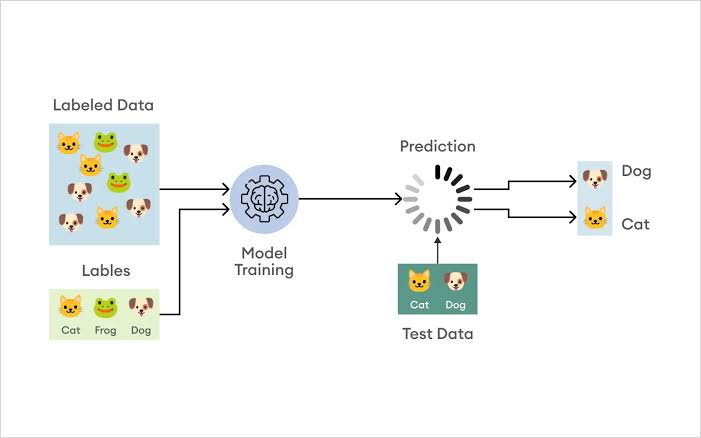
**Project Documentation: Image Classifier on CIFAR Data using a CNN Model**

**Index**

* [Introduction](#introduction)
  + How it’s implemented
* [Project Objectives](#project-objectives)
* [Dataset](#dataset) taken
* [Methodology](#methodology)
  + [Data Preprocessing](#data-preprocessing)
  + [Model Architecture](#model-architecture)
  + [Training the Model](#training-the-model)
  + [Evaluation](#evaluation)
* [Results](#results)
* [Conclusion](#conclusion)
* [References](#references)

**Introduction**

The **CIFAR-10 dataset** ([Canadian Institute For Advanced Research](https://en.wikipedia.org/wiki/Canadian_Institute_for_Advanced_Research)) is a collection of images that are commonly used to train [machine learning](https://en.wikipedia.org/wiki/Machine_learning) and [computer vision](https://en.wikipedia.org/wiki/Computer_vision) algorithms. It is one of the most widely used datasets for machine learning research. The CIFAR-10 dataset contains 60,000 32x32 color images in 10 different classes. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class.



Computer algorithms for recognizing objects in photos often learn by example. CIFAR-10 is a set of images that can be used to teach a computer how to recognize objects. Since the images in CIFAR-10 are low-resolution (32x32), this dataset can allow researchers to quickly try different algorithms to see what works.

CIFAR-10 is a labeled subset of the [80 Million Tiny Images](https://en.wikipedia.org/wiki/80_Million_Tiny_Images) dataset from 2008, published in 2009. When the dataset was created, students were paid to label all of the images.

Various kinds of [convolutional neural networks](https://en.wikipedia.org/wiki/Convolutional_neural_network) tend to be the best at recognizing the images in CIFAR-10.

The goal of this project is to develop a simple image classifier using a Convolutional Neural Network (CNN) to classify images from the CIFAR dataset. The CIFAR dataset consists of 60,000 32x32 color images in 10 different classes, with 6,000 images per class.

**How it is implemented**

This Image Classifier Model is Implemented by using TensorFlow



**Project Objectives**

* To preprocess the CIFAR dataset for input into a CNN model.
* To design and implement a CNN model suitable for image classification tasks.
* To train and validate the CNN model on the CIFAR dataset.
* To evaluate the performance of the trained model using various metrics.
* To document the process, code, outputs, and results.

**Dataset**

The CIFAR dataset used in this project contains images categorized into the following classes:

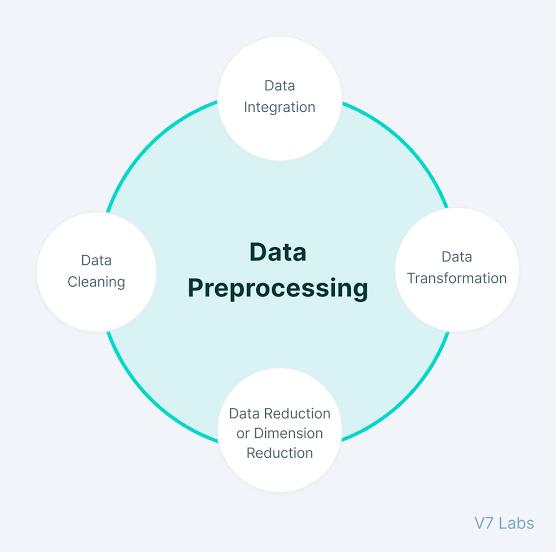
* Airplane
* Automobile
* Bird
* Cat
* Deer
* Dog
* Frog
* Horse
* Ship
* Truck



