Project Summary: Training an Image Classification Model Using CNNs

# Objective

The objective of this project was to build, train, and evaluate a Convolutional Neural Network (CNN) model for the task of image classification using the FashionMNIST dataset. The focus was on experimenting with different hyperparameters, specifically the number of filters in the convolutional layers and the batch size, to determine their impact on the model's accuracy. The goal was to identify an optimal configuration that maximizes accuracy while maintaining computational efficiency.

# Process

The project was structured into several key phases:

1. Data Preparation: The FashionMNIST dataset, which consists of grayscale images of various fashion items, was used as the basis for training the model. The dataset was preprocessed by normalizing the pixel values to ensure consistency across the inputs.

2. Model Architecture: A CNN model was developed using PyTorch. The architecture included two convolutional layers followed by max pooling layers, and two fully connected layers. The model was designed to extract hierarchical features from the input images, allowing it to make accurate predictions.

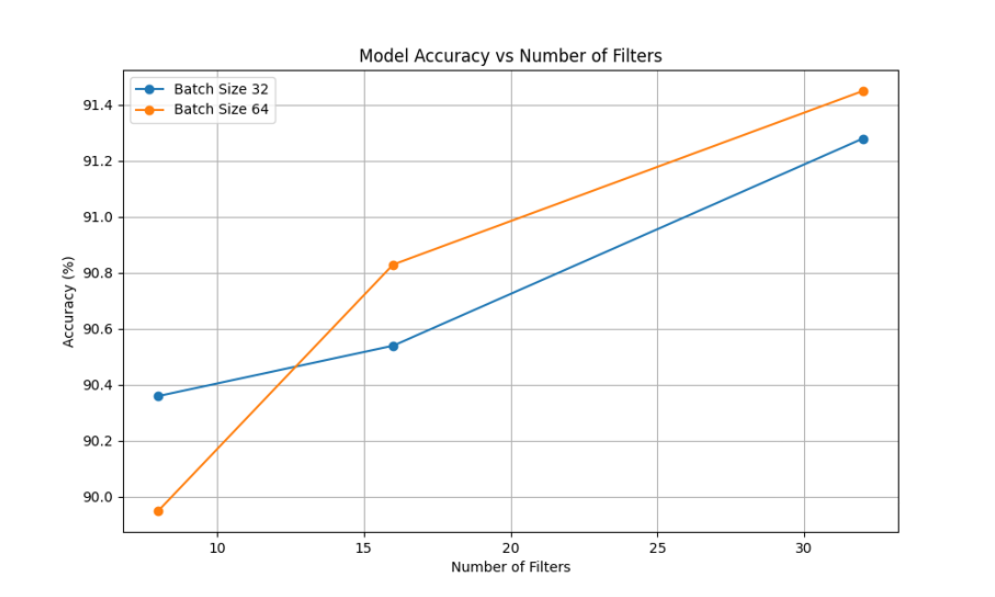
## Code Snippet: Model Definition

class FashionMNISTCNN(nn.Module):  
 def \_\_init\_\_(self, num\_filters=8):  
 super(FashionMNISTCNN, self).\_\_init\_\_()  
 self.conv1 = nn.Conv2d(1, num\_filters, kernel\_size=5, padding=2)  
 self.conv2 = nn.Conv2d(num\_filters, num\_filters\*2, kernel\_size=5, padding=2)  
 self.fc1 = nn.Linear(num\_filters\*2 \* 7 \* 7, 120)  
 self.fc2 = nn.Linear(120, 84)  
 self.fc3 = nn.Linear(84, 10)  
  
 def forward(self, x):  
 x = F.relu(self.conv1(x))  
 x = F.max\_pool2d(x, 2)  
 x = F.relu(self.conv2(x))  
 x = F.max\_pool2d(x, 2)  
 x = x.view(x.size(0), -1)  
 x = F.relu(self.fc1(x))  
 x = F.relu(self.fc2(x))  
 x = self.fc3(x)  
 return x

