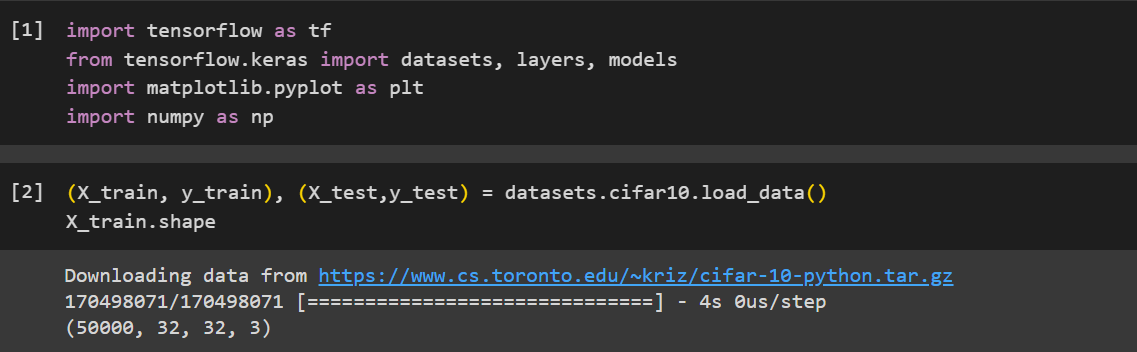
**Assignment: 2**

**Aim:** Image Classification**.**

Step 1: Load and Explore CIFAR-10 Dataset in Google Colab

* Import necessary libraries: TensorFlow, Keras, Matplotlib, and NumPy.
* Load the CIFAR-10 dataset using Keras's built-in function “datasets.cifar10.load\_data()”.
* Check the shape of the training dataset to ensure it contains 50,000 images with dimensions 32x32 pixels and 3 color channels.
* Load and explore the CIFAR-10 dataset using Google Colab, including loading the dataset from Google Drive and visualizing sample images for analysis.



Step 2: Explore Dataset Dimensions and Labels

* Confirm the testing dataset's dimensions: 10,000 images of size 32x32 pixels with 3 color channels.
* Acknowledge dataset sizes: 50,000 training images and 10,000 test images.
* Analyze the structure of training labels (y\_train): initially a 2D array (50000, 1), displaying the first five labels as integers (0 to 9).
* Optimize label representation for classification by converting y\_train to a 1D array.



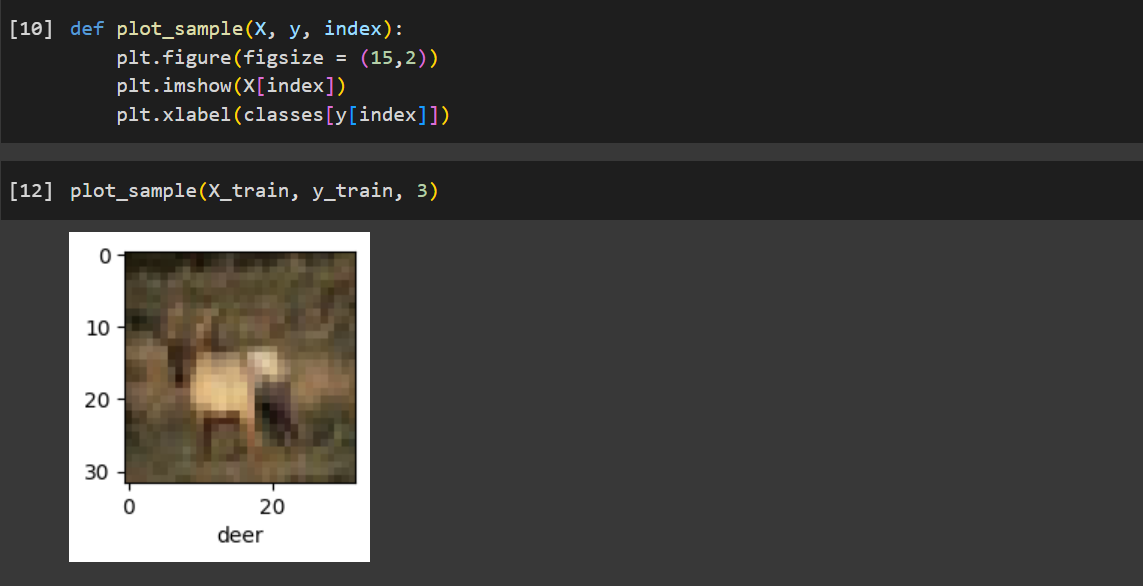
Step 3: Optimize Label Representation

* Reshape y\_train and y\_test arrays to a 1D format for improved classification ease.
* Display the first five labels from the reshaped y\_train array, now represented as integers (0 to 9).
* Define a list classes containing the class names corresponding to the label integers for better interpretation in classification results.



