Assignment - 05

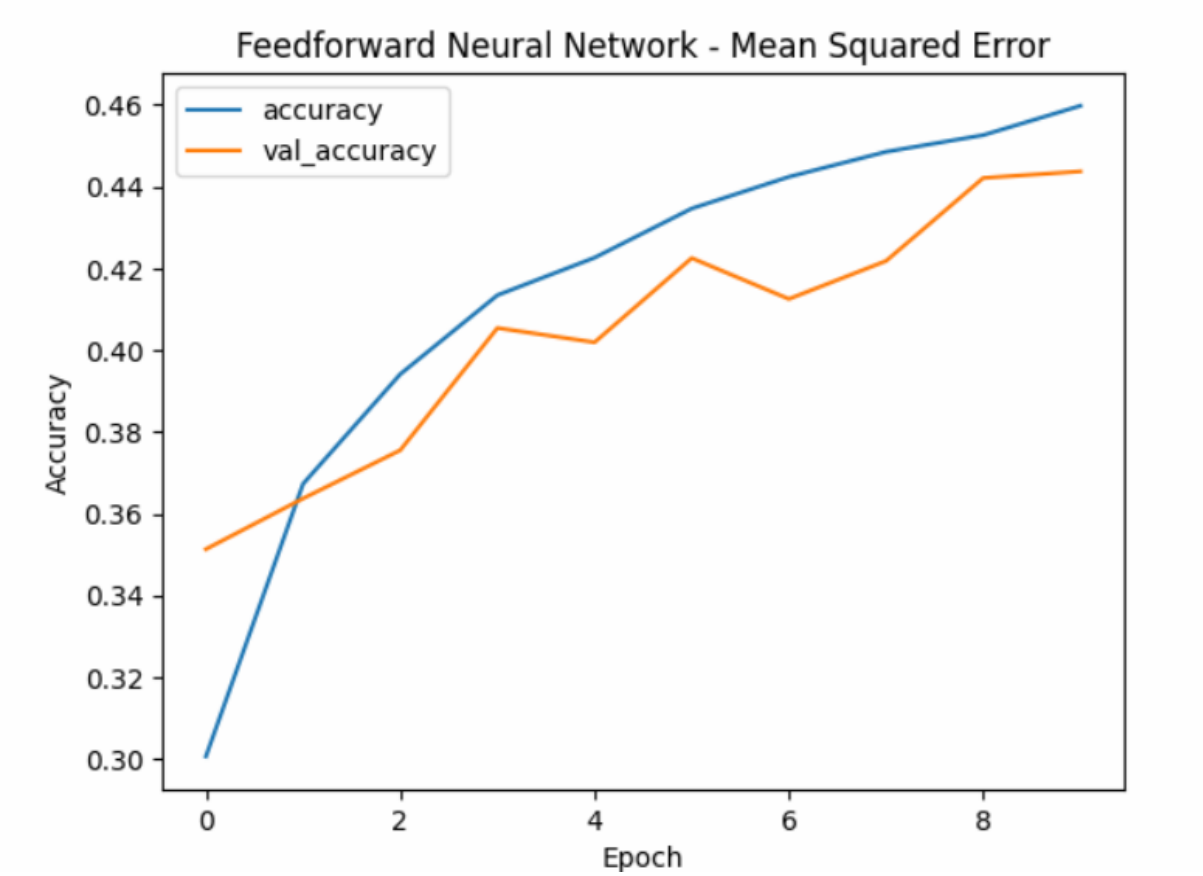
By Nishant Sharma (B22CH017)

