Journal of Airline Operations and Aviation Management

The Netherlands Press

Article

RECOGNITION OF AIRCRAFT IN REMOTE SENSING IMAGES USING CONVOLUTIONAL NEURAL NETWORK

Jai Krishna Sahith

DSchool of Chemical & Bio-process Engineering,
University college Dublin,
Dublin, Ireland
Orchid ID: https://orcid.org/0000-0002-7407-6759

E-mail: krishnasahith355@gmail.com

Abstract.

Picture processing is a stand-alone, non-human image comprehension system that is one of the major breakthroughs (IUS). When one tries to explain what comprehension is, the effort of comprehending images becomes massive. In addition to classical signal processing, pattern recognition and artificial intelligence are applied. The early phases of the picture comprehension process might be called scene analysis methods that use edge and texture segmentation. As a result, putting a man in a signal processing loop at particular sensors, such as remotely piloted vehicles, satellites, and spacecraft, is not acceptable. Smart sensors and semi-automated procedures are being created as a result. Another major use of image processing is land remote sensing. With the debut of shows like Star Wars, this application has taken on a new level of relevance in the military's eyes. This study gives an overview of digital image processing and investigates the reach of remote sensing and IUS technology from the military's perspective. To demonstrate the significance of IUSs, a detailed description of a current autonomous car project in the United States is provided.

Keywords: Segmentation, satellites, aircrafts, Artificial intelligence, Remote sensing images.

Journal of Airline Operations and Aviation Management Volume 1 Issue 1
Received Date: 16 May 2022
Accepted Date: 13 June 2022
Published Date: 25 July 2022

1. Introduction

Remote sensing is the process of detecting and monitoring an area's physical characteristics by measuring its reflected and emitted radiation from afar (usually by satellite or aircraft). Using sophisticated cameras to acquire remote sensing photos, researchers can "smell" the truth about Earth. In machine learning, constitutive neural networks are a key concept for anticipating picture issues [1]. Tensors are used to tackle image difficulties. Tensorflow api includes a number of libraries and techniques for tackling tensor challenges. R-CNN addresses these kinds of issues more quickly and accurately. The network is connected via connections, with each neuron's output serving as an input to another. Each connection is given a weight to indicate its importance. A single neuron might have a number of different information and outcome relationships [2]. It connects neurons from one layer to neurons from another layer or neurons from a comparable layer. Weight self-esteem is always linked to a relationship [3]. The purpose of the preparation is to get this weight figure up to date in order to minimise the loss (error)[4]. A dependable 1 extra contribution to neurons with its own weighted connection. This ensures that the neuron will be locked in regardless of whether all of the sources of information are zero[5]. Learning is the process through which a network adapts to a task by absorbing sample data. The network's weights (and optional criteria) are changed to increase the accuracy of the result. This is performed by lowering the amount of faults that may be detected [6]. When looking at more data doesn't assist reduce the mistake rate, the learning process is over. Indeed, even after learning, the error rate barely drops below 1% on rare occasions [7]. If the organization's error rate is abnormally high after learning, it should be updated on a regular basis. In practise, this is refined by creating an expenditure that works [8].

2. MULTI TYPE REMOTE SENSING IMAGES (MTARSI):

The MTARSI dataset is a collection of 9385 remote sensing aeroplane photos from Google satellite photography. These photos were taken at 36 different airports and feature 20 different types of aircraft. In this dataset, there are 20 different aircraft types: A-10, A-26, B-1, B-2, Boeing, B-29, B-52, C-21, C-130, C-135, C-17, C-5, E-3, F-16, F-22, KC-10, T-6, T-43, U-2, and P-63. Specialists label these photos, and they label them themselves[9].

