# Experimental Results

Faraz Razi - i201866

March 16, 2024

## 1 Introduction

In this report, we present the experimental results of applying various deep learning models to the CIFAR-10 dataset, a widely used benchmark dataset in the field of computer vision. The objective of this assignment is to explore and compare the performance of models such as Convolutional Neural Networks (CNNs), Artificial Neural Networks (ANNs), PixelRNN, PixelCNN, and their variants in tasks such as image classification, image generation, and image similarity calculation.

## 2 Dataset

The CIFAR-10 dataset consists of 60,000 images divided into 10 classes. Before training the LSTM model, several preprocessing steps are performed to prepare the data for input:

1. **Data Loading**: The CIFAR-10 dataset is loaded using Keras, and it is split into training, validation, and test sets. If a subset of the data is required, the dataset is subsetted accordingly.
2. **Validation Split**: A portion of the training data (20% by default) is reserved for validation to monitor the model's performance during training.
3. **Dataset Preview** 1

## 3 Questions

### 3.1 Question: 1. CNN

The CNN (Convolutional Neural Network) model is a widely used architecture for image classification tasks. This model consists of several layers, including convolutional and pooling layers followed by fully connected layers. Below is the description of the CNN model:

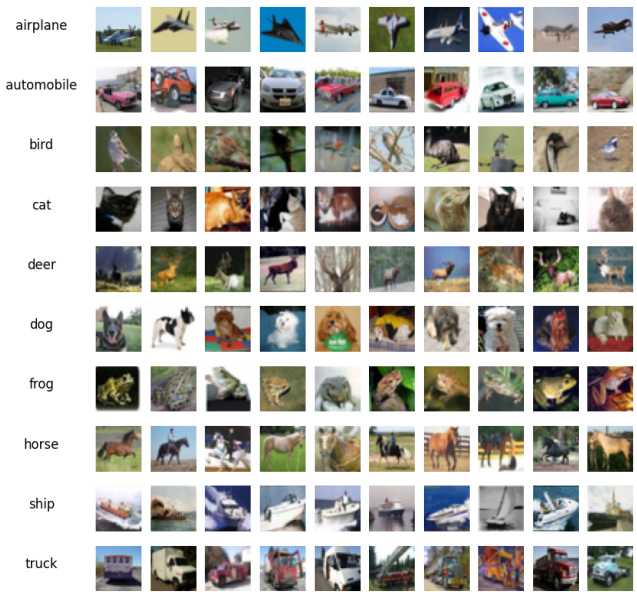


Figure 1: Dataset Preview

- **Input Layer**: The input layer accepts images of size $32 \times 32 \times 3$, where 3 represents the number of color channels (RGB).
- **Convolutional Layers**: The CNN model includes three convolutional layers with 32, 64, and 128 filters of size $3 \times 3$, respectively. Each convolutional layer is followed by a ReLU activation function, which introduces non-linearity into the network.
- **MaxPooling Layers**: After each convolutional layer, a max-pooling layer with a pool size of $2 \times 2$ is applied to reduce the spatial dimensions of the feature maps.
- **Flatten Layer**: The output of the last max-pooling layer is flattened into a one-dimensional vector to be fed into the fully connected layers.
- **Fully Connected Layers**: The flattened output is passed through two fully connected (dense) layers with 128 and 64 units, respectively. Each dense layer is followed by a ReLU activation function.
- **Output Layer**: The output layer consists of 10 units corresponding to the number of classes in the CIFAR-10 dataset. It applies a softmax activation function to output the class probabilities.

The CNN model is compiled with the Adam optimizer using a learning rate of 0.0001 and Sparse Categorical Crossentropy loss function, which is suitable

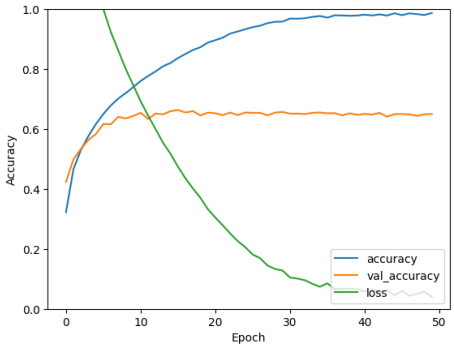


Figure 2: Training History for CNN

for multi-class classification tasks. The accuracy metric is used to monitor the performance of the model during training and evaluation.

