An Investigative Study into the Properties of the GoogLeNet Image Recognition Network

Adam Horvath-Smith

Wright State University

Departmental Honors Program, Department of Computer Science and Engineering

Committee: Drs. Thomas Wischgoll (advisor), Derek Doran, Michael Raymer

An Investigative Study into the Properties of the GoogLeNet Image Recognition Network

**Abstract**

This paper serves to examine various properties and features of the GoogLeNet image recognition network in an attempt to better understand how it arrives at its decisions. GoogLeNet was chosen as the focus of this study because of the large body of existing research into interpreting and understanding its decisions. Building from this previous work, this paper will examine the particular features of an image that GoogLeNet looks at when forming a classification decision. Saliency maps are used to highlight and detect which features are most important to making a decision. The features deemed important are then used to try to improve the ability of adversarial images to “fool” the network into seeing something that is not there. Being able to fool the network shows that these features do, indeed, have influence over GoogLeNet’s classification decisions.

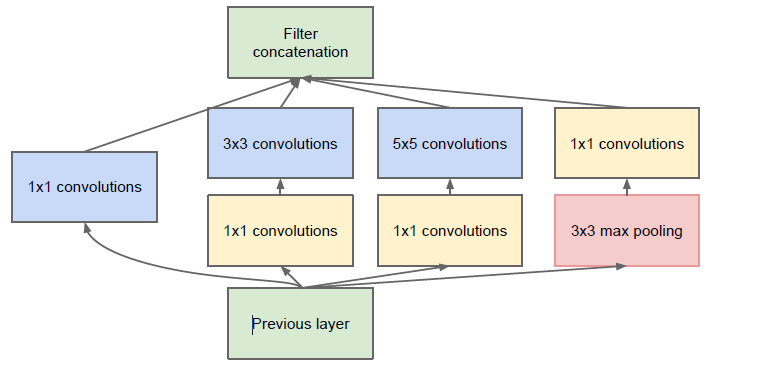
**Introduction**

Interpretability of neural networks is a hot topic in the field of machine learning. Neural networks operate as a sort of black box, taking in inputs and producing reliable outputs, but their complexity makes it difficult to understand exactly how these outputs are produced. The GoogLeNet image recognition network is a frequent subject for research into the interpretability of neural networks. The particular article that served as the inspiration for this project was “The Building Blocks of Interpretability” by Chris Olah et al. [1]. This article uses a number of techniques to visualize what is happening on each layer of the GoogLeNet network and get a better understanding of what each layer contributes to the final classification. One such technique is using saliency maps to understand which pieces of the network are activated when a certain image is presented. These saliency maps work by highlighting the parts of an image that a neural network looks at most closely when making a classification decision. The highlighting is overlaid on the original image to see exactly which areas contribute most to the decision. Figure 1 shows an example. Typically, saliency maps are generated only based on the output of the final layer of the network. Olah et al. found a way to create saliency maps between hidden layers of the network. By doing so, they were able to trace an image through GoogLeNet and see which parts of an image were being most closely examined by each layer.

*Figure 1: An example saliency map, highlighting features contributing to “Labrador retriever” classification.*



Before going further into the research done into GoogLeNet here, it may be helpful to see an overview of how GoogLeNet is structured. The architecture was first described by Christian Szegedy et al. in “Going Deeper with Convolutions” [2]. GoogLeNet is a convolutional neural network based on the Inception architecture Szegedy and his team created. The Inception architecture focuses on stacking convolutional “Inception modules” on top of one another. Rather than relying on a single type of convolution, the Inception module runs several convolutions in parallel, then concatenates each resulting feature map into one. 1x1 convolutions are done before the larger 3x3 and 5x5 convolutions simply to reduce the dimensionality of the inputs, minimizing the number of necessary computations [3]. An Inception module is illustrated in Figure 2. The core of the GoogLeNet architecture consists of nine such Inception modules linked together. GoogLeNet was the first neural network to be based on the Inception architecture; several others have since followed.



*Figure 2: Inception module architecture [2].*

In this project, I wanted to see what a single Inception module could tell us about the neural network’s final classification decision. To do so, I chose the first and most discriminative of the Inception modules, labeled “Mixed3A” in the official documentation of the GoogLeNet architecture (see [2]). I generated a series of saliency maps for the output of Mixed3A on several different images. I chose several images of the same class as well to see if there were any similarities between each. (For example, several images that were classified as a “barn” were examined.) Doing this, I was able to find patterns in saliency maps generated on images from the same class.

