epochs required in practice.

V. Performance Metrics

The model's performance was evaluated using:

- **Accuracy:** It measures the overall classification performance by calculating the proportion of correctly predicted samples out of the total samples. It gives a general idea of how the model is performing but may not capture the imbalances in class performance.
- **Loss:** It ensures stability and convergence, indicating whether the model is training effectively or not and whether it is generalizing well on unseen data.
- **Confusion Matrix:** It identifies per-class performance and highlights issues like class imbalance or misclassification trends.
  - **Precision:** Fraction of correct positive predictions out of all positive predictions.
  - **Recall:** Fraction of true positive samples identified correctly.
  - **F1-Score:** It is the harmonic mean of Precision and Recall, balancing the trade-off between the two.

VI. Results

A. *Training and Validation Accuracy:*

As shown in the "Model Accuracy" graph, the model achieved high accuracy, stabilizing at $\tilde{9}9.4\%$ on the validation set after 20 epochs. The gap between training and validation accuracy remained minimal, indicating effective regularization and minimal overfitting. The consistent high accuracy suggests the model is able to learn patterns effectively and generalizes well to unseen data.

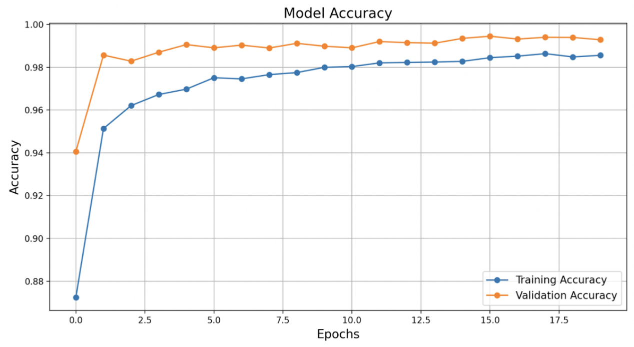


Fig. 1. Visualization of model accuracy over epochs.

B. *Training and Validation Loss:*

The "Model Loss" graph highlights a significant decrease in loss for both training and validation sets, stabilizing after the initial epochs, which further validates the robustness of the model. The training and validation losses remain below 1.0 after the 5th epoch, indicating that the model is effectively learning from the data and generalizing well to unseen data.

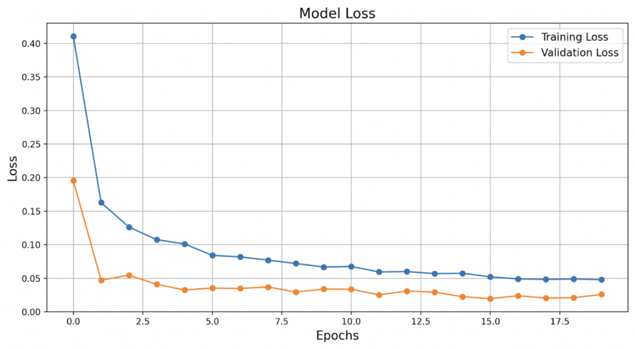
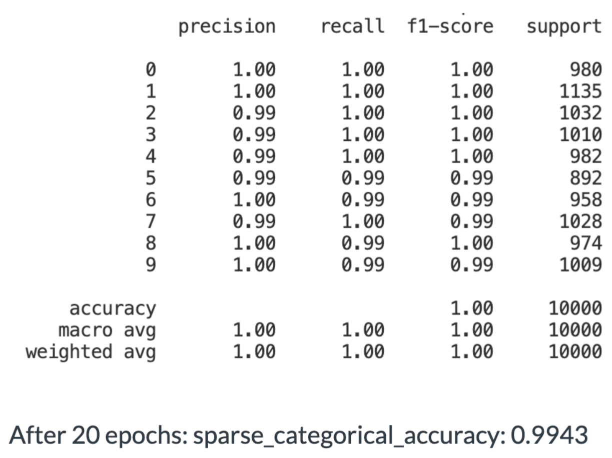


Fig. 2. Visualization of model loss over epochs.

C. *Test Set Performance:*

The final evaluation on the test set yielded an overall accuracy of 99.43%, demonstrating excellent generalization. The classification report confirmed high precision, recall, and F1scores for all classes, with minimal misclassification.



