

CLOUD CLASSIFICATION FOR WEATHER PREDICTION USING DEEP LEARNING

AKSHAYA B

Regno: 210419104005

Under the supervision of

PREMALATHA T K

JULY 2022

ABSTRACT

Weather condition is an important factor that is considered for various decisions. In the industrial world, weather classification is very useful, such as in development of self-driving cars, smart transportation systems, and outdoor vision systems. Manual weather classification by humans is inconsistent and takes a long time. Weather forecast information obtained from the internet is not real time at a specific location. Computer vision is a branch of computer science to recognize or classify images that can assist in classifying cloud images that do not depend on weather forecast information from the internet. **This project aims to classify cloud images using Convolutional Neural Network (CNN) to perform cloud image classification that is used predicting the weather condition.** The proposed method will be applied to the cloud image which consists of eleven classes, with each class denoting the type of cloud, cirrus, cirrostratus, cirrocumulus, altocumulus, altostratus, cumulus, cumulonimbus, nimbostratus, stratocumulus, stratus and contrail were classified in this study.

ACKNOWLEDGEMENT

I am using this opportunity to express my gratitude to everyone who supported me throughout the course of this group project. I am thankful for their aspiring guidance, invaluable constructive criticism and friendly advice during the project work. I am sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.

Further, I am fortunate to have Mrs. PREMALATHA T K as my mentor. She has readily shared her immense knowledge in data analytics and guided me in a manner that the outcome resulted in enhancing my data skills. I wish to thank all the faculties, as this project utilized knowledge gained from every course that formed the DSP program.

I certify that the work done by me for conceptualizing and completing this project is original and authentic.

Date: July 22, 2022

Name: Akshaya B

Place: Chennai

CERTIFICATE OF COMPLETION

I hereby certify that the project titled “**Cloud Classification for Weather Prediction Using Deep Learning**” was undertaken and completed (22nd July 2022)

Mentor: Premalatha T K

Date: July 22, 2022

Place – Chennai

Table of Contents

Abstract	2
Acknowledgement	3
Certificate of completion	4
1. Introduction	6
1.1 Title and objective of study	6
1.2 Need of the study	7
2. Datasources	8
2.1 Data preprocessing	10
2.1.1 Grayscale conversion	10
2.1.2 Flipping	10
2.1.3 RotationData Pre-processing	11
2.1.4 Scaling and Standardizing images	11
3. Fitting models to data	12
3.1 CNN for image classification	12
3.2 Resnet50 for image classification	18
4. Predicting new data	20
4.1 Prediction with CNN model	21
4.2 Prediction with Resnet50	22
5. Conclusion	23
6. References	24

CHAPTER 1

INTRODUCTION

1.1 Title & Objective of the study

In **computer vision**, we have a **convolutional neural network** that is very popular for computer vision tasks like **image classification**, **object detection**, **image segmentation** and a lot more.

The primary aim corresponding to this research is to make cloud image classification using convolutional neural network model and to deploy the model so that user can upload the live cloud data and find out the weather condition.

The research objectives are formulated based on the aim of this study which are as follows:

- a. To train CNN model using the Cirrus Cumulus Stratus Nimbus (CCSN) dataset.
- b. The trained model should correctly classify the test cloud image
- c. To deploy the model so that user can interact.

1.2 Need of the Study

Being able to predict the weather by observing cloud formations is a skill that is somewhat lost on us modern humans. Most of us can easily

