5-2018

Comparison of Google Image Search and ResNet Image Classification Using Image Similarity Metrics

David Smith

Comparison of Google Image Search and ResNet Image Classification Using Image Similarity Metrics

David Smith

April 27th, 2018

Abstract

In this paper, we compare the results of ResNet [1] image classification with the results of Google Image search. We created a collection of 1,000 images by performing ten Google Image searches with a variety of search terms. We classified each of these images using ResNet and inspected the results. The ResNet classifier predicted the category that matched the search term of the image 77.5% of the time. In our best case, with the search term “forklift”, the classifier categorized 92 of the 100 images as forklifts. In the worst case, for the category “hammer”, the classifier matched the search term 61 times out of 100. We also leveraged the prediction confidence levels of the ResNet classifier to determine the relative similarity of an image within a set of images. In typical usage of an image classifier, only the most confident prediction is utilized. By using a larger piece of the output vector of the ResNet classifier, we were able to calculate distances between images in feature space. We created visualizations of the distance between images in sets of 100 images.

1. Introduction

Goal

For this research, we wanted to compare how Google Image search classifies images versus how a convolutional neural network classifies images. Rather than implement a convolutional neural network, we used an existing implementation that was known to be excellent at categorizing images. The specific implementation we chose was ResNet, a residual convolutional neural network. By choosing search terms from the list of categories on which ResNet is trained, we can generate data sets to test the classifier. We analyze the results of the classification to determine how closely the Google Image results match with ResNet’s expectations for that category.

We also noticed that the ResNet classifier returns a vector containing a confidence level for each of the 1,000 categories. We were interested in using the most significant dimensions of that vector to discern similarities between images. If an image is passed into the classifier, but is not a member of one of the available categories, the network will still try and predict the object of the image. Even if the prediction made by the network is not correct, there is still valuable information to be learned from the categories the classifier outputs.

Background

Neural Networks

Neural networks are a construct that take a set of inputs, or features, and after a series of manipulations, produce an output that can be treated as a decision or a prediction of the neural network [3]. The structural organization of neural networks was inspired by the biology of the brain. Inside the brain, neurons send synapses across a network of neurons, creating links between regions. A neural network works similarly. It contains a network of nodes connected via mathematical functions. The inputs on the first nodes in the sequence are modified by the weights on the links, and the output from those nodes become inputs on the next nodes in the sequence.

Once we have a neural network structure, we need to train the network on training data and then test the network using test data. One way of training the network is back propagation. Back propagation works by fixing incorrect predictions made by the network. If the output is incorrect, we can find the percent error and calculate a gradient by which to modify the connections leading to the output. We then repeat this process and modify the weights on the nodes until the percent error is below a certain threshold.

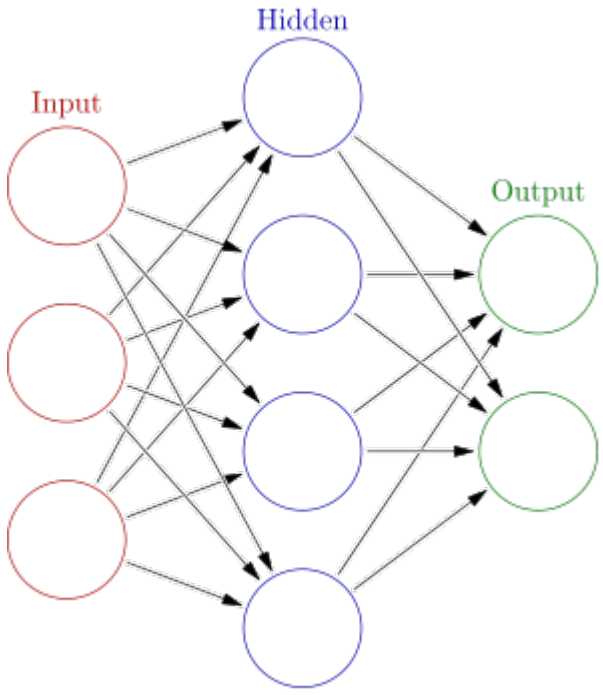

Figure 1. Structure of a neural network [8].

