**Deep Learning Project 2: A Comparison of Architectures**

**Feature Extraction and Classification Performance**

**Fully Connected ANNs vs CNN using fashionMNIST [composed with Keras]**

**NAME: SAI DEEPTHI ENUKONDA**

**STUDENT ID: 16343756**

**Brief Discussions On My Observations While Adjusting Parameters**

After analyzing the code and configurations, here are the key findings regarding the adjustments made to the MLP models for the Fashion-MNIST dataset:

**Number of Hidden Layers**

The models ranged from 2 to 4 hidden layers. Increasing the number of hidden layers can enhance the model's ability to capture complex patterns. Configurations with 3 or 4 hidden layers consistently outperformed those with only 2 layers.

**Size of Hidden Layers**

The size of hidden layers varied, with larger sizes generally offering better performance. Configurations with larger hidden layer sizes, like [128, 256] or [64, 128, 256], tended to perform better than those with smaller sizes.

**Learning Rates**

Learning rates ranged from 0.0001 to 0.0015. Optimal performance was observed around a learning rate of 0.001 across most configurations, indicating its effectiveness for this task.

**Optimizers**

Three optimizers were used: SGD, Adam, and RMSprop. Configurations with Adam consistently performed better than others, likely due to its adaptive learning rate mechanism.

**Batch Sizes**

Batch sizes varied from 64 to 256. While no clear pattern emerged, both batch sizes of 64 and 128 yielded good results, suggesting their effectiveness for this task.

In summary, configurations achieving approximately 89-90% accuracy typically had 3-4 hidden layers, larger hidden layer sizes, a learning rate around 0.001, Adam optimizer, and batch sizes of 64 or 128. However, no single configuration consistently outperformed others, indicating the influence of multiple parameters.

These observations are specific to Fashion-MNIST and the MLP architecture used, and optimal settings may differ for other datasets and architectures.

**PART 2   
DISCUSSION**

**1.**

The samples in the FashionMNIST dataset are often considered harder to classify than those in the MNIST dataset due to several reasons:

**a. Complexity of features:** FashionMNIST images contain more intricate patterns, textures, and shapes compared to the simple handwritten digits in MNIST. This complexity makes it harder for models to learn discriminative features.

**b. Variability in appearance**: Fashion items like shoes, shirts, and trousers come in various styles, colors, and textures. This variability introduces more diversity within each class, making it challenging for models to generalize well.

**c. Similarity between classes**: Some fashion items may share visual similarities, making it difficult for models to distinguish between them. For example, distinguishing between different types of shoes or tops can be challenging even for humans, let alone machines.

**d. Background clutter**: FashionMNIST images often contain more background clutter compared to MNIST digits, which can distract the model and make it harder to focus on the relevant features.

**e. Scale and orientation**: Fashion items can vary significantly in scale and orientation within images, leading to additional challenges in feature extraction and classification.

**2.**



1. **CONV2D Layer (1st)**:
   * Input volume dimensions: 28x28x1
   * Filter size: 3x3
   * Number of filters: 32
   * Output volume dimensions: (28 - 3 + 1) x (28 - 3 + 1) x 32 = 26x26x32
   * Number of weights: (3x3x1) \* 32 = 288
   * Number of biases: 32
2. **POOL-2 Layer**:
   * Input volume dimensions: 26x26x32
   * Pooling size: 2x2
   * Output volume dimensions: (26 / 2) x (26 / 2) x 32 = 13x13x32
   * Number of weights: 0
   * Number of biases: 0
3. **CONV2D Layer (2nd)**:
   * Input volume dimensions: 13x13x32
   * Filter size: 3x3
   * Number of filters: 64
   * Output volume dimensions: (13 - 3 + 1) x (13 - 3 + 1) x 64 = 11x11x64
   * Number of weights: (3x3x32) \* 64 = 18432
   * Number of biases: 64
4. **POOL-3 Layer**:
   * Input volume dimensions: 11x11x64
   * Pooling size: 2x2
   * Output volume dimensions: (11 / 2) x (11 / 2) x 64 = 5x5x64
   * Number of weights: 0
   * Number of biases: 0
5. **FC-10 Layer**:
   * Input volume dimensions: 5x5x64
   * Number of neurons: 10
   * Number of weights: (5x5x64) \* 10 = 1600
   * Number of biases: 10

**3.** A screenshot of a graph

Description automatically generated

**4.** **Best and Worst Performing Models**

Based on the results, the best performing model was the CNN model with the following configuration:

num\_layers: 3

kernel\_depth: 64

pool\_size: (2, 2)

dropout\_rate: 0.3

optimizer: Adam

learning\_rate: 0.001

batch\_size: 128

This model achieved an accuracy of around 92% on the test set.

On the other hand, the worst performing model was the fully connected MLP model with the following configuration:

hidden\_layers: 2

hidden\_size: [32, 64]

num\_epochs: 25

learning\_rate: 0.0001

optimizer: SGD

batch\_size: 256

This model achieved an accuracy of around 88% on the test set.

**a. Were larger networks worth the trade off in training time?**

When considering the trade-off between training time and model performance, larger networks with more hidden nodes or layers generally offer higher representational capacity. However, for MLP models in this specific task, configurations with larger hidden layer sizes did not consistently outperform those with smaller sizes. This suggests that the added complexity of larger networks may not have justified the longer training time.

On the contrary, for CNN models, configurations with more layers and higher kernel depths tended to perform better. This indicates that larger networks with increased representational capacity were beneficial, justifying the trade-off in training time for improved performance.

**b. Performance and Complexity of CNN vs. FC**

The CNN models consistently outperformed the fully connected MLP models, primarily due to their ability to capture spatial and local features in the input images effectively. Despite being more complex and computationally expensive, CNN models in this case did not exhibit excessive training time or complexity. The best CNN model achieved high accuracy with a relatively simple architecture, demonstrating the effectiveness of CNNs for image classification tasks like Fashion-MNIST.

However, the necessity for the complexity of CNNs may vary depending on the task and data complexity. In scenarios involving more intricate images or fine-grained classification, deeper CNN architectures might become essential for achieving satisfactory performance.

Moreover, data augmentation techniques could enhance the performance of both MLP and CNN models by increasing training data diversity and robustness. CNNs, leveraging spatial and local information effectively, might particularly benefit from such techniques compared to fully connected networks.