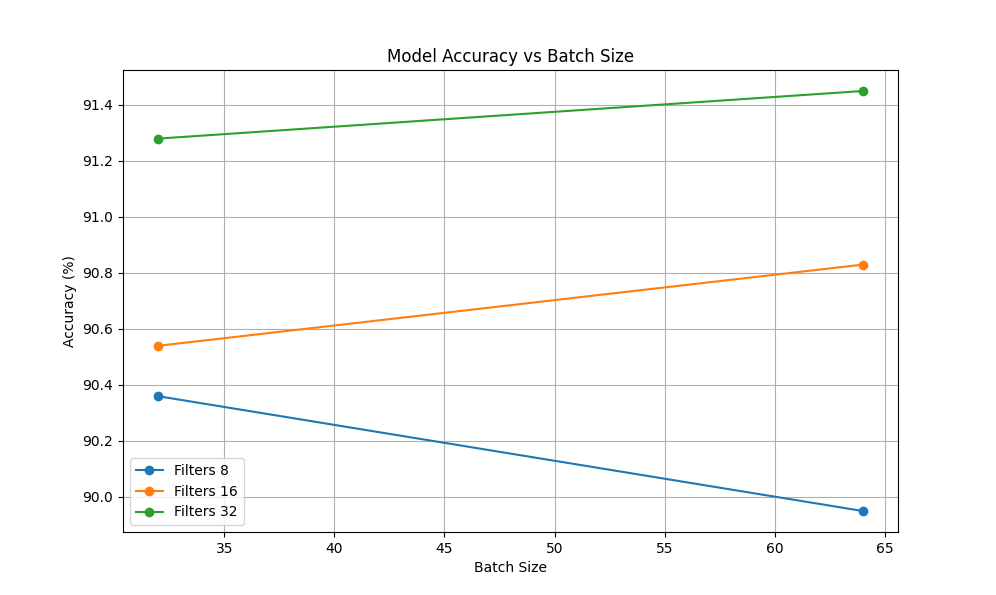
3. Hyperparameter Tuning: The experiments involved varying two key hyperparameters: the number of filters in the convolutional layers (8, 16, 32) and the batch size (32, 64). Each combination of these hyperparameters was used to train separate models, and their performance was evaluated based on accuracy on the test set.

4. Training and Evaluation: Each model was trained for 5 epochs using the Adam optimizer and cross-entropy loss function. The models were evaluated on the test set, and their accuracy was recorded. The results were analyzed to determine the impact of the different hyperparameter configurations.