**Training and Evaluations**

The Training results for CNN are shown in figure 2

Table 1: Classification Report for CNN

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Airplane | 0.63 | 0.76 | 0.69 | 1000 |
| Automobile | 0.79 | 0.75 | 0.77 | 1000 |
| Bird | 0.60 | 0.49 | 0.54 | 1000 |
| Cat | 0.46 | 0.47 | 0.47 | 1000 |
| Deer | 0.61 | 0.57 | 0.59 | 1000 |
| Dog | 0.52 | 0.58 | 0.55 | 1000 |
| Frog | 0.74 | 0.75 | 0.74 | 1000 |
| Horse | 0.72 | 0.67 | 0.69 | 1000 |
| Ship | 0.75 | 0.77 | 0.76 | 1000 |
| Truck | 0.74 | 0.72 | 0.73 | 1000 |
| Accuracy | | | 0.65 | |
| Macro Avg | 0.66 | 0.65 | 0.65 | 10000 |
| Weighted Avg | 0.66 | 0.65 | 0.65 | 10000 |

The Confusion matrix at 3 and ROC at 4

## 3.2 Question: 2. PixelRNN and PixelCNN

### 3.2.1 Data Preprocessing

1. **Reshaping**: Before feeding the data into the LSTM model, the channel dimension of each image is moved to the second position to match the expected input shape of the LSTM model. Then, the data is reshaped to

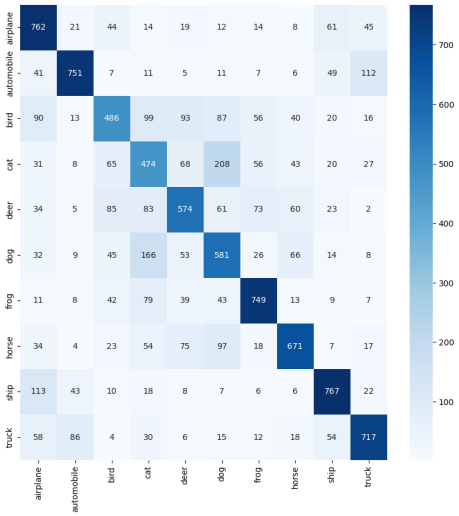


Figure 3: Enter Caption

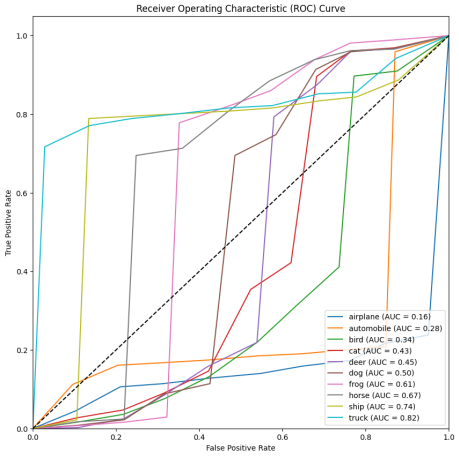


Figure 4: Enter Caption

have a shape of $(\text{num\_samples}, 32, 32 * 3)$, where each sample represents an image with dimensions $32 \times 32$ pixels and 3 color channels (RGB).

2. **Data Type Conversion**: Finally, the data is converted to `float32` to ensure consistency in data types.

These preprocessing steps ensure that the data is properly formatted and ready to be fed into the LSTM model for training.

### 3.2.2 Models

**RowLSTM Model**

The RowLSTM model is designed to process sequential data, making it suitable for tasks such as image classification. This model architecture consists of two LSTM layers followed by a Flatten layer and a Dense output layer with softmax activation. The key features of the RowLSTM model are as follows:

- **LSTM Layers**: The RowLSTM model incorporates two LSTM layers with 128 and 256 units, respectively. These layers enable the model to capture temporal dependencies within the input data efficiently.
- **Flatten Layer**: Following the LSTM layers, a Flatten layer reshapes the output tensor into a one-dimensional vector, preparing it for the final classification layer.
- **Dense Output Layer**: The model's output layer consists of a Dense layer with softmax activation, which predicts the probabilities for each class in the CIFAR-10 dataset.

**Training Process**

The RowLSTM model is trained using the Adam optimizer with default parameters and sparse categorical cross-entropy loss. The training process involves iterating over the training dataset, adjusting the model's weights based on the gradients computed during backpropagation, and monitoring performance on the validation dataset to prevent overfitting.

**Evaluation Metrics**

After training, the RowLSTM model is evaluated on a separate test set using standard evaluation metrics, including accuracy. Additionally, precision, recall, and F1-score are computed to provide insights into the model's performance across different classes. These metrics help assess the model's effectiveness in accurately classifying images from the CIFAR-10 dataset.

The results shows the accuracy of model capped at arround 50% during training. This pattern was observed till epoch 50, Here only 20 epochs are shown. 5

- **Accuracy**: The BiLSTM model achieved an accuracy of 0.0979 on the test set.
- **Loss**: The loss calculated during evaluation was 2.3810.

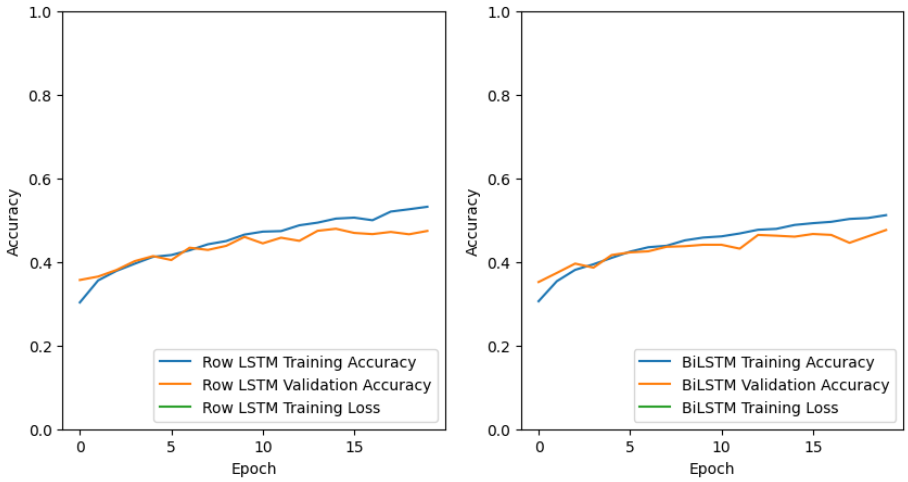


Figure 5: Training History for Row and Bi LSTM

**BiLSTM Model**

The BiLSTM model is a variant of the Long Short-Term Memory (LSTM) model that incorporates bidirectional processing to capture temporal dependencies in both forward and backward directions. This model architecture consists of two Bidirectional LSTM layers followed by a Flatten layer and a Dense output layer with softmax activation. Here's a brief overview of the BiLSTM model:

- **Bidirectional LSTM Layers**: The BiLSTM model utilizes two Bidirectional LSTM layers with 128 and 256 units, respectively. These layers enable the model to capture complex temporal patterns in sequential data, such as images from the CIFAR-10 dataset.
- **Flatten Layer**: Following the Bidirectional LSTM layers, a Flatten layer reshapes the output tensor into a one-dimensional vector, preparing it for the final classification layer.
- **Dense Output Layer**: The model's output layer consists of a Dense layer with softmax activation, which predicts the probabilities for each class in the CIFAR-10 dataset.

**Evaluation Results**

The training results shows the accuracy of model capped at arround 50% during training similarl to RowLSTM. This pattern was observed till epoch 50, Here only 20 epochs are shown. 5

After training, the BiLSTM model was evaluated on the test set of CIFAR-10 dataset. The evaluation results are as follows:

- **Accuracy**: The BiLSTM model achieved an accuracy of 0.0979 on the test set.

- **Loss**: The loss calculated during evaluation was 2.3810.

These evaluation metrics provide insights into the performance of the BiLSTM model in classifying images from the CIFAR-10 dataset. Further analysis and comparison with other models can help determine the effectiveness of the BiLSTM architecture for this task.

**RowLSTMGen Model**

The RowLSTMGen model is designed for image generation using Long Short-Term Memory (LSTM) networks. It takes an input of shape (32, 32, 3) representing CIFAR-10 images and generates images of the same shape. The key components of the RowLSTMGen model include LSTM layers for capturing temporal dependencies and Dense layers for feature extraction and output generation. Here's a summary of the RowLSTMGen model:

- **Input Layer**: The model accepts input of shape (32, 32, 3) representing CIFAR-10 images.
- **Reshape Layer**: The input is reshaped to (32, -1) to fit the LSTM input shape.
- **LSTM Layers**: Two LSTM layers with 128 and 256 units, respectively, capture temporal dependencies in the input data.
- **Flatten Layer**: The output of the LSTM layers is flattened to prepare it for dense layers.
- **Dense Layers**: Two dense layers with relu and None activation functions, respectively, extract features and generate the output.
- **Output Reshape Layer**: The output is reshaped back to the original image shape (32, 32, 3).

**BiLSTMGen Model**

The BiLSTMGen model is similar to the RowLSTMGen model but utilizes Bidirectional LSTM layers to capture temporal dependencies in both forward and backward directions. It also generates images of shape (32, 32, 3) based on CIFAR-10 images. Here's a summary of the BiLSTMGen model:

- **Input Layer**: Similar to RowLSTMGen, the model accepts input of shape (32, 32, 3).
- **Reshape Layer**: The input is reshaped to (32, -1) to fit the Bidirectional LSTM input shape.
- **Bidirectional LSTM Layers**: Two Bidirectional LSTM layers with 128 and 256 units, respectively, capture temporal dependencies in both directions.
- **Flatten Layer**: The output of the Bidirectional LSTM layers is flattened.


Figure 6: Enter Caption

- **Dense Layers**: Two dense layers with relu and None activation functions, respectively, extract features and generate the output.
- **Output Reshape Layer**: The output is reshaped back to the original image shape (32, 32, 3).

**Generation Results**

The Figure 6 is the output generated by RowLSTMGen model. The results are not very promissing but a good start for next models.

## 3.3 Question: 3. PixelCNN Model

The PixelCNN model is a generative model designed for image generation tasks, particularly suited for generating images with complex structures. The model architecture consists of several layers, including PixelConvLayer and ResidualBlock, which are stacked to capture dependencies between pixel values. Referenced form: [Oor+16] Below is the description of the PixelCNN model:

- **PixelConvLayer**: The PixelConvLayer is the first layer in the PixelCNN model. It applies a masked convolution operation to the input image, ensuring that each pixel can only access information from its previous rows and columns. This layer helps in capturing the spatial dependencies within the image.
- **ResidualBlock**: The ResidualBlock is a building block of the PixelCNN model. It consists of multiple convolutional layers followed by skip connections, similar to the ResNet architecture. These residual blocks enable the model to learn complex patterns while mitigating the vanishing gradient problem.
- **Convolutional Output Layer**: The final convolutional layer outputs the generated image. It consists of a single convolutional layer with a kernel size of 1, which reduces the channel dimension to match the desired output shape.

### 3.3.1 PixelCNN Model Variants

We experimented with three different variants of the PixelCNN model, varying the number of residual blocks and pixel convolution layers to observe their impact on the model's performance.

**First Variant** The first variant of the PixelCNN model is configured with 2 residual blocks and 1 pixel convolution layer. This variant aims to explore the performance of the model with a relatively simpler architecture.

**Second Variant** The second variant of the PixelCNN model is configured with 3 residual blocks and 2 pixel convolution layers. This variant introduces additional complexity to the model architecture, allowing for more sophisticated feature learning.

**Third Variant** The third variant of the PixelCNN model is configured with 5 residual blocks and 3 pixel convolution layers. This variant represents a deeper and more intricate model architecture, capable of capturing intricate dependencies within the image data.

These three variants of the PixelCNN model are compiled with the Adam optimizer and trained using binary cross-entropy loss. The model summaries provide insights into the architecture and parameter configurations of each variant.

### 3.3.2 Results

The Size of the model and complexity greatly increases the image generation results as show in figure 7 and 8.


Figure 7: Sample Generation 1


Figure 8: Sample Generation 2

# References

[Oor+16] Aaron van den Oord et al. *Conditional Image Generation with PixelCNN Decoders*. 2016. arXiv: 1606.05328 [cs.CV].