WICHITA STATE UNIVERSITY

CS770 Machine Learning

Assignment 3-1: Fashion MNIST and Neural Networks

04/29/2025

Submitted by: Logan Schraeder (x356t577)

1. Introduction

This report details the development and evaluation of a committee of deep learning models for image classification on the Fashion MNIST dataset. The Fashion MNIST dataset consists of 60,000 training images and 10,000 testing images, each 28×28 pixels, categorized into 10 clothing classes. The objective of this assignment was to develop multiple deep learning models, combine them into a committee, and compare the committee's performance against individual models using various evaluation metrics, including confusion matrices and classification reports. An additional task involved implementing another mechanism that could enhance model performance. In this case an autoencoder used for upsampling was chosen.

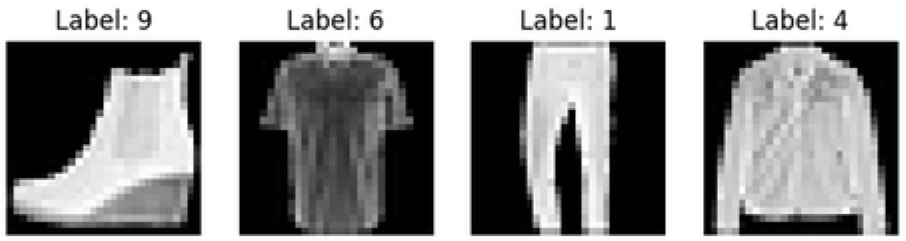

Figure 1. Fashion MNIST Samples.

2. Methodology

Data Exploration & Preparation

The Fashion MNIST dataset was loaded and a sample of images with their corresponding labels were visualized to explore the data. The image pixel values were scaled to the range [0, 1] by dividing by 255.0 for normalization. The labels were converted to one-hot encoded vectors to be suitable for training the neural networks and the classification task output/softmax activation functions. The data was then split into training, validation, and testing datasets, with 80% of the original training data used for training and 20% for validation. The test dataset was kept separate for final evaluation.

Model Development

Three different deep learning models were designed independently and then incorporated into the committee. Table 1 details the Keras/Tensorflow model summaries of each model.

- *Model 1: Shallow Neural Network*
  This model consists of an input layer accepting 28×28 images, followed by a flattening layer to convert the 2D image data into a 1D vector. A dense hidden layer with 128 units and ReLU activation is used, followed by a dense output layer with 10 units and softmax activation for classification across the 10 classes. The choice of ReLU activation in the hidden layer allows the model to learn non-linear relationships, and softmax activation

in the output layer provides a probability distribution over the classes.

- *Model 2: Basic Convolutional Neural Network (CNN)*
  This model incorporates convolutional layers, which are effective for image data. It starts with an input layer for 28×28 images. A Conv2D layer with 32 filters and a 3x3 kernel with ReLU activation is followed by a MaxPooling2D layer with a 2x2 pool size. The output is then flattened and passed through a dense hidden layer with 128 units and ReLU activation, and finally a dense output layer with 10 units and softmax activation. Convolutional and pooling layers help in extracting features from the images.

- *Model 3: Deeper CNN*
  This model is architecturally different from Model 2 by including an additional convolutional layer and dropout. It has an input layer, followed by a Conv2D layer (32 filters, 3x3 kernel, ReLU), MaxPooling2D, another Conv2D layer (64 filters, 3x3 kernel, ReLU), MaxPooling2D, and a flattening layer. It then has a dense layer with 128 units and ReLU activation, a Dropout layer with a rate of 0.5, another dense layer with 64 units and ReLU activation, and a final dense output layer with 10 units and softmax activation. The additional layers and dropout are intended to capture more complex features and help prevent overfitting.

- *Committee Classification*
  A committee of the three models was created for classification. The committee's prediction for an image is the average of the softmax output probabilities from the three individual models. The class with the highest average probability is chosen as the committee's predicted class.

Model Name	**Trainable Params**	**Non-Trainable Param**	**Total Parameters**
1 - Shallow NN	101,770	0	101,770
2 - Basic CNN	693,962	0	693,962
3 - Deeper CNN	232,650	0	232,650
4 - Autoencoder CNN	17,226	241,930	241,930

Table 1. Keras Model Summaries. Note that many of the non-trainable parameters for Model 4 are due to the use of pre-saved weights for the autoencoder, which are not re-trained but are rather trained initially and then frozen.

Bonus Task: Autoencoder Pipeline

As a bonus task to enhance performance, a pipeline model was implemented using an autoencoder. The autoencoder consists of an encoder and a decoder. The encoder's role is to learn a compressed representation (latent vector) of the input images, while the decoder reconstructs the image from this latent vector. The autoencoder was trained separately to ensure effective feature learning via upsampling.

The trained encoder from the autoencoder was then used as the initial layer of a new pipeline model. The weights of the encoder were frozen, meaning they were not updated during the training of the pipeline model. The output of the frozen encoder (the latent vector) was then fed into a classification head, which consists of dense layers (128 units with ReLU and 64 units with ReLU) and a softmax output layer (10 units), similar to the classification part of Model 3. This pipeline effectively uses the autoencoder as a feature extractor for the classification task. Batch Normalization layers were included in the encoder architecture. The pipeline model summary shows that only the parameters in the classification head are trainable, while the encoder parameters are frozen.

Training, Validation & Model Evaluation

Each of the three individual models for the committee was compiled with the Adam optimizer and categorical crossentropy for the loss function. The models were trained for 10 epochs with a batch size of 32, using the validation data to monitor performance and identify potential overfitting. The training time for each model was recorded and is included in Table 2. The autoencoder pipeline model was compiled with the Adam optimizer and sparse categorical crossentropy loss and was trained for 10 epochs with a batch size of 128, using the test data for validation during training. After training, each model, the committee, and the autoencoder pipeline were evaluated on the test dataset to assess their performance. Test loss and accuracy were reported for all models and the confusion matrices can be seen in the figures throughout this report. Table 3 also details the comprehensive evaluation metrics for each model.

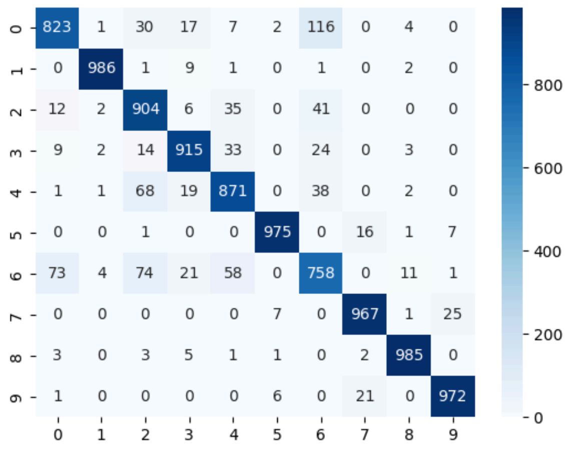

Figure 2. Heatmapped Model 2 (Basic CNN) Confusion Matrix. Notice the slightly but generally higher number of false negatives compared to false positives.

3. Results

The following table summarizes the test accuracy and training time for each individual model, the committee, and the autoencoder pipeline:

Model	Test Accuracy	Training Time (seconds)
Model 1 (Shallow NN)	0.8760	57.74
Model 2 (Basic CNN)	0.9156	82.71
Model 3 (Deeper CNN)	0.9052	74.33
Committee	0.9202	-
Autoencoder CNN Pipeline	0.8013	~28.0

Table 2 & 3. Network Performance. Note: Training time for the committee is not applicable as it combines the outputs of already trained models. The training time for the autoencoder itself is not included in the pipeline training time.

Model	Precision	Recall	F1	Accuracy
1 - Shallow NN	0.88	0.8	0.87	0.88
2 - Basic CNN	0.92	0.92	0.92	0.92
3 - Deeper CNN	0.90	0.91	0.90	0.91
4 - Autoencoder CNN	0.80	0.80	0.80	0.80
Committee	0.92	0.92	0.92	0.92

4. Analysis

The performance of the models varied, with the CNN models generally outperforming the shallow neural network on the Fashion MNIST dataset, as expected due to the nature of image data.

- **Individual Models:**
  - **Model 1 (Shallow NN)** achieved a test accuracy of 0.8760. Its confusion matrix shows a noticeable number of misclassifications between certain classes, such as Pullovers (class 2) and Coats (class 4) being confused with T-shirts/Tops (class 0). Classes like T-shirt/Top (0), Pullover (2), Coat (4), and Sandal (6) show lower f1-scores compared to others, indicating more difficulty in classifying them correctly.
  - **Model 2 (Basic CNN)** performed better with a test accuracy of 0.9156. Its confusion matrix indicates improved classification across most classes compared to Model 1. Classes 0, 2, 4, and 6 still show some significant off-diagonal values in

the confusion matrix, suggesting these classes remain somewhat challenging.

  - **Model 3 (Deeper CNN)** had a test accuracy of 0.9052. While expected to perform better than Model 2 due to its depth and dropout, its test accuracy was slightly lower. This could potentially be due to the specific architecture choices, hyperparameter tuning, or the amount of training data being sufficient for Model 2 to generalize well without the additional complexity of Model 3. The confusion matrix and classification report for Model 3 show similar patterns of misclassification as Model 2, particularly with classes 0, 2, 4, and 6.

- **Committee Approach**:
  The committee, formed by averaging the predictions of the first three models, achieved the highest test accuracy of 0.9202. This indicates that the committee approach provided an improvement over individual models. The confusion matrix and classification report for the committee generally show a slight reduction in misclassifications compared to individual models, suggesting that combining the diverse perspectives of the different architectures helped to improve overall robustness and accuracy. The improvements with the committee support the idea that ensembling models with different strengths can lead to better generalization and robustness.

- **Autoencoder CNN Pipeline**:
  The autoencoder CNN pipeline achieved a test accuracy of 0.8013. This accuracy is lower than the individual CNN models (Model 2 and Model 3) and the committee. This result suggests that, in this case, using the autoencoder for feature extraction and upsampling, and then training a separate classification head on the extracted features did not improve the classification performance compared to training the CNNs end-to-end directly on the image data. Several factors could contribute to this:
  - **Mismatch in Training Objectives:** The autoencoder was likely trained to reconstruct the input images. The objective of reconstruction is to capture all the information needed to recreate the input, which may include details that are not relevant or even detrimental to the classification task. In contrast, the end-to-end trained CNNs (Models 2 and 3) are directly optimized to learn features that are discriminative for classifying the 10 Fashion MNIST categories. The features most useful for reconstruction are not necessarily the features most useful for distinguishing between, for example, a shirt and a t-shirt.
  - **Limited Capacity of the Classification Head:** The classification head placed on top of the frozen encoder, while containing dense layers, might not have had sufficient capacity to learn how to effectively map the autoencoder's latent representation to the 10 different classes. The latent space generated by the encoder has a dimension of 64, which is a significant dimensionality reduction from the original image data. If the autoencoder's learned representation is not easily separable for classification, a relatively simple classification head might struggle to achieve high accuracy.
  - **Suboptimal Feature Space:** The features learned by the autoencoder's encoder,

while effective for reconstruction, might not be the most discriminative features for the Fashion MNIST classes compared to the features learned by the convolutional and dense layers in the end-to-end trained CNNs. The end-to-end training allows the network to jointly optimize the feature extraction process and the classification process, ensuring the learned features are directly relevant to the final task.

- **Training Data Size:** While Fashion MNIST is a reasonably sized dataset, for complex pipelines involving pre-training like an autoencoder, the dataset size can still influence how well the learned features generalize to the downstream task.

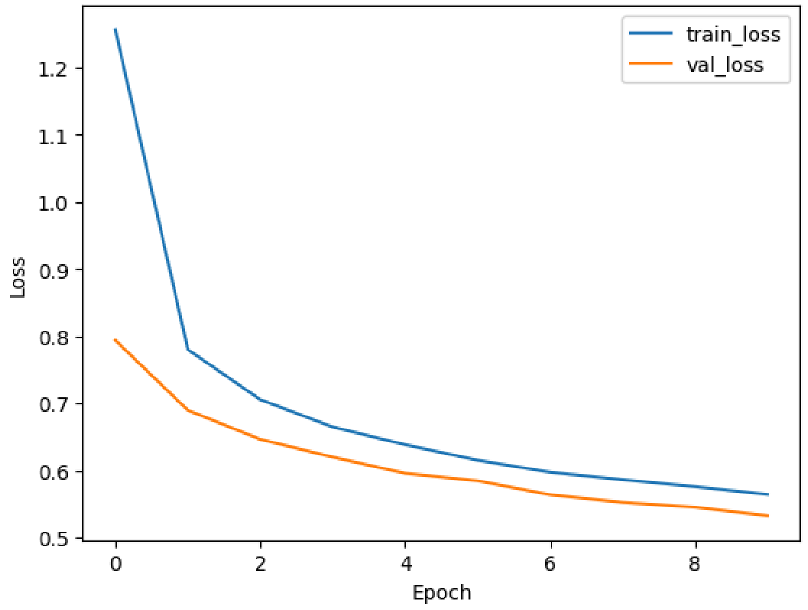

Figure 3. Autoencoder CNN Training Loss Chart. The autoencoder CNN pipeline converges in a generally typical way, and does not indicate any overfitting or underfitting, especially when combined with its performance scores. It does however leave room for improvement.

5. Conclusion

In this report, a committee of deep learning models (shallow NN, basic CNN, and deeper CNN) was developed and evaluated on the Fashion MNIST dataset. The basic CNN (Model 2) and the deeper CNN (Model 3) outperformed the shallow neural network (Model 1), highlighting the effectiveness of convolutional layers for image classification. The committee approach, using simple averaging of model predictions, yielded the highest test accuracy among all models, demonstrating the potential benefits of ensembling. The bonus task explored a pipeline using an autoencoder for feature extraction combined with a classification head. However, this approach resulted in a lower test accuracy compared to the end-to-end trained CNN models and the committee. This suggests that for this specific dataset and model configurations, direct end-to-end training of CNNs was more effective than using an autoencoder for pre-training or feature extraction in this manner. The discrepancy in performance may be attributed to the difference in training objectives between reconstruction and classification, the capacity of the classification head, and the nature of the learned feature space.

Future improvements and experiments could include exploring different ensemble techniques beyond simple averaging, such as weighted averaging based on validation performance. Additionally, techniques to further enhance individual model performance, such as data augmentation, batch normalization (which was included in Model 3 but could be explored more extensively), and hyperparameter tuning for all models and the autoencoder pipeline, could be implemented and evaluated. Different autoencoder architectures or approaches to integrating the autoencoder with the classifier, such as fine-tuning the encoder weights during pipeline training, could also be investigated.