To better understand these patterns, I set two goals for this project. First, I wanted to find a way to extract the most “important” features from an image, as highlighted by the saliency map, to see exactly which were necessary for GoogLeNet to make its decisions. Second, to confirm these features’ role in GoogLeNet’s classification decisions, I tried to leverage these features to better trick the network into seeing something that was not there. Using an adversarial image generation framework, I was able to generate a series of images that looked like a given normal image, but would be misclassified by GoogLeNet as being another type of image. Using the features deemed “important” to a given classification decision, I generated some of these adversarial examples using base images that had similar “important” features, and some that did not. Figure 3 shows an example of such adversarial images. Judging from GoogLeNet’s confidence in the adversarial examples, one can draw conclusions as to the role image features play in predicting an image’s class.

*Figure 3: The first image on the left is an original, unaltered photo of a barn. The middle image was created by the adversarial generation framework to be classified by GoogLeNet as an electric ray, which does not share similar features with the barn class. The third photo is an adversarial image made to be classified as a boathouse, which does share similar visual features with the barn.*



**Inquiry Methods**

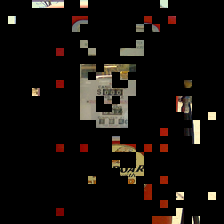
This project began as an open-ended inquiry into the inner workings of GoogLeNet. Using Olah’s “Building Blocks” article as a starting point, I began looking into what aspects of the network I could visualize to better understand its decision-making process. After seeing how this article used saliency maps to, in a way, peer into each of the nine Inception modules of GoogLeNet, I decided to try to leverage this to see if there was any sort of correlation between the areas highlighted on the saliency maps and the final classification predictions the network produced. The initial idea was that, as an image proceeded through the network, classes it was more confident in in the end would have more highlighting in later saliency maps, while classes of less confidence would have less highlighting. I generated several images, like the one in Figure 4, to test this theory. These images were made using code provided by Olah et al. along with the “Building Blocks” article [4]. However, after preliminary testing, this idea proved infeasible. I could not find any real correlation between the amount of area highlighted in a saliency map and the classifier’s confidence in its classification, and Dr. Raymer suspected there should not be any real such correlation. As a result, I decided to abandon this idea altogether.



*Figure 4: Saliency maps generated after each of the nine Inception modules in GoogLeNet.*

Branching off this initial idea, my advisors and I decided it would be best to focus on the features being highlighted by the saliency maps, rather than the saliency maps themselves. I began to look at what features the network seemed to be most interested in and what they could tell us about how GoogLeNet classified a given image. I found a way to extract different features, so-called “features of interest,” from images that GoogLeNet seemed to be most interested in when making a classification decision. The code used to do so was based on that from Olah et al.’s saliency map generator code [4]. Essentially, it would look at the brightest regions of a saliency map generated after Inception module Mixed3A in the GoogLeNet system. This brightness was calculated as pixel intensity, with any area with an intensity value of 0.5 or higher, on a 0-1 scale, being considered a “feature of interest”. From there, it would clip out all portions of an image that aligned with these bright regions, then save these clipped regions into another image. Figure 5 provides an example. By generating several of these clippings from several different images of the same class, I was able to get a fairly good idea of what features GoogLeNet was looking for when making a classification decision.

After extracting the features that GoogLeNet was interested in when making a decision, I wanted to see how these influenced the classification process. There were a few things I tested with this. First, I did a simple test where I would remove certain features from an image to see how that affected the classification confidence. In doing so, I would try both removing features that were deemed important, as well as other features in the image that were not highlighted in my feature selection.



*Figure 5: An example image, the saliency map generated after running through Inception module Mixed3A, and its related areas of interest. Note how the counters on the gas pump are highlighted in the final image, meaning these are an area of interest for the “gas pump” classification.*

Another item I decided to test was how to leverage these important features to see if I could do a better job “tricking” the network into seeing something that was not there. To do so, I used the Expectation Over Transformation (EOT) adversarial image generation algorithm, developed by Athalye, Engstrom, Ilyas, and Kwok [5]. One of the authors, Anish Athalye, wrote a companion article titled “A Step-by-Step Guide to Synthesizing Adversarial Examples” [6], which gave a tutorial on how to implement the EOT algorithm and a framework for doing so. This algorithm was chosen due to its ability to produce clean adversarial images that cause a classifier to mis-classify an image with high confidence.

