Summer Internship Project Report

On

Binary Classification Using Convolutional Neural Network

Submitted by

Chandra Bhusan Rai
Saurabh Kumar Pandey
Chandana N
Somnath Deb

Cambridge Institute of Technology
Bangalore, Karnataka

Under the Guidance of

Mr. M Justin Sagayaraj
Scientists G
Ms. Roshani Dange
Senior Researcher

Electronics and Radar Development Establishment (LRDE)
Defense Research & Development Organization
Ministry of Defense, Government of India, Bangalore

Acknowledgement

Electronics and Radar Development Establishment (LRDE) works in the area of design and development of ground-based, shipborne and airborne complex radar systems and related technologies. Currently, LRDE is also developing technologies for space-based radars. We are highly obliged to Mr. M Justin Sagayaraj, Scientists G, LRDE, Bangalore for allowing us to associate with this esteemed establishment as summer intern in 'AI Group' for a period from 15 April to 8 June 2021. We express our sincere thanks to Ms. Roshani Dange, Senior Researcher for her unflagging guidance throughout the progress of this project as well as valuable contribution in the preparation and compilation of the text. We are thankful to all those who have helped us for the successful completion of our internship at LRDE. Bangalore.

Chandra Bhusan Rai
Saurabh Kumar Pandey
Chandana N
Somnath Deb
B.E (Final year),
Cambridge Institute of Technology

Abstract

This report contains brief description of the internship program in LRDE, DRDO. This report elaborates the working of CNN model. The main focus of this internship is to collect the data, train and build a Convolutional neural network (CNN model) to classify between two classes. One being drone dataset and the other bird dataset.

During this period, I get the real, first-hand experience for working in the actual environment. Most of the theoretical and practical knowledge that I gained during the study is put to test here. It provides a linkage between the student and industry to develop an awareness of industrial approach to problem solving, based on a broad understanding of process.

I had the opportunity to have an experience on many ventures, which increased the sphere of knowledge for me to a great extent. I was entrusted with a real-life project, working on which had finally made me step into the industrial experience and gradually become a part of it.

TABLE OF CONTENT

LIST OF FIGURES

Chapter 1: Introduction

This report is a short description of the internship carried out as a component of B.E program. The internship was carried out in LRDE, DRDO. The internship is based on CNN model building for binary classification. A tremendous interest in deep learning has emerged in recent years.

The most established algorithm among various deep learning models is convolutional neural network (CNN), a class of artificial neural networks that has been a dominant method in computer vision tasks. CNN is a type of deep learning model for processing data that has a grid pattern, such as images and designed to automatically and adaptively learn spatial hierarchies of features, from low- to high-level patterns.

CNN is a mathematical construct that is typically composed of three types of layers (or building blocks): convolution, pooling, and fully connected layers. The first two, convolution and pooling layers, perform feature extraction, whereas the third, a fully connected layer, maps the extracted features into final output, such as classification and apply Binary function to classify an object with probabilistic values between 0 and 1.

