Assignment 1

Group 42

CS-671: Deep Learning and Applications
Coding Assignment 1 – FCNN

Group Members:

Roll No.	Name
b22109	Kanika Choudhary
b22031	Aman Sharma
b22249	Vikky Kumar
b22281	Vershita Yadav
b22051	Mihir Chandra
b22043	Harshit
b22069	Sowmika Rao

Instructor: Dr. Aditya Nigam

Course: CS-671 Deep Learning and Applications

Date: February 08, 2026

Question1

Problem Definition & Methodology

The objective of this study is to analyze the impact of feature scaling on the training behavior of a fully connected neural network for binary classification. The dataset contains features with vastly different numerical ranges, such as Age (small range) and Capital Gain (very large range). A neural network was implemented from scratch using NumPy, trained using binary cross-entropy loss and stochastic gradient descent. Two experiments were conducted: one using raw features and another using Min–Max scaled features. Both experiments used identical model architectures and hyperparameters to ensure a fair comparison.

Hyperparameter Tuning & Architecture Choices

The neural network architecture consists of an input layer, one hidden layer with ReLU activation, and an output layer with sigmoid activation for binary classification. Key hyperparameters include a learning rate of 0.01, batch gradient descent optimization, and a fixed number of training epochs with early stopping based on validation accuracy improvement. Hyperparameters were kept constant across both experiments to isolate the effect of feature scaling.

Results, Visualizations, and Interpretations

The final test accuracy was approximately the same (~75.9%) for both raw and scaled features. However, training dynamics differed significantly. Without scaling, the model started with a very high loss and required many epochs to converge, showing unstable gradients and numerical sensitivity. With Min–Max scaling, the initial loss was close to the theoretical optimum for random guessing (~0.69), convergence was much faster, and training remained stable. Although accuracy did not improve dramatically, the efficiency and robustness of training improved substantially after scaling.

Key Findings from Latent Space Analysis

Analysis of the hidden layer activations (latent space) revealed that, without feature scaling, neurons were disproportionately influenced by high-magnitude features such as Capital Gain, leading to skewed representations and inefficient learning. After Min–Max scaling, the latent space became more balanced, with neurons responding to a combination of features rather than being dominated by a few large-scale inputs. This resulted in smoother gradients, more meaningful feature representations, and improved optimization behavior, even though the final classification accuracy remained similar.

Conclusion

This study demonstrates that feature scaling is critical for stable and efficient neural network training. While it may not always lead to higher final accuracy, scaling ensures faster convergence, prevents gradient instability, and produces more interpretable latent representations. Therefore, feature scaling should be considered a fundamental preprocessing step in neural network pipelines.

Part - 2

Vision & Feature Interpretation

1. Problem Definition

The objective of this study is to analyze how Fully Connected Neural Networks (FCNNs) learn patterns from raw pixel data and to investigate their limitations with respect to spatial understanding. Specifically, the study aims to answer the following questions:

1. What types of visual patterns are learned by the first hidden layer of an FCNN trained on the MNIST dataset?

2. How does the absence of spatial awareness affect the network's ability to classify images when pixel order is disrupted?

To address these questions, two experiments were conducted:

1. Weight Visualization on Normal MNIST

2. Pixel Flattening (Scrambling) Experiment

2. Methodology

The MNIST dataset, consisting of handwritten digit images of size 28×28, was used in CSV format. Each image was flattened into a 784-dimensional vector before being passed to the network.

A Fully Connected Neural Network with two hidden layers was trained using supervised learning. The network was evaluated under two conditions:

1. Training and testing on the original MNIST dataset.

2. Training and testing on a version of MNIST where pixel positions were randomly shuffled using a fixed permutation across all images.

Model performance was evaluated using classification accuracy on the test set.

2. Hyperparameter Tuning and Architecture Choices

Network Architecture

The chosen FCNN architecture is summarized as follows:

1. Input Layer: 784 neurons (flattened image)

2. Hidden Layer 1: 128 neurons, ReLU activation

3. Hidden Layer 2: 64 neurons, ReLU activation

4. Output Layer: 10 neurons (digit classes)

This architecture was selected to balance representational capacity and computational efficiency while remaining simple enough to highlight the inherent limitations of FCNNs.