Convolutional Neural Networks

When using neural networks with images as inputs, we typically need to reduce the input image to a more meaningful set of data before passing it into network. There a few operations that we can use to achieve this [4], the first of which is convolution. We use convolution to extract features from an image. Convolution works by calculating weighted averages of the input pixels to produce an output image. These weights are often called the “convolution filter” or “convolution kernel”. We can use different convolution filters to detect edges, blur or sharpen the image, etc. Convolution turns the image into data that is more useful for the neural network to process.

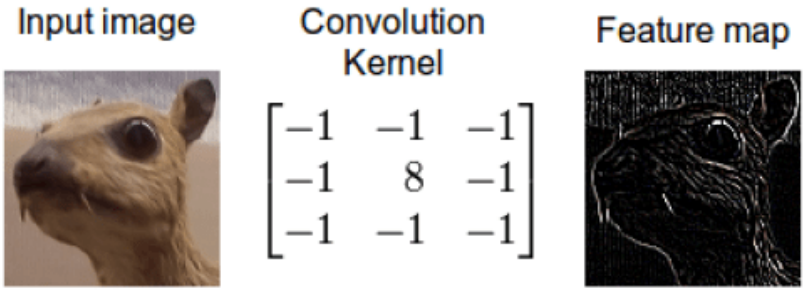

Figure 2. Convolution step [9].

Another operation we use in convolutional neural networks is called ReLU, which is an abbreviation for Rectified Linear Unit. This operation replaces each negative value in the feature maps with a zero. This introduces non-linearity, which will more closely match the operation of neurons in the brain. Neurons do not produce negative output.

$$ReLU(x) = \begin{cases} 0 & if \ \ x < 0 \\ x & if \ \ x \geq 0 \end{cases}$$

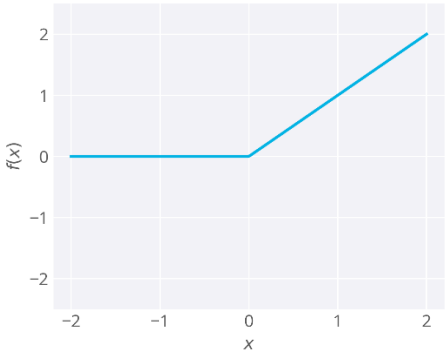

Figure 3. Rectified linear unit (ReLU) [10].

The third operation is called the pooling step. The purpose of this operation is to reduce the dimensionality of the features maps, without obscuring the most important information. Pooling can be done in a few different ways, such as finding the maximum value in each 2x2 matrix of pixels, or finding the average or the sum. When we do this pooling step, we are reducing the size of the feature map by four, but retaining most of the feature information. This makes the input into the neural network more manageable and reduces the number of parameters necessary in the network. This step is also known as the subsampling step.

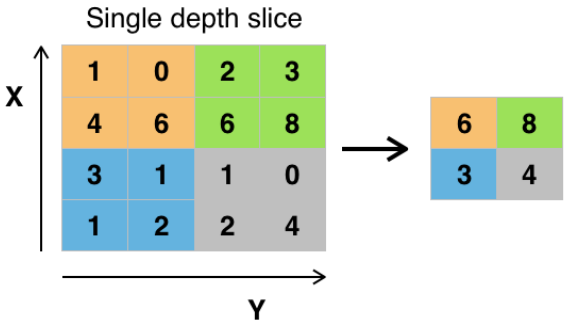

Figure 4. Max pooling [11].

Once we have done the convolution, ReLU, and pooling operations, we have transformed the image into a more useful input, and we can pass that input into the neural network. If we are using the CNN to classify images, we typically use a softmax classifier in the output layer of the network. Softmax is a function that changes the arbitrary values for each category into a vector of values that are between zero and one, and the sum of all dimensions of the vector together is

one. This is similar to a unit length vector, but instead of the length of the vector being one, the sum of the parts is one. It is similar to a probability function in that regard.

