# DroneNet - An Image Classifier for UAV

**Or Argaman**
Tel-Aviv University
orargaman@gmail.com

**Arie Olshanezky**
Tel-Aviv University
arikolsh@gmail.com

## 1 Introduction and Discussion

Over the last couple of years, Unmanned Aerial Vehicles (UAV) have gained popularity for both public and commercial use. Along with the rise of popularity, many governments have enacted laws preventing the use of personal UAVs in a variety of zones such as airfields, military facilities, neighbourhoods and above crowds of people.

In this work we present an image classification model, that is based on images taken from a UAV attached camera. This model classifies whether the UAV is flying over an area that is prohibited by law or whether it is flying in accordance with the law. We base our classification in accordance with the Israeli law, which states that UAVs should keep a distance of at least 250 meters from neighbourhoods, public buildings, and not near crowds of people. This problem can be mostly solved using GPS signals along with rigorous mapping of zones. Nevertheless, image recognition can bridge the gap where GPS and mapping fails. For example, using image recognition we enable a fail-over mechanism in case UAVs lose GPS signals, or a safety net in case a drone operator falsely chooses to fly over non-flight zones.

Consumer drones capture images that differ significantly from common satellite imagery such as Spacenet (Etten et al., 2018) and from standard aerial imagery datasets such as Inria (Maggiori et al., 2017). While satellite and aerial photography present images that are orthogonal to the earth and taken from a high view point (which gives a 2D perspective), UAVs point of view varies in different angles and heights giving a 3D perspective to the images it captures. We use this task to examine and compare a new dataset built on GoogleEarth 3D modeled images which gives a high resemblance to real life UAV images. The reasoning for this is that there are no publicly available datasets which contain UAV images from various heights and various point of views along with a big variety of image features. Therefore, collecting such with real UAVs is a very time consuming task. From exploring online sources, we found out that most UAV images do not fully resemble the live feed you see from a UAV while operating it. This is because they use an angle that is parallel to earth for image capture which is not the angle the UAVs fly in. Also, most of the images are edited as they are published in photography sites. We believe proving the 3D modeled GoogleEarth images can be a replacement for real UAV images. Hopefully, this will prove to be useful for more complicated tasks such as Object Segmentation, as they have few advantages over collecting real UAV images:

- Much less time consuming, as collecting an image can simply be done by snipping an image from GoogleEarth and does not include going anywhere and flying a UAV or scraping the web and then filtering out images that truly resemble a UAV live feed image.
- Images Can be collected using tools such as Mechanical Turk.
- The variety of places where images can be collected is much bigger as Google published 3D modeling of hundreds of cities with their surrounding around the globe. This also allows capturing images from areas where UAVs are prohibited.
- Does not require owning and flying a UAV which may be expensive and requires knowledge of flying one.

## 2 Method - Deep Learning UAV Flight Zone Classification

This section outlines the process of developing an efficient convolutional neural network suitable for classifying UAV images as Flight Zone areas.

### 2.1 Dataset

For the task at hand, we used three different datasets.

- 300 Manually collected GoogleEarth images varying in location, perspective and altitude. For a sample see Figure 1.
- 300 Satellite images taken from the RESISC45 (Cheng et al., 2017). For a sample see Figure 2.
- 540 Manually scraped UAV images from the web. For a sample see Figure 3.

The Data was labeled manually into 2 classes, "NonFlightZone" and "FlightZone". We made sure the data was well balanced between the classes to avoid bias.

### 2.2 Data Processing

Before passing the data into our model we performed the following transformations on each image:

- **Random Horizontal Flip** - Since our data size set is very small, we chose to augment the data to give the model more images to train on. Data augmentation can be done using multiple methods. Nevertheless, we can't use methods that might remove vital information from the image. A bad method for example, would have random cropping, since that could have removed buildings from certain images which clearly characterize the area as a "NonFlightZone".
- **Resize** - Resize image to 3x224x224 pixels, which are the dimensions the pre-trained models we use expect to get.
- **Normalization** - Normalizing the images with Imagenet's Mean and STD.