Question 1: Steps Taken

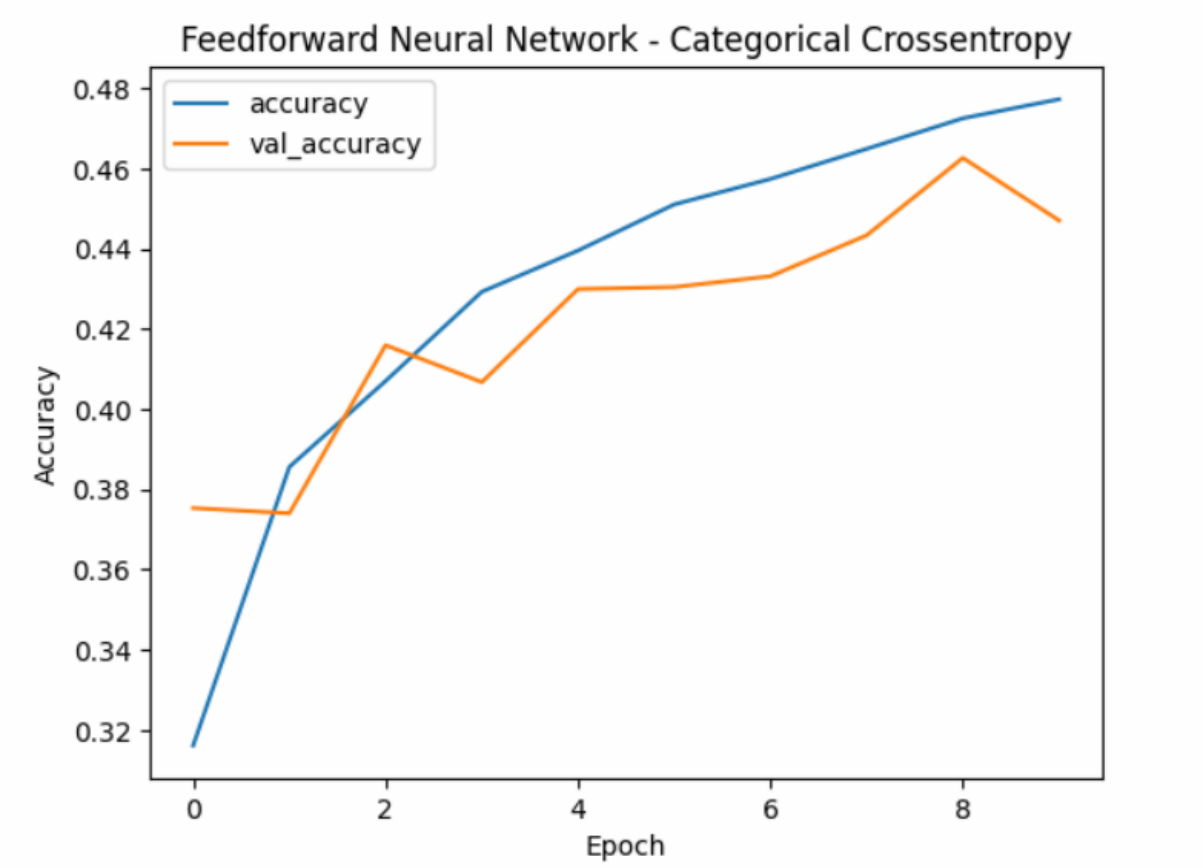
1. Import Libraries: Key libraries for constructing and training neural networks with TensorFlow and Keras were imported.

2. Load CIFAR-10 Dataset: The CIFAR-10 dataset was loaded using TensorFlow's built-in dataset loader.

3. Data Normalization: Pixel values of images were normalized to a range of 0 to 1 for numerical stability during training.



4. Define and Train a Simple Feedforward Neural Network: A basic feedforward neural network was established and trained on the CIFAR-10 dataset, setting the stage for further exploration.



1.1 Neural Network Variations

Simple Feedforward Neural Networks:

Implemented a basic feedforward neural network using Keras' Sequential model.

Introduced a flattening layer, a dense hidden layer with 128 neurons (ReLU activation), and a dense output layer with 10 neurons (softmax activation).

Compiled the model using the Adam optimizer, sparse categorical cross-entropy loss, and accuracy as the metric.

Trained the model for 10 epochs with validation data for performance monitoring.