Hyperparameters

The following hyperparameters were used consistently across both experiments:

1. Batch Size: 64

2. Learning Rate: 0.001

3. Optimizer: Adam

4. Loss Function: Cross-Entropy Loss

5. Number of Epochs: 5

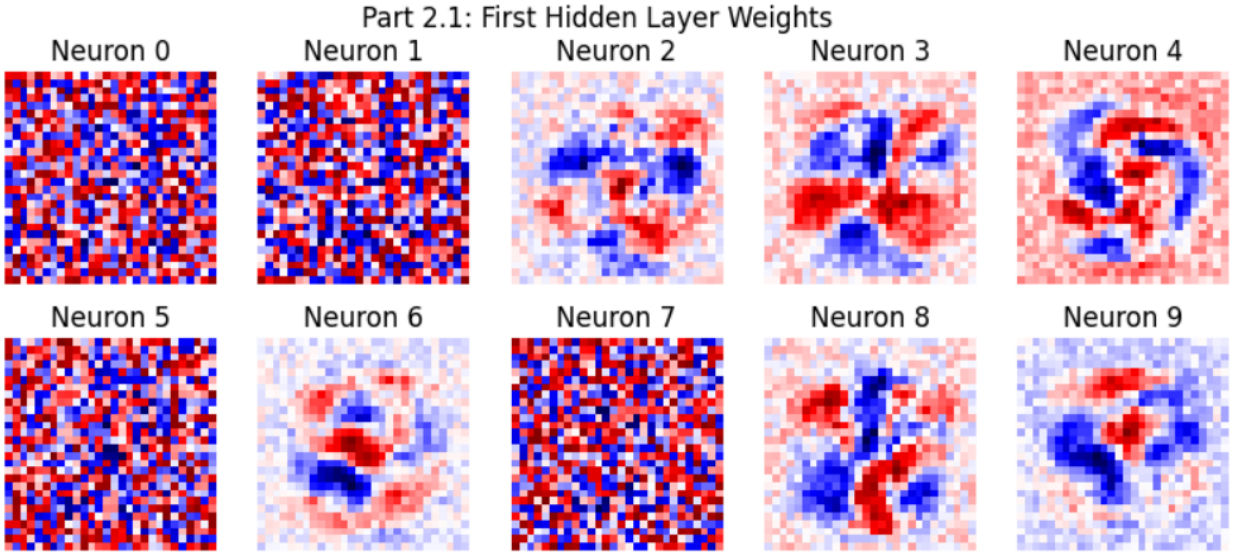
Adam optimizer was chosen due to its fast convergence and robustness. ReLU activation was used to mitigate vanishing gradient issues.

No extensive hyperparameter tuning was performed, as the primary goal was conceptual analysis rather than maximum accuracy.

3. Results, Visualizations, and Interpretations

3.1 Weight Visualization of the First Hidden Layer

The weights of the first hidden layer were extracted after training on the normal MNIST dataset. Each neuron's weight vector was reshaped into a 28×28 grid and visualized as a heatmap.



Interpretation :

The visualized weight patterns reveal that the network learns:

1. Stroke-like structures
2. Curved and linear edge patterns
3. Regions of strong positive and negative pixel correlations