### 2.3 Model

For the task of training, we fine tune the ResNet model (He et al., 2015) which was pre-trained on the huge ImageNet dataset. We do this by first freezing the weights for all the layers except the last fully connected layer which we then replace by

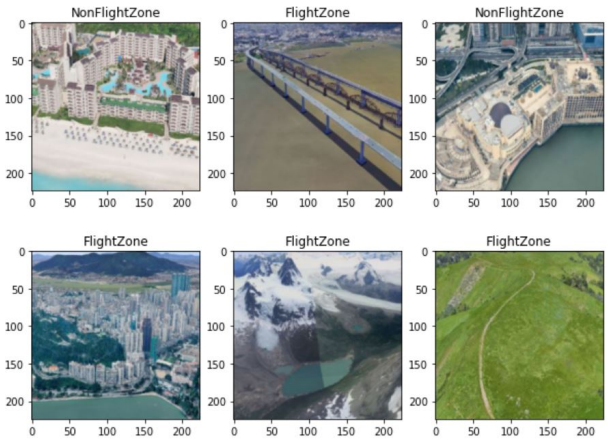


Figure 1: GoogleEarth Perspective Images

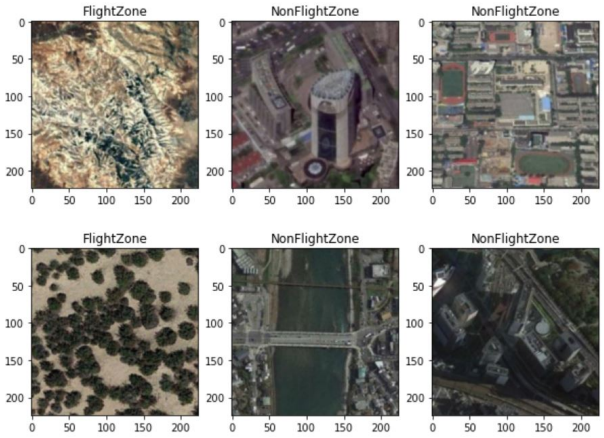


Figure 2: RESISC45 Satellite Images

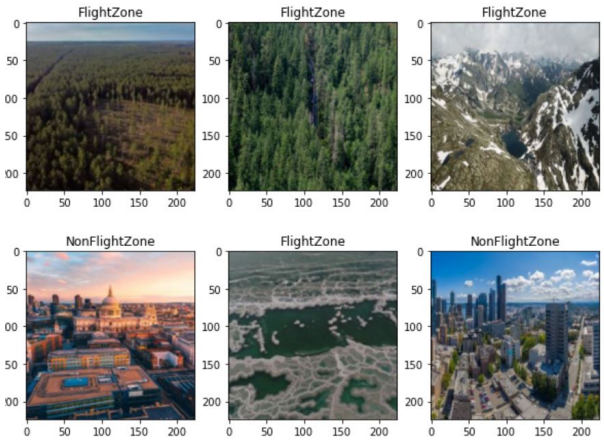


Figure 3: UAV Images

our own feed forward layer. Our feed forward layer consists of a Linear layer, followed by a RELU activation function, then a Dropout layer. The output of the Dropout layer is then passed into another linear layer, followed by LogSoftmax layer. The Softmax layer eventually, classifies our input image into two distinct classes, "FlightZone" and "Non-FlightZone". For an illustration of the structure of our model see Figure 4. We used grid search over our hyper parameters to best optimize our model, this led us to use a dropout probability of 0.2 and a learning rate of 0.001. In addition, we used Adam optimization for gradient descent and Log-Likelihood loss.

### 2.4 Training

Training a CNN for the required classification task requires collecting a suitable dataset. To the best of our knowledge there is no widely used and publicly available dataset for UAV taken images. In comparison, GoogleEarth images exist in abundance. Our premise is that aerial images taken from GoogleEarth bear similar characteristics to images taken by UAVs. Hence we employ transfer learning for our pre-trained model on GoogleEarth images, so that the model would be able to learn said characteristics. Then we feed the output into a custom feed forward layer described in the previous section. Afterwards, we evaluate the accuracy, loss and F1 score in the validation and test phases, using the manually obtained UAV images. Our overall objective is to explore the performance-accuracy trade-offs between two main approaches. One approach consists of training on perspective images collected manually from GoogleEarth. The other approach consists of training on Satellite images from a publicly available dataset. [1]

## 3 Experiments and Results

In the following section, we outline the experiments we have made with our model, exploring different training sets and eventually different CNN architectures to achieve the best result with our limited UAV dataset. We will present the technical results, graphs. Based on said results, we will evaluate conclusions.