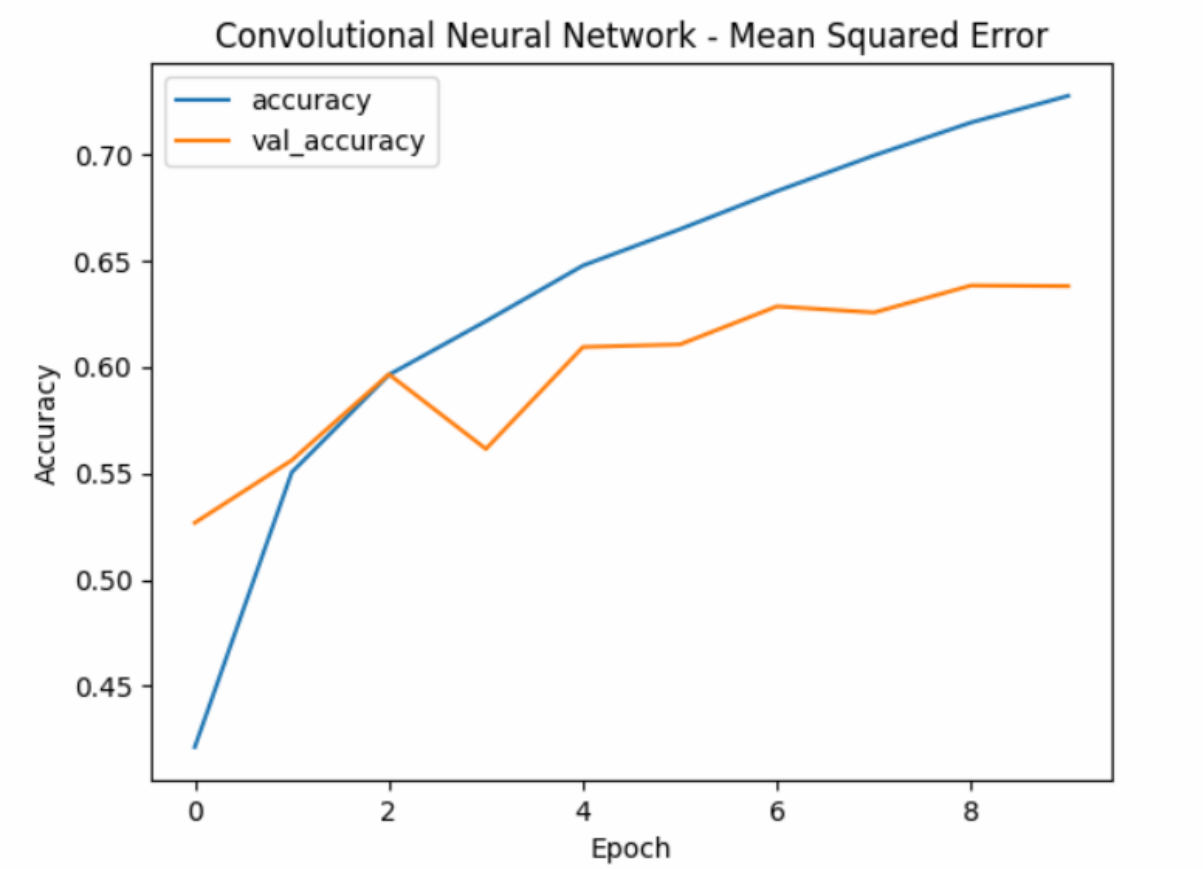
Convolutional Neural Network Definition:

Utilized Keras' Sequential model for a CNN with a convolutional layer (Conv2D), max-pooling layer (MaxPooling2D), and dense layers.

Applied ReLU activation for spatial pattern capture.

Configured the model for image classification with 10 output neurons.

Deep Neural Network Definition:



Constructed a deep neural network with three dense hidden layers (256, 128, 64 neurons) and a dense output layer (10 neurons).

1.2 Loss Functions and Training Protocols

Two Distinct Loss Functions Employed:

Neural Network with Categorical Cross-Entropy Loss.

Neural Network with Mean Squared Error.

Experimentation with Training Protocols:

Created a model with the Adam optimizer, specifying a learning rate of 0.001.

Explored sparse categorical cross-entropy as the loss function for integer-encoded class labels.

1.3 Model Evaluation

Formulated and trained a neural network model named model_sgd using the Stochastic Gradient Descent (SGD) optimizer.

Configured the model with a learning rate of 0.01, sparse categorical cross-entropy loss, and accuracy as the monitoring metric.