From there, we can train the neural network as described above. We use the training data and attempt to classify the images; if the classifier is incorrect, then we can back propagate along the connections. We repeat this process until the classification error is below the threshold.

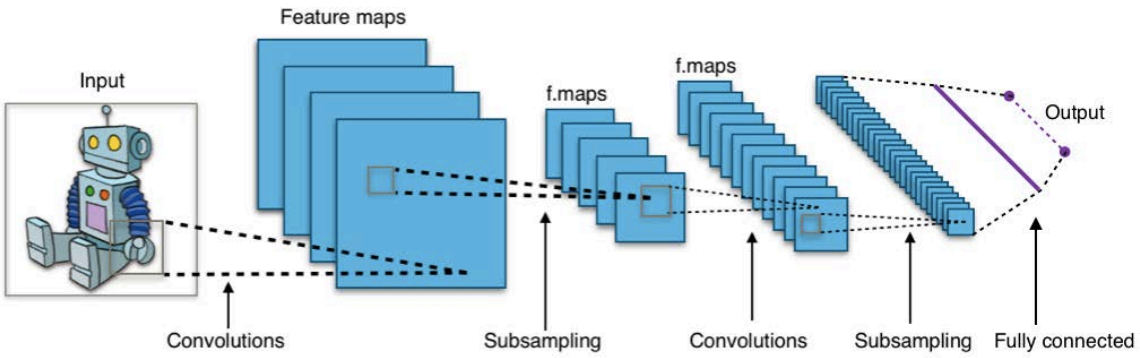

Figure 5. Convolution neural network structure [12].

ResNet

ResNet is the residual convolutional neural network we chose to use for this project. It was developed by a Microsoft research team. This network won 1st place at ILSVRC 2015, which is the ImageNet Large Scale Visual Recognition Challenge [1]. There are a number of conferences that compete on classification challenges; in this instance, they test and train on the same dataset. We knew this model was successful, so we chose this network as our pretrained model. ResNet follows a structure similar to the convolutional neural network described above, but with additional augmentation. In a residual CNN, there are additional shortcut connections between the convolution layers. This allows features to be passed deeper into the network, and increases the performance of the neural network.

ResNet is written in Lua and uses the Torch scientific computing framework. Torch is a LuaJIT framework with an emphasis on machine learning algorithms that utilize GPUs. ResNet is dependent on cuDNN, which is a CUDA library of primitives for deep neural networks. CUDA is a parallel computing platform that streamlines the process of using a GPU for computation. In order to use ResNet in an application, one must train the network on a collection of images.

ResNet was trained and tested on CIFAR-10 and on ImageNet. CIFAR-10 is an image dataset with 60,000 32x32 color images and ten classes [5, 6]. There are 6,000 images per class. CIFAR-10 contains 50,000 training images and 10,000 test images. ImageNet is a much larger dataset with 1,000 classes and larger images [7]. Percent error is obviously much lower on the CIFAR-10 set, as there are only a few categories from which the neural network can choose.

2. Approach for Pre-Trained ResNet Classification

Set Up

Rather than train ResNet on ImageNet ourselves, we instead used pre-trained ResNet models available on the ResNet GitHub [2]. To set this up, we needed a machine with a CUDA-enabled GPU, as well as Lua, Torch, and cuDNN installations and configurations. Once that was set up, we downloaded a few pre-trained models, including the ResNet-18 and ResNet-200 files. These models are eighteen and two hundred layers deep, respectively. They were trained on the ImageNet dataset. They have one crop, top-1 percent errors of 30.43 and 21.66, respectively. In other words, they correctly identify the category of the object 70% and 79% of the time.

When we say one crop, we mean that only one cropped version of the image is passed into the network. The other option would be ten crops, which is where the image is cropped in each corner, as well as a center crop. Then, the image is mirrored horizontally, and cropped in the four corners and the center. These ten crops help to decrease the prediction error in the machine; they reduce the chance that an off-center object in the image is partially cropped or misclassified. Also, by mirroring the image, we mitigate the risk that the network might recognize an object better when oriented a certain direction.