[1]Our code, model and links to the dataset blob: https://github.com/arolshan/cnn$_drone_net$

### 3.1 Experiments

We first presented a baseline of training over the UAV dataset. We then experimented with two approaches as discussed in previous sections. One approach was to train using the RESISC45 SAT images, the other using manually extracted perspective images from GoogleEarth. In Table 1 we show the results between the two methods. We can see a slightly advantage to the perspective images.

We then experimented with various architectures of our model, all trained with the GoogleEarth

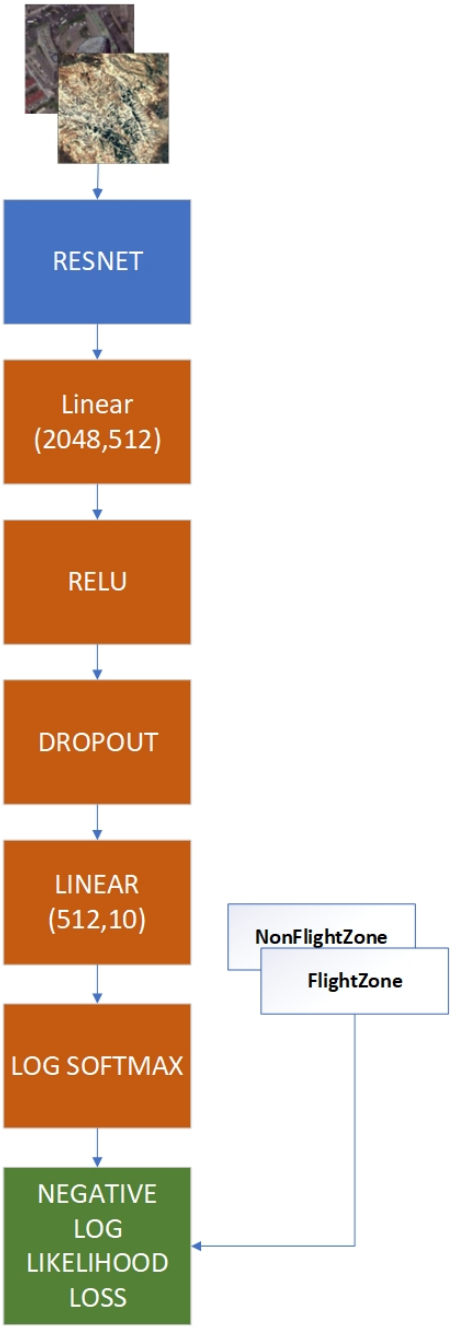


Figure 4: Model Architecture

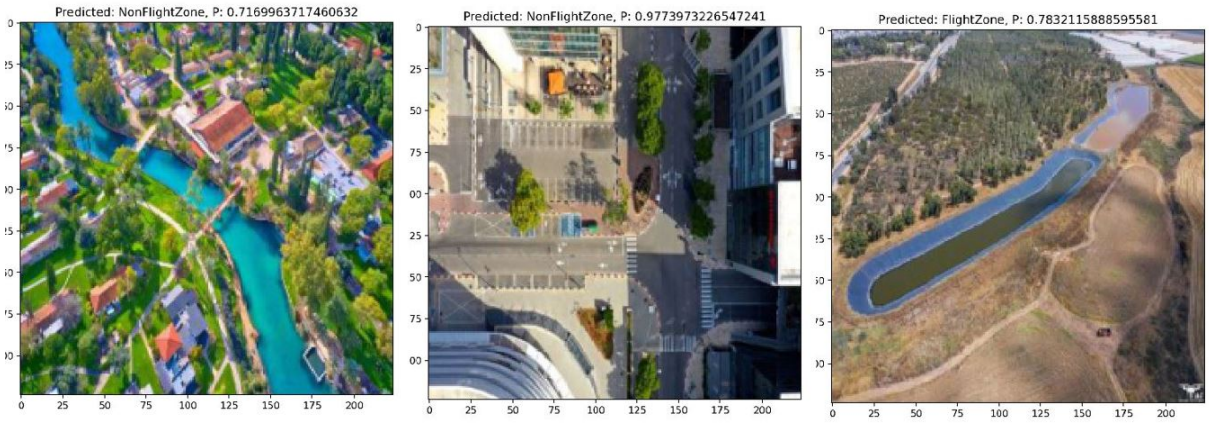


Figure 5: Images correctly predicted by the model along with the models "confidence"

| Training Set | Loss | Accuracy | F1 Score |
|---|---|---|---|
| **UAV** | **0.102** | **0.972** | **0.967** |
| SAT | 0.185 | 0.940 | 0.931 |
| GoogleEarth | 0.170 | 0.968 | 0.961 |

