This project aims to accurately classify images into different categories using artificial neural networks (ANNs) and convolutional neural networks (CNNs). You can remove the gray sentences and fill this document using your information.

**Student Name:** David Haynes (z1756127)

**Data Source:** CIFAR-10 is a built-in image dataset available in Python. It consists of 50,000 32x32 color images in 10 different classes (e.g., airplanes, cars, cats, dogs, etc.). It is often used for image classification tasks. Import it into Google Colab by using the following code.

import tensorflow as tf

# Import CIFAR-10 dataset

cifar10 = tf.keras.datasets.cifar10

(train\_xs, train\_ys), (test\_xs, test\_ys) = cifar10.load\_data()

class\_names = ["Airplane","Automobile","Bird","Cat","Deer","Dog","Frog","Horse","Ship","Truck"]

# Printing the class names

for i, class\_name in enumerate(class\_names, start=0):

    print(f"Class {i}: {class\_name}")

To reduce the complexity of finding and importing appropriate image datasource, please use the above built-in dataset for this project.

**Python Code**:

https://colab.research.google.com/drive/1DW3DvUJ-UZKSsdGUVkvPVGBVo4YwxaOw?usp=sharing

* **Data Exploration**:

1. find the basic information of the data set, including the total number of images in the train and test datasets, the size of each image, and the number of categories (labels/classes).
2. Display sample images with corresponding labels: 25 images in the train set and 25 images in the test set.

Hint: to add labels, you need to convert train\_ys[i] to an integer for this dataset.

plt.xlabel(class\_names[int(train\_ys[i])])  # Convert train\_ys[i] to an integer

* **Data Preprocessing**: perform necessary preprocessing steps such as resizing the images to a consistent size, normalizing pixel values, etc.
* **ANN Model Architecture Design:**

1. Initiate model: design an ANN architecture suitable for image classification. Experiment with different architectures and hyperparameters to optimize performance, such as the number of layers, activation functions, optimizer, etc.
2. model training: train the ANN model on the training dataset.
3. model evaluation: evaluate the trained model using the testing dataset. Calculate loss, accuracy, and/or other metrics to measure the model's performance.
4. check overfitting issues using a loss or accuracy plot. Identify the optimal number of epochs.
5. apply the model to do prediction: predict the label/classification for at least one image.

* **CNN Model Architecture Design:**

1. Initiate model: design a CNN architecture suitable for image classification. Experiment with different architectures and hyperparameters to optimize performance, such as the filter size, activation functions, optimizer, etc.
2. model training: train the CNN model on the training dataset.
3. model evaluation: evaluate the trained model using the testing dataset. Calculate loss, accuracy, and/or other metrics to measure the model's performance.
4. check overfitting issues using a loss or accuracy plot. Identify the optimal number of epochs.
5. apply the model to do prediction: predict the label/classification for at least one image.

* **Compare the two models’ performance:** accuracy, computational efficiency, etc.
  + ANN and CNN were used in this project to help decipher which model is best to classify images. ANN is a general usage neural network that uses activation functions Sigmoid, ReLU , and Softmax to fit the data. CNN specializes in processing image data and does so through convolutional layers, pooling, and flattening to increase accuracy in processing grid like images. For these reasons it is expected that CNN has a higher accuracy rate than ANN and this holds true as seen in the attached python file. Once activation functions, learning rates (0.001), convolutional layers, and epochs (10 to begin) were chosen, ANN produced an accuracy rate of 0.47 and a loss of 1.49. CNN produced an accuracy rate of 0.71 and a loss of 0.87. These numbers display that CNN outperforms ANN in terms of image classification. Also, since CNNs use multiple layers, they are less computationally efficient than ANN’s. However, that is the tradeoff for being accurate.

**Summary**

This project classified images from the CIFAR-10 dataset in python into different categories using ANNs and CNNs. To better understand the data, there are 50,000 training images, 10,000 testing images, dimensions of 32x32x3, and 10 different classes (airplanes, cars, cats, dogs, etc.).

Many parameters were changed in effort to obtain the best model. The default learning rate for ‘adam’ of 0.001 was chosen, however, a smaller learning rate of 0.00000001 and a larger learning rate of 0.01 was tested for both ANN and CNN. With the changes, both models performed slightly worse than with a learning rate set at 0.001, but not but much. Each neural network accuracy changed by less than 2.0. CNN completely outperforms ANN in terms of loss (0.87 vs 1.49) and accuracy (0.71 vs 0.47).

The number of epochs makes a difference in the performance of each model and helps to determine if overfitting is occurring. In the ANN model, when epochs are changed to 30, the loss drops slightly to 1.48 and accuracy increases a bit to 0.49, however, shown in the graph overfitting occurs after roughly epoch 6. When the model is set to an epoch of 6, loss increases slightly to 1.52 and accuracy decreases slightly to 0.49, however overfitting does not occur. Six is the optimal epoch number for ANN. When changing the CNN number of epochs to 30, overfitting occurs, and validation accuracy tapers off at around 10 epochs. When set at 30 epochs, loss also increases and accuracy decreases meaning the CNN model is best fit at 10 epochs.

Different activation functions were used in the ANN model, but ReLU was chosen for hidden layers to ensure positive real numbers are being used since we are dealing with image classification. Also, Softmax is used for the output layer since multiple categories are involved. The number of neurons used in each activation function were changed to add layers with smaller neuron amounts of 128 or starting from a much higher number of neurons of over 1K. Two hidden layers with neurons of 512 and 256 were used. ReLU and Softmax were again used in CNN, however, CNN is more computationally complex, and the model was built on convolution layers.

The biggest takeaway is how different neural networks work better for different datatypes. We can conclude that for image classification, CNN is the right choice.