A-10	A-26	B-1
Boeing	B-29	B-52

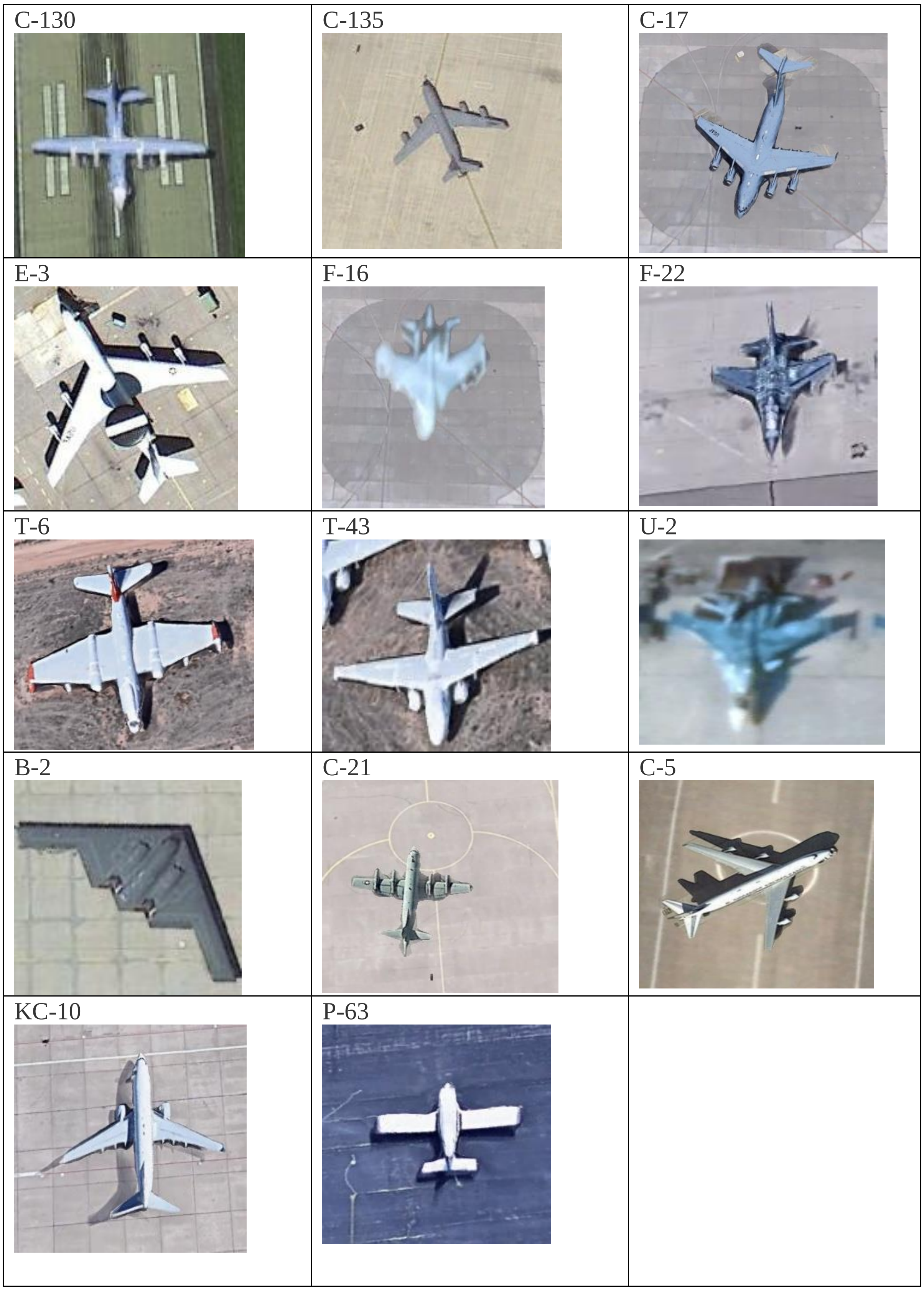
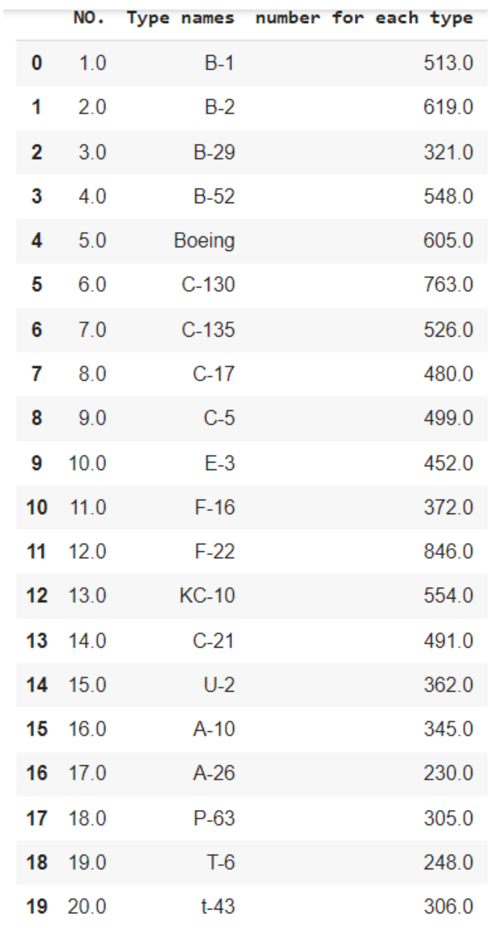


Figure 1: Remote sensing images[9]

The table below shows the number of photos in the dataset for each class. The dataset contains a total of 9385 photos divided into 21 classes[10].



