Maryam Abrahams
ABRMAR043

Neural Networks and Image Classification

Data Processing:

Our goal with data processing is to load and preprocess the FashionMNIST dataset, applying tensor conversion and normalization to prepare the images for the training loops and evaluation of our neural network model.

Design Implementation:

1. Transformations: The images are converted to tensors using **ToTensor(),** and Normalization is applied with **Normalize((0.5,), (0.5,))** to center pixel values around zero. This improves training stability for neural networks.
2. **Loading:** Downloaded the full FashionMNIST dataset. We split the training set into:
    a. **New Training Set:** 50000 images
    b. **Validation Set:** 10000 images
    c. **Test set:** 10000 images (which is downloaded separately for final evaluation

---

Building and Training a Base Model:

Our goal with building and training a baseline model is to implement a simple Multi-Layer Perceptron (MLP) to act as a reference/ baseline performance on the FashionMNIST dataset before experimenting with the flexible multi-layer perceptron model and hyperparameter tuning.

Design Implementation:

1. **FlexibleMLP:** A flexible Multi-Layer Perceptron class allowing customization of hidden layer sizes, dropout rates, and activation functions.
    a. **__init__:** Initializes the layers, dropout, and selected activation function (ReLU or Sigmoid) based on provided hyperparameters.
    b. **forward:** Defines the forward pass through the network. It flattens the input, passes through the hidden layers with activation and dropout, and outputs the logits to be used for classification.
2. **Train_model:** Encapsulates the training loop. It iterates over epochs, computes loss, updates weights, and tracks training and validation losses and accuracy.
3. **BaselineMLP:** A standard MLP used to establish baseline performance.
    a. **__init__:** Initializes a sequential stack of layers with a flattened input, two hidden layers (128 and 64 neurons) with ReLU activations, and an output layer of 10 neurons.
    b. **forward:** Computes the forward pass to produce logits for class predictions.

Results :

```
============================================================
TRAINING BASELINE MODEL
============================================================
Epoch 1/10 - Train Loss: 0.5352, Val Loss: 0.4001, Val Acc: 85.26%
Epoch 2/10 - Train Loss: 0.3893, Val Loss: 0.3620, Val Acc: 86.35%
Epoch 4/10 - Train Loss: 0.3273, Val Loss: 0.3476, Val Acc: 87.09%
Epoch 6/10 - Train Loss: 0.2908, Val Loss: 0.3273, Val Acc: 88.15%
Epoch 8/10 - Train Loss: 0.2624, Val Loss: 0.3263, Val Acc: 88.09%
Epoch 10/10 - Train Loss: 0.2423, Val Loss: 0.3198, Val Acc: 88.29%

 Baseline Model Final Accuracy: 88.29%
```

The baseline MLP achieved a final validation accuracy of **88.29%** after **10** epochs, showing consistent improvement in both training and validation loss throughout the training process. Had we trained for more epochs, we may have seen a further improvement however, the slowing of accuracy improvements implies the model was nearing convergence. The steady decrease in loss with increases in accuracy shows that the network effectively learned to classify FashionMNIST images without overfitting in the initial epochs.

Hyperparameter Optimization:

The goal of hyperparameter optimization is to monitor and fine-tune configuration settings (learning rate, optimizer, and hidden layer size) to improve model performance beyond the baseline MLP accuracy.

Design Implementation:

1. **Experiment 1:** Investigates the impact of different learning rates **[0.1, 0.01, 0.001, 0.0001]** on model convergence and validation accuracy.
2. **Experiment 2:** Compares **Adam, SGD**, and **RMSprop** optimizers while keeping other settings constant to determine which optimizer leads to faster convergence and higher accuracy.
3. **Experiment 3:** Tests the effect of varying the first hidden layer size **[64, 128, 256, 512]** on learning capacity and model performance.

For each experiment, we use the **FlexibleMLP** with the chosen hyperparameter, trained for 10 epochs using cross-entropy loss, and evaluated on the validation set. We store results to find the best-performing configuration.