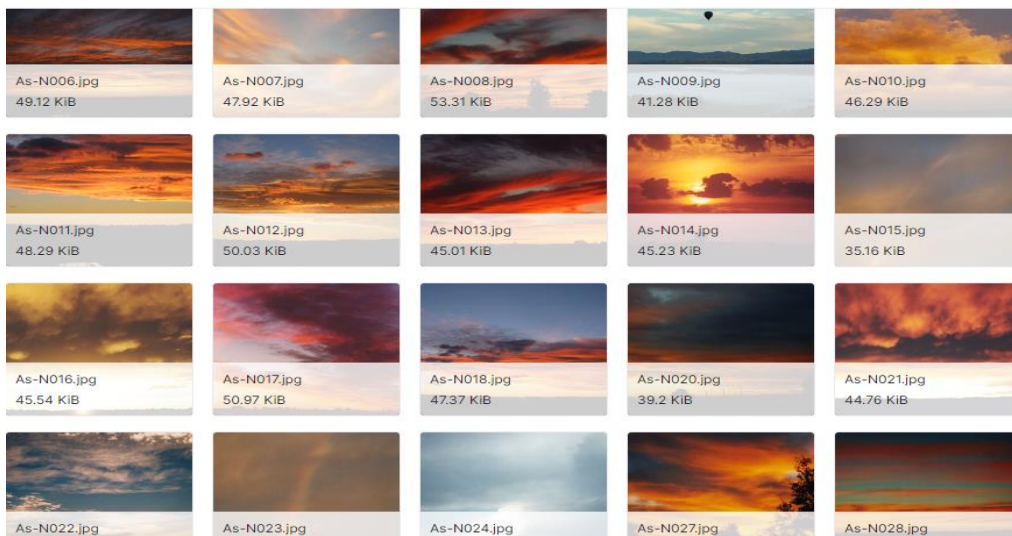
look at a cloud and see the unicorn or ice cream cones, but very few of us can look at clouds and see the approaching cold front. Fortunately, being able to predict the weather is easier than one may think. There are many different types of cloud which can be identified visually in the atmosphere. The modern classification scheme classifies clouds according to the altitude of cloud base, there being three altitude classes: low; mid-level and high. Within each altitude class additional classifications are defined based on four basic types and combinations thereof. Weather Prediction from the type of cloud Can predict a fair weather in the immediate future. They can also be an indicate of a change in weather patterns within the next 24 hours. Every image processing technique or algorithm takes an input, an image or a sequence of images and produces an output, which may be a modified image a description of the input image contents. Image Processing extracts information from images and integrates it for several applications. There are several fields in which image processing applications are relevant. Hence multi class cloud image classification for predicting weather from still image provided by the user has crucial application in real life such as environmental monitoring, accurately distinguishing weather phenomena can improve local agricultural planning etc.

CHAPTER 2

DATA SOURCES

The dataset has features 11 different classes of clouds collected from the Harvard data source, however it's real life data so any system for weather classification must be able to handle this sort of images. The dataset contains about 2543 cloud labelled images. Images are of fixed dimensions and the photos are of same sizes. Each image has only one cloud category and are saved in separate folder as of the labelled class.

As (168 files)



The dataset provides a platform for outdoor weather analysis by extracting various features for recognizing different weather conditions. The CCSN dataset contains 2543 cloud images. According to the World Meteorological Organization's genera-based classification recommendation, divided into 11 different categories :

Ac, Sc, Ns, Cu, Ci, Cc, Cb, As, Ct, Cs, St. where each category represents the type of cloud.

Representative sample images from each category are mentioned as below:

Ci = cirrus; Cs = cirrostratus; Cc = cirrocumulus; Ac = altocumulus; As = altostratus; Cu = cumulus; Cb = cumulonimbus; Ns = nimbostratus; Sc = stratocumulus; St = stratus; Ct = contrail.

All images in the dataset are fixed resolution 256×256 pixels with the JPEG format.

