# Deep Learning Project

TKO_7094-3003 Introduction to Deep Learning

Year 2025

**Submitted By:**

**First name:** Tanuraj

**Lastname:** Saha

**Student Number:** 2411806

**Date of Submission:** 18.05.2025

## Introduction

The CIFAR-100 image dataset provides 100 classes and is an important challenge for training convolutional neural networks (CNNs), especially in constrained environments like Google Colab. As a solution, we extracted smaller subsets of 10 classes to conduct systematic experiments on image classification. Thus, a CNN was to be built from scratch, whose performance was to be improved through iterations of architectural change, activation function, optimizer, and training configuration. The experiments were designed to test the effect of each variation on the final test accuracy, with observations made from original as well as subsequent trials using different subsets of classes.

## 1. Preparation of Dataset

**i) Dataset**: CIFAR-100

**ii) Subset Selection**:

- 1st subset: Class Range (0–9)
- 2nd subset (Repeated): Class Range (20–29)

For the unpaid colab environment, we need to reduce size for faster training and lower memory usage, which simplifies analysis while maintaining variability.

## 2. Baseline Model (Task 3,4)

- **Architecture**: Basic CNN having 2 Conv/MaxPooling layers, 32 and 64 filters, and Dense layers.
- **Training Setup**: 15 epochs and batch size 64
- **Accuracy**: 65.20%
- **Purpose**: Establishing a starting benchmark

## 3. Increasing Model Depth & Size (Task 5.1)

- **Changes**: Increasing the size from 32 to 64 and 128 of convolutional 2D, BatchNorm, and Dense units. Here I have used 15 epochs and 25 epochs for this analysis.
- **Best Accuracy**: 66.60% (15 epochs)
- **Observation**: Deeper models improved accuracy up to a point; very deep ones very slightly underperformed due to overfitting or complexity. For 25 epochs this issue has been found. (66%)

## 4. Convolutional/Max Pooling (Task 5.2)

- **Changes**: I have used more layers from 32 to 64,128 and 256 of convolutional 2D, BatchNorm, and Dense units. Here I have used 15 epochs and 30 epochs for this analysis.
- **Best Accuracy**: 69.40% (30 epochs)
- **Observation**: A higher number of layers means it is a more powerful model with more parameters, and which is more capable of collecting important features of the image data. But here I found the accuracy has been increased in case of 30 epochs.

## 5. Activation Function Experiments (Task 5.3)

- **Functions Tested**: ReLU, Leaky ReLU, Sigmoid, Tanh, Swish
- **Results**:
    - **Leaky ReLU** (67.20%), **Tanh** (68.20%), **Swish** (65.20%)
    - **Sigmoid** underperformed in all cases (~40%)
- **Observation**: Tanh tops the charge. ReLU variants and Swish outperformed Sigmoid because of having better gradient flow and learning capability.

## 6. Effect of Activation Functions and Their Interaction with Network Depth and Size

Different activation functions matter a lot for model performance. Your experiments gave me the following results:

- Tanh came out on top, giving 68.20% accuracy (initial subset), truly showing its prowess in non-linear transformations.
- Leaky ReLU managed to do very well as well, scoring up to 67.20% (initial) and 66.20% (new subset).
- Swish had erratic behaviour, 65.20% moderate under initial and 63.80% under new subset.
- Sigmoid really lagged behind, giving scores of only around 40%, probably due to the vanishing gradient problem in deeper nets.

When activation function choice was combined with network sizes and depths, the results revealed that activations like Tanh and Leaky ReLU better scale with deep networks. For example, in very deep architectures, Tanh and Leaky ReLU kept high accuracies while Sigmoid could not benefit from the additional depth due to saturation of gradients.

Thus, the effectiveness of an activation function is not just an inherent property of its own, but one that is, to some extent, dictated by the chosen network architecture. In this sense, pairing activation functions like Tanh or Leaky ReLU with deeper networks that are well regularized yielded stronger performance gains.

## 7. Optimizer Experiments (Task 5.5)

- **Optimizers Tested**: Adam, SGD, RMSprop
- **Best**: Adam (64.40%)
- **Observation**: Adam provided stable convergence; RMSprop with low LR did not provide good result (47.80%).