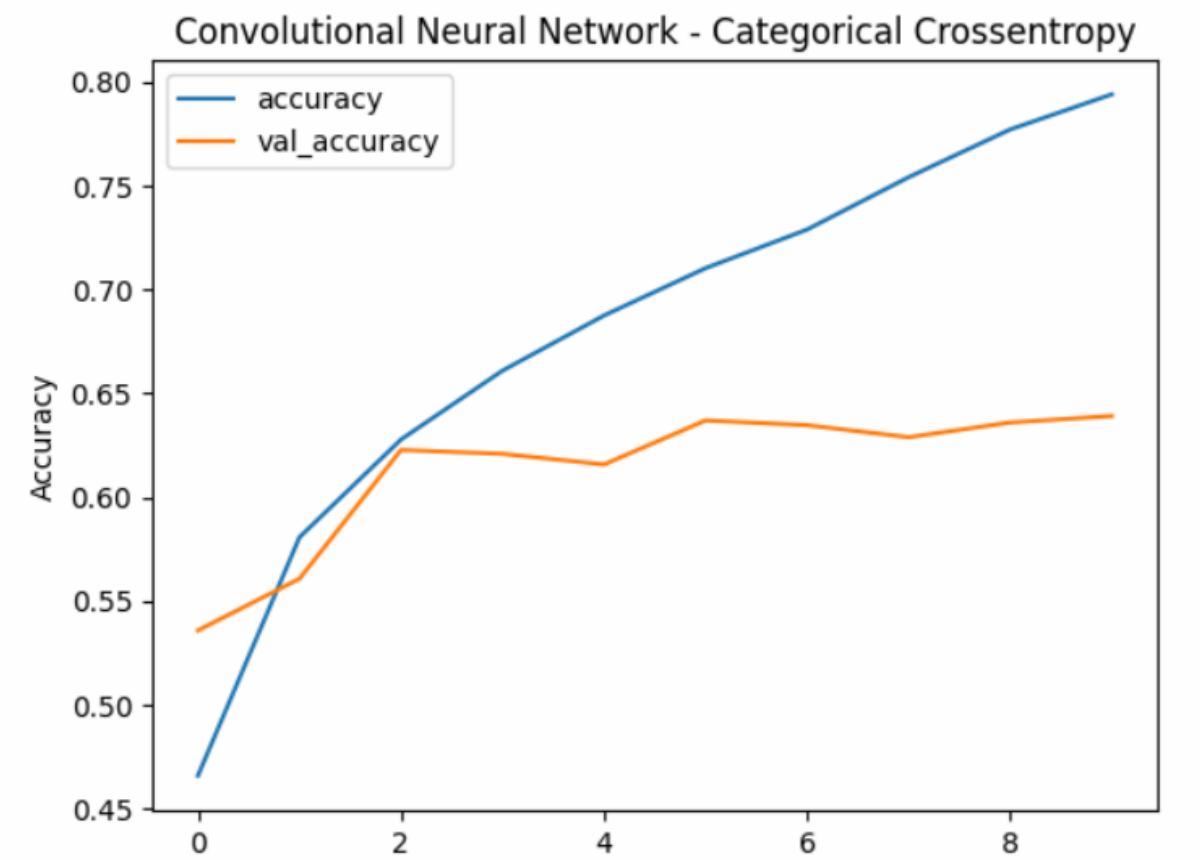
Trained the model for 10 epochs, evaluating performance on the validation set after each epoch.

Question 2: CNN Architectures

2.1 Implementation and Exploration

Objective: Evaluate three distinct Convolutional Neural Network (CNN) architectures for image classification.

Variations Explored:



Filter Size: Investigated different sizes of filters in convolutional layers.

Number of Filters: Explored the effects of varying filter quantities.

Pooling and Fully Connected Layers Structure: Investigated configurations of pooling layers and the arrangement of fully connected layers.

Objective: Gain insights into the impact of modifications in CNN architecture on image classification performance.

2.2 Hyperparameter Tuning

Objective: Conduct hyperparameter tuning for CNN models.

Hyperparameters Explored:

Learning Rates: Explored different learning rates.

Batch Sizes: Varied batch sizes during training.

Dropout Rates: Investigated the influence of dropout regularization.

Evaluation: Assessed how hyperparameter variations influenced model performance.

2.3 Data Augmentation

Objective: Implement data augmentation techniques.

Techniques Applied:

Rotation, scaling, and horizontal flips to augment training data.

Assessment: Evaluated the impact of data augmentation on model performance, focusing on generalization and handling dataset variations.

Analysis and Discussion

Neural Network vs. CNN

Comparison:

Traditional Neural Networks (NNs) demonstrated simplicity and faster training, while CNNs excelled in image classification.

Advantages of CNNs included spatial hierarchy capture, translation invariance, and feature learning.

Limitations of CNNs included computational complexity, especially in deep architectures, and potential overfitting concerns.

Hyperparameter Impact

Variations Explored:

Number of neurons and layers, filter sizes, and learning rates.

Insights:

Increased neurons and layers improved model capacity but raised overfitting risks.

Filter size affected spatial feature capture.

Optimal learning rates were crucial for convergence speed and overall performance.

Data Augmentation Benefits

Impact of Data Augmentation:

Enhanced Generalization:

Augmentation increased dataset diversity, reducing overfitting risks.

Improved extrapolation of learned features to new, unseen data.

Robustness to Variations:

Transformations enhanced robustness against object variations in real-world scenarios.

Better performance on objects at different orientations or scales.

Decreased Sensitivity to Noise:

Reduced sensitivity to minor dataset variations, enhancing reliability.

Conclusion

Traditional NNs offer simplicity and quick training, while CNNs excel in image-related tasks.

Hyperparameter tuning requires a delicate balance between model complexity and efficiency.

Data augmentation proves valuable for enhancing generalization and robustness in CNNs, especially in scenarios with limited datasets.

In-depth analysis and experimentation have provided valuable insights into the intricacies of neural network architectures, hyperparameter choices, and the advantages of data augmentation in the context of image classification.