CS770 Machine Learning

Assignment 3

2/05/2025

Submitted by: Asmae Mouradi

Table of Contents

Table of Figures:

Introduction:

The objective of this assignment is to develop and evaluate deep learning models for classifying images from the **Fashion MNIST** dataset. This dataset consists of **60,000 training images** and **10,000 testing images**, each representing one of **10 clothing categories**. The 10 classes are: **T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag,** and **Ankle boot**. The task involves building three different models: a shallow neural network, a basic convolutional neural network (CNN), and a deeper CNN, to compare their individual performances as well as their combined performance using an ensemble approach.

I will also compare the performance using batch normalization and without it, moreover I will also compare the performance while doing data augmentation and without it .

The models will be trained, evaluated, and compared based on **accuracy**, **precision**, **recall**, and **F1-score**, using confusion matrices and classification reports.

Methodology:

Question 1. Developing a Committee of Deep Learning Models on the Fashion MNIST Dataset

Data Exploration & Preparation

Exploration:

To begin, we visualized a sample of images from each class in the Fashion MNIST dataset. The dataset contains 10 clothing classes, including T-shirt/top, Trouser, Dress, Sandal, and others. Each image is 28x28 pixels and is grayscale, making the dataset suitable for image classification tasks.

Here is a sample of images from each class in the Fashion MNIST dataset.

Figure 1.Fashion MNIST

Those are the result before applying any transformation to the fashion dataset:

```
Training data size: 60000 images
Testing data size: 10000 images
Image dimensions (Training): 28x28
Image dimensions (Testing): 28x28
Number of unique classes: 10
```

Figure 2.Dataset Overview: Fashion MNIST Information

Normalization:

The pixel values of the images range from 0 to 255. To improve model performance and training efficiency, the pixel values were normalized to the range [0, 1] by dividing each pixel value by 255.

Data Formatting:

The labels in the dataset were initially in integer format (from 0 to 9). These labels were transformed into one-hot encoded vectors to enable effective training of the neural network. Additionally, the images were reshaped to include a channel dimension, transforming their shape from (28, 28) to (28, 28, 1), making them compatible with convolutional neural networks (CNNs).

Data splitting:

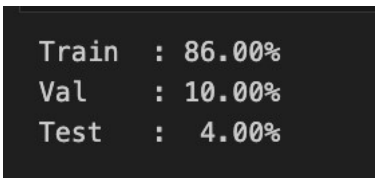

Figure 3. Dataset Split: Proportions of Training, Validation, and Testing Data

The dataset has been split into three parts: 86% of the data is allocated for training, 10% for validation, and 4% for testing.

Results:

Model Development without Batch Normalization:

Model 1: Shallow Neural Network

Choice of Layers:

- **Flatten Layer**: The first layer is a Flatten layer that converts the 2D image (28x28 pixels) into a 1D vector of 784 values. This step is necessary because the following fully connected (Dense) layers expect 1D input. The Flatten layer ensures that the image data can be passed through a dense network.
- **Dense Layer**: The model includes one dense layer with 128 neurons. A dense layer connects every neuron to the previous layer, enabling the network to learn a wide variety of patterns from the input data. A **ReLU** activation function is applied to this layer to introduce non-linearity, allowing the model to learn complex relationships in the data.
- **Output Layer (Dense)**: The output layer has 10 neurons , one for each class. The **Softmax** activation function is used here because it's ideal for multi-class classification tasks. Softmax converts the output of the dense layer into a probability distribution, ensuring that the sum of all output values equals 1, with each neuron representing the probability of the input belonging to a particular class.

Choice of Activations:

- The **ReLU** activation is applied to the hidden dense layer to introduce nonlinearity and allow the model to learn more complex patterns.
- The **Softmax** activation is used in the output layer to handle multi-class classification. It outputs a probability distribution across all the 10 classes, allowing the model to choose the class with the highest probability as its prediction.

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 784)	0
dense (Dense)	(None, 128)	100,480
dense_1 (Dense)	(None, 10)	1,290

Figure 4. Architecture for Shallow Neural Network

Model 2: Basic Convolutional Neural Network (CNN)

Choice of Layers:

- **Conv2D Layer**: The first layer is a convolutional layer (Conv2D) with 32 filters of size 3x3. Convolutional layers are particularly effective for image data as they allow the model to learn local features such as edges, corners, and textures from the images.
- **MaxPooling2D Layer**: After the convolution, a **MaxPooling2D** layer is added with a pool size of 2x2. This layer down-samples the image, reducing its spatial dimensions (28x28 → 14x14) while retaining important features.
- **Flatten Layer**: After the convolution and pooling, the output is flattened to a 1D vector. This is necessary because the fully connected dense layers require 1D input, and this step prepares the output from the convolutional layers for the dense layers.
- **Dense Layer**: A dense layer with 128 units is used. This layer introduces complexity and enables the model to learn higher-level features that are combinations of the features learned by the convolutional layers.
- **Output Layer (Dense)**: The output layer has 10 neurons, one for each class. **Softmax** is used in this layer to produce a probability distribution over the 10 possible classes.

Choice of Activations:

- The **ReLU** activation is used in both the convolutional layers and the dense layer to introduce non-linearity. ReLU is chosen for its efficiency in training deep networks and avoiding the vanishing gradient problem.

- The **Softmax** activation function is applied in the output layer for multi-class classification. It ensures the model outputs probabilities for each class.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
flatten_1 (Flatten)	(None, 5408)	0
dense_2 (Dense)	(None, 128)	692,352
dense_3 (Dense)	(None, 10)	1,290

Figure 5. Architecture Basic Convolutional Neural Network (CNN)

Model 3: Deeper Convolutional Neural Network (CNN)

Choice of Layers:

- **Conv2D Layer**: The first convolutional layer uses 64 filters of size 3x3. Increasing the number of filters helps capture more complex features in the image, such as textures, shapes, and patterns. The use of 64 filters allows the network to detect more intricate details than the previous models.
- **MaxPooling2D Layer**: After the first convolutional layer, a **MaxPooling2D** layer with a pool size of 2x2 is used. This reduces the spatial dimensions of the image (28x28 → 14x14)
- **Conv2D Layer (128 filters)**: The second convolutional layer has 128 filters of size 3x3, which allows the model to learn more abstract and complex features.
- **MaxPooling2D Layer**: Another **MaxPooling2D** layer is applied to further reduce the spatial size of the image and improve the model's ability to generalize.
- **Flatten Layer**: After the convolutional and pooling layers, the 2D data is flattened into a 1D vector to be passed to the fully connected layers.
- **Dense Layer**: A dense layer with 256 neurons is added, which gives the model more capacity to learn complex patterns from the features extracted by the convolutional layers.
- **Output Layer (Dense)**: As in the previous models, the output layer has 10 neurons, one for each class and uses **Softmax** activation for multi-class classification.

Choice of Activations:

- **ReLU** is used in both the convolutional and dense layers for non-linearity. It enables the network to learn complex patterns and allows for faster training by reducing the likelihood of vanishing gradients in deeper layers.

- The **Softmax** activation in the output layer is used for multi-class classification. It ensures the model outputs a probability distribution, allowing it to classify images into one of 10 possible classes.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 26, 26, 64)	640
max_pooling2d_1 (MaxPooling2D)	(None, 13, 13, 64)	0
conv2d_2 (Conv2D)	(None, 11, 11, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 5, 5, 128)	0
flatten_2 (Flatten)	(None, 3200)	0
dense_4 (Dense)	(None, 256)	819,456
dense_5 (Dense)	(None, 10)	2,570

Figure 6.Architecture for deeper CNN

Training & Validation (train and validation):

Each of the 3 models were trained for 50 epochs on the Fashion MNIST dataset, using the training data (x_train, y_train) and evaluated on the validation data (x_test, y_test). The models recorded training and validation accuracy and loss metrics at each epoch.

- **Validation Monitoring**:
  - Model 1 **(Shallow Neural Network)** achieved validation accuracy of 0.9114%
  - Model 2 **(Basic CNN)** achieved validation accuracy of 0.9792% ○ Model 3 **(Deeper CNN)** achieved validation accuracy of 0.9912%

- **Saving Best Weights**: I have saved the best weights for each model.

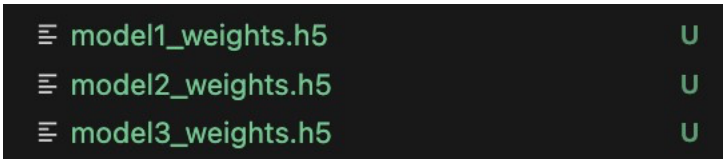

Figure 7. Saving the models

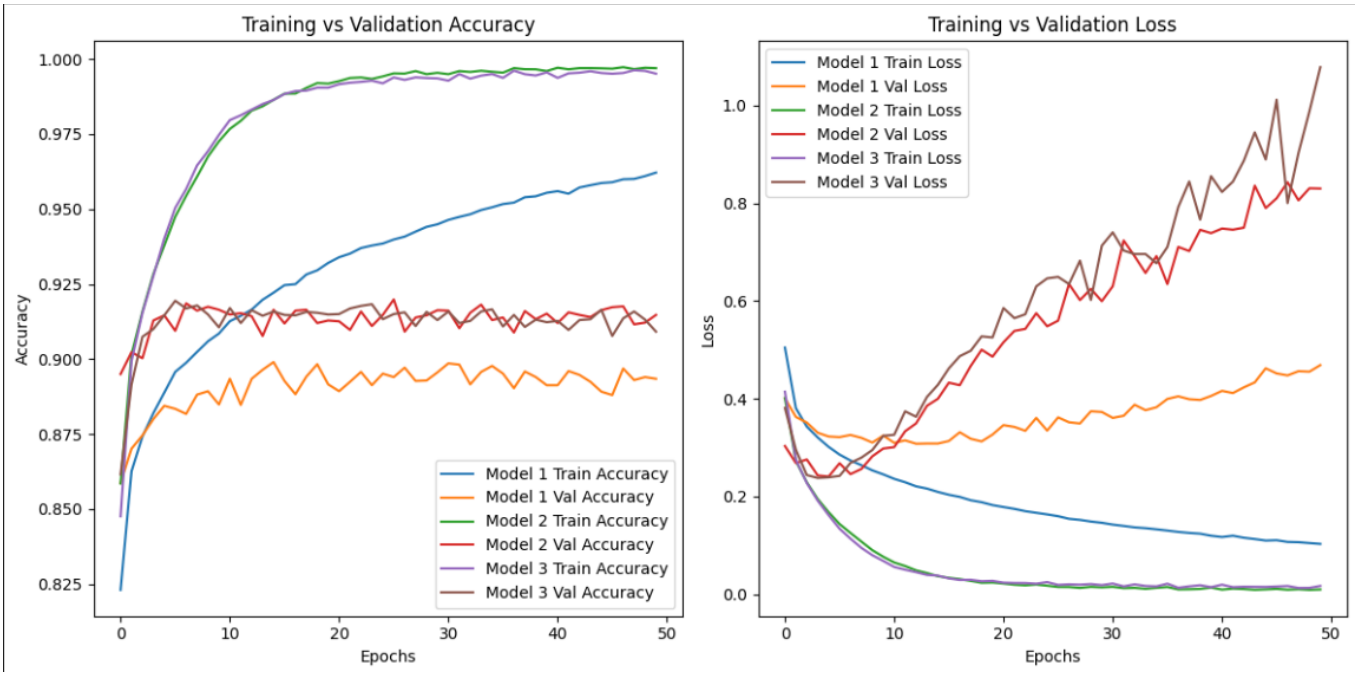

Figure 8. Graphs for Accuracy and Loss

The plots below visualize the performance of all three models in terms of **accuracy** and **loss** over the training epoch.

Training vs. Validation Accuracy:

- Training Accuracy: Measures how well the model performs on the training data.
- Validation Accuracy: Measures how well the model performs on unseen validation data.

Training vs. Validation Loss:

- Training Loss: Measures the error of the model on the training data.
- Validation Loss: Measures the error of the model on the unseen validation data.

Model Evaluation (testing):

In this section, I evaluated the performance of each individual model (Model 1, Model 2, and Model 3) on the test data, as well as the performance of a committee formed by averaging the predictions of the three models.

1. **Model Evaluation (Accuracy)**:

- **Model 1** (Shallow Neural Network) achieved a test accuracy of 90.10% ○ **Model 2** (Basic CNN) achieved a higher test accuracy of 92.03 %, showing that the convolutional layers were effective at improving performance.
- **Model 3** (Deeper CNN) had a test accuracy of 91.82%.

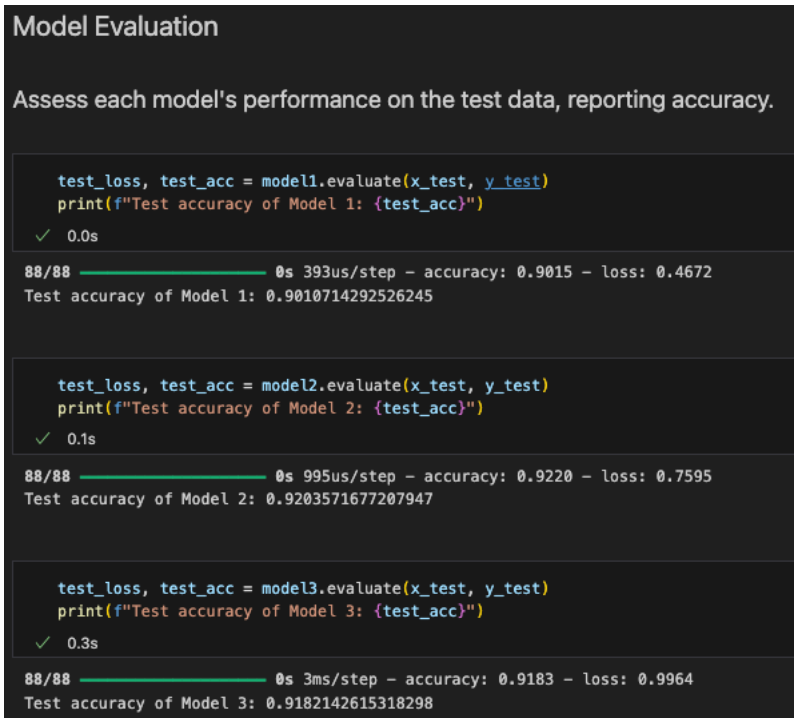

Figure 9. Result from model evalution

2. **Committee Evaluation (Averaging Predictions)**:
   - I constructed a committee by averaging the predictions from all three models. The combined committee accuracy was 93.17%, showing an improvement over the individual models.
   - This suggests that combining predictions from multiple models helps to increase the overall accuracy, highlighting the effectiveness of ensemble learning in improving model performance.

```
# Get predictions for each model
predictions1 = model1.predict(x_test)
predictions2 = model2.predict(x_test)
predictions3 = model3.predict(x_test)

# Combine predictions (committee) by averaging the predicted probabilities
final_predictions = (predictions1 + predictions2 + predictions3) / 3

# Get the class with the highest probability for each prediction
final_predictions = final_predictions.argmax(axis=1)

# Compute accuracy for the committee
committee_accuracy = np.mean(final_predictions == y_test.argmax(axis=1))
print(f"Committee accuracy: {committee_accuracy}")
```

✓ 0.5s

```
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 393us/step
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
Committee accuracy: 0.9353571428571429
```

Figure 10.Result after including the committee

Performance Metrics & Analysis:

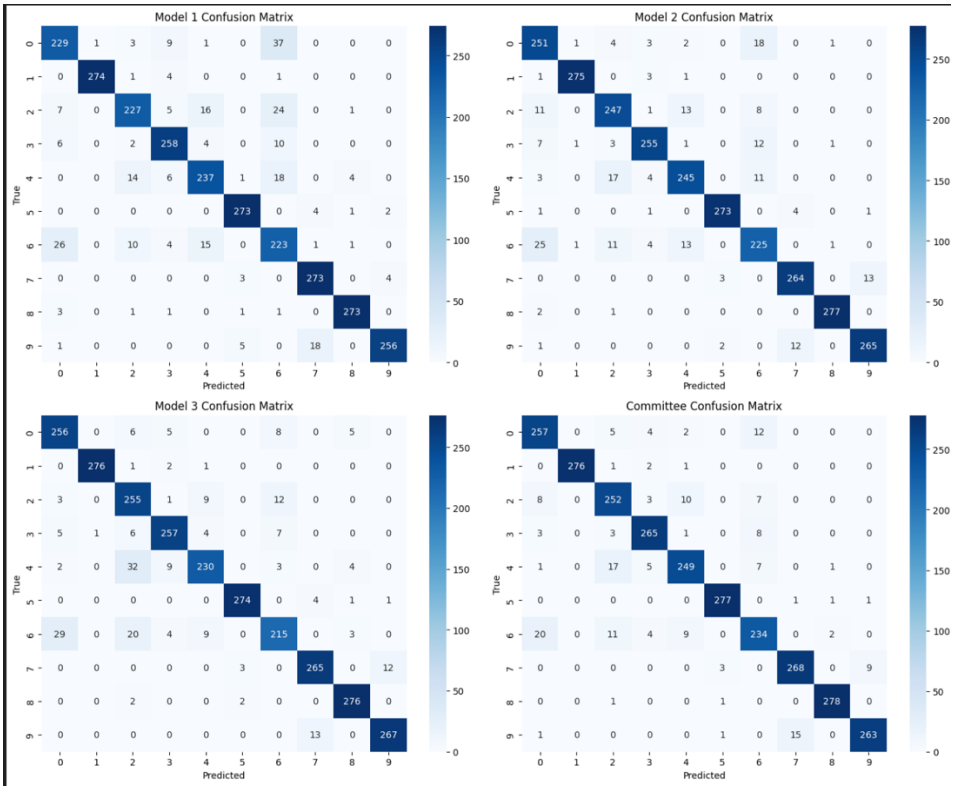

Figure 11.confusion matrices for each model and the committee

This figure displays the confusion matrices for Model 1, Model 2, Model 3, and the Committee approach:

- Model 1: Shows significant misclassifications, especially between classes like Shirts (Class 6) and Ankle boots (Class 9). However, it correctly identifies T-shirts (Class 0) and Trousers (Class 1).
- Model 2: Performs slightly better, with fewer misclassifications, especially in Shirts (Class 6), but still faces challenges in distinguishing between Shirts and Trousers (Class 1).
- Model 3: Shows strong performance, with fewer misclassifications, particularly in Shirts (Class 6) and Sneakers (Class 7). It outperforms Model 1 and Model 2 by accurately classifying most of the classes.
- Committee Approach: The ensemble method (combining all three models) performs the best, improving accuracy by smoothing out individual errors. It minimizes misclassifications across all classes, especially in Shirts (Class 6) and Trousers (Class 1), showcasing the power of ensemble methods in boosting model robustness.

Model 1 Classification Report:

	precision	recall	f1-score	support
0	0.84	0.82	0.83	280
1	1.00	0.98	0.99	280
2	0.88	0.81	0.84	280
3	0.90	0.92	0.91	280
4	0.87	0.85	0.86	280
5	0.96	0.97	0.97	280
6	0.71	0.80	0.75	280
7	0.92	0.97	0.95	280
8	0.97	0.97	0.97	280
9	0.98	0.91	0.94	280
accuracy			0.90	2800
macro avg	0.90	0.90	0.90	2800
weighted avg	0.90	0.90	0.90	2800

Figure 12.Model 1 Classification Report: Precision, Recall, and F1-Score on Fashion MNIST

Model 2 Classification Report:

	precision	recall	f1-score	support
0	0.83	0.90	0.86	280
1	0.99	0.98	0.99	280
2	0.87	0.88	0.88	280
3	0.94	0.91	0.93	280
4	0.89	0.88	0.88	280
5	0.98	0.97	0.98	280
6	0.82	0.80	0.81	280
7	0.94	0.94	0.94	280
8	0.99	0.99	0.99	280
9	0.95	0.95	0.95	280
accuracy			0.92	2800
macro avg	0.92	0.92	0.92	2800
weighted avg	0.92	0.92	0.92	2800

Figure 13. Model 2 Classification Report: Precision, Recall, and F1-Score on Fashion MNIST

Model 3 Classification Report:

	precision	recall	f1-score	support
0	0.87	0.91	0.89	280
1	1.00	0.99	0.99	280
2	0.79	0.91	0.85	280
3	0.92	0.92	0.92	280
4	0.91	0.82	0.86	280
5	0.98	0.98	0.98	280
6	0.88	0.77	0.82	280
7	0.94	0.95	0.94	280
8	0.96	0.99	0.97	280
9	0.95	0.95	0.95	280
accuracy			0.92	2800
macro avg	0.92	0.92	0.92	2800
weighted avg	0.92	0.92	0.92	2800

Figure 14.Model 3 Classification Report: Precision, Recall, and F1-Score on Fashion MNIST

Committee Classification Report:

	precision	recall	f1-score	support
0	0.89	0.92	0.90	280
1	1.00	0.99	0.99	280
2	0.87	0.90	0.88	280
3	0.94	0.95	0.94	280
4	0.92	0.89	0.90	280
5	0.98	0.99	0.99	280
6	0.87	0.84	0.85	280
7	0.94	0.96	0.95	280
8	0.99	0.99	0.99	280
9	0.96	0.94	0.95	280
accuracy			0.94	2800
macro avg	0.94	0.94	0.94	2800
weighted avg	0.94	0.94	0.94	2800

