DRONE IMAGERY AND DEEP LEARNING FOR ROAD INSPECTION

Aditya Gautam — Haoyang Yuan — Raquel Rodriguez Jimenez

1. INTRODUCTION

This project evaluates the feasibility of deploying drones and deep learning together to solve problems that humans are not well suited to perform because of safety issues, or because these are time consuming tasks.

The main obstacle in this application is that drones are not smart enough yet. We need to combine the hardware that is available with powerful algorithms like Deep Learning. In this process, we will make smarter and more complicated image processing.

Even though we had the scope of the project, we went through a lot in choosing the specific project objective. We considered railway inspection, however it is not practical in this case since the railway will usually need more inspection metrics than just images. We also considered bridge inspection, but we found bridges are usually unique and they are much harder to evaluate compared with roads.

Thus we decided to evaluate real time road infrastructure monitoring based on publicly available road images. The main advantage of drone inspection is that drones can reach the areas that may not be reached by inspection vehicles. This condition will especially occur after the disaster such as mountain roads and icy roads, and it will also occur during the war time.

The main reason for choosing this problem was the need of a dataset that was easy to acquire. We used the publicly available road images to train our Deep Learning model, and tested the model against real road images from the perspective of drones.

Our project goal was to experiment and prove that we can use Deep Learning model for the real time road inspection purpose.

2. MOTIVATION

Drones have become a bursting point in the current technology. Many applications have become possible with the help of drones.

One of the most common application is to use drones to inspect or manage infrastructure. As the infrastructure is usually static and hard to inspect from the necessary point of view, drones are a good option for this situation.

Traditionally, road inspection has been performed in two ways, community driven information and human inspection.

Community driven information is the crowdsourced information that is available. For example, communities will share the road information within each other and authorities when necessary. Also, there are crowdsourced applications like Waze where people can report road conditions.

An example of human inspection is when the road authority hires people to inspect the road's conditions.

However, these two methods need a human in between, thus they can not provide information fast enough, since roads are usually several miles long. These methods will also be unavailable for bad condition road areas when emergency situations happen.

Thus we are using the drone images as well as a Deep Learning model to provide an agile and smart solution to this problem. Our project will build up the model as the proof of concept.

3. RELATED WORK

Most of the related work available is either done by humans monitoring infrastructure using drones without any automatic features or done in the machine learning field without considering hardware like drones.

In 2013, extensive work in the field of machine learning for aerial image labeling was done [1]. Mnih shows how deep neural networks implemented on modern GPUs can be used to efficiently learn highly discriminative image features. Publicly available images were used in this thesis to identify roads, but not to identify the condition of these roads. Also, no practical application using drones is mentioned. The work is focused on finding new loss functions for training neural networks that are partially robust to incomplete and poorly registered target maps.

Several states have been using drones to inspect their infrastructure. For example, Minnesota did a study to demonstrate the effectiveness of utilizing UAV technology as it could apply to bridge safety inspections. The research team investigated the technology on four bridges located throughout Minnesota [2]. In this study, engineers went to site and inspected the bridges using the drones on site, which is different from our objective of automatic inspection. Other similar studies have been made testing the feasibility to use drones for several inspection chores, none of them used deep learning to automatically perform the task[3][4].

4. SYSTEM DESIGN

The project is having a straightforward design that can build Deep Learning model to leverage the publicly available road images. Our design contains three components, the images collection, the cloud environment setup and the neural network model building as seen in Figure 1.

5. SYSTEM IMPLEMENTATION

5.1 Image Collection

We used OpenStreetMap(OSM) to locate the road coordinates, and downloaded the Street View images from Google Street View. The detailed steps can be described as follows:

1. We used the keyword and tag from OSM API to gather the road coordinates. The example keyword could be “highway” and the example tag could be “motorway”.
2. Requesting the Overpass API from OSM, we downloaded a set of target coordinates between set of coordinates we input, which is called a bounding box.
3. We sent requests to Google Street View with these coordinates as parameters to download the Street View images. During this process, we also customized vertical and horizontal parameters to adjust the scope of street.
4. The last step is to clean the noisy images and label the images by two boolean value categories, good and bad.

Some challenges will come out in these processes.

1. Illumination condition: the road images are taken at different time, so the season and daytime when the images are taken will cause different illumination condition.
2. Image noises: the Google Street View is not dedicated for the road, in the road images

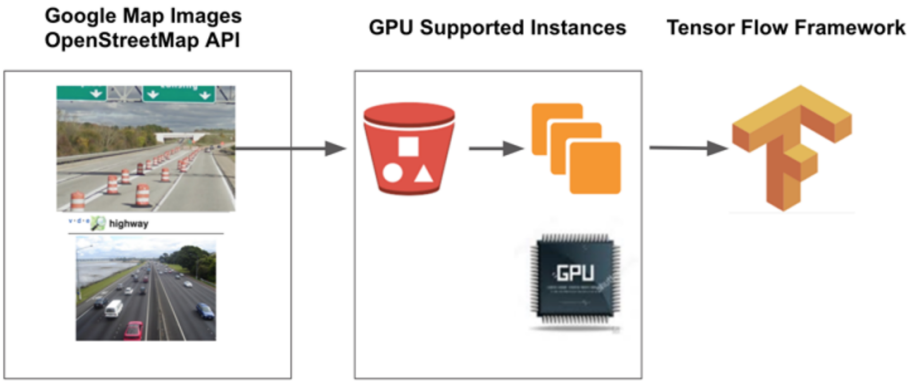

Figure 1: System design to build the deep learning model

scopes, we have noises such as trees, houses, shadows and even the Google car.

3. Lack of domain knowledge: since we don't have enough domain knowledge in road engineering, some of the image labels may not be accurate. We tried to tell a good road condition if there is no crack of potholes, cracks, faded zebra crossing etc. But it is challenging for us to decide a bad road condition when cracks existing. There is no specific metrics yet for us to easily classify a bad road.

For these challenges, we have several ways to address them, but we can not eliminate them.

1. We gradually realized that the roads from the same city is more like to share the similar illumination and noisy features. So we tried to collect images for both good and bad roads from a set of cities
2. Some of the noisy images help in the model training process once we turn on the dropout function.
3. Each image is viewed by all the three team members to reach a mutual agreement on there road quality classification.

After the collection process, we gathered several batches of images. They are around 2000 in each category.

Batch	Cities	Number of Images
1	Oklahoma City, Milwaukee, Bridgeport	~200 each category
2	Sacramento, Oklahoma City, Springfield, Boston	~800 each category
3	San diego (part 1)	~500 each category
4	San diego (part 2)	~500 each category

Table 1: Batches of Images

5.2 Model Training

Environment:

We set the environment using AWS. We used two types of Ubuntu Server 16.04 LTS instances, the compute optimized c3.8xlarge with 60 Gigas of memory and 32 cores and the g2.8xlarge GPU instance, with 32 cores and 60 Gigas of memory.

To set the CPU instance we installed Anaconda2-4.2.0-Linux-x86_64 and using PIP we installed TensorFlow for Python 2.7. We learned that it was important to install the version 2.7 because at this moment, the tools for connecting the instances to AWS and Google cloud are not fully supported and we had quite a few errors when trying to use TensorFlow for Python 3.5.

For the GPU instance we used the Deep Learning AMI v1.5, provided by Amazon Marketplace, with CUDA and TensorFlow preinstalled.

We also had all the images in S3 and used Boto to connect to the buckets.

Deep Learning Framework:

We considered the following frameworks before making a decision on which one we were going to use:

FW	Pros	Cons
Theano	• Well suited to data exploration and intended for research • Python + Numpy • Computational graph is nice abstraction	• Raw Theano is somewhat low-level • Large models can have long compile times • Error messages can be unhelpful • Buggy on AWS • Slow

FW	Pros	Cons
Torch	• Lots of modular pieces that are easy to combine • Easy to write your own layer types and run on GPU • Lots of pretrained models	• Written in Lua • Write your own training code • No commercial support • Documentation is not so good
Caffe	• Good for feedforward networks and image processing • Train models without writing any code • Python interface	• Need to write C++ / CUDA for new GPU layers • No commercial support • Chiefly used as a source of pre-trained models hosted on its Model Zoo site
Tensor Flow	• Flexible Architecture • Big Community - Github - Stack Overflow • Easily supports GPU and CPU Easy to install and start implementing • Good documentation and tutorials • It's becoming the state of the art for deep learning models	• Drops out to Python to load each new training batch • Much "fatter" than Torch; more magic • Computational graph is pure Python, therefore slow

Table 2: Deep learning framework comparison

Code Implementation:

As seen in Figure 2, we used Boto to connect to S3 and download the images in Numpy arrays that will then let us create a modular batching to test the results and change the implementation from CPU to GPU.

Then, we created a set of variables and operations that would be performed by the TensorFlow's graph.

Once we initialize the TensorFlow's session, we iterate over the batches we created before and we start the training, testing and visualizations.

5.3 Deep Learning Model and Parameters

To identify road into a good and bad category, we have to specific or extract the features which distinguishes bad and the good road. These features are potholes, cracks, faded zebra crossing etc. which can defined the quality of the road. To extract these features, we can either do by computer vision algorithm combined with machine learning which would have its own challenges and would require good feature engineering.

Network for road classification:

To solve our problem, the best way would be to use Convolution Neural Network(CNN) and let the network extract the good features by itself and train a model with weights at each layer, so that it can assign a good probability of good and bad images through softmax layer. Standard classification problem like cats vs dog, houses vs cars etc are pretty simple through CNN as almost all the features are different in the objects.

Likewise, it is easy for a network to extract those features through convolution filters and compress/ reduce the dimensionality reduction through max or average pooling. In our case, most of the features are similar, only difference is in cracks on the roads, which the network need to filter out.This makes the problems much more challenging as compared to other image classification problems.

With about 90% of the features overlap that occurs in the image like road side, white and yellow line etc., we are left with only less than

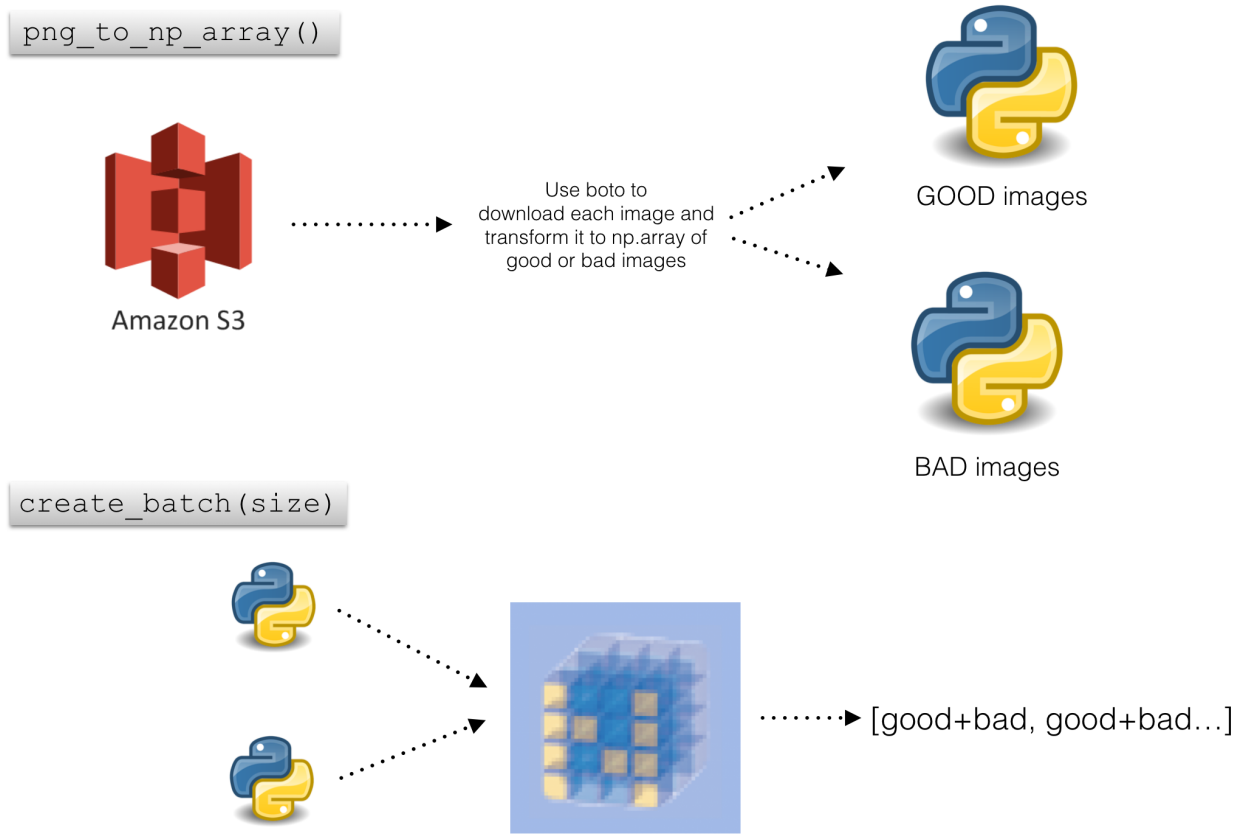

Figure 2: Main functions for batching images

10% of the features i.e. cracks that distinguish good and bad images.

Deep neural network layers and architecture:

To tackle our problems of cracks identification and scoring based on that, we have used four convolution layers with four polling layer along with a fully connected layer of 1024 reLu neurons to learn about the cracks and predict the probability of belonging to a particular class through softmax layer.

The input image size was 500*500 with 3 channels(RGB) which was converted into the tensor. For convolutional layer, we used the filter size of 5*5 with stride as 1. This was good enough for us to extract features as seen by the visualization of images at hidden layer. The total number of output filters that we used at first convolution layer was 64. 64 is based on experimentation as we get decent result with this. Before this we have tried 32 output filters at first convolution layer and could see slight increase in accuracy increase. For 2nd,3rd and 4h convolution layer, the filter and stride size were same but the total number of output filters were increased to 64,128 and 128 respectively. For the pooling, we have used max pooling after reading some literature and understanding that the cracks can be well learned by the max pooling instead of average pooling.

For the fully connected layer, we just kept 1024 ReLu activation neurons which is based on the research reading and learning from standard CNN. With the intention to improve the accuracy further,we have tried two 1024 neurons Relu activation layers before softmax cross entropy loss layer. We didn't see any improvement in the accuracy with indicating that we have to focus on

increasing the number of dataset and/or improving the existing dataset of images we have.

The final layer or softmax layer has two output neurons which gives a normalized score of belonging to a particular class i.e. good and bad. The optimization used is the Adam optimizer which has different inbuilt parameters, which takes care of the momentum and the speed of gradient descent.

Experimentation and Analysis:

We ran various experiments to learn about the complexity of the problem, number of input images, impact of filter size, number of hidden layers etc. and learned about the network which can best perform this our problem.

Our initial experiment was pretty straightforward which contains two convolutional layers and two pooling layers. Filter size at convolutional layer was set to 5*5 with stride as 1 and pooling used was max pooling. Number of images that we used for this experiment was 500 per category. We used this network to see our initial results and the accuracy was similar to random coin toss. This implies that network was not deep enough to understand the features i.e. cracks and thus can't distinguish between good and bad roads properly.

So, the next thing we experimented was to increase the number of images and increase the number of hidden layers. As per the images, we almost doubled the size to 1000 per category. The network has four more layers added to it, two convolution and two pooling layers. This enables network to learn better about the features and it improved our accuracy to 76% on testing data. Our testing accuracy though was very less i.e. around 54% with dropout probability of 25%. This implies that we are still lacking in either the complexity of the network or number of images. So, we further doubled the number of images per category and re-run the experiment.

Number of Images	Model	Training Acc	Test Acc	Drop out
1000	5 (2 Con, 2 Pool, 1 fcc)	Random toss	Rand	0%
2000	9 (4 Con, 4 Pool, 1 fcc)	76%	54%	25%
4000	9 (4 Con, 4 Pool, 1 fcc)	98%	85%	0%
4000	9 (4 Con, 4 Pool, 1 fcc)	92%	90%	25%

Table 3: Different accuracy based on the number of images, hidden layers and dropout

We see that the network has learned really well and it started giving near perfect accuracy on the training set. It clearly overfit the network but a good progress in our approach since the first step would be to get a good accuracy on the testing samples and then generalize it to improve the testing accuracy. The real accuracy or the test accuracy for this network was around 85% and it was expected also since there was no regularization added to the network while training.

Regularization:

To improve the testing accuracy, we added dropouts as regularization with keep probability of 75%. This means that neurons will be deactivated at the fully connected layer with a probability of 25%. Likewise, the weights learned by the network will take into account some of the missing features.

To take different real life scenarios into consideration and make the model more generalized, we added lots of variation in the images like the images with trees, shadows, cars, house, concrete etc. were being feed into the network in both good and the bad images. Along with this, the images with different angles were also being feed into the network while training.

As the result of these two regularization, with the same deep neural mode, the accuracy was improved by 90% from 85%. We also experimented with the change in dropout probability from 25% to 15%, there was not much change in the accuracy.