Table 1: Comparing result over training with different datasets on Resnet50

| Model | Loss | Accuracy | F1 Score |
|---|---|---|---|
| Resnet50 | 0.170 | 0.968 | 0.961 |
| **VGG16** | **0.100** | **0.974** | **0.963** |
| MobilenetV2 | 0.156 | 0.974 | 0.963 |

Table 2: Comparing different models trained with GoogleEarth Images

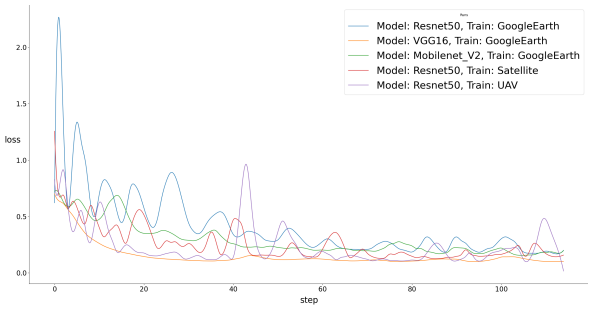


Figure 6: Loss comparison

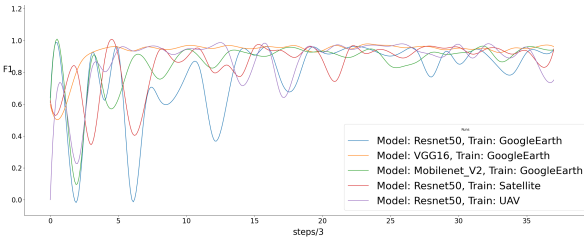


Figure 7: F1 comparison

dataset. In Table 2 we can see the results showing that all models gave very similar results with VGG16 showing slightly better results.

### 3.2 Results

We notice that the loss decreases at a pretty early stage. For F1 and loss graphs compared between different runs, see Figure 5 and Figure 6. In addition, we received validation accuracy of over 95% after two epochs.

### 3.3 Conclusion

Overall we achieved promising results with a relatively low number of training epochs. We believe the good results are attributed to the following reasons:

1. The model we used was trained for many epochs on the huge ImageNet data set. This made the model a very a good image feature extractor for a vast variety of images, including images of landscapes and urban scenes which share the same characteristics of the images we wished to classify. This allowed our job domain to focus on training the model's weights mainly on the high-level features and on the classification layer. In other words, fine-tuning the model was sufficient for the task at hand.

2. GoogleEarth and UAV have similar features, both in low-level and high-level. This makes the trained weights of the model relevant to the UAV images as well.

While the GoogleEarth dataset has shown slightly better results over the Satellite dataset, we have not

proven that it is a better way of training the model as Satellite images are much easier to collect and we can create a much bigger dataset. Nevertheless, we are still confident that Satellite based training set is inferior to GoogleEarth based training set. This is due to the fact that Satellite images are 2D in nature, which puts them in a big disadvantage to GoogleEarth images, which comprise of 3D images as well which are more similar to UAV taken images.

We believe that in order to prove the benefits of training over GoogleEarth modeled images over Satellite images we need to further test this method on a more complicated task such as Object Segmentation. Another experiment that may show the advantage of the GoogleEarth dataset may be on a similar task shown in this paper, but with a more complicated test set containing "edge cases" where the UAV is near the border between flying over a non-flight zone to flying in a flight zone. For the GoogleEarth dataset we can imitate these scenarios and train him to distinguish between them, while with our Satellite dataset we can not properly imitate these scenarios which may cause it to fail at these exact complex cases.

All in all, our project shows feasibility of training for UAV using GoogleEarth images, which concurs with our initial premise. In addition, whilst we have not managed to prove it beyond question, we are still confident that Satellite based training set is inferior to GoogleEarth based training set.

## References


Gong Cheng, Junwei Han, and Xiaoqiang Lu. 2017. Remote sensing image scene classification: Benchmark and state of the art. *CoRR*, abs/1703.00121.

Adam Van Etten, Dave Lindenbaum, and Todd M. Bacastow. 2018. Spacenet: A remote sensing dataset and challenge series. *CoRR*, abs/1807.01232.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. 2017. Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE.