Atmospheric Contrail Detection with a Deep Learning Algorithm

Nasir Siddiqui
University of Minnesota - Morris, siddi065@morris.umn.edu

Atmospheric contrail detection with a deep learning algorithm

Nasir Siddiqui, Sylke Boyd

University of Minnesota Morris

April 2020

Abstract

Aircraft contrail emission is widely believed to be a contributing factor to global climate change. We have used machine learning techniques on images containing contrails in hopes of being able to identify those which contain contrails and those that do not. The developed algorithm processes data on contrail characteristics as captured by long-term image records. Images collected by the United States Deparment of Energy's Atmospheric Radiation Management user facility(ARM) were used to train a deep convolutional neural network for the purpose of this contrail classification. The neural network model was trained with 1600 images taken by the Total Sky Imager(TSI) from March 2017 and achieved an accuracy of 97.5% on the training set of images and an accuracy of 98.5% on the validation set.

1 Introduction

Atmospheric contrails, also known as vapor trails are best visually identified as elongated, white streaks in the sky. The study of contrails and their effects is an active area of research in atmospheric science with studies theorizing that the presence of contrails contributes to the earth's greenhouse effect and ultimately climate change. This mainly occurs due to radiative forcing caused by the the cirrus clouds that arise as a result of contrail formation. In the context of air traffic, the contribution of contrail cirrus clouds to radiative forcing surpasses even that of emitted carbon dioxide's[1]. Contrails are comprised of ice crystals formed when aircraft exhaust condenses in the cold temperatures of the upper atmosphere. Depending on humidity conditions, these ice crystals can grow and thus persist for up to several hours. As time progresses, they can undergo changes in their microphysical properties and at some point become indistinguishable from regular cirrus clouds[2] upon visual inspection.

The growth and advancement of machine learning in recent times has been conducive to significant progress in several fields of applied and natural sciences where huge volumes of data specific to many different situations are now readily available. In this paper, we discuss the application of machine learning to distinguish contrail cirrus clouds from regular cirrus clouds using a convolutional neural network , a statistical model that has become especially popular for general image classification purposes. Here, we have used images belonging to the U.S Department of Energy's Atmospheric Radiation Management user facility(ARM) captured by Total Sky Imagers(TSI) at the Southern Great Plains facility in 2017 [3].

We have used Python 3 for this implementation with the aid of the packages Numpy, Pandas, Matplotlib, opencv2 and the deep learning library Tensorflow.

2 Procedure

2.1 Data

The images used were captured by a TSI unit. The unit itself is comprised of a camera placed vertically above a convex mirror. This allows visualization of the sky from zenith to horizon. Images are captured every 30 seconds and are in JPEG format[4]. In general, the process of locating contrails in these images is time consuming and too subjective if only done by eye. However a computer algorithm if trained well enough could automate this process with a lower error rate. An algorithm suited to this task is a convolutional neural network as it excels in its ability to extract important high level features and abstractions in images. To train this neural network, 1600 images for the month of March 2017 were collected and used on the bases of visually high variability in cloud formation and precipitation patterns and most frequent contrail occurrences. Below is an example of a randomly selected TSI image.

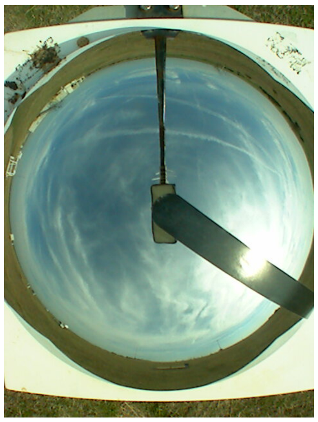


Fig 1. A randomly selected TSI image containing contrails

2.2 Image selection

Image classification is a supervised process. In other words, images with categorical labels distinguishing one category from the other are required for training the algorithm. All images were sifted through and assigned labels by eyeballing the presence of contrails. It should be reiterated that such a method is inherently prone to human error and needs to be done diligently. Oftentimes it was difficult to either locate contrails because some had low visibility or were obscured by other clouds, or because it was difficult to distinguish them from other cirrus like clouds. Another difficulty that was encountered was visual distortion introduced by the TSI"s convex mirror. Observed objects in images towards the edge of the mirror become elongated along the periphery and appeared to shrink in the radial direction as they approached the center.