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       980
           1       1.00      1.00      1.00      1135
           2       0.99      1.00      1.00      1032
           3       0.99      1.00      1.00      1010
           4       0.99      1.00      1.00       982
           5       0.99      0.99      0.99       892
           6       1.00      0.99      0.99       958
           7       0.99      1.00      0.99      1028
           8       1.00      0.99      1.00       974
           9       1.00      0.99      0.99      1009

    accuracy                           1.00     10000
   macro avg       1.00      1.00      1.00     10000
weighted avg       1.00      1.00      1.00     10000
```

After 20 epochs: sparse_categorical_accuracy: 0.9943

Fig. 3. Performance metrics on the test set.

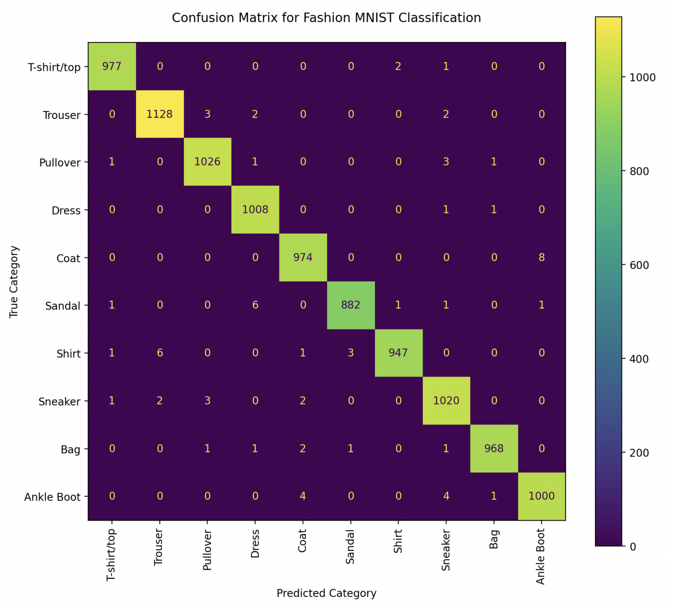


Fig. 4. Visualization of the confusion matrix showing per-class performance.

D. Confusion Matrix:

The confusion matrix revealed strong performance across all categories, with most diagonal values close to 1,000. Minor misclassifications were observed, primarily between similar categories like "Shirt" and "Pullover."

The highest values are along the diagonal, indicating good classification performance for most categories. For instance, 977 data points are correctly classified for the T-shirt category out of 1000 samples, and 1132 data points are correctly classified for the trousers category out of 1135 samples. Misclassifications, such as 'Sandal' being occasionally classified as 'Dress' or 'T-shirt/top' being classified as 'Pullover', likely due to similar shapes and features. To improve future projects, consider further refining the model, such as adding more convolutional layers or attention mechanisms. Future work could also explore transfer learning to leverage features learned from larger and more diverse datasets, investigate misclassified samples, and incorporate additional targeted training data for challenging classes. Investigating different class-aware loss functions could also help emphasize harder-to-classify categories, and exploring additional metrics beyond the confusion matrix could provide deeper insights into the model's performance.

VII. Discussion and Summary

The results underscore the effectiveness of preprocessing strategies and the proposed CNN architecture in achieving state-of-the-art performance on the Fashion MNIST dataset. Data augmentation and dropout layers played a crucial role in improving generalization, while batch normalization stabilized and accelerated training. The low validation loss and minimal overfitting indicate the model's robustness. This project demonstrates the feasibility of using CNNs for image classification tasks and highlights key considerations for designing efficient and accurate deep learning pipelines.

References

[1] Xiao, Han, Kashif Rasul, and Roland Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms." arXiv preprint arXiv:1708.07747 (2017).

[2] Kayed, Mohammed, Ahmed Anter, and Hadeer Mohamed. "Classification of garments from fashion MNIST dataset using CNN LeNet-5 architecture." 2020 international conference on innovative trends in communication and computer engineering (ITCE). IEEE, 2020.

[3] Xhaferra, Edmira, Elda Cina, and Luçiana Toti. "Classification of standard fashion MNIST dataset using deep learning based CNN algorithms." 2022 international symposium on multidisciplinary studies and innovative technologies (ISMSIT). IEEE, 2022.