Results :

```
Best Configuration: Hidden1=128
Best Accuracy: 89.26% (Baseline: 88.29%)
Total experiments run: 11
```

```
--- EXPERIMENT 1: Learning Rate ---

 Training: LR=0.1
Epoch 1/10 - Train Loss: 2.6034, Val Loss: 2.3058, Val Acc: 9.59%
Epoch 2/10 - Train Loss: 2.3122, Val Loss: 2.3138, Val Acc: 9.64%
Epoch 4/10 - Train Loss: 2.3124, Val Loss: 2.3113, Val Acc: 9.59%
Epoch 6/10 - Train Loss: 2.3120, Val Loss: 2.3060, Val Acc: 9.59%
Epoch 8/10 - Train Loss: 2.3116, Val Loss: 2.3114, Val Acc: 10.10%
Epoch 10/10 - Train Loss: 2.3117, Val Loss: 2.3156, Val Acc: 10.52%

 Training: LR=0.01
Epoch 1/10 - Train Loss: 0.5611, Val Loss: 0.4707, Val Acc: 82.49%
Epoch 2/10 - Train Loss: 0.4617, Val Loss: 0.4833, Val Acc: 82.82%
Epoch 4/10 - Train Loss: 0.4185, Val Loss: 0.4601, Val Acc: 83.85%
Epoch 6/10 - Train Loss: 0.3992, Val Loss: 0.4537, Val Acc: 84.19%
Epoch 8/10 - Train Loss: 0.3771, Val Loss: 0.4249, Val Acc: 85.37%
Epoch 10/10 - Train Loss: 0.3743, Val Loss: 0.4353, Val Acc: 85.44%

 Training: LR=0.001
Epoch 1/10 - Train Loss: 0.5361, Val Loss: 0.4340, Val Acc: 83.84%
Epoch 2/10 - Train Loss: 0.3854, Val Loss: 0.4320, Val Acc: 83.98%
Epoch 4/10 - Train Loss: 0.3234, Val Loss: 0.3400, Val Acc: 86.98%
Epoch 6/10 - Train Loss: 0.2903, Val Loss: 0.3497, Val Acc: 86.74%
Epoch 8/10 - Train Loss: 0.2651, Val Loss: 0.3312, Val Acc: 87.88%
Epoch 10/10 - Train Loss: 0.2440, Val Loss: 0.3390, Val Acc: 88.22%

 Training: LR=0.0001
Epoch 1/10 - Train Loss: 0.8326, Val Loss: 0.5412, Val Acc: 80.84%
Epoch 2/10 - Train Loss: 0.4974, Val Loss: 0.4654, Val Acc: 83.63%
Epoch 4/10 - Train Loss: 0.4250, Val Loss: 0.4178, Val Acc: 84.92%
Epoch 6/10 - Train Loss: 0.3904, Val Loss: 0.3971, Val Acc: 85.53%
Epoch 8/10 - Train Loss: 0.3675, Val Loss: 0.3839, Val Acc: 85.72%
Epoch 10/10 - Train Loss: 0.3485, Val Loss: 0.3629, Val Acc: 86.70%

--- EXPERIMENT 3: Hidden Layer 1 Neurons ---

 Training: Hidden1=64
Epoch 1/10 - Train Loss: 0.5545, Val Loss: 0.4215, Val Acc: 84.86%
Epoch 2/10 - Train Loss: 0.4029, Val Loss: 0.4001, Val Acc: 85.40%
Epoch 4/10 - Train Loss: 0.3403, Val Loss: 0.3866, Val Acc: 85.40%
Epoch 6/10 - Train Loss: 0.3056, Val Loss: 0.3494, Val Acc: 87.16%
Epoch 8/10 - Train Loss: 0.2805, Val Loss: 0.3266, Val Acc: 88.12%
Epoch 10/10 - Train Loss: 0.2620, Val Loss: 0.3300, Val Acc: 87.94%

 Training: Hidden1=128
Epoch 1/10 - Train Loss: 0.5344, Val Loss: 0.4929, Val Acc: 82.06%
Epoch 2/10 - Train Loss: 0.3906, Val Loss: 0.3847, Val Acc: 85.68%