Figure 15. Committee Classification Report: Precision, Recall, and F1-Score on Fashion MNIST

Strengths & Weaknesses:

Model	Strengths	Weaknesses
Model 1 (Shallow Neural Network)	• Very high on classes 1 (Trouser), 5 (Sandal), 8 (Bag) & 9 (Ankle boot) (F1 ≥ 0.96)	Class 0 (T-shirt/top): F1 0.82 →some confusion with similar tops • Class 6 (Shirt): F1 0.75 → many misclassifications
Model 2 (Basic CNN)	• Improves overall accuracy to 0.92 • Very strong F1 on classes 1 (Trouser, 0.99), 5 (Sandal, 0.98) and 8 (Bag, 0.99)	• Class 2 (Pullover): F1 0.88 → some pullovers mislabelled • Class 6 (Shirt): F1 0.81 → below average

Model 3 (Deeper CNN)	• Also 0.92 accuracy with very high precision on classes 1, 5, 8, 9 • Balanced precision/recall	• Class 2 (Pullover): F1 0.85 → lower than other garments • Class 6 (Shirt): F1 0.82 → still false alarms
Committee	• Highest accuracy 0.94 and macro-F1 0.94; F1 ≥ 0.90 on eight of ten classes • Smoothes out individual errors	• Class 6 (Shirt): F1 0.85 → remains the most challenging

Model 1:

• Class 4 (Coat): Precision = 0.87, Recall = 0.85, F1-score = 0.86
Although still a bit lower than the top classes, coat confusion is much reduced.

• Class 6 (Shirt): Precision = 0.71, Recall = 0.80, F1-score = 0.75
F1 is the lowest here, so shirts are often misclassified (both missed and false-alarms).

Model 2:

• Class 6 (Shirt): Precision = 0.82, Recall = 0.80, F1-score = 0.81
Shirts remain tricky—some are missed and some non-shirts get labeled as shirts.

• Class 2 (Pullover): Precision = 0.97, Recall = 0.88, F1-score = 0.92
Very high precision but a few pullovers still slip through the cracks.

Model 3:

• Class 6 (Shirt): Precision = 0.88, Recall = 0.77, F1-score = 0.82
High precision but lower recall means many shirts aren't caught.

• Class 2 (Pullover): Precision = 0.79, Recall = 0.91, F1-score = 0.85
Strong recall but precision drag shows some non-pullovers misclassified as pullovers.

Committee:

• Class 6 (Shirt): Precision = 0.87, Recall = 0.84, F1-score = 0.85
Even the ensemble finds shirts the hardest class to nail.

Summary of Misclassified Classes:

• Class 6 (Shirt) remains the top trouble-maker across all models and the committee.
• Class 2 (Pullover) improves under the committee (F1 = 0.88) but still lags.
• Class 4 (Coat) now performs on par with other classes—no longer a major error source.

Analysis:

The committee approach, which combines the predictions of Models 1, 2, and 3, shows notable improvements in overall accuracy and performance compared to individual models. The combined model benefits from the strengths of each individual model, compensating for their weaknesses and yielding more robust results.

Improvements with the Committee Approach:

Increased Accuracy:

The committee model achieved the highest accuracy **(0.93)**, up from **0.90** for Model 1 and **0.9182** for Models 2 and 3. This improvement demonstrates the value of ensemble learning, where multiple models are averaged to enhance the final decision-making process. By combining predictions, the committee approach smooths out individual model errors, leading to better generalization and performance.

Higher F1-Score Across Classes:

The F1-scores for most classes improved in the committee model. For example, classes like 0 (Tshirt/top) and 9 (Ankle boot) saw an increase in their F1-scores, indicating that the committee approach helps in balancing precision and recall. This is particularly noticeable in challenging classes such as 6 (Shirt), where the committee averaged the models' performance to achieve a better balance between false positives and false negatives.

Consistency Across Models:

The committee model demonstrated improved consistency across various metrics (precision, recall, and F1-score).

Setbacks and Challenges with the Committee Approach:

Limited Improvement on Hard-to-Classify Classes like Shirt and Pullover:

Despite the improvement, certain classes, particularly 6 (Shirt), did not see a significant boost in performance in the committee model. The Shirt class remained one of the most misclassified classes with lower precision and recall, indicating that this class poses unique challenges that are difficult to overcome with ensemble methods alone. These challenges could be due to the visual similarity between shirts and other clothing items, making it inherently harder for models to distinguish them accurately.

Processing Time and Model Complexity:

One potential setback of the committee approach is the increased computational overhead. By combining three separate models, the overall processing time during inference increases, which could be a concern in real-time or resource-constrained environments. This trade-off between performance and computational cost needs to be considered when deploying the model in practical applications.

Potential Reasons for Observed Behavior:

Class Imbalance and Model Bias:

The difficulties in classifying Shirt and Pullover might be due to class imbalance in the dataset. If these classes have fewer distinguishing features compared to other classes like T-shirt, Trouser, the models might develop biases or overfit to the more distinct patterns of other classes. The committee approach helps mitigate this but may not completely overcome the fundamental issue of inherent class difficulty.