Using the EOT algorithm, I generated several adversarial images from a set of 100 base images, gathered from the ImageNet dataset. These adversarial images were split into two classes. In one class were images generated to look like one of three random classes, as chosen from the list of 1,000 classes GoogLeNet is capable of recognizing. The other class of images was generated based on whether the adversarial class shared “features of interest” with the base class. The idea in doing so was to run the images from both classes of adversarial examples back through the GoogLeNet classifier to see which class had the higher average confidence in identifying the adversarial class. Should the second class of images, those generated with shared “features of interest,” have a higher average confidence, it would show that even through the distortions caused by the adversarial framework, the similar features of the base and adversarial classes could together produce an adversarial example of high misclassification confidence.

**Experimental Results**

Using the EOT method as described above, I generated a total of 300 adversarial images based on random adversarial classes. These images were split into three equal groups, each with a different adversarial class. The classes used to generate these adversarial images were “electric ray,” “wild boar,” and “saltshaker.” A set of 100 images was then generated based on adversarial classes with shared features of interest between it and the base class. A mapping of base classes to similar adversarial classes can be found in the Appendix. This set of images is noticeably smaller than that used to generate adversarial images from random classes. This is due to maintenance on ImageNet servers that limited the images available for download. As the project required finding classes of images that shared similar features to a base class to create “similar” adversarial examples, I needed to be able to download example images from these similar classes to run through my feature selection algorithm. As most of the ImageNet database was unavailable, I was limited in the similar images that I was able to download.

When comparing GoogLeNet’s classification confidences between the adversarial images based on random adversarial classes and adversarial images based on more “similar” classes, we get the following averages:

Each group of bars above represents a given adversarial class (identified in the Appendix by their textual name), followed by a 0 or 1, indicating which of two groups the base image came from. (There is no significance to this group other than the way the dataset was randomly split into two groups of 50 when the images were first collected.) In the Random Classification Confidence chart, all 100 of the base images are used to generate an adversarial image based on one of the three adversarial classes described earlier. In the “Similar” Classification Confidence chart, only base images deemed similar to the listed adversarial class were used to generate adversarial images. As there were so many more groups of “similar” adversarial images with a smaller number of base images, there are several more bars on this chart. The blue bars represent the average classification confidence in the “correct class” based on the base image. The orange bars represent the average classification confidence in the adversarial class based on the adversarial image.

Averaging only the adversarial confidences between the randomly-generated adversarial images and the “similar” adversarial images, we see the following:

|  |  |
| --- | --- |
| Adversarial Image Type | Average Confidence in Adversarial Class |
| Random | 0.9384 |
| Similar | 0.9962 |

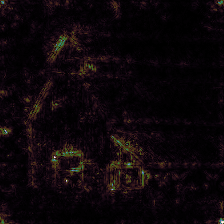
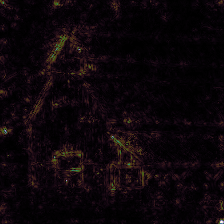
The above averages show that adversarial images generated based on “similar” classes to the base class produce confidences about 0.058 units higher than those generated based on random adversarial classes. While not a large margin, this is somewhat significant given that majority of the “similar” adversarial images produced near-100% confidence in the adversarial class, while randomly-classed adversarial images were closer to 94%. This noticeable increase in classification confidence seems to indicate that there is some use in choosing an adversarial class that shares features of interest with the original base class. The features GoogLeNet considers to be important when making a classification decision can be leveraged to better fool the network with high-quality adversarial images.

**Future Work**

Due to time constraints, there were some questions that appeared during my inquiry that remain unanswered. For one, I noticed that in most cases, the adversarial examples produced higher confidence in the adversarial class than the original image had in the original class. Most adversarial generation techniques seemed to produce images that had low confidence in the adversarial class, but were still convincing enough to the neural network to change its output confidence. The EOT algorithm used for my work seemed to consistently produce adversarial images that would yield high confidence from the neural network in the adversarial classification. It would be worthwhile to do a follow-up study to see why EOT was able to produce such high confidence-yielding images, and what the drawbacks are of using this algorithm compared to other adversarial generation techniques.