The image below shows how an image would be divided when using ten crops. The figure shows how the first five sub-images would be created, and then the subsequent five are created using a horizontal mirror of the image. In this specific image, the center crop only encompasses a portion of the car, whereas the bottom right corner crop contains the entire vehicle. When we pass the sub-images into the network, the bottom right crop will most likely have a more confident prediction than the center crop.

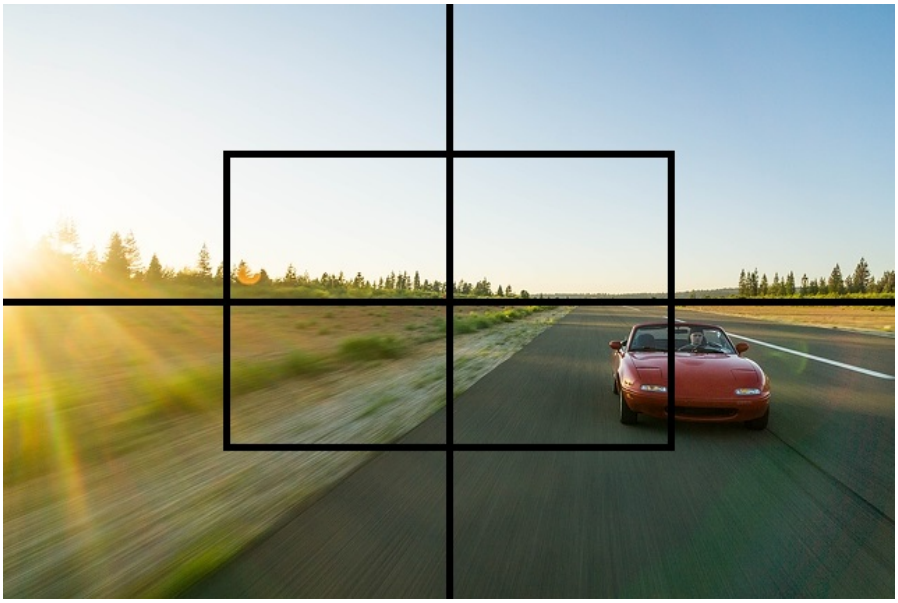

Figure 6. Example of corner crops and center crop.

Data Gathering

Once we set up the Resnet-18 and Resnet-200 models, we needed to obtain test images. We picked ten categories of the 1,000 available, and then we used Google Image Search to get 100 images for each of the ten categories. Once we had a page of search results, we saved the web page and extracted the images from the downloaded directory. These images were the thumbnails generated by Google, so they were approximately 250px by 250px and around 10kb apiece. The categories we picked were barn, forklift, hammer, strawberry, library, pretzel, sports car, street sign, vending machine, and volcano. An example of the search results for “barn” is below. Notice that some of the images are cartoons, or have objects in the foreground. Others could hardly be classified as a barn.