Step 4: Visualize Sample Images

* Define a function plot\_sample(X, y, index) to display a sample image along with its corresponding label.
* Set the figure size to 15x2 for better visualization.
* Plot the image at the specified index from the training dataset (X\_train).
* Label the image with the corresponding class name using the classes list and the label (y\_train[index]).





Step 5: Neural Network Training Overview

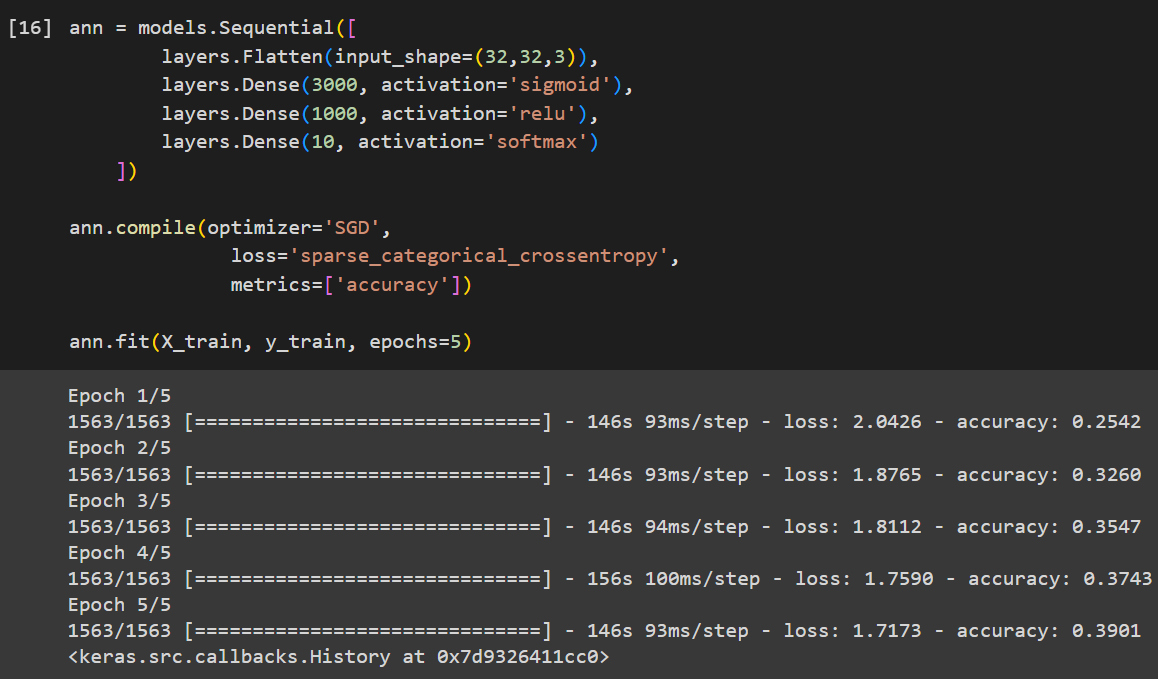
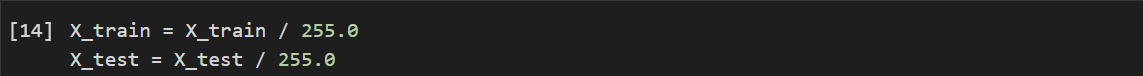
* Normalize pixel values of training and testing data to a range of 0 to 1.
* Define a sequential model with:

1) Flatten layer to convert 3D image data into a 1D array.

2) Dense layers with 3000 and 1000 neurons using sigmoid and ReLU activations respectively.