### 8. Batch Size & Epoch Variations (Task 5.6)

- **Variations Tested**: (Size: 32,64,128) and (Epochs :10,15,20)
- **Best Configuration**: Batch size 64, 15 epochs → 68.60%
- **Observation**: In this case, the best accuracy observed for 64 batch size and 15 epochs. This is because of better iteration as tested on higher batch size and more epochs. Though for batch size 128 and 20 epoch decreased due to overfitting

### 9. Task Repetition with New Subset (Classes 20–29)

In this case, I had to perform all the tasks with new subsets (20-29) again to check the performance and compare with previous subsets (0-9). ImageNet was not used due to having resource constraint of my device.

- **Baseline**: 67.40%
- **Deeper Model (Increase Size and Depth)**: 71.40%
- **Very deep model (Convolutional/Maxpooling for more layers) :** 70.20%
- **Best Activations**: Tanh (67.00%), Leaky ReLU (63.80%)
- **Optimizers:** Adam is leading here with 66.60% too following SGD (61.40%)
- **Experiment with Batch Size and Epochs:** With Batch 128 and Epoch 20, I got 68.60% (Highest)

**Overall Observation**: Experiments on CIFAR-100 subsets (classes 0–9 and 20–29) confirmed consistent trends in CNN performance. Accuracy increased with model depth and layer count to a point, whereas models, which were too deep, were prone to overfitting. Tanh and Leaky ReLU activation functions fared the best, with the Sigmoid function performing the worst. Adam was found to be the most stable and accurate optimizer. The larger the batch size (128) and the longer the training (20–30 epochs), the better the performance. The initial findings were confirmed when the experiments were repeated on another subset, confirming the robustness of the experimental protocol. For acknowledgement, I have used LLM (ChatGpt) in case of some codes to train parameters which I found difficult to get from Keras.

### Experiment Results Summary Table

| Task Number | Experiment/Model Type | Description | Final Test Accuracy (%) |
|---|---|---|---|
| TASK-3/4 | Basic CNN Model (20 Epochs, Batch Size-64) | Initial CNN architecture with 2 blocks (32,64) trained for 30 epochs. | 65.20% |
| TASK-5.1 | Deeper Model (Increasing Size and Depth) (15 Epochs, Batch Size-64) | Model with additional blocks (64, 128), Batch Normaliza-tion, and more dense units. | 66.60% |
| TASK-5.1 | Deeper Model (Increasing Size and Depth) (25 Epochs, Batch Size-64) | Model with additional blocks (64, 128), Batch Normaliza-tion, and more dense units. | 66.00% |
| TASK-5.2 | Very Deep Model (More Conv/Pool Layers) (20 Epochs, Batch Size-64) | Model with additional Conv/Pool blocks (32,64,128). Trained for 20 epochs. | 68.80% |