	NO.	Type names	number for each type
0	1.0	B-1	513.0
1	2.0	B-2	619.0
2	3.0	B-29	321.0
3	4.0	B-52	548.0
4	5.0	Boeing	605.0
5	6.0	C-130	763.0
6	7.0	C-135	526.0
7	8.0	C-17	480.0
8	9.0	C-5	499.0
9	10.0	E-3	452.0
10	11.0	F-16	372.0
11	12.0	F-22	846.0
12	13.0	KC-10	554.0
13	14.0	C-21	491.0
14	15.0	U-2	362.0
15	16.0	A-10	345.0
16	17.0	A-26	230.0
17	18.0	P-63	305.0
18	19.0	T-6	248.0
19	20.0	t-43	306.0

2.1. PRE-PROCESSING THE IMAGES:

Images in the dataset may be of various sizes, but they should all be of the same size before being fed into neural networks. The rescale command in the python library is used to scale the photographs to the same size, and all of the scaled images are saved somewhere else or in a new folder.

Using the Numpy library, all of the pixels in the photos are divided by 255 to get pixels that range from 0 to 1. For values in the range of 0 to 1, neural networks often function well. These characteristics are sent into the neural network, which then trains the model.

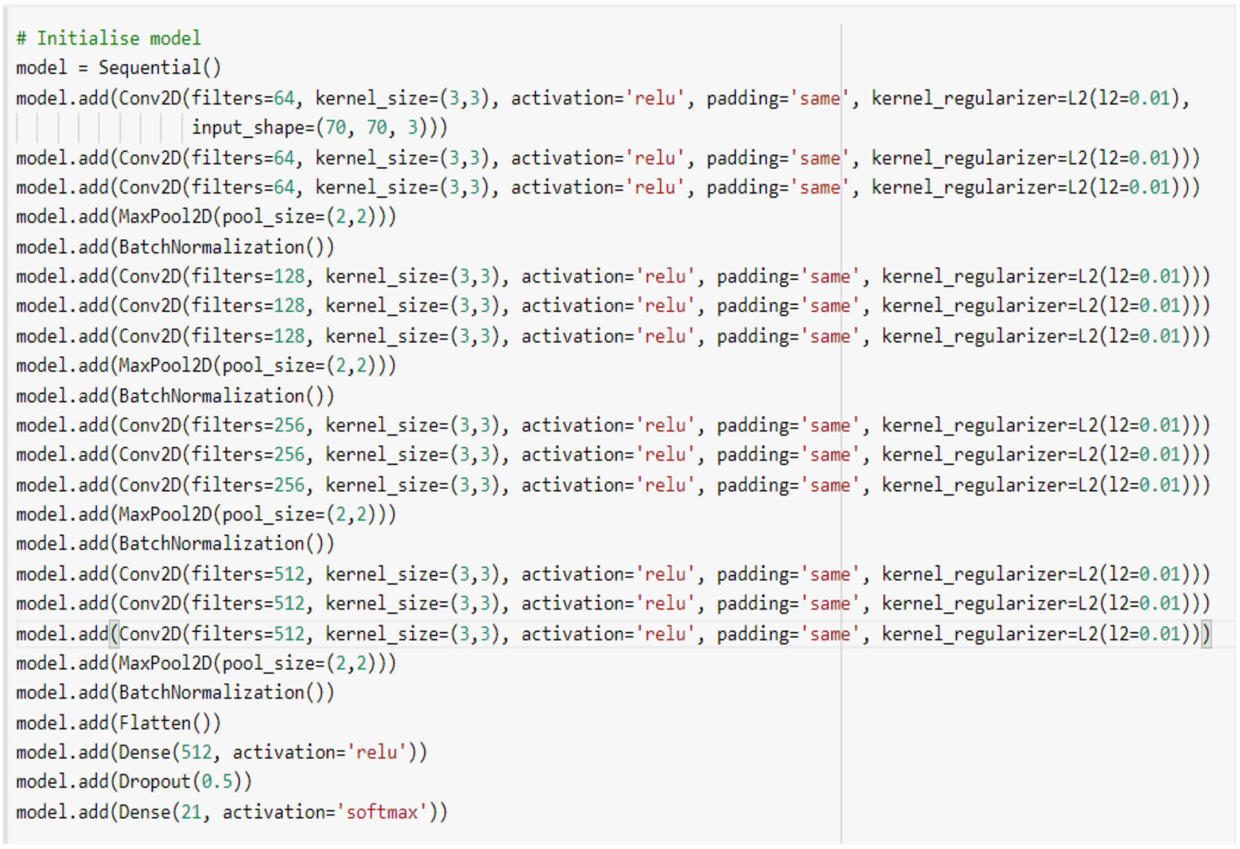
The data is split into test and train data for model assessment, with the train data being utilised to train the model.