While all of this was problematic, it was only so in the process of visually locating contrails in the images. Contrail formation also depends on surrounding atmospheric conditions. If just a single contrail is credibly located in an image, then within the image is also contained visual properties reflective of the physical conditions associated with the contrail's formation. Thus, if we were to train a model with images containing just a few well defined contrails and, the images a few moments before they were formed and a few moments after they dissipated, then our model would not only be trained to detect the geometries of contrails, but also other conditions in the atmosphere that influence their presence. This is the approach that was undertaken. Images that contained a contrail were to be given a label of 1, and those that did not were to be given a label of 0. In the end, 1600 images were set aside for training the model with equal numbers of images corresponding to 1 and 0. Shown below are some images that were sorted accordingly.

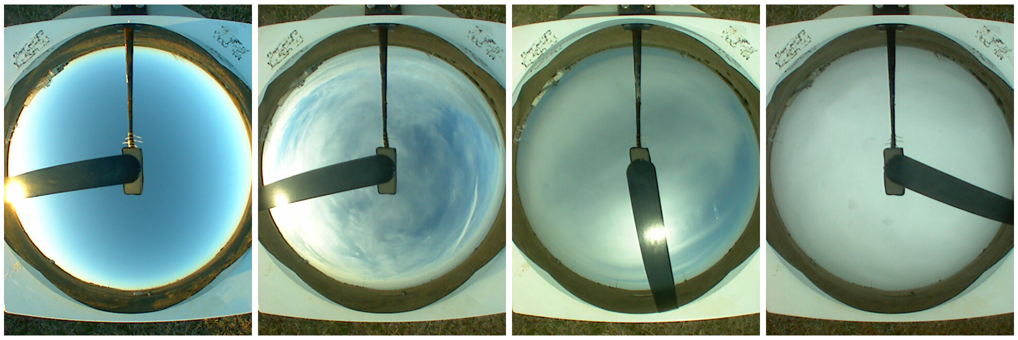


Fig 2. Selected images with no visually perceived contrails to be given labels of 0

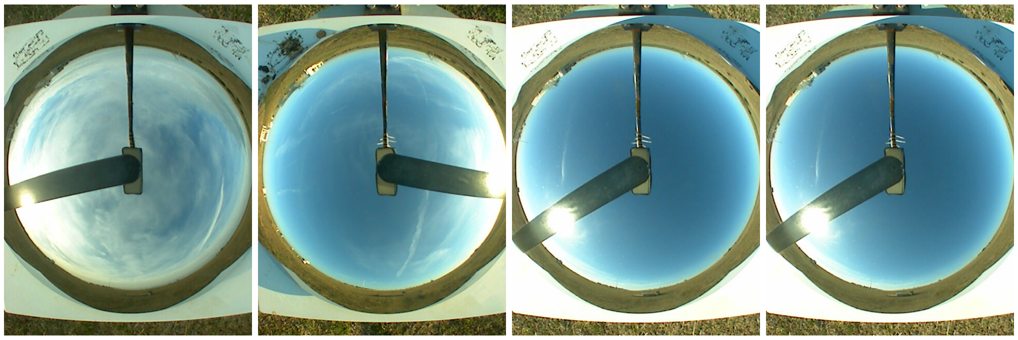


Fig 3. Selected images with visually perceived contrails to be given labels of 1

2.3 Image preprocessing

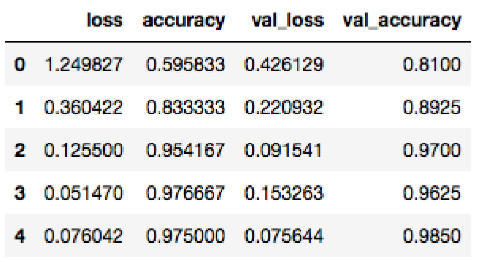
After having organized the labelled data, they were then imported into Python's Jupyter Notebook IDE as three dimensional tensors. In this step, all contrail containing image tensors were given the labels of 1 while all non contrail containing images tensors were given labels of 0. The number of contrail versus no contrail images are both 800. Using an 75% to 25% ratio, these image tensors along with their respective labels were split into training and validation sets and saved as a Numpy array[1] to be used to train and test the model.