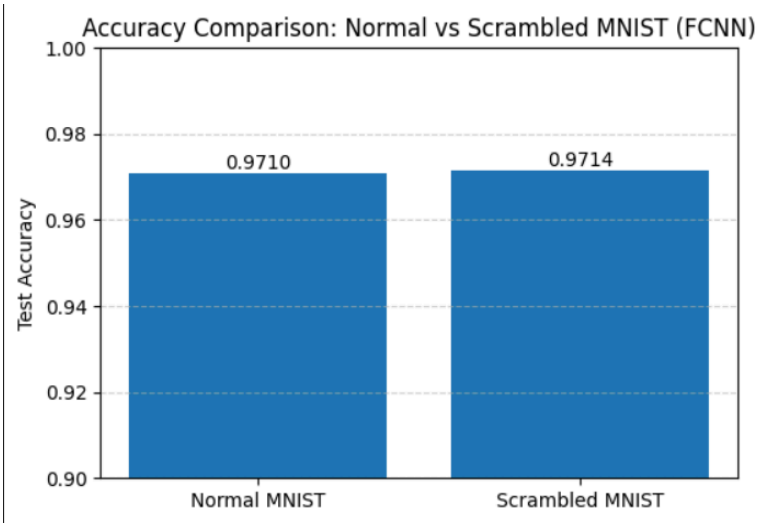
These patterns do not correspond to complete digits. Instead, they resemble partial digit fragments or low-level visual features, similar to edges or strokes. Some neurons appear noisy, indicating weak or distributed correlations across pixels.

This proves the fact that the FCNN learns statistical pixel dependencies rather than semantically meaningful shapes.

3.2 Performance Comparison: Normal vs Scrambled MNIST

The model achieved the following test accuracies:

- Normal MNIST Accuracy: 97.10%
- Scrambled MNIST Accuracy: 97.14%



Interpretation :

Despite the complete destruction of spatial structure in the scrambled dataset, the FCNN achieves nearly identical accuracy. This occurs because the network processes inputs as unordered vectors and does not encode spatial relationships between pixels.

As long as the same permutation is applied consistently across all samples, the network can learn a deterministic mapping from pixel values to class labels. This behavior sharply contrasts with human perception, which relies heavily on spatial continuity and shape structure.

4. Key Findings from Latent Space Analysis

Although the FCNN does not explicitly construct a geometric latent space like convolutional networks, the activations in the hidden layers can be interpreted as a latent representation of the input data.

Key observations include:

1. The latent representations encode combinations of pixel activations, not spatial features.

2. Digit separability arises from statistical correlations rather than shape understanding.

3. Scrambling pixels alters the semantics of the latent space but preserves class separability due to consistent feature-label mapping.

4. The latent space remains discriminative despite losing visual interpretability.

This indicates that high classification accuracy does not imply meaningful feature learning, especially when spatial priors are absent.

5. Conclusion

This study demonstrates that Fully Connected Neural Networks can achieve high classification accuracy on image datasets without learning spatial structure. Weight visualization shows that FCNNs learn low-level pixel correlation patterns rather than complete visual forms. The flattening experiment further reveals that FCNNs are invariant to pixel permutations, provided the permutation is consistent.

These findings highlight a critical limitation of FCNNs and motivate the use of Convolutional Neural Networks, which explicitly incorporate spatial locality and translation invariance. The experiments emphasize the distinction between statistical pattern recognition and true visual understanding.

CS-671: Deep Learning and Applications

Coding Assignment1: Fully Connected Neural Network (FCNN)

Stress Testing and Robustness Analysis on Tiny ImageNet

QUESTION: 3

Instructor: Aditya Nigam

Course: CS-671 Deep Learning and Applications

February 08, 2026

February 8, 2026

1 Introduction

Deep neural networks often face challenges such as vanishing gradients, unstable training, and sensitivity to hyperparameters. This study investigates the robustness of very deep fully connected neural networks using the Tiny ImageNet-10 dataset.

The experiment focuses on:

- Investigating vanishing gradient behavior in deep networks
- Comparing sigmoid activation with ReLU + Batch Normalization
- Performing an ablation study on optimizer, dropout, and learning rate

2 Dataset

The Tiny ImageNet-10 dataset (subset of Tiny ImageNet-200) was used. Images are of size 64×64 with 10 classes. Data was normalized and split into training and validation sets.

3 Question 3.1: Vanishing Gradient Experiment

3.1 Model Architecture

A very deep Fully Connected Neural Network (FCNN) was implemented:

- Input size: $64 \times 64 \times 3$
- Hidden layers: 8+ layers, 512 neurons each
- Optimizer: Adam
- Loss: Cross Entropy
- Epochs: 15

3.2 Experiment A: Sigmoid Activation

A deep network using sigmoid activation in all layers was trained.