Ci-N011.jpg (60.68 KiB)



Cirrus

Ct-N009.jpg (19.48 KiB)



Contrail

As-N014.jpg (45.23 KiB)



Altostratus

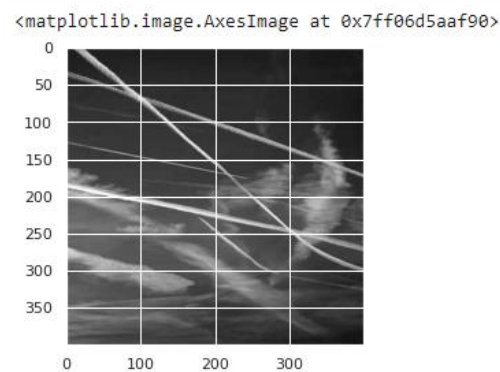
2.1 Data Pre-processing

Since images exist in different formats, i.e., natural, fake, grayscale, etc., we need to take into consideration and standardize them before feeding them into a neural network. some of the image preprocessing techniques are:

1. Grayscale conversion
2. Normalisation
3. Data Augmentation
4. Image standardization

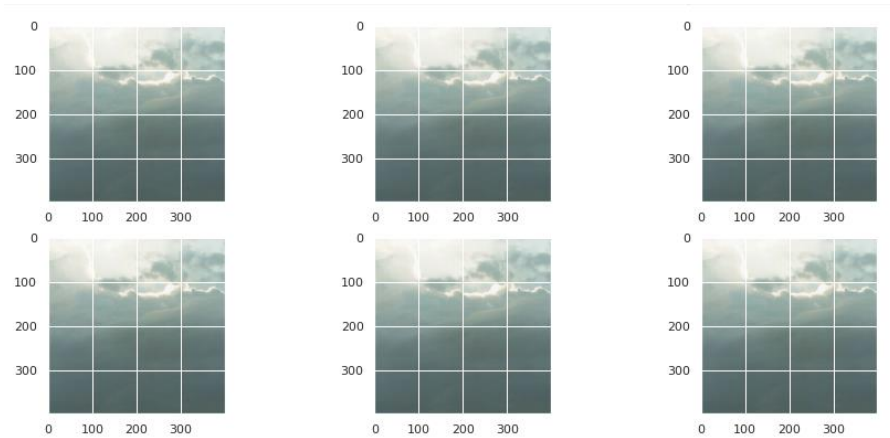
2.1.1 Grayscale conversion

Grayscale conversion means converting an image into gray representation, omitting other colors. This can be done by applying mathematical formulation on the RGB values and replacing them in the image or creating new image. Grayscale images simplifies the algorithm and reduces computational requirements.



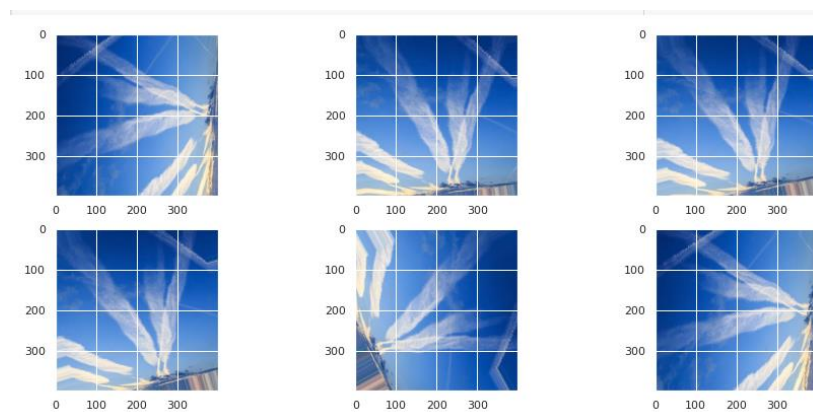
2.1.2 Flipping

Reversing the entire rows and columns of an image pixels in vertically and horizontally is called Vertical flip and horizontal flip augmentation. In this model horizontal and vertical flip has been made on training set.



2.1.3 Rotation

This process involves rotating an image by a specified degree. For this Image Data Generator is used.