2.4 Model Performance and Evaluation

A deep convolutional neural network was used on the Numpy array created in the previous step and trained for 5 iterations. Tensorflow's high level API package Keras was used for constructing the model. Since this is a binary classification problem, a binary cross entropy loss function is used, and the optimizer function is adaptive momentum. Shown below are some of the resultant metrics[2] for the process where "loss" and "accuracy" are the loss and accuracy on the training set respectively, and "val loss" and "val accuracy" are the loss and accuracy on the validation set.

[1]Relevant code for this step is listed in Appendix I

[2]A loss function is a measure of how much the model's outputs differ from the actual data values. In this case, the binary cross entropy loss function measures how "badly" the model performs in predicting image labels against their true labels. The adaptive momentum optimizer function minimizes this loss function by constantly tuning parameters within the structure of the neural network. Accuracy is the ratio of the number of the model's correct image label classifications to the actual image labels themselves.



	loss	accuracy	val_loss	val_accuracy
0	1.249827	0.595833	0.426129	0.8100
1	0.360422	0.833333	0.220932	0.8925
2	0.125500	0.954167	0.091541	0.9700
3	0.051470	0.976667	0.153263	0.9625
4	0.076042	0.975000	0.075644	0.9850

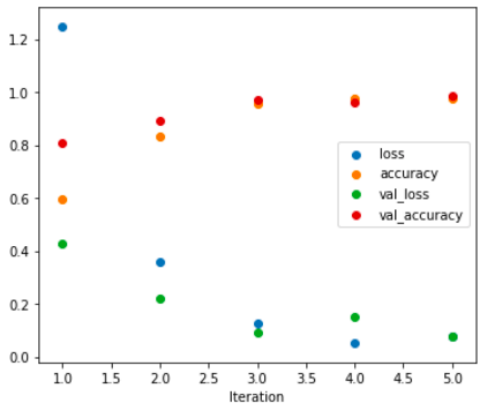
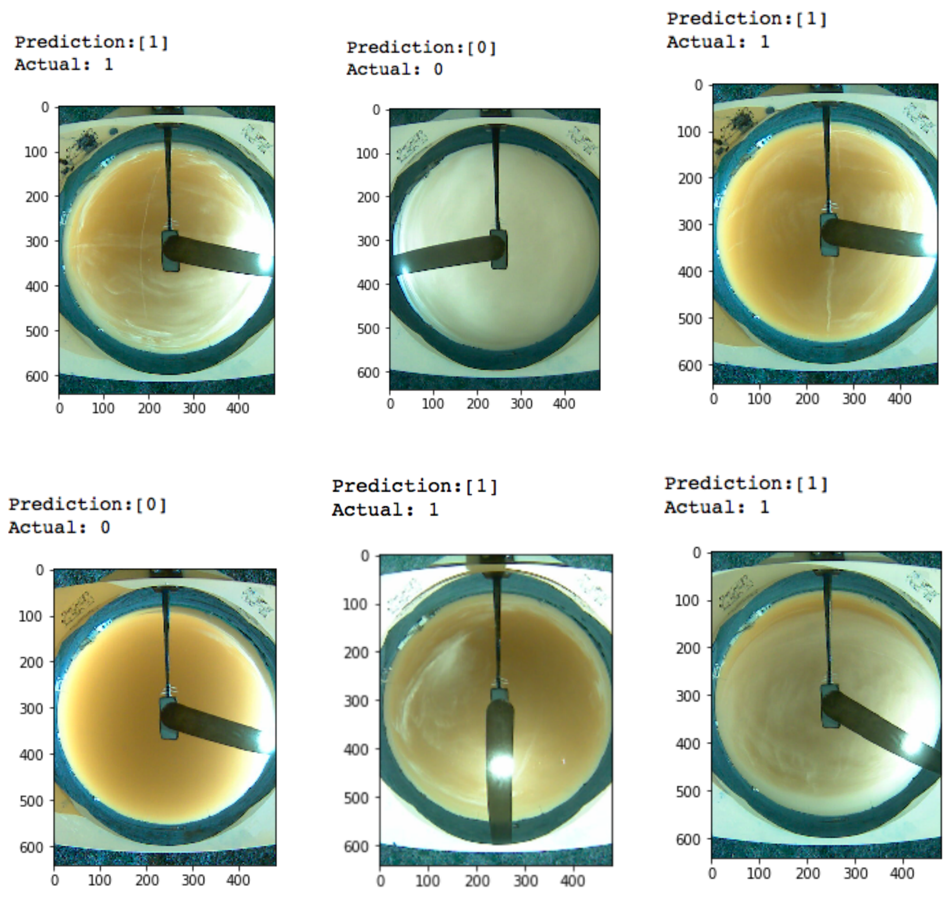


Fig 4. Model performance metrics after 5 training iterations where in the table, 0 is the index of the first iteration and 4 is the index of the last. The graph illustrates the results of the table.