Epoch 4/10 - Train Loss: 0.3248, Val Loss: 0.3524, Val Acc: 86.87%
Epoch 6/10 - Train Loss: 0.2905, Val Loss: 0.3350, Val Acc: 87.78%
Epoch 8/10 - Train Loss: 0.2673, Val Loss: 0.3290, Val Acc: 88.23%
Epoch 10/10 - Train Loss: 0.2477, Val Loss: 0.3031, Val Acc: 89.26%

 Training: Hidden1=256
Epoch 1/10 - Train Loss: 0.5190, Val Loss: 0.4131, Val Acc: 84.63%
Epoch 2/10 - Train Loss: 0.3779, Val Loss: 0.3637, Val Acc: 86.51%
Epoch 4/10 - Train Loss: 0.3158, Val Loss: 0.3397, Val Acc: 87.00%
Epoch 6/10 - Train Loss: 0.2786, Val Loss: 0.3447, Val Acc: 87.29%
Epoch 8/10 - Train Loss: 0.2483, Val Loss: 0.3139, Val Acc: 89.02%
Epoch 10/10 - Train Loss: 0.2280, Val Loss: 0.3104, Val Acc: 89.24%

 Training: Hidden1=512
Epoch 1/10 - Train Loss: 0.5077, Val Loss: 0.4167, Val Acc: 84.27%
Epoch 2/10 - Train Loss: 0.3744, Val Loss: 0.3798, Val Acc: 85.58%
Epoch 4/10 - Train Loss: 0.3092, Val Loss: 0.3434, Val Acc: 87.41%
Epoch 6/10 - Train Loss: 0.2728, Val Loss: 0.3349, Val Acc: 87.77%
Epoch 8/10 - Train Loss: 0.2449, Val Loss: 0.2988, Val Acc: 89.03%
Epoch 10/10 - Train Loss: 0.2229, Val Loss: 0.3179, Val Acc: 88.91%
```

```
--- EXPERIMENT 2: Optimizer ---

 Training: Optimizer=Adam
Epoch 1/10 - Train Loss: 0.5362, Val Loss: 0.4091, Val Acc: 84.90%
Epoch 2/10 - Train Loss: 0.3888, Val Loss: 0.3692, Val Acc: 86.41%
Epoch 4/10 - Train Loss: 0.3251, Val Loss: 0.3300, Val Acc: 87.81%
Epoch 6/10 - Train Loss: 0.2881, Val Loss: 0.3166, Val Acc: 88.19%
Epoch 8/10 - Train Loss: 0.2617, Val Loss: 0.3317, Val Acc: 87.81%
Epoch 10/10 - Train Loss: 0.2406, Val Loss: 0.3147, Val Acc: 88.55%

 Training: Optimizer=SGD
Epoch 1/10 - Train Loss: 2.1960, Val Loss: 2.0678, Val Acc: 36.45%
Epoch 2/10 - Train Loss: 1.8839, Val Loss: 1.6682, Val Acc: 51.13%
Epoch 4/10 - Train Loss: 1.1336, Val Loss: 1.0195, Val Acc: 69.59%
Epoch 6/10 - Train Loss: 0.8326, Val Loss: 0.7906, Val Acc: 73.24%
Epoch 8/10 - Train Loss: 0.7214, Val Loss: 0.7012, Val Acc: 74.62%
Epoch 10/10 - Train Loss: 0.6664, Val Loss: 0.6531, Val Acc: 76.23%

 Training: Optimizer=RMSprop
Epoch 1/10 - Train Loss: 0.5188, Val Loss: 0.5296, Val Acc: 80.24%
Epoch 2/10 - Train Loss: 0.3856, Val Loss: 0.4482, Val Acc: 83.22%
Epoch 4/10 - Train Loss: 0.3202, Val Loss: 0.3395, Val Acc: 87.08%
Epoch 6/10 - Train Loss: 0.2858, Val Loss: 0.3993, Val Acc: 85.85%
Epoch 8/10 - Train Loss: 0.2605, Val Loss: 0.3222, Val Acc: 88.31%
Epoch 10/10 - Train Loss: 0.2391, Val Loss: 0.3840, Val Acc: 86.45%
```