2.2. BUILDING CONVOLUTIONAL NEURAL NETWORK MODEL:

The name of the model is sequential, and it comes from the tensorflow API. A convolutional layer is inserted first, followed by a sequential layer using the 'ReLu' activation function. And the activation regularisation for this layer is L2, with a value of 0.01. Following the pooling layer of the kernel size with a 2x2 window and the application of the pooling layer, Batch normalisation is performed to normalise the integers between 0 and 1. These blocks are arranged in the sequential model with a total of 4 blocks, and at the end of the sequential model, a Dense layer with 32 output neurons is added, with a dropout function of 50% applied to the dense layer, which contains the ReLu activation function, and then another Dense layer is added to the sequential model with a total of 21 neurons, as the number of classes in the dataset is 21.

2.2.1 MODEL 2:

Because of the low accuracy of model 1, the model should be revised after seeing the outcomes of model 1. The new design is Following the three convolutional layers, a pooling layer with a kernel size of 2x2 is added, followed by the Batch Normalization function. The kernel regularizer used in the convolutional layer is L2 with a value of 0.01. These blocks are arranged in a four-block sequence, after which a Dense layer with a layer size of 512 neurons is added, where all the values from the previous layer are flattened and passed through the dense layer, and finally a final output layer with a dense layer size of 21 is added, as the dataset contains 21 classes. The output layer's activation function is the softmax function. The completed model is seen in the diagram below.



```
# Initialise model
model = Sequential()
model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding='same', kernel_regularizer=L2(l2=0.01),
                 input_shape=(70, 70, 3)))
model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding='same', kernel_regularizer=L2(l2=0.01)))
model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding='same', kernel_regularizer=L2(l2=0.01)))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(filters=128, kernel_size=(3,3), activation='relu', padding='same', kernel_regularizer=L2(l2=0.01)))
model.add(Conv2D(filters=128, kernel_size=(3,3), activation='relu', padding='same', kernel_regularizer=L2(l2=0.01)))
model.add(Conv2D(filters=128, kernel_size=(3,3), activation='relu', padding='same', kernel_regularizer=L2(l2=0.01)))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(filters=256, kernel_size=(3,3), activation='relu', padding='same', kernel_regularizer=L2(l2=0.01)))
model.add(Conv2D(filters=256, kernel_size=(3,3), activation='relu', padding='same', kernel_regularizer=L2(l2=0.01)))
model.add(Conv2D(filters=256, kernel_size=(3,3), activation='relu', padding='same', kernel_regularizer=L2(l2=0.01)))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(filters=512, kernel_size=(3,3), activation='relu', padding='same', kernel_regularizer=L2(l2=0.01)))
model.add(Conv2D(filters=512, kernel_size=(3,3), activation='relu', padding='same', kernel_regularizer=L2(l2=0.01)))
model.add(Conv2D(filters=512, kernel_size=(3,3), activation='relu', padding='same', kernel_regularizer=L2(l2=0.01)))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(21, activation='softmax'))
```