2.1.4 Scaling and Standardizing images

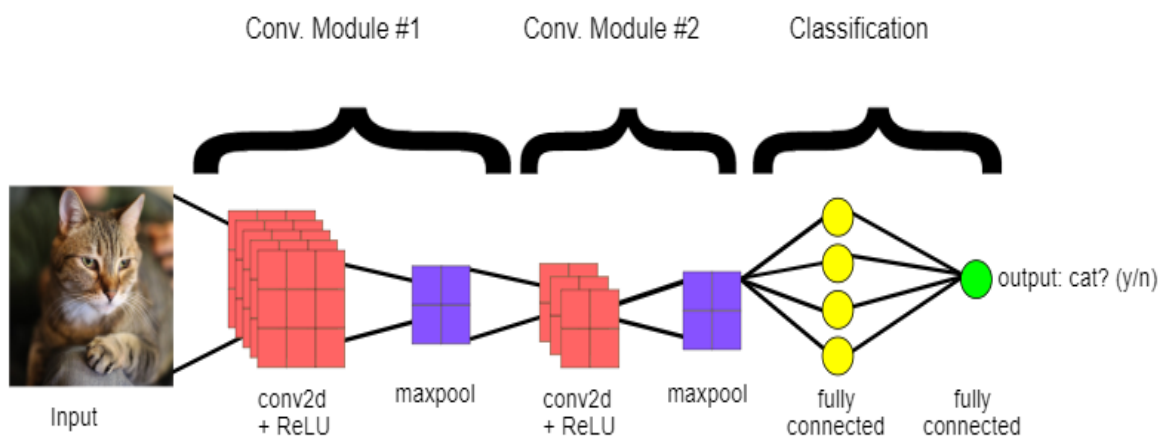
Standardization is a method that scales and preprocesses images to have similar heights and widths. It re-scales data to have a standard deviation of 1 and a mean of 0. Also referred to as *data re-scaling*, it is the process of projecting image data pixels to a predefined range. This is commonly used on different data formats to convert an image's pixel values to a typical or more familiar sense.

CHAPTER 3

FITTING MODELS TO DATA

3.1 Convolutional neural network for image classification.

Image classification is one of the most needed techniques in today's era, it is used in various domains like healthcare, business, and a lot more. Early computer vision models relied on raw pixel data as the input to the model. However, raw pixel data alone doesn't provide a sufficiently stable representation to encompass the myriad variations of an object as captured in an image. The position of the object, background behind the object, ambient lighting, camera angle, and camera focus all can produce fluctuation in raw pixel data; these differences are significant enough that they cannot be corrected for by taking weighted averages of pixel RGB values. To model objects more flexibly, classic computer vision models added new features derived from pixel data, such as colour histograms, textures, and shapes. A convolution extracts tiles of the input feature map, and applies filters to them to compute new features, producing an output feature map. During training, the CNN learns the optimal values for the filter matrices that enable it to extract meaningful features (textures, edges, shapes) from the input feature map. As the number of filters (output feature map depth) applied to the input increases. With the help of CNN model, we can build a model that classifies the given still image of a cloud to its correct type and weather pattern associated with it.



Architecture of CNN

Here the model is built by using Convolutional neural network, a CNN takes just the image's raw pixel data as input and learns how to extract these features, and ultimately infer what object they constitute.

To start with, the CNN receives an input feature map: a three-dimensional matrix where the size of the first two dimensions corresponds to the length and width of the images in pixels. The size of the third dimension is 3; corresponding to the 3 channels of a colour image: red, green, and blue. The CNN comprises a stack of modules, each of which performs three operations.

Convolution

A convolution extracts tiles of the input feature map, and applies filters to them to compute new features, producing an output feature map, or convolved feature. Convolutions are defined by two parameters:

- Size of the tiles that are extracted

- The depth of the output feature map, which corresponds to the number of filters that are applied.