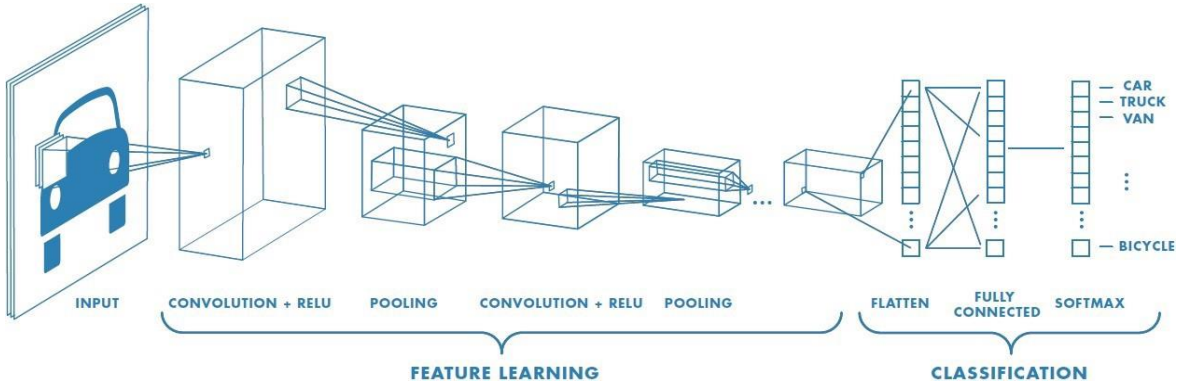

Figure 1: CNN Model

Our internship mainly focuses on binary classification where only two classes are being used. CNN image classification takes an input image, process it and classify it under certain categories i.e., drone or bird. Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d (h = Height, w = Width, d = Dimension). E.g., An image of 128 x 128 x 3 array of matrix of RGB (3 refers to RGB values). The above figure is a complete flow of CNN to process an input image and classifies the objects based on values.

Goals:

- To develop a CNN model for binary classification which includes two classes, drones and birds.
- To understand the functioning of the model.
- To see what it is like to work in a professional environment.
- To better use and understand the academic knowledge in practical environment
- To get real world experience in an environment which was unknown for us.

Objectives:

The main objectives of the internship are:

- Acquire knowledge regarding Convolutional neural network.
- To individually develop a CNN model for binary classification.
- To work individually.

This internship report contains my activities that have contributed to achieve the above stated goals. It provides a linkage between the student and industry to develop an awareness of industrial approach to problem solving.

Chapter 2: Literature Survey

In deep learning, a convolutional neural network **(**CNN**)** is a class of deep neural networks, most commonly applied to analyze visual imagery. Now imagine of a neural network, it seems to be matrix multiplications but that is not the case with CNN. It uses a special technique called Convolution.

R. Chauhan, K. K. Ghanshala and R. C. Joshi, "Convolutional Neural Network (CNN) for Image Detection and Recognition," 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC), 2018, pp. 278-282, doi: 10.1109/ICSCCC.2018.8703316- Convolutional neural networks are deep learning algorithms that can train large datasets with millions of parameters, in form of 2D images as input and convolve it with filters to produce the desired outputs. In this article, CNN models are built to evaluate its performance on image recognition and detection datasets.

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. 2017 International Conference on Engineering and Technology (ICET). doi:10.1109/icengtechnol.2017.8308186: In this paper it is explained and defined all the elements and important issues related to CNN, and how these elements work. In addition, it also states the parameters that effect CNN efficiency. This paper assumes that the readers have adequate knowledge about both machine learning and artificial neural network.

T. Agarwal and H. Mittal, "Performance Comparison of Deep Neural Networks on Image Datasets," 2019 Twelfth International Conference on Contemporary Computing (IC3), 2019, pp. 1-6, doi:10.1109/IC3.2019.8844924: With the introduction of convolutional neural networks, error rate of 2.5 percent has been successfully achieved on the image classification. In this paper, four popular convolutional neural network-based models namely, VGG16, mobileNet, Resnet50 and InceptionV3 have been considered for the comparative study. The efficiency of the considered models is evaluated on various image classification datasets namely; cats and dogs for binary classification and plant seedling dataset for multiclass classification in terms of accuracy.

Chapter 3: Architecture of CNN

Our project is building a CNN model for binary classification of two classes namely Drone and Bird. The first step is to design a python module to college images of Drones and Birds from various open sources like Kaggle. The next module is pre-processing of images into standard format, assuming the pixel size to be 128* 128. Once pre-processing is done, CNN model is then built for classification of two classes using Keras.

3.1. Block Diagram

The block diagram below shows the architecture of CNN model.

