IBM MACHINE LEARNING: DEEP LEARNING AND REINFORCEMENT LEARNING PROJECT

Project: Airbus Turbine Classification

1. Objective:

Wind Turbines generate electricity with the help of wind energy. The turbine rotates when wind pushes the blades and this generates electricity. Wind energy is an alternative method to produce electricity. Conventional energy like coal or hydrocarbons which creates global warming and pollution. Governments are planning to harvest energy from natural sources like sun, water and wind which are also called as renewable energy. So, it is inevitable to build huge turbines and power stations. Knowing where this wind turbines are located is essential and could help in energy production forecast. Deep Learning could help identify wind turbines on satellite image.

In this project we use data given by Airbus DS GEO S.A. of satellite imagery of turbine images. The dataset contains two classes of images with turbines and images without turbines. The goal is to build a deep learning network to classify the images into their respective classes

2. Data Description

The data contains two classes viz. images with turbines called as 'turbine' and images with no turbine called as 'background'. The images are color images of size 128 and 128. Images are extracted from satellite acquisitions from the Airbus SPOT6 and SPOT7 satellites which resolution is 1.5 meters per pixel. So, extracts of 128 x 128 pixels represents roughly 192 meters on the ground which is compatible with the typical size of a wind turbine.

3. Data Exploration/Data Cleaning

The images downloaded come in two different folders called as 'train' and 'val'. The images are also come with improper names. In order to give them a proper name, we define a python function which extracts the given file names and renames the images into proper names.

After renaming, we take a sample of the images and store it as 'test' data for testing the model's performance on unseen data. Thus, we have three datasets viz. Train set for training, Validation set for hyperparameter tuning and Test set to check model performance on unseen data.

The training set contains a total image of 356006 images, the validation set contains 60174 images and test set contains 11120 images.

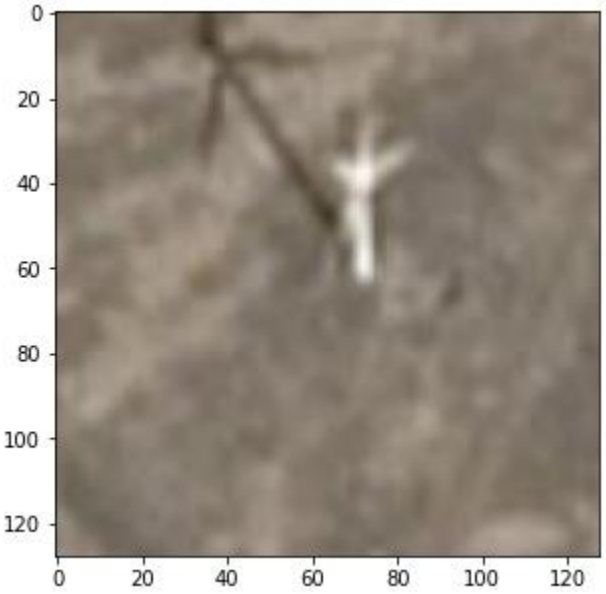

Figure 1. Turbine image of size 128 x 128.

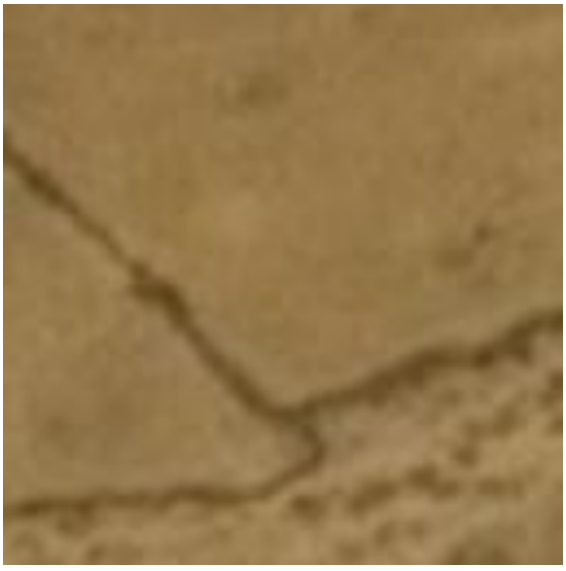

Figure 2. Background Image,

4. Methodology

Convolution Neural Network is used to classify images in this project. The model is created with TensorFlow's Keras API using the Sequential Model. The model architecture is given below:

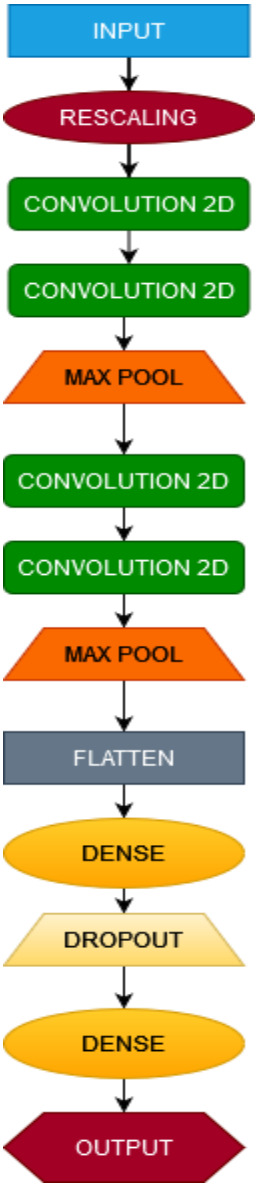

Figure 3. Architecture of the network.

The input image is resized to a shape of 64 x 64. Then it is normalized and fed into the convolution layer. The convolution layer has filters of 64 and kernel size of 4 with 'relu' activation with no padding. The pooling layer is max pooling with kernel size of 3 and no padding. The Dense layers has 128 and 32 units with 'relu' activation. The dropout is introduced to avoid overfitting with dropout rate as 0.2. Finally, the output layer is a sigmoid activated dense layer with 2 units.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling (Rescaling)        (None, 64, 64, 3)         0
_________________________________________________________________
conv2d (Conv2D)              (None, 61, 61, 64)        3136
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 58, 58, 64)        65600
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 19, 19, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 64)        65600
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 13, 13, 64)        65600
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 128)               131200
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                4128
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 66
=================================================================
Total params: 335,330
Trainable params: 335,330
Non-trainable params: 0
_________________________________________________________________
```

Figure 4. Summary of Model.

The model is compiled with Binary Cross-Entropy as the loss function with Adam Optimizer and accuracy as Categorical Accuracy. Two callbacks are introduced. One callback is to record the losses and accuracy which is called as CSV Logger. The other callback is to stop training when validation accuracy reaches to 97%.

```python
class StopOnThreshold(tf.keras.callbacks.Callback):
    def __init__(self, threshold):
        super(StopOnThreshold, self).__init__()
        self.threshold = threshold
    def on_epoch_end(self, epoch, logs=None):
        val_acc = logs['val_categorical_accuracy']
        if val_acc >= self.threshold:
            print(f'Stopping Training! Validation Accuracy Reached Threshold of {val_acc*100}.')
            self.model.stop_training = True
# Call
stop_on_threshold = StopOnThreshold(0.97)
```

Figure 5. Stopping Function Callback.

5. Results

The model was set to train for 10 epochs. But the model reached training accuracy and validation accuracy to 0.9818 and 0.9916 respectively in just one epoch. Due to callback, the training was stopped. But note that one epoch took nearly 12 hours to complete.

