Project

LINK:
https://colab.research.google.com/drive/1b4hdAYR0utvPmYXB1J2iSdPsXkWF5phu?usp=sharing

dataset CIFAR-10 with:

-60.000 img (32x32) → 50,000 train, 10,000 test

-segmented into 10 distinct classes (objects/animals)

-I use in this project just a part of the images for the test

Objective:

Image recognition with right classification

Technologies used:

Google Colab → development environment

OpenCV → for image processing

Numpy → for matrix calculations

Matplotlib → pyplot for plots

Tensorflow → for data augmentation

Data Pre-processing

Normalization of Image Data:

- convert the pixel values data type to **float32** type
- normalizes them with mean and standard deviation

One-Hot Encoding of Labels:

- transform the categorical labels into a format suitable for multi-class classification

Data Augmentation:

- Rotation
- Width and height shift
- Orizontal flip
- Zoom
- Brightness Range
- Shear Intensity
- Channel Shift Intensity

CNN model

- A pair of **Conv2D** layers (32 filters of size 3x3)
- **Batch Normalization** layer → regoularization
- **MaxPooling2D** layer → reduces computational complexity
- **Dropout** layer → random set a fraction of the input units to 0

➢This pattern repeats three more times

  ➢The number of filters in the Conv2D layers doubles with each repetition (32-64-128-256)

  ➢The Dropout rate increases at each step, from 0.2 to 0.5

- **Flatten layer** → to convert 2D outputs into a 1D vector
- **Dense** (fully-connected) layer → for classification

  ➢**Softmax** activation function → to convert the outputs to probability scores for each class

Train

- **ReduceLROnPlateau callback:** used to reduce the learning rate by half (factor=0.5) whenever the validation loss does not improve for 10 consecutive epochs
- **EarlyStopping callback:** to monitor the validation loss and halt the training process when there hasn't been any improvement for a certain number of epochs
- **Loss Function** : CrossEntropy
- **Optimizer**: Adam (0.0005) : to update the model's weights to minimize thet loss during training
- **Epochs** = 5
- **Batch size** = 16

} For test

Visualizing and Evaluating

Plots for visualizing train and val loss, and accuracy evolution over epochs

- The model achieved increasing accuracy, reaching 84.6% on training and 86.1% on validation after 5 epochs
- However, the validation loss remained high and unstable, indicating possible overfitting or optimization issues (I could try increasing data augmentation)

At the end:
Accuracy: 0.8590
Loss: 0.7332
Test Accuracy: 0.8561
Test Loss: 0.7393