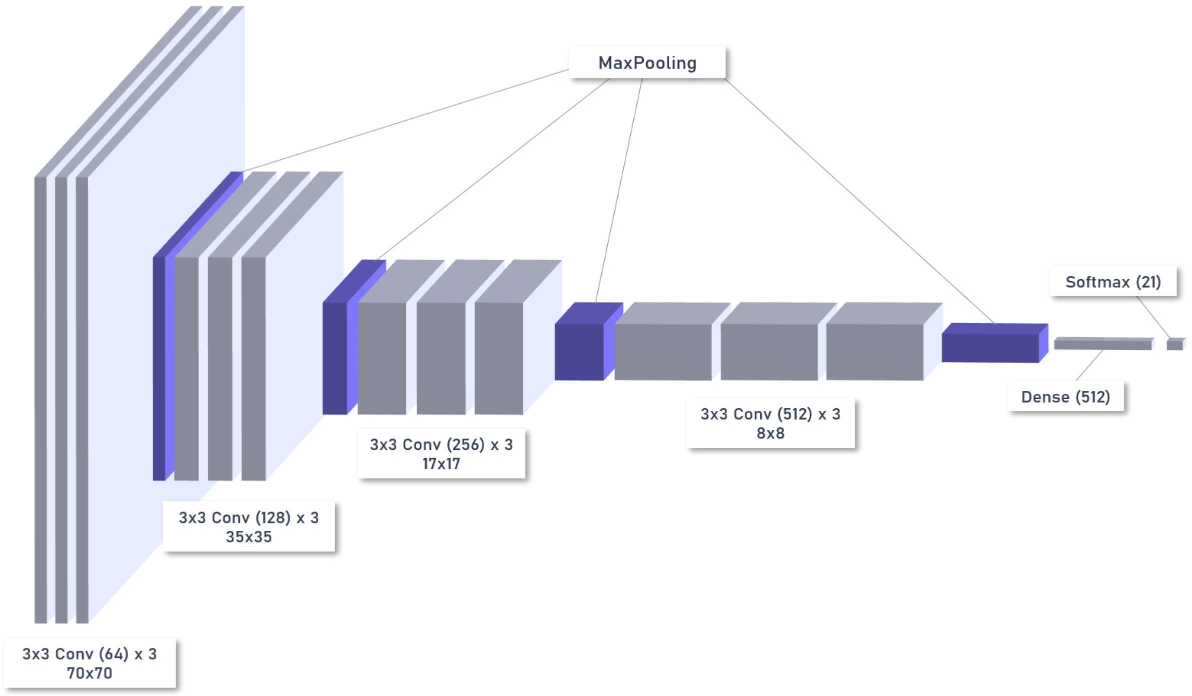
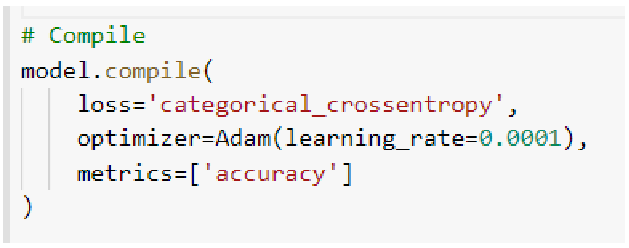


Figure 2: CNN Architecture

3. RESULTS AND DISCUSSIONS

3.1. TRAINING THE MODEL:

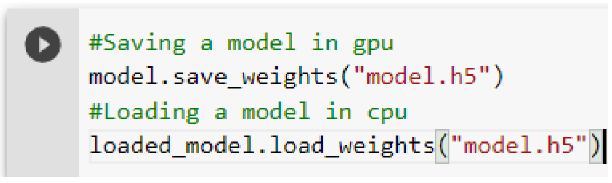
Categorical cross-entropy was utilised as the loss function, while Adam was employed as the optimizer, with a learning rate of 0.001 and accuracy as the model assessment measure. The error or loss is backpropagated and improved using an optimizer in the forward pass of the neural network, and this back propagation is done for a given number of epochs, with the final epoch result of accuracy being the training accuracy acquired.



```
# Compile
model.compile(
    loss='categorical_crossentropy',
    optimizer=Adam(learning_rate=0.0001),
    metrics=['accuracy']
)
```

3.2. SAVING THE MODEL:

The last epoch's weights and bias are preserved in the.h5 file format, and these weights can be utilised instead of training many times. In this example, the photos are tested by saving the model in GPU and loading it in CPU.



```
#Saving a model in gpu
model.save_weights("model.h5")
#Loading a model in cpu
loaded_model.load_weights("model.h5")
```

The loss and accuracy graphs are displayed using the matplotlib library using the first model and training the model with the photos from the dataset. The accuracy attained by using model 1 is 82.5 percent, and the loss is strictly decreasing. However, the model may be adjusted by modifying the model's layers, and the final model is shown above, with the final model's findings differing considerably from the initial model.