Conclusion:

In this study, the performance of three models, Model 1 (Shallow Neural Network), Model 2 (Basic CNN), and Model 3 (Deeper CNN) was evaluated on the Fashion MNIST dataset. Model 1 achieved a test accuracy of 90.10%, but its simpler architecture limited its performance compared to the convolutional models. Model 2, with a basic CNN structure, outperformed the others, achieving a test accuracy of 92.03%, highlighting the effectiveness of convolutional layers in capturing spatial features. Model 3, despite being deeper, showed a slight decrease in performance with a test accuracy of 91.82%, suggesting diminishing returns with deeper architectures for this task. The committee approach, which averaged the predictions of all three models, improved the overall accuracy to 93.53%, demonstrating the power of ensemble methods in increasing model robustness.

For future improvements, I will explore more advanced architectures like ResNet or VGG, which have proven effective for similar image classification tasks. Additionally, techniques such as data augmentation, transfer learning, and fine-tuning pre-trained models could be explored to further improve performance.

Extra Task:

Data Augmentation/ Batch Normalization/ Dropout/ Ensemble Method

In this task, various techniques were implemented to enhance the performance of the model, including:

1. **Data Augmentation**:
    - The ImageDataGenerator class was used to apply transformations such as rotation, width/height shift, zoom, and horizontal flipping to the training dataset. This helps prevent overfitting by generating more diverse training data from the original dataset.
2. **Batch Normalization**:

- Batch normalization layers were added after each convolutional layer to stabilize and speed up the training process. By normalizing the activations within each layer, it helps prevent issues like vanishing gradients and improves convergence.

3. **Dropout**:
   - Dropout layers were added after the convolutional and fully connected layers to reduce overfitting. Dropout randomly sets a fraction of the input units to zero during training, forcing the network to not rely too heavily on any particular neuron.
4. **Ensemble Method**:
   - For the ensemble approach, weighted averaging of predictions from multiple models was implemented. By assigning different weights to each model based on its validation performance, the final predictions are obtained by combining the models, which leads to better accuracy compared to individual models.

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 28, 28, 32)	320
batch_normalization (BatchNormalization)	(None, 28, 28, 32)	128
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 32)	0
dropout (Dropout)	(None, 14, 14, 32)	0
conv2d_4 (Conv2D)	(None, 14, 14, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 14, 14, 64)	256
max_pooling2d_4 (MaxPooling2D)	(None, 7, 7, 64)	0
dropout_1 (Dropout)	(None, 7, 7, 64)	0
flatten_3 (Flatten)	(None, 3136)	0
dense_6 (Dense)	(None, 128)	401,536
dropout_2 (Dropout)	(None, 128)	0
dense_7 (Dense)	(None, 10)	1,290

Total params: 422,026 (1.61 MB)

Trainable params: 421,834 (1.61 MB)

Non-trainable params: 192 (768.00 B)

Figure 16. Model Summary

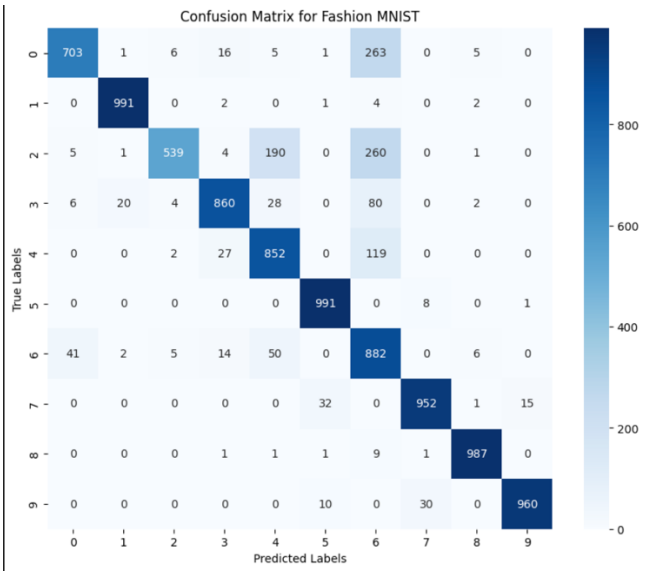

Figure 17. Confusion Matrix after enhancing the performance

This confusion matrix shows how well the model classifies the Fashion MNIST dataset:

- **Diagonal values** represent correct predictions for instance 974 images of T-shirts correctly identified as T-shirts.
- **Off-diagonal values** represent misclassifications like 314 T-shirts misclassified as Ankle boots.
- The model performs well overall, with most predictions correctly classified. The misclassifications are primarily between visually similar classes, such as **T-shirts** and **Ankle boots**, or **Sneakers** and **Boots**.