Another open question relates to which features the neural network looks at the most when classifying an adversarial image. Due to complications with the way I originally extracted these features of interest, I was unable to extract these features from the adversarial images outputted from the EOT algorithm. In an attempt to get some understanding of which features GoogLeNet was looking at when classifying these images, I created some saliency maps to see if I could highlight some of these features. Again, due to complications with my original saliency generation algorithm, I could not use this on the adversarial outputs. However, I was able to find another saliency map generation algorithm, which did work on these images. An example of its outputs is shown in Figure 6. Even on the adversarial images, the features highlighted match very closely to those of the base image, so these are not very telling. It would be beneficial to extract these features of interest from the adversarial images to understand how the network was fooled, but due to complications with the original extraction algorithm and time constraints, this was not possible during this study.





*Figure 6: The first image is the same unaltered barn image from Figure 3, along with the corresponding saliency map to its immediate right. The third image is the adversarial barn classified as an electric ray, along with its saliency map to the right.*

**Conclusion**

This study produced some interesting insights into how the GoogLeNet image recognition network arrives at its decisions. Apparently critical to classifying an object are a number of distinct regions in a given image that relate to one specific class. Leveraging the features in these regions, we can produce adversarial images that accurately and reliably fool the network with a higher confidence than with adversarial images that disregard the features GoogLeNet looks for. This reliance on certain features in an image is telling about the way GoogLeNet works and may provide clues on how to better build neural networks to avoid its pitfalls. Further research is necessary to better understand why the adversarial images produce such high confidence in their classifications, and which features in the adversarial image contribute to the adversarial classification. In the end, this inquiry shows that when making a decision, GoogLeNet relies heavily on distinct features of an image to determine its classification. So closely tied are certain features to a given class of images that it is easy to leverage these to fool the network. In the future, it may be worthwhile to make a neural network that is able to classify images based on other aspects, such as color patterns, outlines of images, or other features that are harder to mimic in an adversarial system.

**Appendix: Mapping of Base Classes to Similar Adversarial Classes**

The following table lists the mapping of base classes (the ground-truth classes of the original, unaltered images) to the adversarial classes deemed to share “features of interest” with the base class. The numbers in parenthesis are the line number listing the class in the JSON file identifying all classes recognized by GoogLeNet [7]. The EOT algorithm extracts adversarial classes by this number. Images used in this study, and all other information, can be found at https://github.com/adamhs1997/googlenet-study.

|  |  |
| --- | --- |
| Base Class | Adversarial Class |
| Barn (425) | Boathouse (449), Church (497) |
| Bullet Train (466) | Bobsled (450), Passenger Car (705) |
| Christmas Stocking (496) | Mitten (658), Sock (806) |
| Colobus (375) | Gibbon (368), Guenon (370) |
| Dam (525) | Breakwater (460) |
| Gas Pump (571) | ATM (480), Payphone (707) |
| Mashed Potato (935) | Cauliflower (938) |
| Nematode (111) | Gar (395) |
| Polaroid (732) | Projector (745), Reflex Camera (759) |
| Rotary Telephone (528) | Payphone (707) |
| School Bus (779) | Ambulance (407) |
| Slide Rule (798) | Computer Keyboard (508), Ruler (769) |
| Water Tower (900) | Hot-Air Balloon (417), Planetarium (727) |
| Weevil (307) | Leaf Beetle (304), Long-Horned Beetle (303) |

Works Cited

[1] C. Olah et al. “The Building Blocks of Interpretability.” Internet: https://doi.org/10.23915/distill.00010, March 6, 2018 [March 16, 2019].

[2] C. Szegedy et al. “Going Deeper with Convolutions.” Internet: https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf, N.D. [March 16, 2019].

[3] N.A. “Paper Explanation: Going Deeper with Convolutions (GoogLeNet).” Internet: https://mohitjain.me/2018/06/09/googlenet/, June 9, 2018 [March 18, 2019].

[4] C. Olah et al. “Spatial Attribution -- Building Blocks of Interpretability.” Internet: https://colab.research.google.com/github/tensorflow/lucid/blob/master/notebooks/building-blocks/AttrSpatial.ipynb, March 6, 2018 [March 24, 2019].

[5] A. Athalye, L. Engstrom, A. Ilyas, K. Kwok. “Synthesizing Robust Adversarial Examples.” Internet: https://arxiv.org/pdf/1707.07397.pdf, June 7, 2018 [March 26, 2019].

[6] A. Athalye. “A Step-by-Step Guide to Synthesizing Adversarial Examples.” Internet: https://www.anishathalye.com/2017/07/25/synthesizing-adversarial-examples/, July 25, 2017 [March 25, 2019].

[7] A. Athalye, Internet: https://www.anishathalye.com/media/2017/07/25/imagenet.json, July 25, 2017 [March 31, 2019].