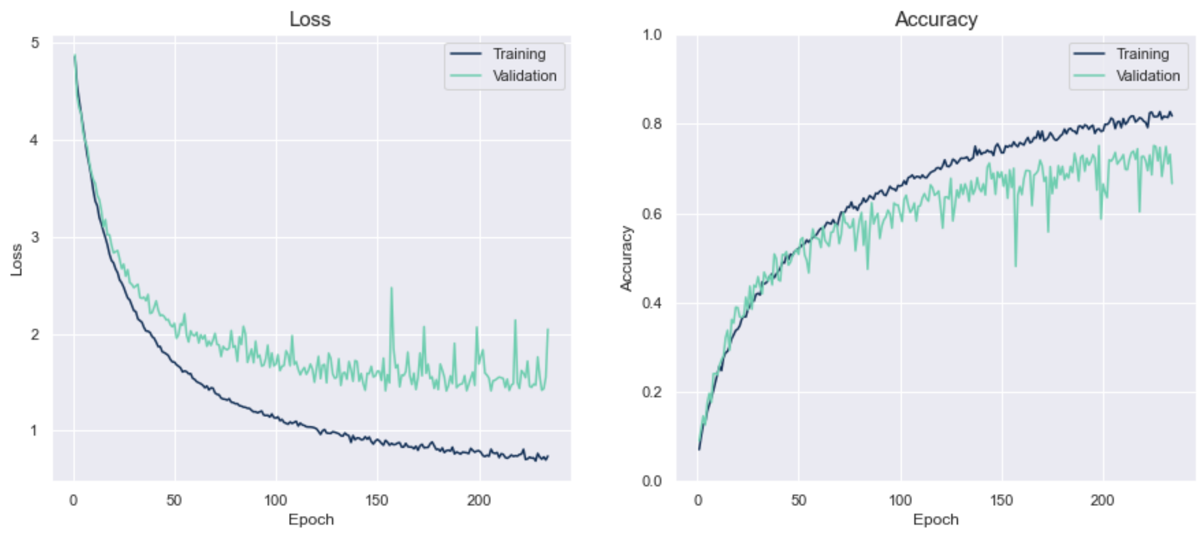


Figure 1: Model 1 loss and accuracy

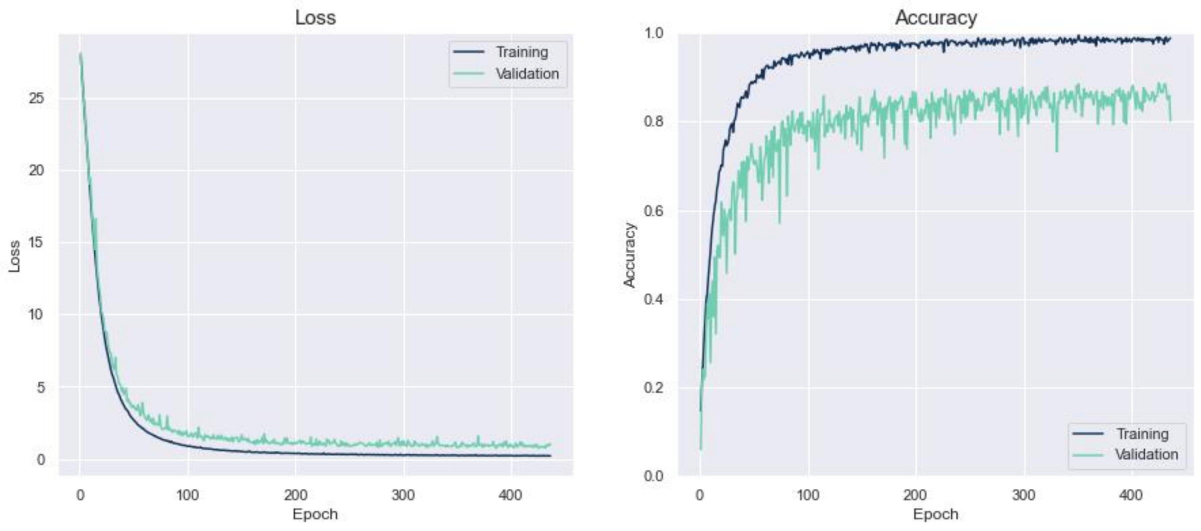
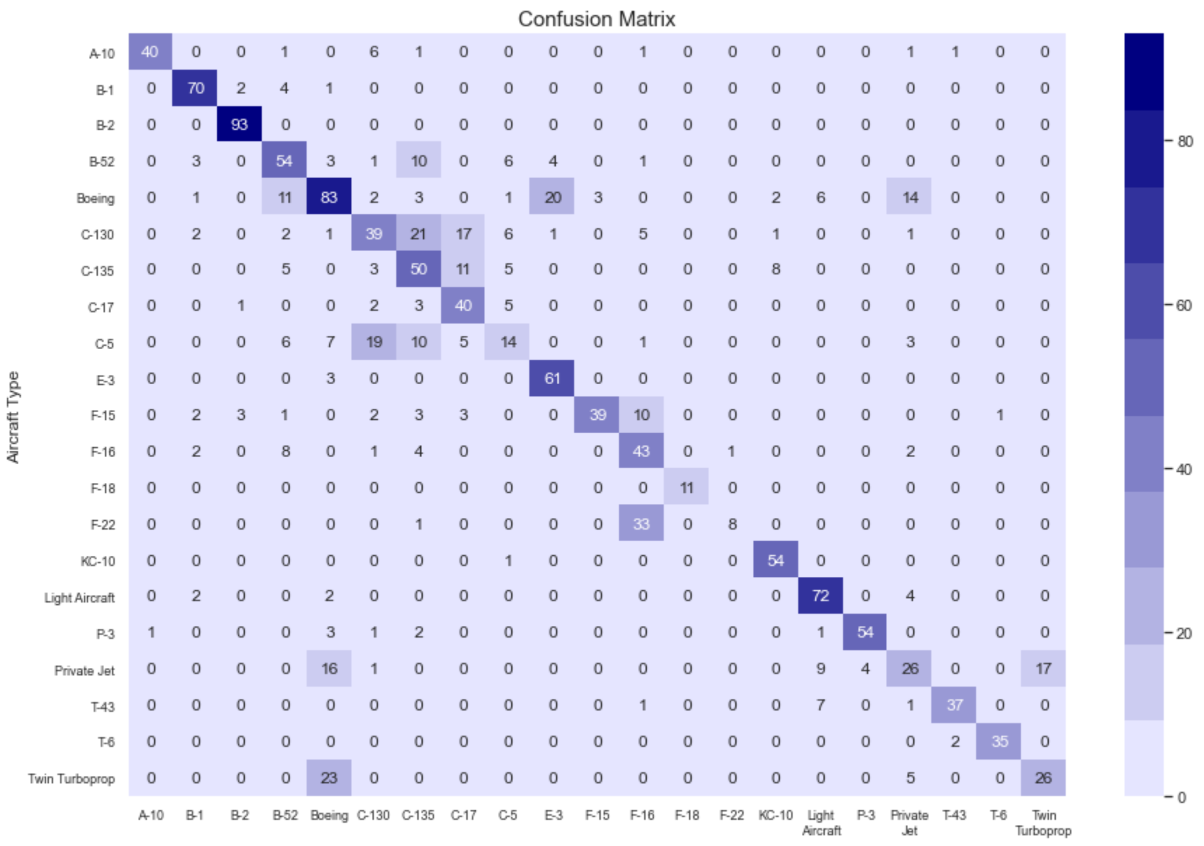