Training Loss Behavior:

- Initial Loss: 2.62
- Final Loss: 2.31
- Training converged slowly

Average Gradient Norm:

$$\textbf{Sigmoid: } 1.84 \times 10^{-7}$$

This extremely small gradient confirms the vanishing gradient problem.

3.3 Experiment B: ReLU + Batch Normalization

The same architecture was trained using ReLU activation and Batch Normalization.

Training Loss Behavior:

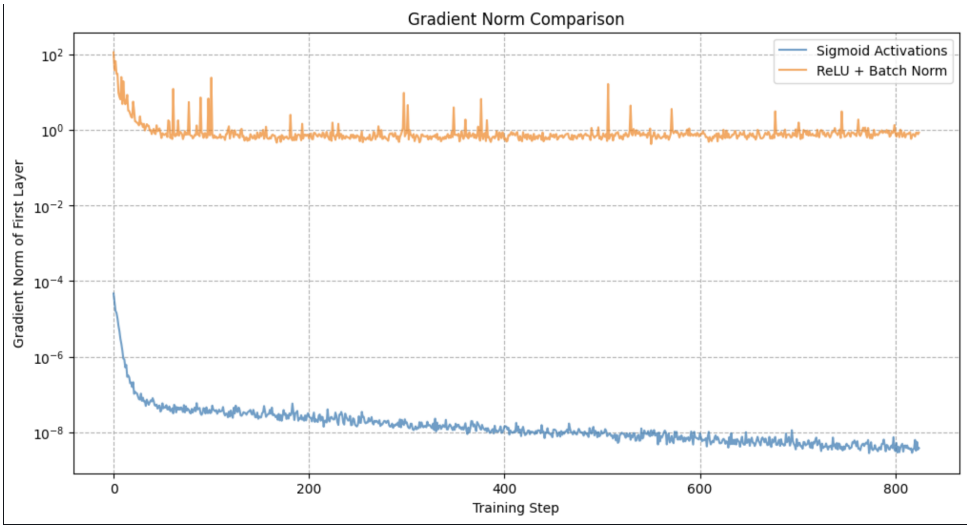
- Initial Loss: 2.34
- Final Loss: 1.10
- Faster convergence observed

Average Gradient Norm:

$$\textbf{ReLU + BN: 1.3370}$$

This shows stable gradient flow and faster learning.

3.4 Gradient Norm Comparison



Observation: Sigmoid gradients quickly approach zero, indicating vanishing gradients. ReLU with batch normalization maintains stable gradients throughout training.

Explanation: Sigmoid saturates for large positive or negative inputs, causing gradients to approach zero. ReLU avoids saturation in the positive region and batch normalization stabilizes activations, enabling effective gradient propagation in deep networks.

4 Question 3.2: Ablation Study

4.1 Objective

To analyze the effect of dropout, learning rate, and optimizer on model performance.

4.2 Baseline Configuration

- Optimizer: Adam
- Learning Rate: 0.001
- Dropout: 0.5

Best Validation Accuracy:

$$\textbf{Baseline} = \textbf{0.364}$$

4.3 Results Summary

Configuration	Final Accuracy	Change	Impact
Vanilla SGD	0.106	-0.258	0.258
Low LR (0.0001)	0.294	-0.070	0.070
No Dropout	0.412	+0.048	0.048
High LR (0.01)	0.334	-0.030	0.030
Baseline	0.364	0.000	0.000

4.4 Observations

- Removing dropout increased accuracy to 0.412 but caused instability in later epochs due to overfitting.
- Low learning rate caused slow convergence and underfitting.
- High learning rate caused unstable training.
- Switching from Adam to SGD produced the largest drop in performance.

4.5 Biggest Impact

The optimizer had the greatest impact on performance. Using vanilla SGD reduced accuracy drastically compared to Adam, highlighting the importance of adaptive optimization.

