Capstone Project - Final Report

Mike Nebesniuk - April 10, 2021

Problem Statement / Background

With machine learning, is it possible to predict if a landscape photo is capable of winning a photo contest. This is the question I will try to prove with this project. I plan to use a CNN (Convoluted Neural Network) to train itself on photos I download from various landscape photography contests in an attempt to prove this point. I will download Images that could be considered contest winners, and I will then download images that did not win.

As a photographer myself, I have entered plenty of contests and often found that photos that win photo contest are really not all that great. Often the photos I see win these contests I would consider failures and not let anyone see them let alone enter them into a contest.

As a photographer I feel that this could potentially be an interesting project that may help other photographers pre-judge their images. I will attempt to use CNN networks to examine photos that have either won, or placed in the top 1 percent of photo contests verse photos that did not. I will use the website viewbug.com to get my photos from as they have monthly contests and will be a potentially rich source of photographs.

About the Data

To begin with, I tried using python and some beautiful soup to web scrape the photos off of the ViewBug.com website. After some research and trial and error I was able to get some preliminary results from some of the contests, but any contests that let me access the 30,000 plus photos, the python scraping would not allow me to access the photos from the contest. I then began to research other alternatives for web scraping.

FoxyImage was the one alternative I found for web scraping images. It is a FireFox addon that will scrape images that are shown on websites. To start with I looked into some of the smaller landscape and best of contests to get some truly award winning photographs. The FoxyImage was able to download them well, but there was a lot of other photos that did not belong like avatars from the people who uploaded the photos, and images to links to other contests. A quick visual look at these photos was enough to determine which ones stayed and which ones left. After this I began looking at the contests where I had entered the photo contest. When the contest was over they would send me a link to the contest page where I could look at all the images winners to last place. This is where I thought I would be able to download more than enough photos for the project. I was mistaken, as it turns out the website throttles the download and viewing of photographs somewhere after about the 2000th place. After many attempts to get past this point the furthest I got was to about 4200 photographs. So I took it, and a few from other contests too, I ended up with about 5200 photos that would probably not win a photography contest, and 4300 images that would. At this point I was happy with the data I had and I began to do EDA on the photo sets.

EDA (Exploratory Data Analysis) and Data Cleaning

The first step in the EDA and data cleaning process was to remove the avatars that each photographer had when they uploaded their photos. I supposed I could have done this in python, but because I was already looking at the photos on windows, it was a simple fix to remove all the avatars. Each avatar had the text 75x75 in the name of the avatar, so a quick search and delete and they were all gone.

Next it was time to remove all the errant photos, photos of pets or models that would cause weird outliers in my modelling. This was a more difficult task but I found if I looked at smaller groups of photos it went quicker.

Removing duplicates was next. This is something I could have done in python, but because the images had the same names, arranging them by name, and removing all the ones with the (2). (3), etc beside them made the task quick and easy to do in windows. Many people enter different contests with the same images, because I took the top photos from many contests, I ended up with many versions of the same image.

Resolving issues where photos would not change aspect ratio. A small group of maybe 200 photos in both the award winning and non award winning photos would not all python to change their aspect ratio to 256x256 pixels, or really any other options. I ended up using a windows based image resizer to resize them to 256x256 pixels it worked great, it showed me which ones would not resize, and I deleted them. This will also make sending the images over the website easier as they will be much smaller now.

Modelling

For the CNN model, I began with setting the variables. In order to approach this challenge, I split the photos (data) I downloaded into two classes, Photos that could potentially win a photo contest, and those that could not.

For testing purposes I resized the images to 64x64 pixels, this allowed the epochs to run quicker and for me to get the answers quicker.. When it came time to run the model for good I chose 256x256 pixels. This is about comparing good quality photos with one another, I wanted the model to have as much data as it could. I didn't go any bigger than 256x256 because the photos in their native format were 400x170 pixels for the most cases.

After evaluating several image networks, I felt the VGG16 was the best one to use. It rated highest by most of the articles I read, and it had more images than most of them out there as well.

For the model itself I decided to use a Sequential model, it was the best fit for photo or image data.

For the optimizer I went with the Adam Optimizer. It works well with problems that are large in terms of data, so I figured it would work well with almost 10,000 photos.

Conclusions

To start out, this was a very ambitious project as it turned out I spent countless hours downloading images from the viewbug website. The website throttling that viewbug has setup on their site eventually got the best of me. I spent 5 hours a day and 5+ hours a night for 3 weeks trying to download photos. I was further hampered with computer issues such as Windows 10 updates that sprang up automatically. Accidentally hitting the back button on the mouse. Browser freezing, and issues with my new Mac Mini not downloading all the photos it said it was. In retrospect I could have used my own photos, I have lots, and it would have taken less time. The downside is it would have been my perspective on what is good or not, where as the ViewBug site had a panel of professionals to judge the images for me.

Results with image size: When comparing image size, the images scaled down to 64x64 pixels performed worse than those at 256x256 pixels. The larger images had higher accuracies.

Validation / Training Size: A split of 20% yielded an accuracy score of 64%, where as a split of 10% yielded 58%. I decided to split at 25%, I didn't want to split too far, but I figured this was a good compromise, at 25% I am seeing about 67%. Also at 10% I was only getting 6 epoch to run before the model stopped learning anything new. At 20% it stopped at 10 epochs. Now I am at 25% I was able to get to at least 18 Epochs.

Introducing New Photos: Now that the model has been trained I tried to run several photos through it to see if it could predict if they were award winning type photos or not. After running a very good photo, a terrible photo, and a photo from the non award winning photo set, all three were predicted as award winning. This confirms my fears from above, the data I am downloading from above is too similar. Photos that places 1st are too similar to those that placed 4000th. A more diverse set of photos would be required.

Final Thoughts: This isn't so much a failure as it just confirms some things. I would need access to lower quality photos to make better predictions. Even a computer trained against the large VGG16 network couldn't tell the difference from 1st place to 4000th place. These contests are attracting lots of excellent talented photographers and choosing a winner must be very difficult.
In future I could try to gather many photos that exemplified basic photographic standards of excellence such as 'rules of thirds', ''guiding lines", and 'composition' to run each photo against. This would take a very long time to collect the data.

Appendices

Attachments

1. Readme-txt - A readme file that outlines all the files included in the zip file.

Project Documents:

1. Mike Nebesniuk - Capstone Project - Report.pdf - Final Report
2. Mike Nebesniuk - CapStone Slide Deck - Final Presentation Final.pdf - Final Presentation on Internal Demo Day
3. Mike Nebesniuk - CapStone Slide Deck - First Presentation.pdf - First Presentation
4. Mike Nebesniuk - BrainStation CapStone Final Loom Presentation.pdf - Presentation from submission to BrainStation website.
5. Mike Nebesniuk - BrainStation CapStone Final Loom Presentation - Google Slides.mp4

Jupyter Notebooks:

1. Mike Nebesniuk - Capstone Notebook 1 EDA and Web Scrapping.ipynb - First Jupyter lab notebook containing EDA and Web Scrapping.
2. Mike Nebesniuk - Capstone Notebook 2 - CNN Network Modelling.ipynb - Second Jupyter lab notebook containing modelling and conclusions.

Image Files:

https://drive.google.com/drive/folders/1eSJHtTF0wgWPdzv04j2DarL1R8ppmcDH?usp=sharing

Please follow the link above, paste it and the 2 jupyter notebooks in the same directory and everything should work fine.