Figure 2: Model 2 loss and accuracy

Precision is obtained by employing the confusion matrix, f1 score, recall, and precision. The results are obtained by testing with some image in the dataset, and the testing images are given to the model, and all are predicted well because the accuracy obtained for the final model is good. For example, an image from the A-10 class is given, and it is predicted correctly by passing through the model using the convolutional neural network images can be predicted.



	A-10	B-1	B-2	B-52	Boeing	C-130	C-135	C-17	C-5	E-3	F-15	F-16	F-18	F-22	KC-10	Light Aircraft	P-3	Private Jet	T-43	T-6	Twin Turboprop
A-10	40	0	0	1	0	6	1	0	0	0	0	1	0	0	0	0	0	1	1	0	0
B-1	0	70	2	4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B-2	0	0	93	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B-52	0	3	0	54	3	1	10	0	6	4	0	1	0	0	0	0	0	0	0	0	0
Boeing	0	1	0	11	83	2	3	0	1	20	3	0	0	0	2	6	0	14	0	0	0
C-130	0	2	0	2	1	39	21	17	6	1	0	5	0	0	1	0	0	1	0	0	0
C-135	0	0	0	5	0	3	50	11	5	0	0	0	0	0	8	0	0	0	0	0	0
C-17	0	0	1	0	0	2	3	40	5	0	0	0	0	0	0	0	0	0	0	0	0
C-5	0	0	0	6	7	19	10	5	14	0	0	1	0	0	0	0	0	3	0	0	0
E-3	0	0	0	0	3	0	0	0	0	61	0	0	0	0	0	0	0	0	0	0	0
F-15	0	2	3	1	0	2	3	3	0	0	39	10	0	0	0	0	0	0	0	1	0
F-16	0	2	0	8	0	1	4	0	0	0	0	43	0	1	0	0	0	2	0	0	0
F-18	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0
F-22	0	0	0	0	0	0	1	0	0	0	0	33	0	8	0	0	0	0	0	0	0
KC-10	0	0	0	0	0	0	0	0	1	0	0	0	0	0	54	0	0	0	0	0	0
Light Aircraft	0	2	0	0	2	0	0	0	0	0	0	0	0	0	0	72	0	4	0	0	0
P-3	1	0	0	0	3	1	2	0	0	0	0	0	0	0	0	1	54	0	0	0	0
Private Jet	0	0	0	0	16	1	0	0	0	0	0	0	0	0	0	9	4	26	0	0	17
T-43	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	7	0	1	37	0	0
T-6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	35	0
Twin Turboprop	0	0	0	0	23	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	26

Figure 5: Confusion Matrix

The probability of predicting the pictures is determined for each class, and the probability for each class is given in the graph below. Almost all of the probabilities are close to 1, and the wrong probability is quite low, therefore the model produces superior results for remote sensing image prediction.