**Methodology**

**Data Preprocessing**

Data preprocessing involves normalizing the pixel values of the images andperforming any necessary data augmentation to enhance the dataset.

****

**Code:**

import tensorflow as tf

from tensorflow.keras.datasets import cifar10

from tensorflow.keras.utils import to\_categorical

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Loading the dataset

(x\_train, y\_train), (x\_test, y\_test) = cifar10.load\_data()

# Normalizing the images

x\_train = x\_train.astype('float32') / 255.0

x\_test = x\_test.astype('float32') / 255.0

# One-hot encoding the labels

y\_train = to\_categorical(y\_train, 10)

y\_test = to\_categorical(y\_test, 10)

# Data augmentation

datagen = ImageDataGenerator(

width\_shift\_range=0.1,

height\_shift\_range=0.1,

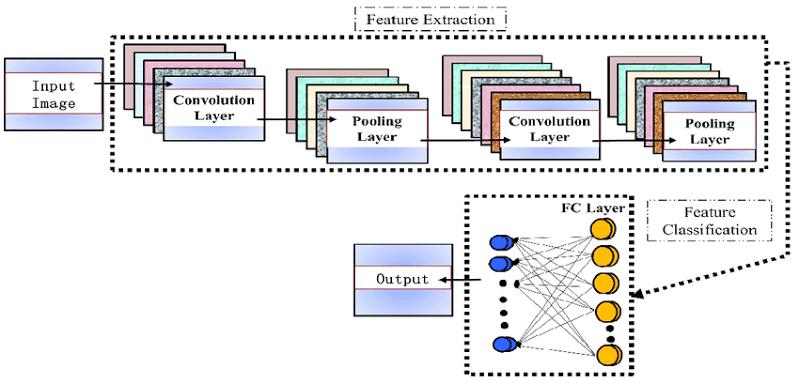
horizontal\_flip=True

)

datagen.fit(x\_train)

**Model Architecture**

The CNN model architecture is designed to effectively capture spatial hierarchies in the image data through convolutional layers, pooling layers, and fully connected layers.



**Code:**

from tensorflow import keras

from keras.models import Sequential

from keras.layers import Dense, BatchNormalization, Conv2D, MaxPooling2D, Activation, Flatten, Dropout

from keras.datasets import cifar10

import tensorflow.keras.utils

from keras.regularizers import l2

import numpy as np

import matplotlib.pyplot as plt

(xtrain, ytrain), (xtest, ytest) = cifar10.load\_data()

ytrain = tensorflow.keras.utils.to\_categorical(ytrain, 10)

ytest = tensorflow.keras.utils.to\_categorical(ytest, 10)

model = Sequential()

model.add(Conv2D(64, kernel\_size = (3,3),

                 input\_shape = (xtrain.shape[1:]),

                 padding = "same"))

model.add(Activation("relu"))

model.add(BatchNormalization())

model.add(Conv2D(64, kernel\_size = (3,3)))

model.add(Activation("relu"))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool\_size = (2,2)))

model.add(Dropout(0.25))

model.add(Conv2D(128, (3,3),

                 padding = "same"))

model.add(Activation("relu"))

model.add(BatchNormalization())

model.add(Conv2D(128, (3,3)))

model.add(Activation("relu"))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool\_size = (2,2)))

model.add(Dropout(0.5))

model.add(Flatten())

model.add(Dense(512, kernel\_regularizer = l2(0.01)))

model.add(Activation("relu"))

model.add(Dropout(0.5))

model.add(Dense(512))

model.add(Activation("relu"))

model.add(Dropout(0.5))

model.add(Dense(10))

model.add(Activation("softmax"))

model.summary()