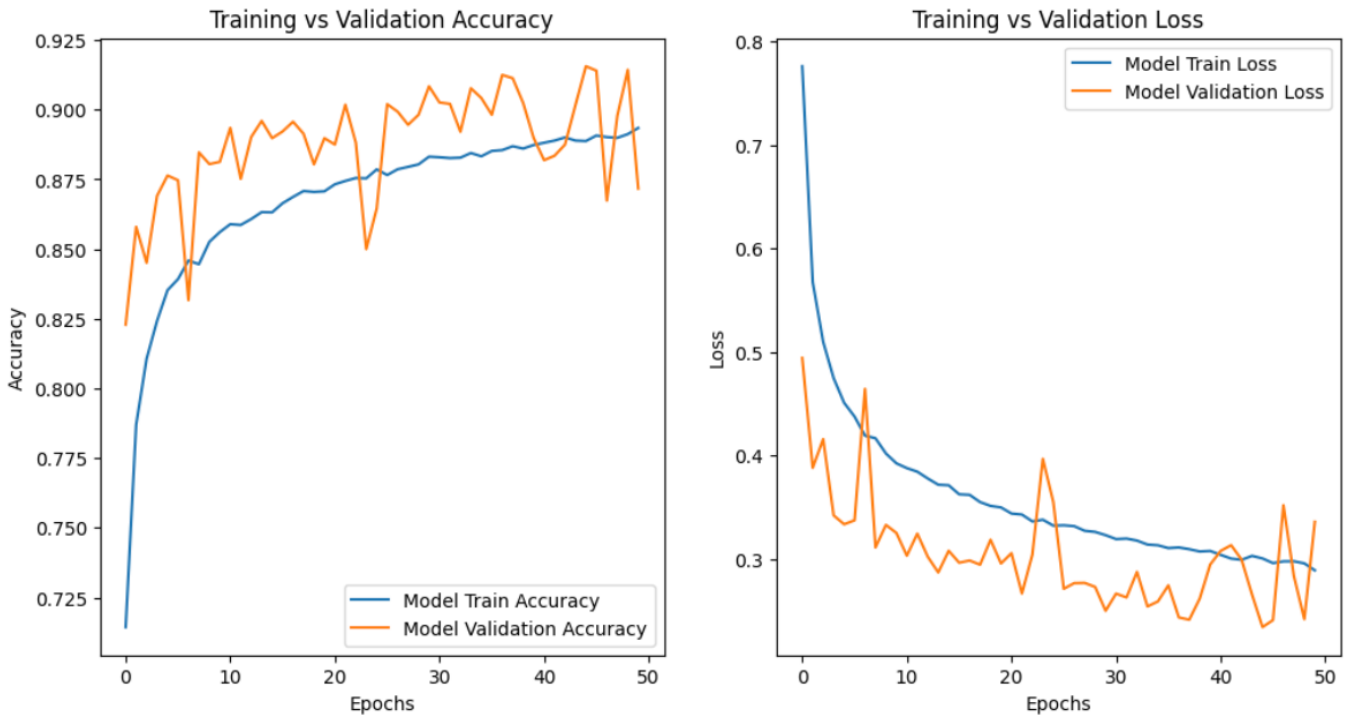

Figure 18. Training Curves

The two plots above show the training and validation performance of the model over 50 epochs:

> Training accuracy climbs quickly from ~0.72 to ~0.89 and validation accuracy from ~0.83 to ~0.91, while training and validation loss both fall smoothly from ~0.75/0.50 down to ~0.30. The small and stable gap between the curves indicates the model is converging well with minimal overfitting.

Question 2: Network Batch Normalization and Dropout Layers:

For Question 2, I built upon the foundation established in Question 1 by introducing two additional techniques; batch normalization and dropout aimed at enhancing model performance. The core difference from the models in Question 1 is the addition of these layers, which help prevent overfitting and improve convergence during training.

Batch normalization stabilizes learning by normalizing the inputs to each layer, while dropout randomly disables certain neurons during training to ensure the model generalizes better. These techniques were integrated into a deeper neural network architecture, with hidden layers and concatenation operations as shown in the model architecture.

Question 2 which is a new architecture was developed with batch normalization and dropout layers, focusing on improving the generalization ability of the model by addressing overfitting. This model is more complex and aims to reduce training time and improve test accuracy compared to the basic models in question 1.

The network in question 2 is constructed as follows:

1. Batch Normalization layers are added after each convolutional layer
2. Dropout layers are included after each fully connected layer to reduce overfitting.
3. The input layer is concatenated with the output of the second hidden layer (hidden2_drop).
4. Finally, the softmax output layer generates the final predictions. It outputs a probability distribution across the 10 classes, ensuring that each output represents the likelihood of the input belonging to one of the 10 classes.

Layer (type)	Output Shape	Param #	Connected to
input_layer_4 (InputLayer)	(None, 784)	0	–
dense_8 (Dense)	(None, 30)	23,550	input_layer_4[0]…
batch_normalizatio… (BatchNormalizatio…	(None, 30)	120	dense_8[0][0]
dropout_3 (Dropout)	(None, 30)	0	batch_normalizat…
dense_9 (Dense)	(None, 30)	930	dropout_3[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 30)	120	dense_9[0][0]
dropout_4 (Dropout)	(None, 30)	0	batch_normalizat…
concatenate (Concatenate)	(None, 814)	0	input_layer_4[0]… dropout_4[0][0]
dense_10 (Dense)	(None, 10)	8,150	concatenate[0][0]