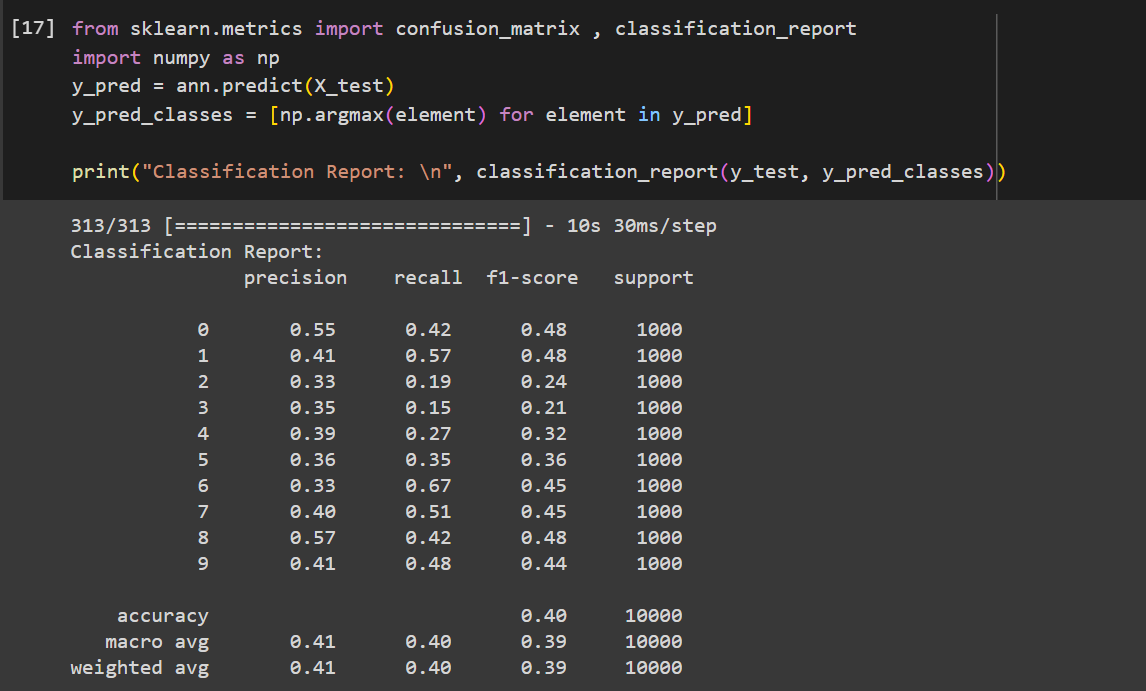
3) Output layer with softmax activation for classification.

* Compile the model using stochastic gradient descent (SGD) optimizer, sparse categorical crossentropy loss, and accuracy metric.
* Train the model on the training data for 5 epochs.
* Review training progress, reaching approximately 39% accuracy after 5 epochs.



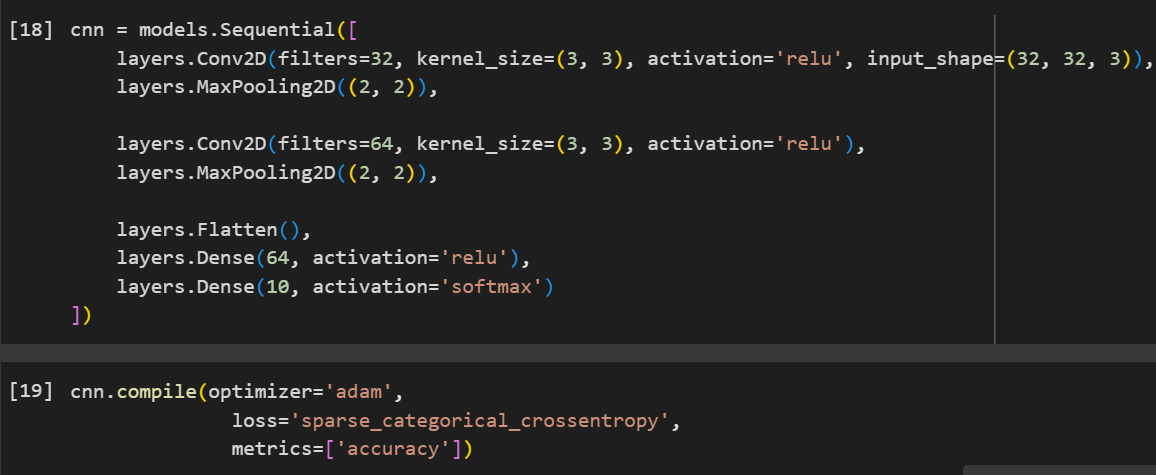
Step 6: Model Evaluation with Classification Metrics

* Import necessary libraries: confusion\_matrix, classification\_report from sklearn.metrics, and numpy.
* Predict classes for the testing dataset using the trained model.
* Calculate predicted classes by selecting the index of the maximum probability from each prediction.
* Generate and display a classification report showing precision, recall, and F1-score for each class, along with overall accuracy.
* The report indicates the precision, recall, and F1-score for each class, as well as macro and weighted averages.