With these experiments, we learned a lot about the model, appropriate number of images,

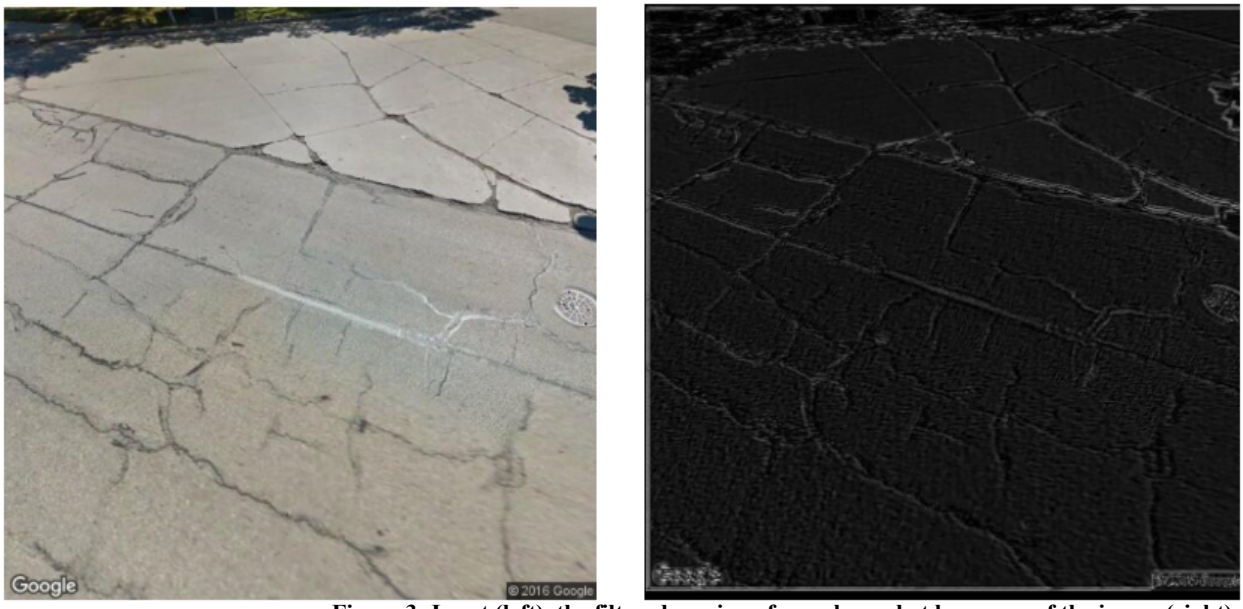

Figure 3: Input (left), the filtered version of one channel at layer one of the image (right)

regularization etc. which can be used for the future research and experimentation.

Image visualization:

To have a better understanding of what is going on in the hidden layers, we analyzed the filters associated with the 1st convolution layer. We saved the final weights learned and pass it to some sample images. We learned about how the images are being passed from one layer to another and the features that was extracted at them.

The network learned to extract the cracks,edges and other relevant things from the images as seen in Figure 3.

5. KEY TAKEAWAYS

Some of the key takeaways from the project were :

1. Number of images needed for road classification was 90% accuracy was around 4000. With more images we can further improve the accuracy.
2. Apart from dropout, another way to do the regularization is to feed images with some features missing.
3. The quality of images influences the model's accuracy a lot. For real practice, we recommend road images from uniform scope and direction. It is highly possible for configured drone tasks.
4. The noise in the images can be addressed in certain degree as long as we have enough training dataset.

FUTURE WORK

1. The future work can leverage the more uniform real drone road images to train and test the model. It will be better if the images are already clean data set labeled by expert.
2. Another work is to build a more generalized model and verify that it can process images with various road conditions from different places. With huge learning from this project, some of the key areas would be to see how well the model perform in different regions like the characteristics and features of a bad road in one city would be different from another city, so accuracy may change. Doing experimentation and building a cluster of cities which are similar and then sampling out good and bad images from each cluster is expected to build a robust model. So, doing this would be really improve the robustness of the model and the ability to regularize.
3. Moreover, the multilevel classification of the road image would be another thing that could

be done. Like having a category of 1 to 5, where 1 implies best road condition and 5 implies the worst road. This would require much more images than two class classification as number of classes are increased. Also, this would need labelling expertise from the government people etc. who are experts in the field.

REFERENCES

[1] Mnih, V. 2013. *"Machine Learning for Aerial Image Labeling"*. Doctor of Philosophy Thesis. University of Toronto.

[2] Lovelace, B. 2015. *"Unmanned Aerial Vehicle Bridge Inspection Demonstration Project"*. Minnesota Department of Transportation, Research Services.

[3] Otero, L.D. 2015. "*Proof of Concept for Using Unmanned Aerial Vehicles for High Mast Pole and Bridge Inspections*". State of Florida Department of Transportation.

[4] http://waypoint.sensefly.com/using-drones-uas-for-bridge-inspection/ accessed October 15 2016