Assignment 3 Report

CNN ARCHITECTURE COMPARISON: BASENET VS RESNET

# Objective

The objective of this assignment is to implement and compare two CNN architectures, BaseNet and ResNet, on the Fashion-MNIST dataset. The experiments focus on:

1. Evaluating the effect of increasing the layer count in the BaseNet architecture by comparing BaseNet-10 and BaseNet-16
2. Comparing the performance of BaseNet (plain CNN) and ResNet (Residual Network) architectures
3. Investigating how different optimizers, including learning schedule, momentum, and Adam, affect the convergence of BaseNet-16 and ResNet-18 models

# Experimental Setup

## Data Preparation

* Dataset
  + Fashion-MNIST
* Subset Size
  + 10,000 training samples (out of the full 60,000)
* Data Augmentation (Applied to only the training set)
  + Horizontal flip
  + Rotation: range 20
  + Scaling: range 0.4
  + Resizing: 36
  + Random crop: 5 pixels
* Preprocessing
  + Normalization: mean = 0.5 and std = 0.5
* Batch Size
  + 50 for all experiments

## BaseNet Architecture

* BaseNet-10
  + First layer
    - Convolution with 8 filters of size 3x3, same padding, stride 1
  + 3 modules with 2 convolution layers each:
    - Module 1: 8 filters, stride 1
    - Module 2: 16 filters, first layer with stride 2, others with stride 1
    - Module 3: 32 filters, first layer with stride 2, others with stride 1
  + Final layers
    - Global average pooling followed by fully connected softmax layer with 10 neurons
* BaseNet-16
  + Similar to BaseNet-10 but with 4 convolution layers per module instead of 2
* Common Features
  + Batch Normalization after each convolution layer
  + ReLU activation function
  + He initialization for weights

## ResNet Architecture

* ResNet-18
  + First layer
    - Convolution with 8 filters of size 3x3, same padding, stride 1
  + 4 residual modules with 2 residual blocks each:
    - Module 1: 8 filters, stride 1
    - Module 2: 16 filters, first convolution layer with stride 2, others with stride 1
    - Module 3: 32 filters, first convolution layer with stride 2, others with stride 1
    - Module 4: 64 filters, first convolution layer with stride 2, others with stride 1
  + Final layer
    - Global average pooling followed by fully connected softmax layer with 10 neurons
* Residual Block Structure
  + Two convolution layers with batch normalization and ReLU activation
  + Skip connection that adds the input to the output of the second convolution layer
  + Shortcut connection with 1x1 convolution when input and output dimensions are different

## Training Parameters

* Loss Function
  + Cross-entropy loss
* Optimizers
  + SGD with learning rate schedule: starting at 0.1, divided by 10 every 50 epochs
  + SGD with momentum: learning rate 0.01, momentum 0.9
  + Adam: learning rate 0.001
* Number of Epochs
  + 75 epochs for experiments 1 and 2
  + 5 epochs for experiment 3 (due to limited training resources)
* Evaluation Metrics
  + Training and testing accuracy, training and testing loss

# Experiments and Results

## 1st Experiment: Comparing the Performance on Layering a Model

### Objective:

* Evaluate the impact of increasing number of layers in the BaseNet architecture

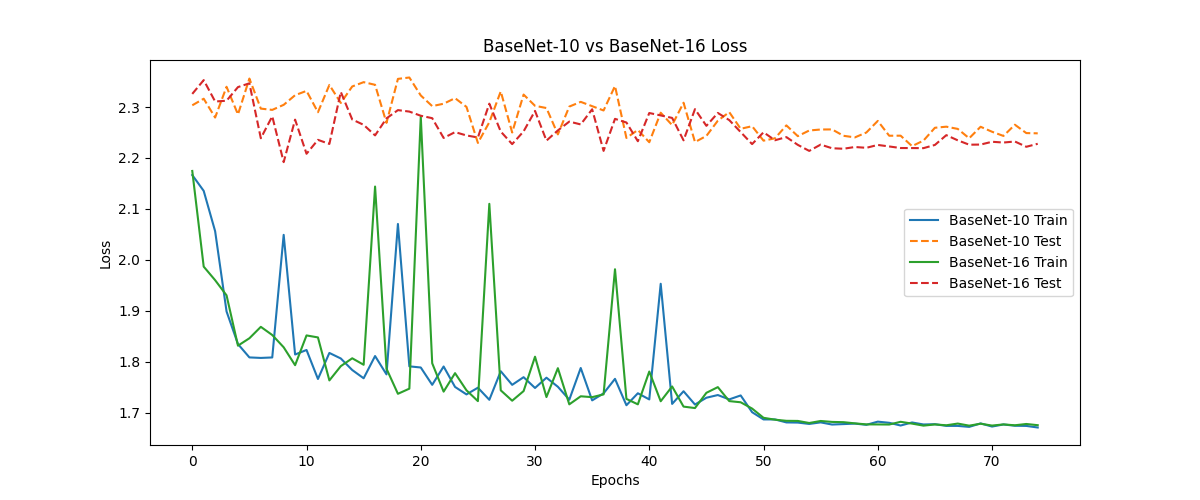
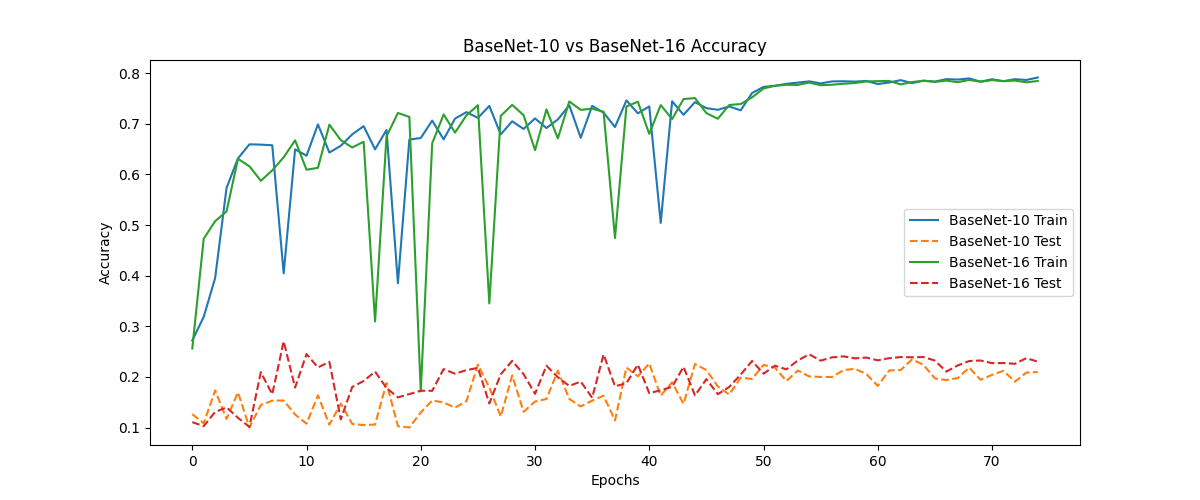
### Experimental Setup:

* Models
  + BaseNet-10: 2 convolution layers per module
  + BaseNet-16: 4 convolution layers per module
* Optimizer
  + SGD with learning rate schedule: starting at 0.1, divided by 10 every 50 epochs)
  + Batch size: 50
  + Number of epochs 75

### Observations:

* Both BaseNet-10 and BaseNet-16 show similar training accuracy patterns, reaching approximately 80% accuracy by the end of training
* The training curves for both models show significant fluctuations, especially in the early epochs, which indicates some instability during training
* The test accuracy for both models, being around 20-25%, suggests substantial overfitting
* The training loss for both models steadily decreases, reaching similar final values around 1.7
* The test loss remains high, around 2.2-2.3 for both models, confirming the overfitting issue

### Results:



## 2nd Experiment: Comparing the Performance of BaseNet and ResNet

### Objective:

* Compare the performance of BaseNet (plain CNN) and ResNet (Residual Network) architectures

### Experimental Setup:

* Models: BaseNet-10, BaseNet-16, and ResNet-18
* Optimizer:
  + SGD with learning rate schedule: starting at 0.1, divided by 10 every 50 epochs
* Batch size: 50
* Number of epochs 75

### Observations:

* ResNet-18 achieves slightly higher training accuracy, around 80%, compared to both BaseNet models
* ResNet-18 shows more stable training behavior with less dramatic drops in accuracy compared to BaseNet models
* The test accuracy for all models remains low, 20-25%, indicating overfitting across all architectures
* ResNet-18 shows marginally better test accuracy than both BaseNet models, specifically in later epochs
* The training curves for all models show significant fluctuations, but ResNet-18 appears to have slightly less severe drops

### Results: A graph of different colored lines AI-generated content may be incorrect.A graph of a graph AI-generated content may be incorrect.