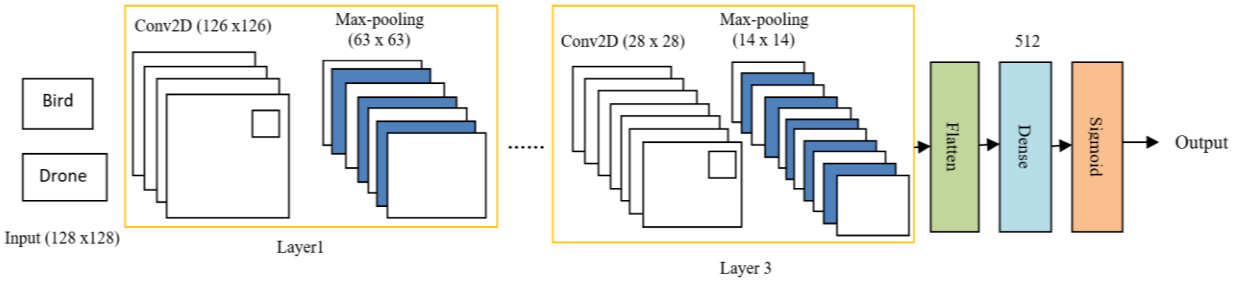

Figure 3: Architecture of CNN Model

There are two main parts to a CNN architecture.

- A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction
- A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.

3.1.1. Convolution Layer

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM).
The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

3.1.2. Pooling Layer

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations.
In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer.

3.1.3. Fully Connected Layer

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

In this, the input image from the previous layers is flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place.

3.1.4. Activation Functions

Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network.

It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, Binary, tanH and the Sigmoid functions. Each of these functions have a specific usage. For a binary classification CNN model, sigmoid and Binary functions are preferred for a multi-class classification, generally Binary us used.

Chapter 4: Implementation

4.1. Making CNN Model

To build a CNN model one needs datasets. As we are doing Binary classification, we took 1300 images of two classes each of bird and drone. The datasets were divided into training and validation datasets. The classes are mutually exclusive and there is no overlap between them.
Our datasets consist of three section:

- Training data consist of:
  -1200 of images Birds.
  -1200 of images Drones
- Validation Data consist of:
  -100 of images Birds
  -100 of images Drones
- Test Data consist of:
  -225 random images of birds and drone

Steps we followed to build CNN Model:

Step 1: Loading the datasets

```
train_dataset=train.flow_from_directory('train',
                                        target_size=(128,128),
                                        batch_size = 64,
                                        class_mode = 'binary')
validation_dataset = validation.flow_from_directory('test',
                                            target_size = (128, 128),
                                            batch_size = 64,
                                            class_mode = 'binary')
```

Fig 4.1.1: Datasets

Step 2: Create the convolution base and add dense layers on top

```
model=tf.keras.models.Sequential([tf.keras.layers.Conv2D(16,(3,3),activation='relu',input_shape=(128,128,3)),
                                  tf.keras.layers.MaxPool2D(2,2),
                                  #
                                  tf.keras.layers.Conv2D(32,(3,3),activation='relu'),
                                  tf.keras.layers.MaxPool2D(2,2),
                                  #
                                  tf.keras.layers.Conv2D(64,(3,3),activation='relu'),
                                  tf.keras.layers.MaxPool2D(2,2),
                                  ##
                                  tf.keras.layers.Flatten(),
                                  #
                                  tf.keras.layers.Dense(512,activation='relu'),
                                  #
                                  tf.keras.layers.Dense(1,activation='sigmoid')
                                  ])
```

Fig 4.1.2: Layers

Step 3: Compile and train the model

At this point, we have training data and a fully configured neural networkto train with set data. All that is left is to pass the data to the model for the training process to commence, a process

which is completed by iterating on the training data. Training beingins by calling the fit() method.
This step takes a few minutes based on the epochs given. Epoch is the number of times you would like to train the machine with the given dataset. Steps per epoch is calculated as train_length / batch_size.

Step 4: Saving the model
Our Model is saved in '.h5' format, An H5 file is a data file saved in the Hierarchical Data Format (HDF).