We see that our model performs with an accuracy of 97.5% on the training set and 98.5% on the validation set. This means that out of the 1200 images that were in the training set, the model gives 1170 correct predictions on as to whether an image corresponds to the label of 1 or 0. But perhaps more important are the results on the validation set since it contains images that the model had not "seen" before. It contained 400 images out of which it made 394 correct classifications. Shown below are some selected images[3] from this set along with their original labels and the model's predicted labels.



```
Prediction:[1]
Actual: 1
```

```
Prediction:[0]
Actual: 0
```

```
Prediction:[1]
Actual: 1
```

```
Prediction:[0]
Actual: 0
```

```
Prediction:[1]
Actual: 1
```

```
Prediction:[1]
Actual: 1
```

[3]When viewing and constructing the images within Python from their respective tensors there is an unwanted colorization that occurs for reasons that were unknown to us. This however did not seem to affect the final performance in any adverse way.

Fig 5. Images with their original labels along with the model's predicted labels appended

2.5 Discussion of results

It was found that contrail predictions often occurred under sky conditions that also showed other cirrus features. This is true in the training set as well as in the validation set. In Fig. 5 the first two correct predictions of label 1, i.e the first and third pictures going from top left to right, the presence of contrails is visually evident along with the presence of other cirrus clouds in the background. For these cases, it is most likely that the model uses the linear characteristic of contrails to distinguish them from regular cirrus clouds. The cases for the fifth and sixth pictures are more interesting because the presence of cirrus like clouds seems more evident than that of linear contrails. The most likely is that the contrails dissipated into cirrus like clouds, and as stated in section 2.2 images that led up to the formation of contrails and those that concluded with their dissipation were also used for training the model. Even before training it was seen that contrails that dissipated generally began to resemble cirrus like clouds. Given the high accuracy of the model on the set that contained these images, it likely to have been capable of distinguishing contrail cirri from regular cirrus on the basis of more subtle and abstract features such as differences in optical properties, shapes, drifting patterns and so forth. These details are captured in the properties of the images that would be next to impossible to pick out if solely equipped with the human eye.

3 Conclusion

We have seen that a convolutional neural network proves to be a very effective tool for the purpose of distinguishing contrail cirri from regular cirri on the TSI images. A high accuracy of 98.5% on the validation set means that the model was able to learn important features of contrails from their visual properties alone such as their linearity and other sets of abstract features that set them apart from regular cirrus clouds. This model can now used for a variety of practical purposes to study contrails. Potential applications include running it on images for other months of data and investigating the frequencies of contrail occurrences which will give further insight into air traffic trends or how physical conditions such as temperature and humidity vary seasonally and how they might affect contrail formation.

References

[1] Bock, L. and Burkhardt, U.: Contrail cirrus radiative forcing for future air traffic, Atmos. Chem. Phys., 19, 8163–8174, https://doi.org/10.5194/acp-19-8163-2019, 2019

[2] Gruber, S., Unterstrasser, S., Bechtold, J., Vogel, H., Jung, M., Pak, H., and Vogel, B.: Contrails and their impact on shortwave radiation and photovoltaic power production – a regional model study, Atmos. Chem. Phys., 18, 6393–6411, https://doi.org/10.5194/acp-18-6393-2018, 2018.

[3] Atmospheric Radiation Measurement (ARM): Climate Research Facility. updated hourly. Total Sky Imager (TSISKYIMAGE). Southern Great Plains (SGP) Central Facility, Lamont, OK (C1), compiled by: Morris, V.

[4] Boyd, S., Sorenson, S., Richard, S., King, M., and Greenslit, M.: Analysis algorithm for sky type and ice halo recognition in all-sky images, Atmos. Meas. Tech., 12, 4241–4259, https://doi.org/10.5194/amt-12-4241-2019, 2019.