Input Feature Map					Convolutional Filter		
3	5	2	8	1	1	0	0
9	7	5	4	3	1	1	0
2	0	6	1	6	0	0	1
6	3	7	9	2			
1	4	9	5	1			

Input Feature Map					Output Feature Map		
3×1	5×0	2×0	8	1	25	18	17
9×1	7×1	5×0	4	3	18	22	14
2×0	0×0	6×1	1	6	20	15	23
6	3	7	9	2			
1	4	9	5	1			

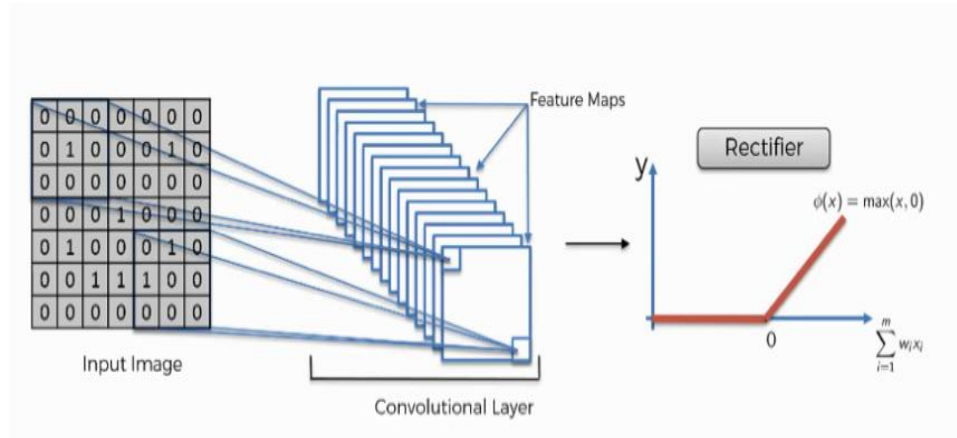
3+0+0+9+7+0+0+0+6

The 3x3 convolution is performed on the 5x5 input feature map

During a convolution, the filters (matrices the same size as the tile size) effectively slide over the input feature map's grid horizontally and vertically, one pixel at a time.

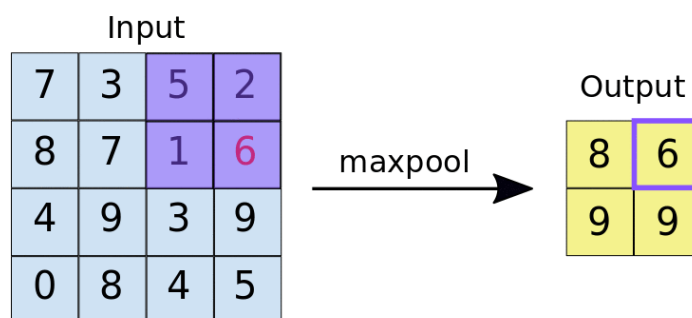
ReLU

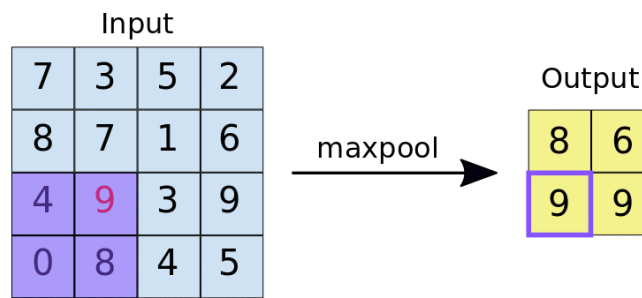
The CNN applies a Rectified Linear Unit (ReLU) transformation to the convolved feature, in order to introduce nonlinearity into the model.



Pooling

After ReLU comes a pooling step, in which the CNN down samples the convolved feature reducing the number of dimensions of the feature map, while still preserving the most critical feature information. A common algorithm used for this process is called max pooling.





Max pooling performed over a 4x4 feature map with a 2x2 filter and stride of 2.

Max pooling operates in a similar fashion to convolution. We slide over the feature map and extract tiles of a specified size. For each tile, the maximum value is output to a new feature map, and all other values are discarded. Max pooling operations take two parameters:

- **Size** of the max-pooling filter (typically 2x2 pixels)
- **Stride**: the distance, in pixels, separating each extracted tile. Unlike with convolution, where filters slide over the feature map pixel by pixel, in max pooling, the stride determines the locations where each tile is extracted. For a 2x2 filter, a stride of 2 specifies that the max pooling operation will extract all nonoverlapping 2x2 tiles from the feature map

At the end of a convolutional neural network are one or more fully connected layers. Their job is to perform classification based on the features extracted by the convolutions.