4.2. Deployment

Step 1**:** Creating Flask Application
Flask is a web framework in python language. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a machine learning webapp.
Once our model is ready**,** we will have to install Flask. Flask is a micro-framework in Python which is extensively used to deploy ML models on the web, we will have to install flask using the following command.

```
!pip install flask-ngrok
```

The dependencies required to run flask app on local host are shown below.

```
from flask import Flask, redirect, url_for, request, render_template
from werkzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer
from flask_ngrok import run_with_ngrok
```

Figure 4.2.1: Flask libraries

Step 2**:** Defining a flask app

```
# Define a flask app
app = Flask(__name__)
run_with_ngrok(app)
```

Figure 4.2.2: Flask App

Step 3**:** Create a HTML Page
Coming to the webpage we want to build. Create a HTML page according to your preference, we have created a basic webpage with a button to upload an image and a button to predict what the image is. This code sends the image as a post request and stores it in the folder "uploads/" where a prediction is made from here. The code is divided into 2 files, namely a .js file an a .html file.
App routing is used to map the URL with the index.html file using a get request.

```
@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('index.html')
```

Fig 4.2.3: Index

This code sends the image as a post request and stores it in the folder ". /uploads" where a prediction is made from here.

```
# Save the file to ./uploads
basepath = os.path.dirname(__file__)
file_path = os.path.join(
    basepath, 'uploads', secure_filename(f.filename))
f.save(file_path)
```

Fig 4.2.4: Filepath

Step 4**:** Loading the model and a Prediction Function

```
# Load your trained model
model = load_model(MODEL_PATH)
model.make_predict_function()

def model_predict(img_path, model):
    img = image.load_img(img_path, target_size=(128, 128))

    # Preprocessing the image
    x = image.img_to_array(img)
    # x = np.true_divide(x, 255)
    x = np.expand_dims(x, axis=0)

    # Be careful how your trained model deals with the input
    # otherwise, it won't make correct prediction!
    x = preprocess_input(x, mode='caffe')

    preds = model.predict(x)
    return preds
```

Fig 4.2.5: Prediction function

Step 5: Output

Finally, all we need to do is run our deploy.py file and go to http://8bec4c03c45d.ngrok.io see the output.

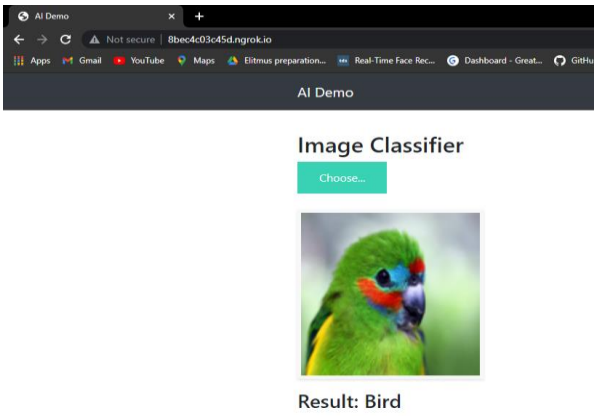

Fig 4.2.6: Output_1

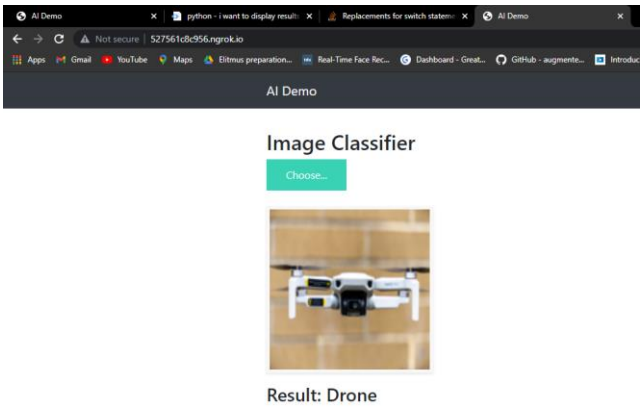

Fig 4.2.7: Output_2

Chapter 5: Result and Conclusion

Convolutional neural networks (CNN) have accomplished astonishing achievements across variety of domain. Although deep learning has become a dominant method in a variety of complex tasks such as image classification and object detection, it is not a panacea.