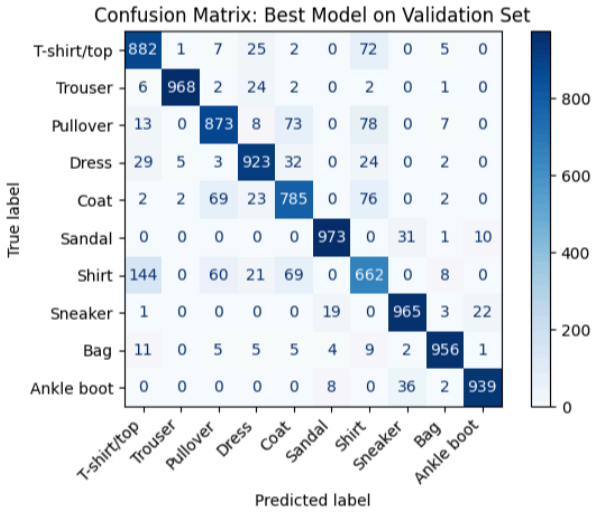
From our observations, we can see that slight increases in hidden layer size beyond 128 did not yield significant gains, and a learning rate of 0.001 provided stable convergence.

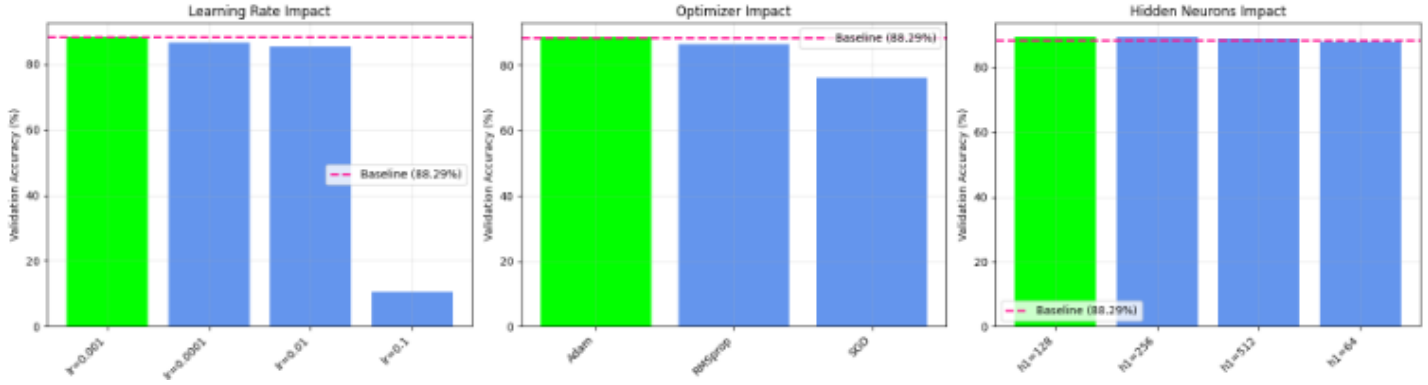
The best configuration to use was a learning rate around 0.001, a Hidden layer 1 with 128 neurons, and with the Adam optimizer, although very barely when compared to the RMSprop optimizer.