Using CNN as a deep learning model with an activation function as ReLU and max pooling to reduce the size of the feature map, the model is fitted for 85 epochs, and the training accuracy turns out to be 44.05%.


```

model = Sequential([
    Conv2D(16, (3, 3), activation='relu', input_shape=(img_height, img_width, 3)), MaxPooling2D(2, 2),
    Conv2D(32, (3, 3), activation='relu'), MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation='relu'),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Conv2D(128, (3, 3), activation='relu'),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Conv2D(256, (3, 3), activation='relu'),
    Conv2D(256, (3, 3), activation='relu'),
    Conv2D(256, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(512, activation='relu'),
    Dense(11, activation='softmax')
])
model.summary()

```

Building sequential model

```

history = model.fit_generator(train_generator,
                             epochs=85,
                             verbose=3,
                             validation_data=validation_generator,
                             callbacks = [best_model]
                             )

```

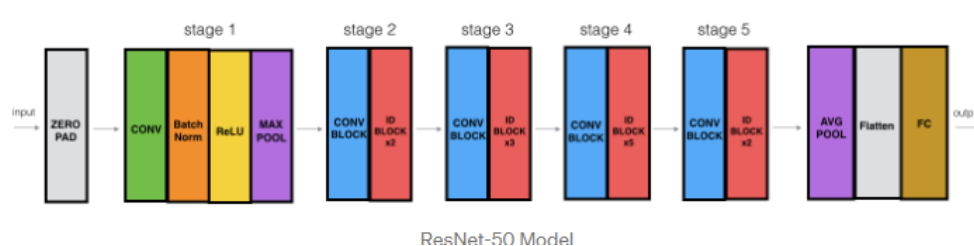
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
 Epoch 2/85
 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
 Epoch 3/85
 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
 Epoch 4/85
 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
 Epoch 5/85
 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
 Epoch 6/85
 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
 Epoch 7/85
 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
 Epoch 8/85
 WARNING:tensorflow:Can save best model only with val_acc available, skipping.

Fitting data into CNN

3.2 Image Classification with ResNet50 Model

ResNet-50 is a convolutional neural network that is 50 layers deep (48 Convolution layers along with 1 MaxPool and 1 Average Pool layer). A residual neural network (ResNet) is an artificial neural network (ANN) of a kind that stacks residual blocks on top of each other to form a network.

The ResNet-50 model consists of 5 stages each with a convolution and Identity block. Each convolution block has 3 convolution layers and each identity block also has 3 convolution layers. The ResNet-50 has over 23 million trainable parameters. ResNet first introduced the concept of skip connection that They allow the model to learn an identity function which ensures that the higher layer will perform at least as good as the lower layer.



```
[27] resnet = ResNet50(
    input_shape = IMAGE_SIZE + [3], # Making the image into 3 Channel, so concating 3.
    weights = 'imagenet', # Default weights.
    include_top = False #
)

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94773248/94765736 [=====] - 0s 0us/step
94781440/94765736 [=====] - 0s 0us/step

resnet.summary()
conv1_conv (Conv2D) (None, 112, 112, 64 9472 ['conv1_pad[0][0]']
conv1_bn (BatchNormalization) (None, 112, 112, 64 256 ['conv1_conv[0][0]']
conv1_relu (Activation) (None, 112, 112, 64 0 ['conv1_bn[0][0]']
pool1_pad (ZeroPadding2D) (None, 114, 114, 64 0 ['conv1_relu[0][0]']
pool1_pool (MaxPooling2D) (None, 56, 56, 64) 0 ['pool1_pad[0][0]']
conv2_block1_1_conv (Conv2D) (None, 56, 56, 64) 4160 ['pool1_pool[0][0]']
conv2_block1_1_bn (BatchNormal (None, 56, 56, 64) 256 ['conv2_block1_1_conv[0][0]']
conv2_block1_1_relu (Activatio (None, 56, 56, 64) 0 ['conv2_block1_1_bn[0][0]']
conv2_block1_2_conv (Conv2D) (None, 56, 56, 64) 36928 ['conv2_block1_1_relu[0][0]']
conv2_block1_2_bn (BatchNormal (None, 56, 56, 64) 256 ['conv2_block1_2_conv[0][0]']
conv2_block1_2_relu (Activatio (None, 56, 56, 64) 0 ['conv2_block1_2_bn[0][0]']
conv2_block1_3_conv (Conv2D) (None, 56, 56, 256) 16640 ['pool1_pool[0][0]']
conv2_block1_3_relu (Conv2D) (None, 56, 56, 256) 16640 ['conv2_block1_2_relu[0][0]']
```