In this internship program, we had hands on experience on the creating a CNN model for classification between two classes and deploying the same. The dataset is fed to the network, the model extracts feature of both the classes and then classify images. The results gathered establish the state of validation accuracy to 93.25% using a deep neural network on the dataset. All results obtained from training the network for binary classification has shown an improvement in the overall performance. We conclude that CNNs give the best performance in image classification problems and even outperforms humans in certain cases.

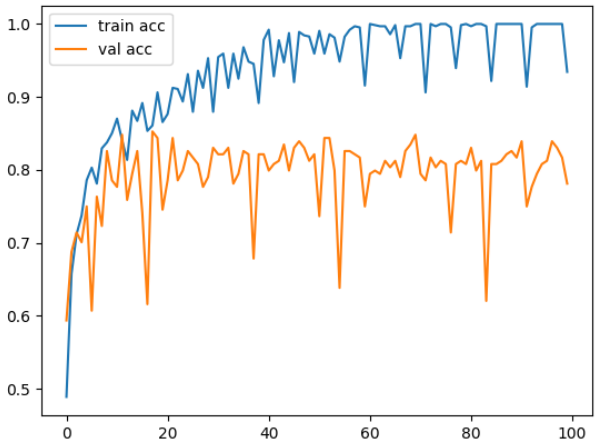

Figure 5.1: Model accuracy

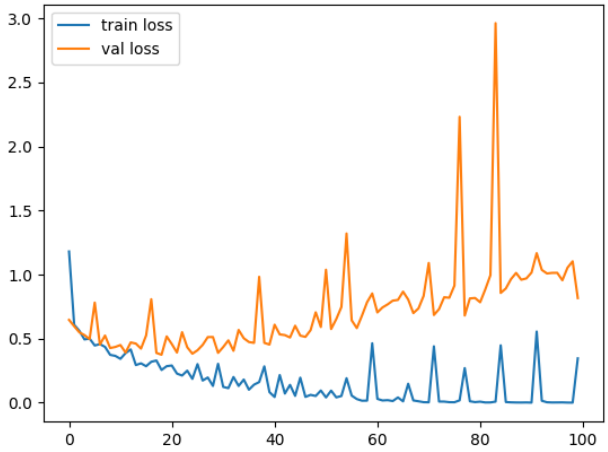

Figure 5.2: Model loss

We also had experience on deploying the model. Deploying the model in order to perform inference means that you've trained a model, tested its performance, and decided to use it to make predictions on new data points. The session remains active for 2 hrs. after running the python file.

Chapter 6: Bibliography

1] Fig1 https://images.app.goo.gl/JoyAqbzpinPYZZhE9

2] R. Chauhan, K. K. Ghanshala and R. C. Joshi, "Convolutional Neural Network (CNN) for Image Detection and Recognition," 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC), 2018, pp. 278-282, doi:

3] Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. 2017 International Conference on Engineering and Technology (ICET). doi:10.1109/icengtechnol.2017.8308186:

4] T. Agarwal and H. Mittal, "Performance Comparison of Deep Neural Networks on Image Datasets," 2019 Twelfth International Conference on Contemporary Computing (IC3), 2019, pp. 1-6, doi:10.1109/IC3.2019.8844924:

5] https://www.geeksforgeeks.org/deploying-a-tensorflow-2-1-cnn-model-on-the-web-with-flask/

6] https://www.youtube.com/watch?v=CSEmUmkfb8Q

7] https://www.youtube.com/watch?v=FmpDIaiMIeA

8] https://data-flair.training/blogs/cats-dogs-classification-deep-learning-project-beginners/

9] https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697

10]https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to-classify-photos-of-dogs-and-cats/