4 Appendix 1

```
In [1]: #WE BEGIN BY ORGANIZING ALL OF OUR DATA INTO THE FORM WE NEED.
        #WE SELECT 800 RANDOM NO CONTRAILS, AND 800 RANDOM CONTRAILS FROM THE THEIR RESPECTIVE DIRECTOR

        #BEGIN BY IMPORTING SOME RELEVANT LIBRARIES

        import pandas as pd
        import numpy as np
        import cv2

In [2]: import matplotlib.pyplot as plt

In [3]: import os, os.path
        import glob

        #WE IMPORT THE DIRECTORIES CONTAINING THE IMAGES THAT HAVE BEEN SELECTED

In [4]: training_dir_no_contrails = "/Users/nasirsiddiqui/Desktop/training_data/NO_CONTRAILS_FILTERED"

In [5]: training_path_no_contrails = os.path.join(training_dir_no_contrails , "*g")

In [6]: training_files_no_contrails = glob.glob(training_path_no_contrails)

In [7]: #FIRST WE LOAD THE FOLDER CONTAINING NO CONTRAILS; THERE ARE 872 OF THEM.

        data = []

        for file in training_files_no_contrails:
            img = cv2.imread(file)
            img = cv2.resize(img, (480,640))
            data.append(img)


In [8]: len(data)

In [9]: #LET'S PICK 800 RANDOM SAMPLES OF NO CONTRAILS

       np.random.seed(42)

        some_random_numbers = np.random.randint(1 , 800 , (800,))

        random_no_contrails = []
        for i in some_random_numbers:
            random_no_contrails.append(data[i])


In [14]: #NOW WE ASSIGN LABELS. LET 0 DENOTE "NO CONTRAIL". WE NEED 800 OF THEM.


         labels = []
         for i in range(800):
             decision = 0
             labels.append(decision)
         labels = np.asarray(labels)

In [15]: #COMBINE THEM INTO AN OBJECT
```

```
        no_contrail_w_label = (random_no_contrails , labels)

In [24]: #NOW REPEAT PROCEDURE FOR THE CONTRAIL IMAGES

         training_dir_contrails = "/Users/nasirsiddiqui/Desktop/training_data/CONTRAIL"
         training_path_contrails = os.path.join(training_dir_contrails , "*g")
         training_files_contrails = glob.glob(training_path_contrails)

In [25]: data_contrails = []
         for file in training_files_contrails:
             img = cv2.imread(file)
             img = cv2.resize(img,(480,640))
             data_contrails.append(img)

In [26]: random_contrails = []
         for i in some_random_numbers:
             random_contrails.append(data_contrails[i])

In [28]: len(random_contrails)

Out[28]: 800

In [29]: #LET 1 DENOTE THE LABEL FOR A CONTRAIL

         labels = []
         for i in range(800):
             labels.append(1)
         len(labels)
         labels = np.asarray(labels)

In [30]: contrails_with_labels = (random_contrails , labels)

In [46]: from sklearn.model_selection import train_test_split

In [47]: #NOW WE CONSTRUCT THE TRAINING AND TEST SETS. THE TRAINING DATA WILL BE USED TO TRAIN THE MODE
         #TEST SET WILL CONTAIN DATA THAT THE MODEL HAS NOT SEEN BEFORE SO WE CAN FAIRLY EVALUATE ITS P

In [64]: relevant_image_arrays = np.concatenate([contrails_with_labels[0] , no_contrail_w_label[0]])

In [66]: relevant_labels = np.concatenate([contrails_with_labels[1] , no_contrail_w_label[1]])

In [74]: X = relevant_image_arrays
         y = relevant_labels

In [75]: #WE'LL DO A 75% - 25% SPLIT; THIS MEANS 75% OF ALL OUR DATA WILL BE USED TO TRAIN THE MODEL, A
         #USED TO VALIDATE THE MODEL. SINCE THERE ARE 1600 IMAGES IN TOTAL, 1200 WILL BE USED TO TRAIN
         #400 IMAGES WILL BE USED FOR VALIDATION.

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

In [76]: #COMBINE THE TRAINING IMAGES WITH THEIR RESPECTIVE LABELS AND THE TEST IMAGES AND THEIR RESPEC
         #INTO A SINGLE OBJECT


         data_final_hopefully = (X_train , y_train) , (X_test , y_test)

In [83]: #SAVE THIS. THIS IS THE FORM WE NEED OUR DATA TO BE IN FOR THE NEXT TIME WHERE WE ACTUALLY BUI

         np.save('convnet_trial_1' , data_final_hopefully , allow_pickle= True)
```