Building Resnet50 model

```
[49] history = model.fit_generator(
    training_set,
    validation_data = test_set,
    epochs = 100,
    steps_per_epoch = 120,
    validation_steps = len(test_set)
)

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please u

Epoch 1/100
81/120 [=====>.....] - ETA: 2:59 - loss: 2.7348 - accuracy: 0.2570WARNING:tensorflow:Your input ran out of data; interrupting training. Make s
120/120 [=====] - 417s 3s/step - loss: 2.7348 - accuracy: 0.2570 - val_loss: 2.1726 - val_accuracy: 0.3039

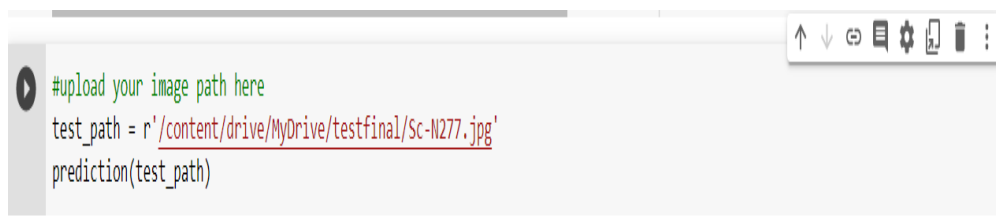
print("Accuracy of the model on train data is {:.2f}%".format(history.history["accuracy"][-1]*100))
Accuracy of the model on train data is 25.70%
```

Fitting data into Resnet50

CHAPTER 4

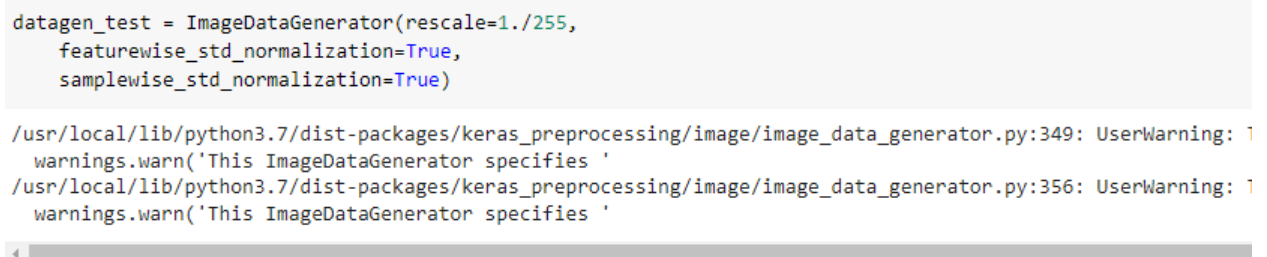
PREDICTING NEW DATA

Once the model has been trained and validated, the model is now feeded with unknown set of samples to check the accuracy



```
#upload your image path here
test_path = r'/content/drive/MyDrive/testfinal/Sc-N277.jpg'
prediction(test_path)
```