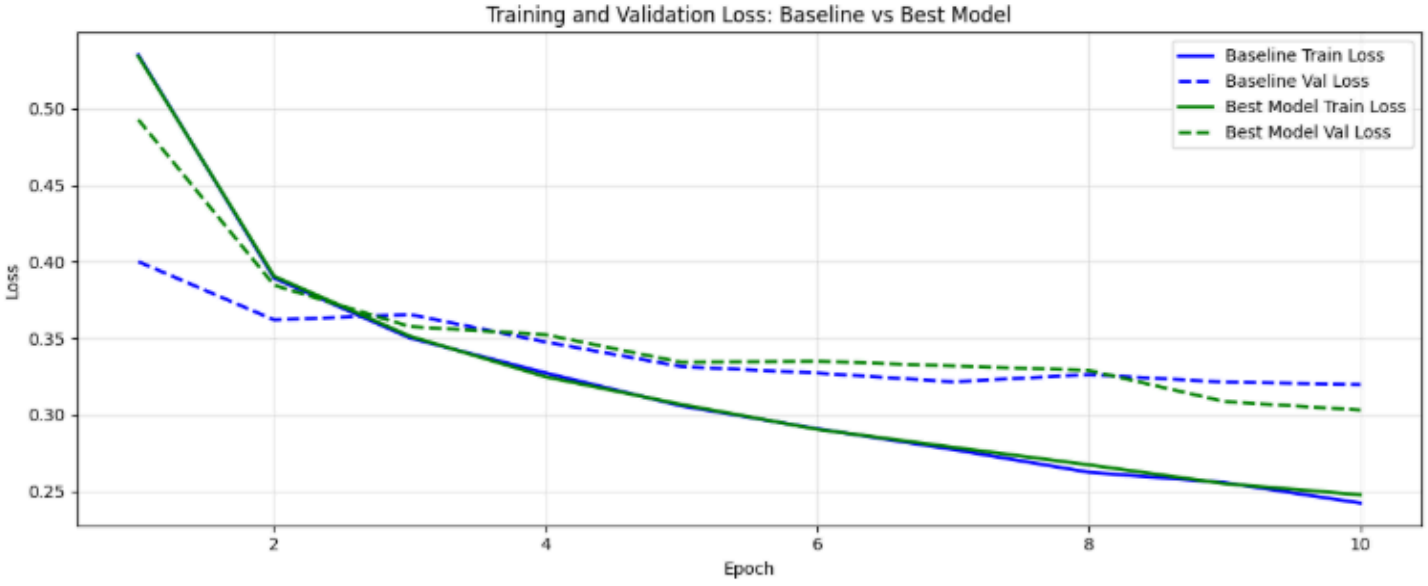
Results Analysis:

Results :

```
============================================================
RESULTS TABLE
============================================================
    learning_rate optimizer  hidden1_neurons  final_accuracy
8          0.0010      Adam              128           89.26
9          0.0010      Adam              256           89.24
10         0.0010      Adam              512           88.91
4          0.0010      Adam              128           88.55
2          0.0010      Adam              128           88.22
7          0.0010      Adam               64           87.94
3          0.0001      Adam              128           86.70
6          0.0010   RMSprop              128           86.45
1          0.0100      Adam              128           85.44
5          0.0010       SGD              128           76.23
0          0.1000      Adam              128           10.52
```







The best model (Run 8: lr=0.001, Adam, 128 neurons) achieved **89.26%** validation accuracy, representing a significant improvement of **+31.01** percentage points over the baseline MLP model (**58.25%** accuracy). This demonstrates that proper hyperparameter tuning is vital for optimal model performance.

Learning rate had the most dramatic impact on performance. Extreme values (0.1) caused complete training failure (**10.52%** accuracy), while very small values (0.0001) led to slow convergence (**86.70%**). The optimal range was 0.001, balancing convergence speed and stability. Among optimizers, **Adam** consistently outperformed **RMSprop** and **SGD** across all learning rates, but very minimally so when compared to **RMSprop**. The Adam/0.001 combination yielded the top 4 results.

The loss curves show that both baseline and best models exhibit some overfitting (validation loss higher than training loss), but the best model demonstrates:

- Lower final validation loss (~**0.35** vs ~**0.45**)
- Closer alignment between training and validation curves
- More stable convergence

This indicates that hyperparameter tuning (particularly optimal learning rate and Adam optimizer) helped prevent overfitting compared to the baseline.

The confusion matrix shows us that the following are frequently confused:

- Shirts misclassified as T-shirts/tops (**144** errors)
- Shirts, pullovers, and coats as well are often confused (**68 - 78** misclassifications)

These confusions are expected, since even people are likely to mistake the two due to their similar functions, visuals, and wear.