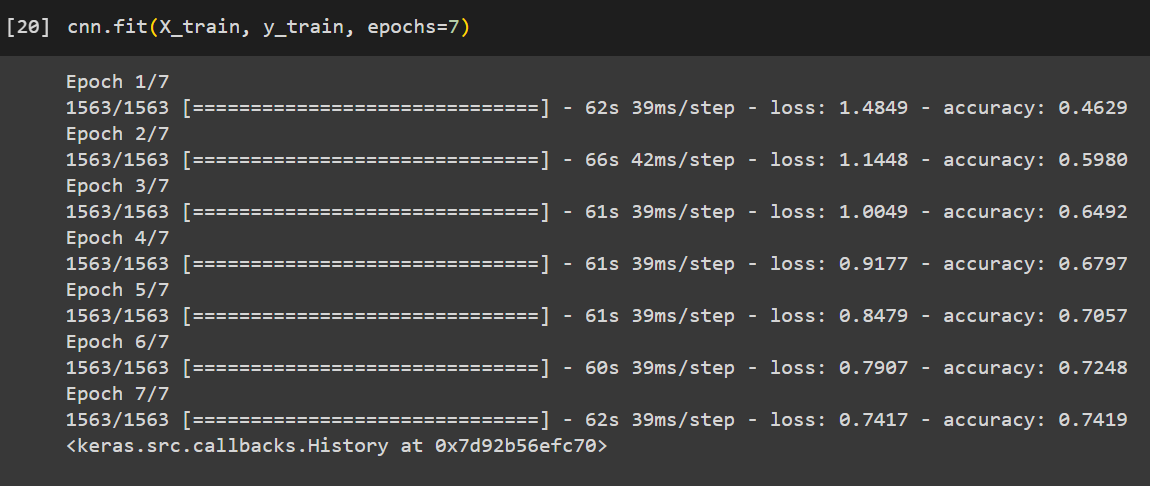
Step 7: Convolutional Neural Network (CNN) Model Construction

* Define a sequential model (CNN) for a Convolutional Neural Network (CNN) architecture.
* Utilize Conv2D layers with ReLU activation to extract features from input images.
* Apply MaxPooling2D layers to reduce spatial dimensions and capture dominant features.
* Flatten the output from convolutional layers into a 1D array.
* Introduce fully connected Dense layers with ReLU activation to perform classification.
* Compile the model using the Adam optimizer, sparse categorical crossentropy loss, and accuracy metric.



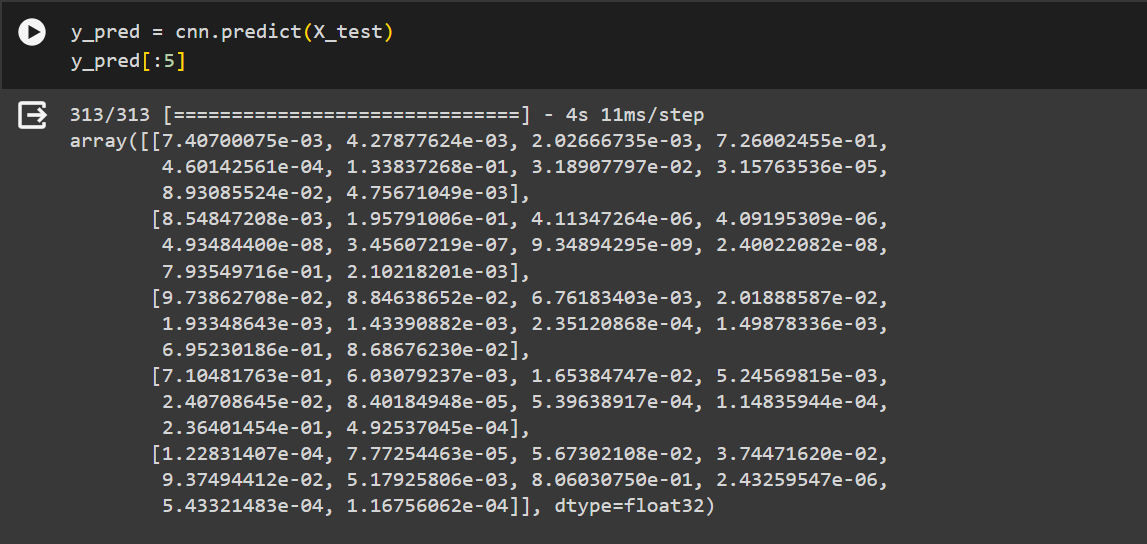
Step 8: CNN Model Training

* Train the CNN model (CNN) on the training data (X\_train, y\_train) for 7 epochs.
* Review the training progress, with each epoch displaying loss decreases and accuracy improves, indicating the model's learning process.