| | | | |
|---|---|---|---|
| TASK-5.2 | Very Deep Model (More Conv/Pool Layers) (30 Epochs, Batch Size-64) | Model with additional Conv/Pool blocks (32,64,128). Trained for 30 epochs. | 69.40% |
| TASK-5.3 | Activation Experiment (Leaky ReLU) (20 Epochs, Initial Subset) | Very deep model structure with Leaky ReLU activation. Trained for 20 epochs on classes 0–9. | 67.20% |
| TASK-5.3 | Activation Experiment (Sigmoid) (Initial Subset) | Very deep model structure with Sigmoid activation. Trained for 20 epochs on classes 0–9. | 40.60% |
| TASK-5.3 | Activation Experiment (Tanh) (Initial Subset) | Very deep model structure with Tanh activation. Trained for 20 epochs on classes 0–9. | 68.20% |
| TASK-5.3 | Activation Experiment (Swish) (Initial Subset) | Very deep model structure with Swish activation. Trained for 20 epochs on classes 0–9. | 65.20% |
| TASK-5.5 | Optimizer Experiment (Adam) (Initial Subset) | Simple model trained with Adam optimizer (lr=0.001) for 20 epochs on classes 0–9. | 64.40% |
| TASK-5.5 | Optimizer Experiment (SGD) (Initial Subset) | Simple model trained with SGD optimizer (lr=0.01, mom=0.9) for 20 epochs on classes 0–9. | 59.40% |
| TASK-5.5 | Optimizer Experiment (RMSprop) (Initial Subset) | Simple model trained with RMSprop optimizer (lr=0.0001) for 20 epochs on classes 0–9. | 47.80% |
| TASK-5.6 | Batch Size/Epochs (Batch 32, Epochs 10) (Initial Subset) | Deeper model with 3 Conv/Pool blocks. Trained with batch size 32 for 10 epochs on classes 0–9. | 64.60% |
| TASK-5.6 | Batch Size/Epochs (Batch 64, Epochs 15) (Initial Subset) | Deeper model with 3 Conv/Pool blocks. Trained with batch size 64 for 15 epochs on classes 0–9. | 68.60% |
| TASK-5.6 | Batch Size/Epochs (Batch 128, Epochs 20) (Initial Subset) | Deeper model with 3 Conv/Pool blocks. Trained with batch size 128 for 20 epochs on classes 0–9. | 66.00% |
| **Repeating Task** | | | |
| TASK-6 | Base Model (New Subset 20–29) (25 Epochs, Batch Size-64) | The original basic CNN model trained on the new subset of classes (20–29) for 25 epochs. | 67.40% |
| TASK-6/5.1 | Deeper Model (Increasing Size/Depth) (New Subset 20–29) | The improved deeper model architecture trained on the new subset of classes (20–29) for 15 epochs. | 71.40% |
| TASK-6/5.2 | Very Deep Model (More Conv/Pool) (New Subset 20–29) | The model with more Conv/Pool layers trained on the new subset of classes (20–29) for 15 epochs. | 70.20% |
| TASK-6/5.3 | Activation Experiment (ReLU) (New Subset 20–29) | Very deep model structure with ReLU activation. Trained for 20 epochs on classes 20–29. | 63.80% |

| | | | |
|---|---|---|---|
| TASK-6/5.3 | Activation Experiment (Sigmoid) (New Subset 20–29) | Very deep model structure with Sigmoid activation. Trained for 15 epochs on classes 20–29. | 44% |
| TASK-6/5.3 | Activation Experiment (Swish) (New Subset 20–29) | Very deep model structure with Swish activation. Trained for 15 epochs on classes 20–29. | 63.80% |
| TASK-6/5.3 | Activation Experiment (Tanh) (New Subset 20–29) | Very deep model structure with Tanh activation. Trained for 15 epochs on classes 20–29. | 67.00% |
| TASK-6/5.3 | Activation Experiment (Leaky ReLU) (New Subset 20–29) | Very deep model structure with Leaky ReLU activation. Trained for 15 epochs on classes 20–29. | 67.20% |
| TASK-6/5.5 | Optimizer Experiment (Adam) (Initial Subset) | Simple model trained with Adam optimizer (lr=0.001) for 20 epochs on classes 0–9. | 66.60% |
| TASK-6/5.5 | Optimizer Experiment (SGD) (Initial Subset) | Simple model trained with SGD optimizer (lr=0.01, mom=0.9) for 20 epochs on classes 0–9. | 61.40% |
| TASK-6/5.5 | Optimizer Experiment (RMSprop) (Initial Subset) | Simple model trained with RMSprop optimizer (lr=0.0001) for 20 epochs on classes 0–9. | 50.60% |
| TASK-6/5.6 | Batch Size/Epochs (Batch 32, Epochs 10) (Initial Subset) | Deeper model with 3 Conv/Pool blocks. Trained with batch size 32 for 10 epochs on classes 0–9. | 66.80% |
| TASK-6/5.6 | Batch Size/Epochs (Batch 64, Epochs 15) (Initial Subset) | Deeper model with 3 Conv/Pool blocks. Trained with batch size 64 for 15 epochs on classes 0–9. | 68.20% |
| TASK-6/5.6 | Batch Size/Epochs (Batch 128, Epochs 20) (Initial Subset) | Deeper model with 3 Conv/Pool blocks. Trained with batch size 128 for 20 epochs on classes 0–9. | 68.60% |