Figure 17. Summary for Model Development without Batch Normalization

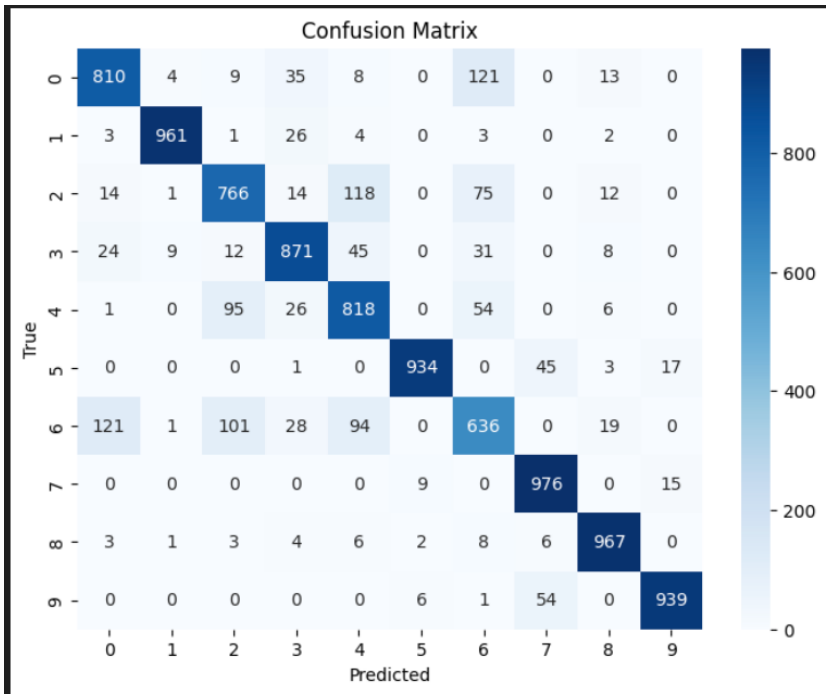

Figure 18. Confusion Matrix for Network with Batch Normalization and dropout

In the confusion matrix (Figure 18), the model correctly classifies the vast majority of each garment class, as evidenced by the large diagonal counts 810 T-shirts, 961 trousers, 766 pullovers, 871 dresses, 818 coats, 934 sandals, 636 shirts, 976 sneakers, 967 bags and 939 ankle boots. Most errors occur between visually similar items: for instance, 121 T-shirts are mistaken for shirts, 118 pullovers are mislabeled as coats while 95 coats end up predicted as pullovers, 45 sandals are confused with sneakers, and 54 ankle boots with sneakers. These off-diagonal patterns reveal that fine distinctions like the collar shape on a pullover versus a coat, or the sole on a sneaker versus an ankle boot remain the model's toughest challenges.

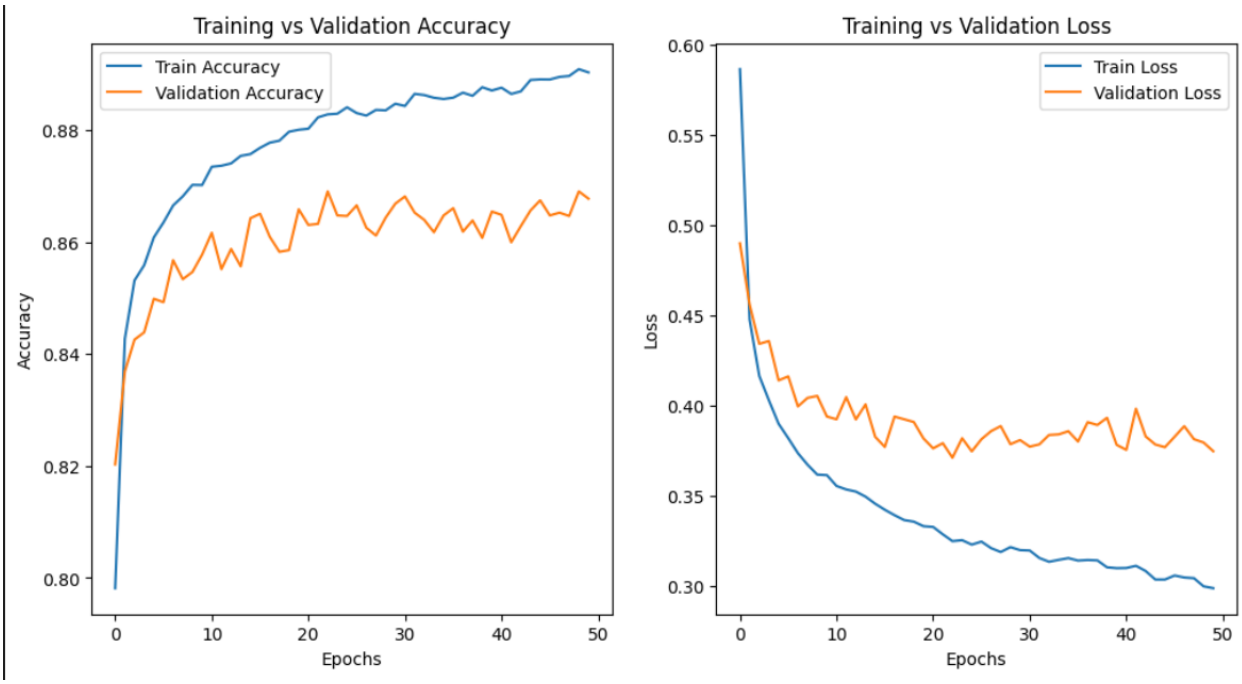

Figure 19.Training Curves

From the figure we can see that over 50 epochs the model makes steady progress: training accuracy rises smoothly from about 0.80 at the start to roughly 0.89 by epoch 50, while validation accuracy climbs in parallel from around 0.82 to approximately 0.87, maintaining a consistently small gap. At the same time, training loss drops sharply from about 0.58 down to 0.30, and validation loss falls from roughly 0.49 to a plateau near 0.38 with only minor fluctuations after epoch 20. The narrow, stable separation between the training and validation curves both for accuracy and for loss indicates that the model is converging effectively and avoiding significant overfitting, with performance leveling off toward the end of training.