Image Classification using CIFAR-10 and CIFAR-100 datasets

Version {FILL}

17th December 2023

Tanish Afre, Kevin {FILL}

D00242568@student.dkit.ie, {FILL}

Dundalk Institute Of Technology

Dundalk (A91 K584)

# Abstract:

This project presents a comprehensive exploration of image classification utilizing the CIFAR-10 and CIFAR-100 datasets. This assignment was part of the Smart Technology module where we were asked to develop a model capable of classifying images into 24 distinct categories containing different animals to vehicles. The model collects data from CIFAR-10 and CIFAR-100 respectively and presents an effective training model. During our research, we looked closely at the size and types of images in CIFAR datasets to understand it better. The main part of our project was building Convolutional Neural Network (CNN). The model was carefully examined and addressed underfitting and overfitting problems, ensuring a reliable code. The model results effectiveness in accurately classifying a diverse range of images, showcasing the potential of CNNs in complex image recognition tasks.

# Data Pre-Processing

## 1.1 Loading Combined Data

### 1.1.1 Image Loading and Extraction

Writing a simple function to load the images which is written in ‘data\_loader.py’. We imported ‘numPy’ which is a library used for performing numeric operations. We also imported ‘tensorflow.karas.datasets’ for loading CIFAR 10 and CIFAR 100 datasets. We extracted training data from both the datasets. We filtered the database as project requirements sheet.

CIFAR 10 : automobile, bird, cat, deer, dog, horse, and truck.

CIFAR 100 : cattle, fox, baby, boy, girl, man, woman, rabbit, squirrel, trees (superclass), bicycle, bus, motorcycle, pickup truck, train, lawn-mower and tractor.

### 1.1.2 Removing classes that are not used

We removed the unused classes while writing the code in filter\_and\_combine\_dataset. We then check if the label on the current image using ‘lbl[0]’ is specified in ‘cifar10\_classes.value()’. If it is, the image and its corresponding class name are added to separate lists ‘cifar10\_filtered\_images’ and ‘cifar10\_filtered\_labels’. Similar process is repeated for CIFAR 100.

### 1.1.3 Combining two datasets

Creating 'combined\_images' containing images from both datasets and ‘combined\_labels’ which contains labels from both datasets.

## 1.2 Preprocessing techniques

The images then undergo several processing steps such in ‘image\_processor.py’. We then wrote a function ‘process\_image’ where the image is converted into **grayscale** using openCV’s ‘cv2.cvtColor()’. This converts the image into black and white for simplifying the process. Then the images histogram equalizing which enhances the contrast of the image. The image is then reshaped such that it has 32x32 pixels with a single channel. The images are then undergo normalization to keep the image between the range 0 and 1. Later on Gaussian blur is applied to the images which helps in reducing noise in the existing image.

# 2. Data Exploration

{FILL}

# 3. Building the model

In ‘data\_models.py’ we created a model which was able to classify between 24 different classes with an accuracy of more than 98%. Firstly we did ‘label\_mapping’ which is very important to convert the class labels into numerical format for training the model.

## 3.1 Model Architecture

Defining a CNN model using the TensorFlow Keras Sequential API. Model uses three convolutional layers with increasing filter sizes (32, 64, and 128) and ReLU activation functions. They are responcible for learning image features. After each convolutional layer, there is a max-pooling layer 'MaxPooling2D' with a (2, 2) pool size. Max-pooling reduces the spatial dimensions of the feature maps. A flatten layer is used to flatten the 2D feature maps into a 1D vector. Two densely connected layers (Dense) with 3000 and 1000 units, which are responsible for learning high-level representations. The output layer has 22 units (assuming 22 classes) with a softmax activation function which outputs the class probabilities.

The model then is compiled using Adam optimizer, which is a popular optimization algorithm for training neural networks. The loss function is set to 'sparse\_categorical\_crossentropy', which is appropriate for multi-class classification tasks where the target labels are integers.

The training process runs for the specified number of epochs (default is 15) and uses a batch size of 32 by default. Here the model learns to classify images into one of the 22 classes based on the provided training data.

## 3.2 Underfitting Model

For this model, we used flatten layer that keeps the input image in 32x32 size and flattens them into 1-dimension vectors. No max pooling layers are used for this model, there are only dense layers which have very less value such as 30 and 10. Due to this, the accuracy of this model is very low and hence will be considered as an under-fitting model.

## 3.3 Over Fitting Model

The model is very deep with multiple convolutional and dense layers, which provides it with a high capacity to learn complex patterns. The dropout layers are included to help mitigate overfitting, but they may not be sufficient given the model's complexity. The model is trained for a potentially large number of epochs (15 by default), which can allow it to excessively fine-tune its parameters to the training data, including noise and irrelevant details. Overfitting is more likely to occur when the model is too complex relative to the size of the training dataset, as the model can memorize the training examples rather than generalize from them.

## 3.4 Data Augmentation

We created a function names ‘augment\_image’ for increasing the diversity of the training data by applying various transformation to the original image also known as data augmentation. This helps in improving the models generalization and robustness. We could make the following changes to the images:

Randomly shifts the image horizontally by up to 10% of its width.

Randomly shifts the image vertically by up to 10% of its height.

Randomly zooms in or out on the image by up to 20%.

Randomly applies shear transformation, which distorts the image by up to 10 degrees.

Randomly rotates the image by up to 10 degrees.

We then apply this augmentation to each image which is completely random.

# 4. Testing Model

{FILL}

# 5. GitHub Activities

We frequently made commits and also wrote well defined commit messages for understanding what steps we went through along with how we dealt with problems.   
You could check the Git repository by [clicking Here!](https://github.com/TanishAfre/SmartTechCA1.git)