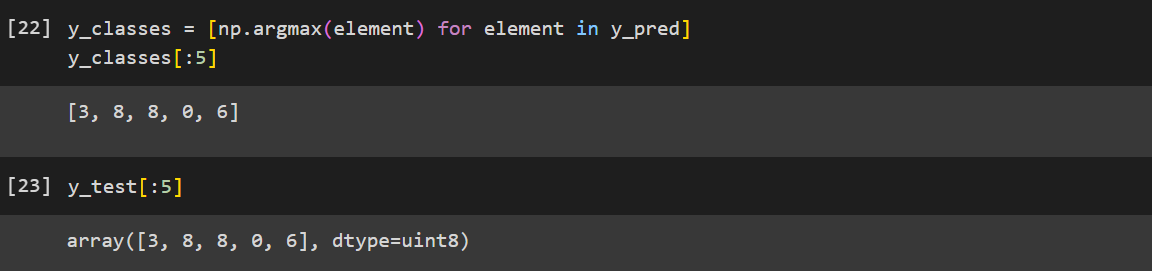
Step 9: Predicted Probabilities from CNN Model

* The output array displays the predicted probabilities for the first five test images across ten classes, generated by the trained Convolutional Neural Network (CNN) model.
* Each row represents a test image, and each column represents the probability of the image belonging to a specific class, ranging from 0 to 9.



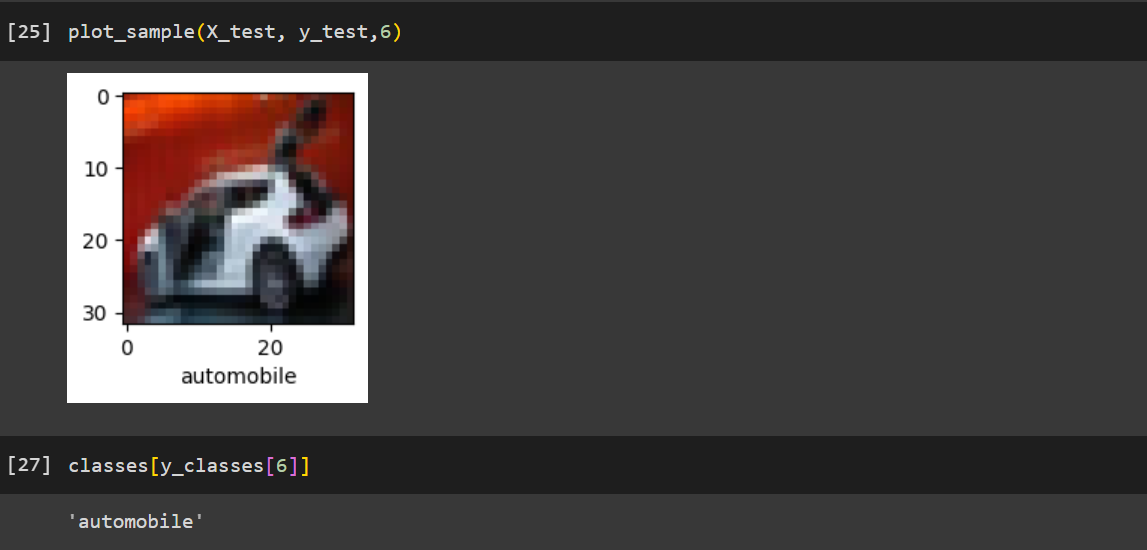
Step 10: Predicted and True Classes Comparison

* Compare the predicted classes (y\_classes) generated by the CNN model with the true classes (y\_test) from the test dataset for the first five images.
* The arrays y\_classes and y\_test show the predicted and true classes respectively, indicating successful predictions when they match.



Step 11: Visualization and Prediction Verification

* Visualize the 6th image from the test dataset (X\_test) along with its true label, showing an automobile.
* Verify the prediction result by checking the class name corresponding to the predicted class index, which matches the true label, confirming the prediction accuracy.



* Visualize the 13th image from the test dataset (X\_test) along with its true label, showing a Horse.