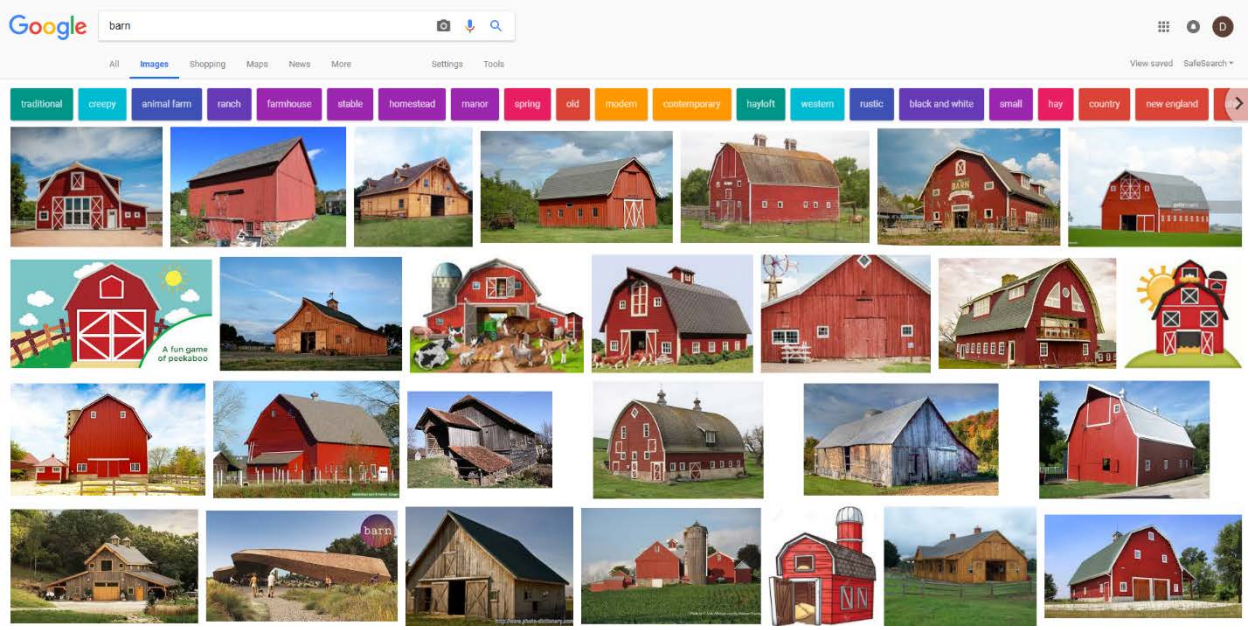

Figure 7. Search results for “barn”.

Running the Experiment

Once we had the images on the machine, we created scripts to automate the classification process. One of the scripts we created allowed us to run the classifier on batches of images. We separated the images into directories with the name of the categories. The script takes a category name as a parameter, and then runs the classifier on each of the test images. We also extended the Lua code for the classifier. We created a procedure to print out the results for the top five predictions per image in a tab separated format, so that it would be easy to duplicate the data in a spreadsheet. We created one last script that would run the classifier on each of the ten categories, and the output of that script was transferred into a spreadsheet for analysis.

The classifier scales the images down to where the smaller edge is at most 256 pixels. It then performs a color normalization based on the mean and standard deviation that were used to train the model originally. After that, it performs a center crop of the image down to 224 pixels. Then, each modified image is passed into the pre-trained residual network, goes through the convolution layers, the neural network, and finally the softmax classifier. We can then output the top predictions of the network and iterate to the next image.

3. Classification Evaluation

Results

For the ten categories, we manually inspected the classification results and counted the number of correct predictions out of the one hundred test images. The highest result was an impressive 92 predictions correct in the “forklift” category, and the lowest result was 61 in the “hammer” category. Out of the 1,000 total images, the network correctly guessed the proper category 775 times. Our results are shown in the figure below.