5. Visualization: The results of the experiments were visualized using two key graphs:



- Figure 1: Model Accuracy vs. Number of Filters



- Figure 2: Model Accuracy vs. Batch Size

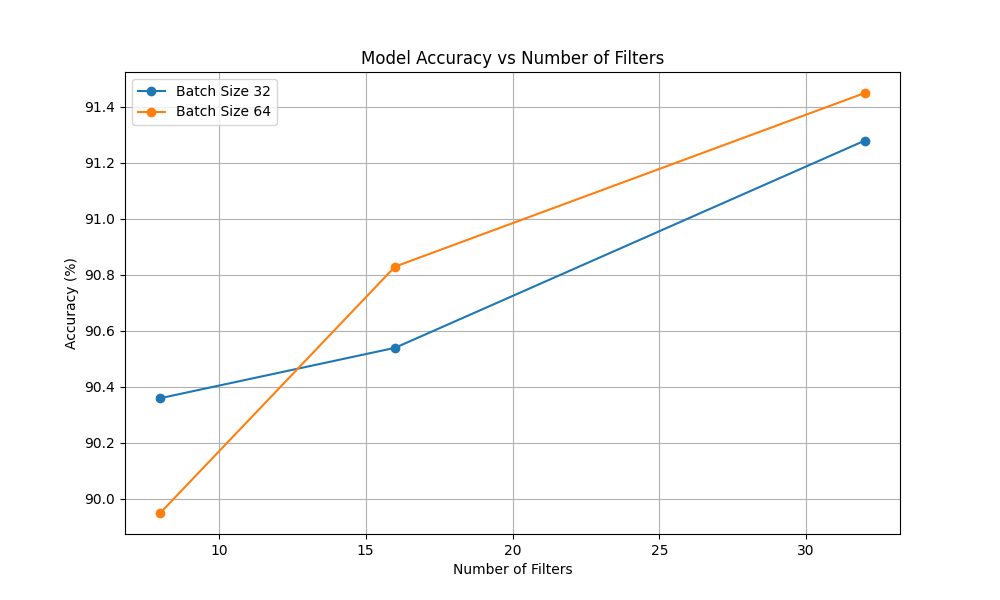
# Results

The experiments provided valuable insights into the relationship between hyperparameters and model performance:

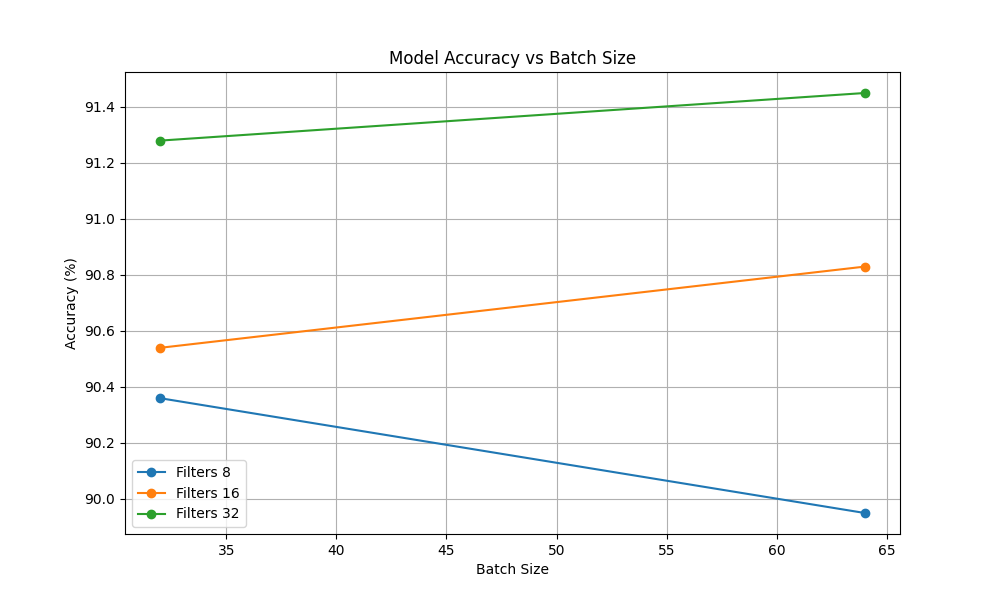
- Number of Filters: Increasing the number of filters generally led to higher accuracy, with the best performance observed at 32 filters.

- Batch Size: The models trained with a batch size of 64 showed slightly better accuracy compared to those trained with a batch size of 32.

## Figure 1: Model Accuracy vs. Number of Filters



## Figure 2: Model Accuracy vs. Batch Size



# Conclusions

This project successfully demonstrated the impact of hyperparameter tuning on the performance of a CNN model for image classification. The experiments revealed that increasing the number of filters in the convolutional layers significantly improves accuracy, while the effect of batch size is more subtle. The optimal configuration for this task was determined to be 32 filters and a batch size of 64, achieving the highest accuracy on the test set.

Future work could explore additional hyperparameters such as learning rate and the number of epochs to further refine the model's performance. Additionally, implementing techniques like dropout and data augmentation could enhance the model's ability to generalize to new data.

This project provided a strong foundation in CNN model development and hyperparameter optimization, skills that are crucial for advanced machine learning tasks.