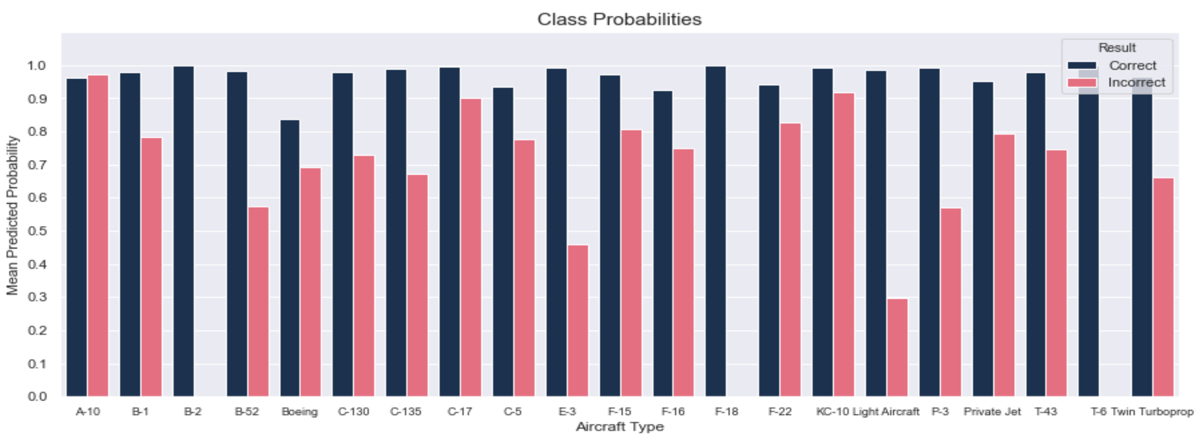


Figure 6: Class probabilities

4. CONCLUSION:

This research presents a complicated neural network-based aeroplane detection methodology. Transfer learning and aircraft samples from Google Earth were used to construct an end-to-end trainable aircraft recognition system. A rotation procedure is incorporated while adding data to boost the variety of the training samples. For large-scale remote sensing photos as input, an altered neural network model is provided, which successfully overcomes the problem of tiny feature loss caused by simple image scaling. The suggested aeroplane identification framework is highly successful since it can be utilised for a variety of object recognition applications in remote sensing image processing. A single convolutional neural network has a quicker detection speed than a two-step approach. Training is efficient and easy to converge thanks to the advantages of an end-to-end formatable architecture. The aircraft identification approach described in this work likewise has a good detection performance, according to our test findings.

References

[1] Filippidis, A.; Jain, L.C.; Martin, N. Fusion of intelligent agents for the detection of aircraft in SAR images. IEEE Trans. Pattern Anal. Mach. Intell. 2000, 22, 378–384.

[2] Yao, J.; Zhang, Z. Semi-supervised learning based object detection in aerial imagery. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA, 20–25 June 2005; pp. 1011–1016.

[3] Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Neural Information Processing Systems (NIPS), Lake Tahoe, NV, USA, 3–8 December 2012; Volume 2, pp. 1097–1105.

[4] Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F.F. Imagenet: A large-scale hierarchical image database. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.

[5] Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. IJCV 2004, 60, 91–110.

[6] Dalal, N.; Triggs, B. Histograms of Oriented Gradients for Human Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA, 20–25 June 2005; pp. 886–893.

[7] Szegedy, C.; Liu, W.; Jia, Y.Q.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.

[8] Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. arXiv, 2014.

[9] Z. Wu, "Muti-type Aircraft of Remote Sensing Images: MTARSI | Zenodo," Zenodo, May 18, 2019. https://zenodo.org/record/3464319#.YrKD_3ZBy3A (accessed Jun. 22, 2022).

[10] [8]"A benchmark data set for aircraft type recognition from remote sensing images - ScienceDirect," A benchmark data set for aircraft type recognition from remote sensing images -ScienceDirect,Jan.31,2020. https://www.sciencedirect.com/science/article/abs/pii/S1568494620300727 (accessed Jun. 22, 2022).