5 Problem Definition and Methodology

The objective of this assignment is to design and analyze a very deep Fully Connected Neural Network (FCNN) for image classification on the Tiny ImageNet-10 dataset. The study focuses on evaluating training stability, gradient flow, and robustness of deep neural networks under different architectural and optimization settings.

Two primary investigations were performed. First, the vanishing gradient problem was analyzed by comparing deep networks using sigmoid activation with networks using ReLU and batch normalization. Second, an ablation study was conducted to evaluate the effect of dropout, optimizer choice, and learning rate variations on final model performance.

The dataset was preprocessed using normalization and loaded using PyTorch data loaders. All models were trained using cross-entropy loss and evaluated on validation accuracy. Gradient norms of the first layer were tracked to analyze gradient flow in deep architectures.

6 Hyperparameter Tuning and Architecture Choices

A very deep Fully Connected Neural Network with more than eight hidden layers was implemented. Each hidden layer consisted of 512 neurons. The input size was 64×64×3 corresponding to Tiny ImageNet images.

Architecture Details:

- Input Layer: Flattened image vector
- Hidden Layers: 8+ fully connected layers (512 units each)
- Output Layer: Softmax classifier

Training Configuration:

- Loss Function: Cross-Entropy Loss
- Optimizer: Adam (baseline)
- Batch Size: 64
- Epochs: 15

Hyperparameters explored in ablation study:

- Learning Rate: 0.0001, 0.001, 0.01
- Optimizer: Adam vs Vanilla SGD
- Dropout: 0.5 vs No Dropout
- Activation Functions: Sigmoid vs ReLU + BatchNorm

These hyperparameters were chosen to analyze their impact on gradient stability and final accuracy.

7 Results, Visualizations, and Interpretation

7.1 Vanishing Gradient Analysis

The deep sigmoid network showed extremely small gradient norms in early layers, confirming the vanishing gradient problem. The average gradient norm observed was:

$$\text{Sigmoid: } 1.84 \times 10^{-7}$$

In contrast, the ReLU network with batch normalization maintained strong and stable gradients:

ReLU + BatchNorm: 1.3370

This demonstrates that ReLU activation combined with batch normalization enables effective gradient propagation in very deep networks.

7.2 Training Loss Comparison

Sigmoid network converged slowly and remained near high loss values, while ReLU + BatchNorm showed rapid loss reduction and better convergence.

7.3 Ablation Study Results

Configuration	Validation Accuracy	Impact
Baseline (Adam, LR=0.001)	0.364	—
No Dropout	0.412	Improved but unstable
Low LR (0.0001)	0.294	Underfitting
High LR (0.01)	0.334	Unstable training
Vanilla SGD	0.106	Major performance drop

Interpretation: Optimizer choice had the greatest impact on performance. Adam provided stable convergence, while SGD resulted in significantly lower accuracy. Removing dropout increased accuracy initially but caused overfitting and instability in later epochs.

8 Key Findings from Representation Analysis

The experiments reveal important insights into representation learning in deep networks. Sigmoid activations compress representations into a narrow range, causing gradient shrinkage and poor feature propagation across layers. This leads to weak latent representations in early layers.

ReLU activation preserves positive activations and avoids saturation, allowing richer feature representations to propagate through depth. Batch normalization further stabilizes internal feature distributions, improving representation quality and training stability.

The ablation study confirms that optimization strategy significantly affects learned representations. Adaptive optimizers such as Adam enable better exploration of the loss landscape and improved feature learning compared to vanilla SGD.

9 Final Conclusion

This study demonstrated the vanishing gradient problem in deep sigmoid networks and showed that ReLU with batch normalization significantly improves gradient flow and training speed.

The ablation study revealed that optimizer choice has the largest effect on performance, followed by learning rate and dropout. Removing dropout improved accuracy but introduced instability due to overfitting.

Overall, modern deep learning techniques such as ReLU activation, batch normalization, and Adam optimization are essential for stable and efficient training of very deep neural networks.