**Compiling the model:**

model.compile(optimizer = "adam",

              loss = "categorical\_crossentropy",

              metrics = ["accuracy"])

xtrain = xtrain.astype("float32")

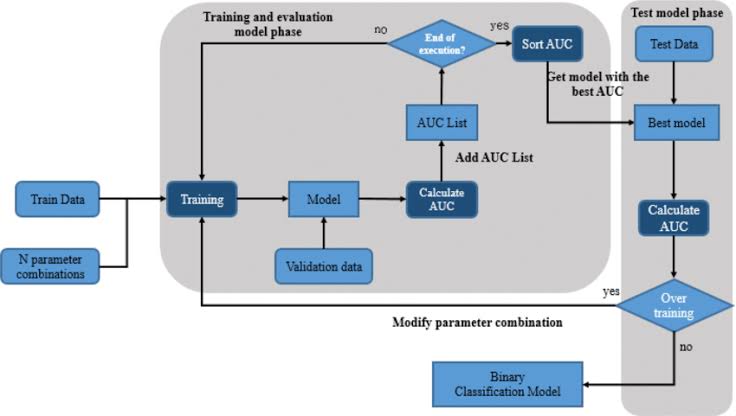
xtest = xtest.astype("float32")

xtrain = xtrain/225

xtest = xtest/225

**Model Training and Evaluation**

history = model.fit(xtrain, ytrain, batch\_size = 32, epochs = 50, validation\_data = (xtest, ytest),shuffle  =True)



**Results**

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

plt.show()

# summarize history for loss

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

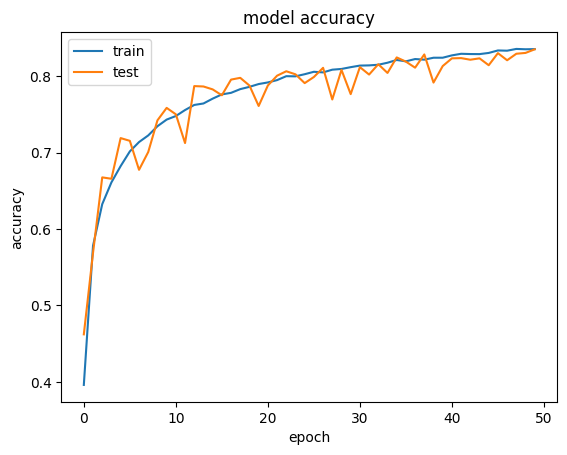
plt.title('model loss')

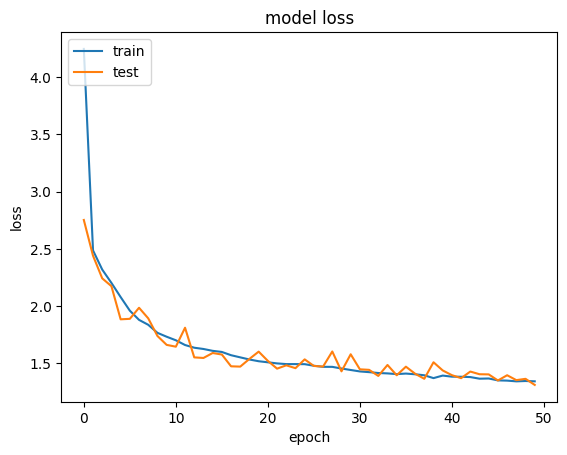
plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

plt.show()





**Conclusion**

In this project, a CNN model was successfully implemented to classify images from the CIFAR dataset. The model was trained and evaluated, achieving a test accuracy of 83%. This demonstrates the capability of CNNs to effectively handle image classification tasks.

**References**

* CIFAR-10 dataset
* [TensorFlow Documentation](https://www.tensorflow.org/)

The below are the link for dataset :

https://www.kaggle.com/c/cifar-10/