### Inferences:

* The residual connections in ResNet-18 provide a small but noticeable improvement in both training stability and final performance compared to the plain BaseNet architectures
* The skip connections in ResNet likely help with gradient flow during backpropagation, leading to more stable training
* Despite the architectural differences, all models suffer from significant overfitting on this dataset
* The relatively small performance difference between ResNet and BaseNet models suggests that for this dataset and training configuration, the advantages of residual connections are present but modest

## 3rd Experiment: How Different Optimizers Affect Convergence

### Objective:

* Evaluate the impact of different optimization algorithms on the training of CNN models

### Experimental Setup:

* Models: BaseNet-16 and ResNet-18
* Optimizer:
  + SGD with learning rate schedule: starting at 0.1, divided by 10 every 50 epochs
  + SGD with momentum: learning rate 0.01, momentum 0.9
  + Adam: learning rate 0.001
* Batch size: 50
* Number of epochs 5 (shown in plots)

### Observations:

* Adam optimizer consistently outperforms both SGD variants for both BaseNet-16 and ResNet-18 models
* For BaseNet-16, Adam reaches approximately 65% training accuracy within 4 epochs, while SGD with learning schedule achieves around 50%, and SGD with momentum only reaches about 25%
* Similarly, for ResNet-18, Adam achieves around 65% training accuracy, SGD with learning schedule reaches about 60%, and SGD with momentum about 45%
* Adam shows the fastest convergence in terms of loss reduction for both models
* SGD with momentum shows the slowest convergence but exhibits steady improvement
* SGD with learning schedule shows intermediate performance between Adam and SGD with momentum

### Results:

A graph of a graph

AI-generated content may be incorrect.

A graph of different colored lines

AI-generated content may be incorrect.A graph of a line graph

AI-generated content may be incorrect.A graph of different colored lines

AI-generated content may be incorrect.

### Inferences:

* Adam's adaptive learning rate and momentum properties make it particularly effective for training CNN models on this dataset
* The superior performance of Adam is likely due to its ability to adapt the learning rate for each parameter individually, which helps navigate the complex loss landscape of deep neural networks
* SGD with momentum performs poorly with the chosen hyperparameters, suggesting that the learning rate of 0.01 might be too low for effective training
* The learning rate schedule improves SGD performance significantly compared to fixed learning rate with momentum, highlighting the importance of learning rate adjustment during training
* The relative performance of optimizers is consistent across both architectures, indicating that these observations about optimizer behavior are generalizable

# Conclusion

* The experiments conducted in this assignment provide valuable insights into CNN architecture design and training:

### Effect of Model Depth:

* Increasing the number of layers from BaseNet-10 to BaseNet-16 does not significantly improve performance on the Fashion-MNIST dataset with the given training configuration
* Both models show similar training accuracy patterns but suffer from significant overfitting

### Residual vs. Plain Networks:

* ResNet-18 shows slightly better performance and training stability compared to both BaseNet models
* The skip connections in ResNet help with gradient flow during backpropagation, leading to more stable training
* However, the performance difference is modest, suggesting that for this dataset and training configuration, the advantages of residual connections are present but not dramatic

### Optimizer Impact:

* Adam consistently outperforms both SGD variants for both architectures
* SGD with learning rate schedule performs better than SGD with fixed learning rate and momentum
* The choice of optimizer has a significant impact on training speed and final performance

### Overfitting Challenges:

* All models show a large gap between training and testing performance, indicating significant overfitting
* This suggests that additional regularization techniques beyond batch normalization might be necessary for better generalization

## Summary

These findings highlight the importance of architectural choices, optimization strategies, and regularization techniques in CNN design. While deeper networks and residual connections can improve performance, their benefits may be limited without proper regularization, especially on smaller datasets. Additionally, the choice of optimizer can significantly impact training dynamics and final model performance.

# References

* He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).*
* Xiao, H., Rasul, K., & Vollgraf, R. (2017). *Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747.*
* Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.*
* Ioffe, S., & Szegedy, C. (2015). *Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456).*
* He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).*
* *Course Lecture Slides and Notes on Convolutional Neural Networks and Deep Learning on Brightspace*