Loading the test data path



```
datagen_test = ImageDataGenerator(rescale=1./255,
    featurewise_std_normalization=True,
    samplewise_std_normalization=True)
```

```
/usr/local/lib/python3.7/dist-packages/keras_preprocessing/image/image_data_generator.py:349: UserWarning: 1
warnings.warn('This ImageDataGenerator specifies '
/usr/local/lib/python3.7/dist-packages/keras_preprocessing/image/image_data_generator.py:356: UserWarning: 1
warnings.warn('This ImageDataGenerator specifies ')
```

Standardization and normalization in test data

The test image is passed on to the function where some of the preprocessing happens and the image is rescaled to 256*256 pixels and finally the cloud type and its weather condition is showed

4.1 Prediction with CNN model

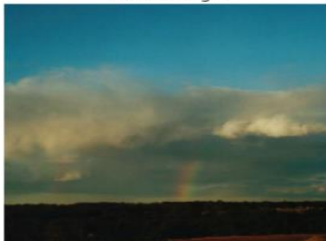
```
def prediction(test_path):  
    img = image.load_img(test_path , target_size = (256, 256))  
    img = image.img_to_array(img, dtype=np.uint8)  
    img = np.array(img)/255.0  
  
    plt.title('Cloud Image')  
    plt.axis('off')  
    plt.imshow(img.squeeze())  
  
    predict = model.predict(img[np.newaxis , ...])  
    predicted_class = labels[np.argmax(predict[0] , axis = -1)]  
  
    print('Prediction Value: ' , np.max(predict[0] , axis = -1))  
  
    print("Classified:",predicted_class)
```

```
test_path = r'/content/drive/MyDrive/testfinal/Ct-N116.jpg'  
prediction(test_path)  
  
Prediction Value: 0.90071796  
Classified: Ct  
Cloud Type: contrail  
short-lived contrail indicates low-humidity air at high altitude, a  
Cloud Image
```



The model correctly classifies the cloud type.

```
Prediction Value: 0.51014936  
Classified: Sc  
Cloud Type: stratocumulus  
foretell bad weather  
Cloud Image
```



As we can see that, the model misclassifies the cloud type and outputs the wrong class label. Hence the prediction value is low

4.2 Prediction with Resnet50 model

```
test_path = r'/content/drive/MyDrive/testfinal/St-N004.jpg'  
prediction(test_path)
```

Prediction Value: 0.32074833
Classified: St
Cloud Type: stratus
These clouds combine in a dense gray overcast that promises light to heavy rain



Although the model predicted the class label of the given image correctly, due to its low accuracy on training data the model's prediction value is very low

CHAPTER 5

CONCLUSION

I have evaluated the deep learning model for cloud classification for predicting weather using a convolutional neural network. In my project, I have exploited and evaluated the application of CNN approach to identify the different type of clouds. I have used about 2543 images on training and 245 images for testing in CNN architecture. This model is trained only upon eleven categories of cloud. The model cannot predict new type of cloud other than the given trained cloud labels. The model's prediction on the weather by predicting the type of cloud will not be much accurate to real time because of fact scud clouds that often bring misinterpretation of weather Sometimes the model could predict incorrect class labels.

In the future, we will continue our research to develop new architectures for detecting and segmenting the clouds on a larger database and to use computer vision to dynamically identify different types of clouds at the same time.

CHAPTER 6

REFERENCES

- Jaswal, Deepika & Vishvanathan, Sowmya & Kp, Soman. (2014). Image Classification Using Convolutional Neural Networks. International Journal of Scientific and Engineering Research. 5. 1661-1668. 10.14299/ijser.2014.06.002.
- Xin, M., Wang, Y. Research on image classification model based on deep convolution neural network. *J Image Video Proc.* **2019**, 40 (2019).
- Source for cloud images [Online]dataverse.harvard.edu
- An introduction to convolutional neural networks available at: white.stanford.edu/teach/index.php/An_Introduction_to_Convolutional_Neural_Networks
- Image Classification Using Convolutional Neural Networks With Multi-stage Feature Junho Yim, Jeongwoo Ju, Heechul Jung, and Junmo Kim