**Module: (ECS659P/ECS7026P) Neural Networks and Deep Learning**

**Student name: Sahib Bhatti**

**Student ID: 190319889**

**Introduction:**

This report focuses on the steps in designing a Convolutional Neural Network with the purpose of classifying 60,000 32x32 photos from the CIFAR-10 dataset. My approach involved constructing a basic architecture and from there improving the accuracy through training, implementing methods and tuning parameters and then assessing the accuracy through visualisations.

**Architecture overview and improvements from the basic architecture:**

My neural network architecture implements various advancements in comparison to the basic architecture model. In the basic neural network architecture, two blocks and two convolutional layers are used without the implementation of regularisation or normalisation techniques. Thus, my improved neural network architecture comprises of 3 blocks with 4 convolutional layers in each block. My model class specifies my improved neural network in the following way. Block 1 takes an input with 3 channels (RGB image) and then with 4 convolutional layers outputs it to 64 channels. Block 2 receives the 64 channels as an input from block 1 and with 4 convolutional layers processes it and produces 128 channels as the output. Finally, Block 3 takes the 128 channels as input and processes it into 256 channels also using 4 convolutional layers. This was a worthy improvement since adding an additional intermediate block in the architecture and convolutional layers helped alter channel depths between layers enabling greater flexibility for tailored and effective feature extraction of the CIFAR-10. Thus, more accurate classification.

Following each convolutional layer, I firstly implemented **batch normalisation** to normalise the inputs of each layer maintaining the outputs of each layer in a standard range. Secondly, I implemented the **ReLu** activation function to introduce non-linearity into the model. This feature is crucial because it allows the network to learn complex patterns and makes learned features more meaningful.

Additionally, after the fully connected layers, I implemented the **dropout** layer tuned to 0.5 for regularisation to prevent overfitting and increasing robustness. Then below this I implemented the **adaptive average max pool 2d** which helped standardise the output.

Within the sequential intermediate block class, **sigmoid** function was used as a re-weighting feature. This function computes the weights from the fully connected layer outputs. Then after they are summed.

My output block captures **global average pooling** concentrates each feature map to a single value, reducing model complexity and preventing overfitting by streamlining the transition to fully connected layers.

**Hyperparameters and Training Techniques:**

**Batch Size:** Configured at 64 for general tasks and increased to 100 for training, optimizing the balance between processing speed and the detail of model updates.

**Optimizer:** Adam optimizer with the **hyperparameter ‘learning rate’** of 0.01, facilitating rapid convergence. The learning rate determines the size of each step taken towards minimizing the loss function during each iteration.

**Loss Function**: **Cross-entropy loss** used for multi-class classification tasks evaluating the classification model output of a probability value between 0 and 1.

**Epochs:** The model was trained for 50 epochs, providing sufficient iterations for the network to learn from the entire dataset effectively. The more epochs that passed, the more accurate the model became and the lower the loss.

**Regularization:** **Dropout** with a rate of 50% in the sequential block to prevent overfitting by reducing variations among neurons.

**Average Max Pooling** used to emphasize prominent features by reducing spatial dimensions, enhancing model performance.

**Results and Visualisations:**

The highest accuracy obtained was 0.8539 (85.39%) at the 44th Epoch. The accuracy passed the 85% mark at 40th epoch at 85.09%. I received the 85% mark a few times. When viewing the code, do not truncate the epoch output and list all to view the 85% accuracy.

A screenshot of a computer code

Description automatically generated

A screenshot of a computer code

Description automatically generated

**Figure 1: Highest accuracy achieved**

**Figure 2: All 50 epochs**

**Plots for:**

1. **The loss in training at each epoch:** The loss steeply decreases and begins to gradually settle as the epochs pass.
2. **A graph of loss and accuracy

   Description automatically generatedThe** **training and testing accuracy at each epoch:** Both training and testing accuracies increase as epochs pass. The training accuracy becomes slightly higher as epochs progress. However, the testing accuracy proves that the majority of images are classified correctly.