```
Epoch 1/10
3561/3561 [==============================] - 42723s 12s/step - loss: 0.0526 - categorical_accuracy: 0.9819 - v
al_loss: 0.0240 - val_categorical_accuracy: 0.9916
Stopping Training! Validation Accuracy Reached Threshold of 99.15910363197327.
```

Figure 6. Training details.

	epoch	categorical_accuracy	loss	val_categorical_accuracy	val_loss
0	0	0.981947	0.05263	0.991591	0.024021

Figure 7. CSV Logger Details.

The test set is used to evaluate model performance on unseen data. The performance of test set was obtained to be 99.12 %.

5.1 Prediction

A python function is written which predicts any image's class which is passed to it. A couple of results is given below

Reality: This image is a Turbine

Prediction: This image is a turbine.
Prediction Accuracy: 100.0%

Reality: This image is a Background

Prediction: This image is a background.
Prediction Accuracy: 99.949997663497928

Figure 8. Predictions of Turbine and Background.

5.2 Confusion Matrix

The confusion matrix gives us an idea of how many images were classified correctly and how many were not. Given below is the confusion matrix obtained. 75 turbine images were classified as 'Background' and 22 background images were classified as 'Turbines'.

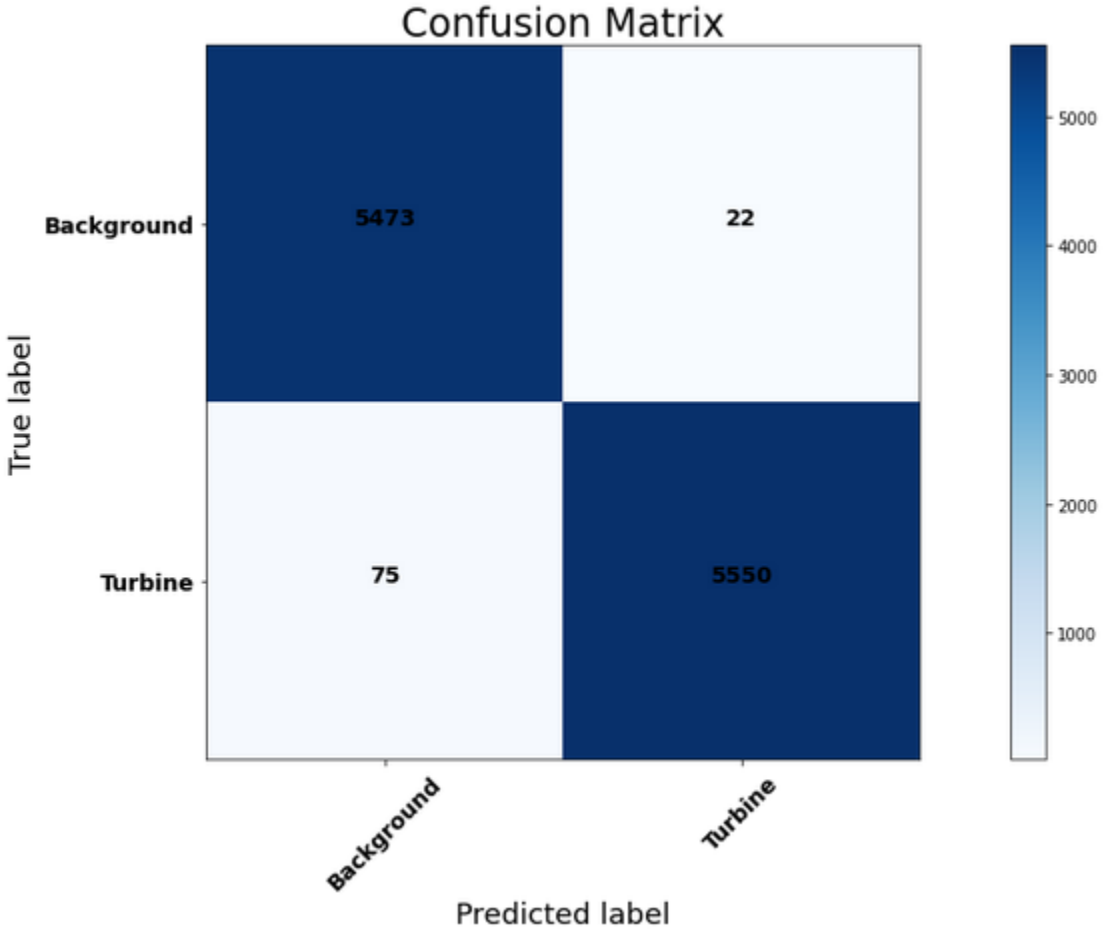

Figure 9. Confusion Matrix.

6. Conclusion

The model was finally trained with just one epoch. Since there are so many images in one class and there are only two classes, the training took only one epoch. Different networks are not experimented since we reached high accuracy and low loss.

Finally, the deep learning model developed is a very good model as the accuracy and predictions are accurate and precise. If we want to still improve the model, the model can be trained with the actual image size of 128 x 128.

7. Data Reference:

Data: The data is obtained from Kaggle website. The data is developed by Airbus Innovation Team DS GEO S.A.

Website: https://www.kaggle.com/airbusgeo/airbus-wind-turbines-patches