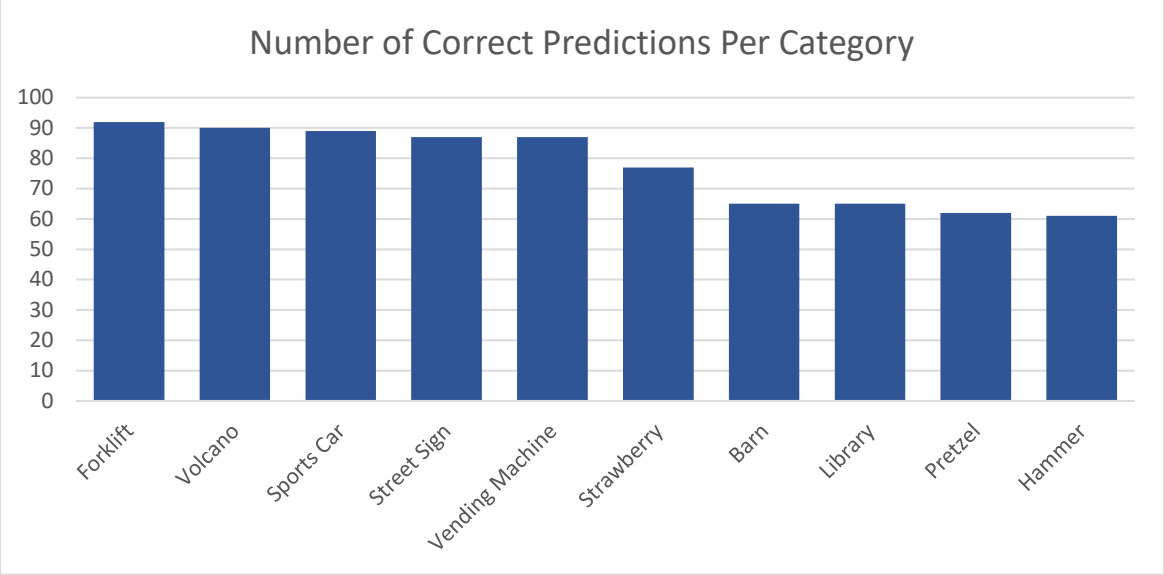

Figure 8. Number of correct predictions per category.

To better understand the ResNet classifications of our Google Image Search results, we created graphs representing the 100 predictions made by the network. Our results for the “barn” images are shown below. The lighter shade indicates the network predicted correctly, while the darker shade indicates an incorrect prediction. The height of the bar represents the confidence level of the prediction. Our results are sorted according to confidence level.

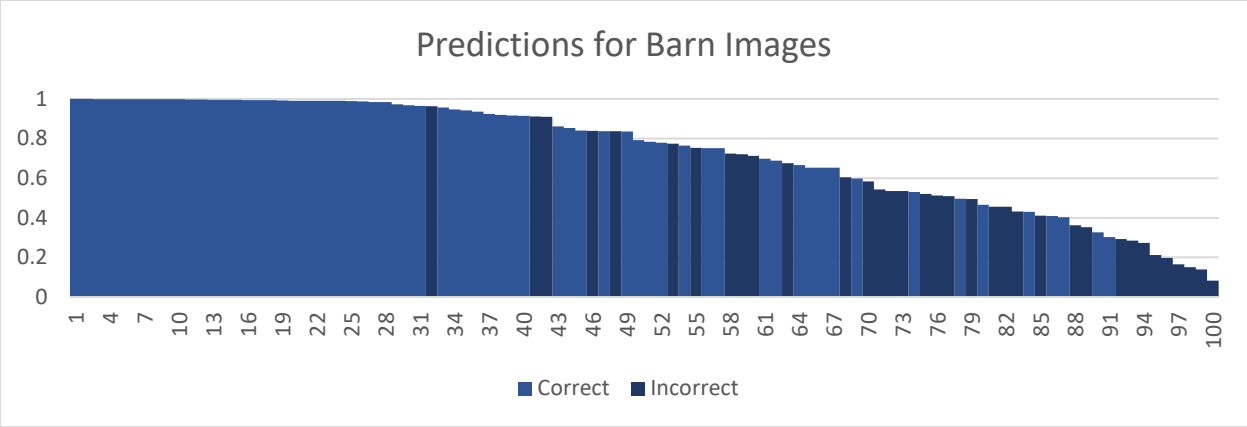

Figure 9. Results for the barn images.

There are two images in this set that were both misclassified, one as a mobile home, and the other as a boathouse. The peculiar part of this occurance is the similarity of the images. The angle of the image, the shape of the building, and even the window placement is nearly identical on the two pictures. However, the network classifies neither of them as barns, and in fact, is quite confident with the incorrect predictions. The network is 91.1% confident the image on the left is a mobile home, and 90.9% the image on the right is a boathouse.

Figure 10. “Mobile home” vs “boathouse”.

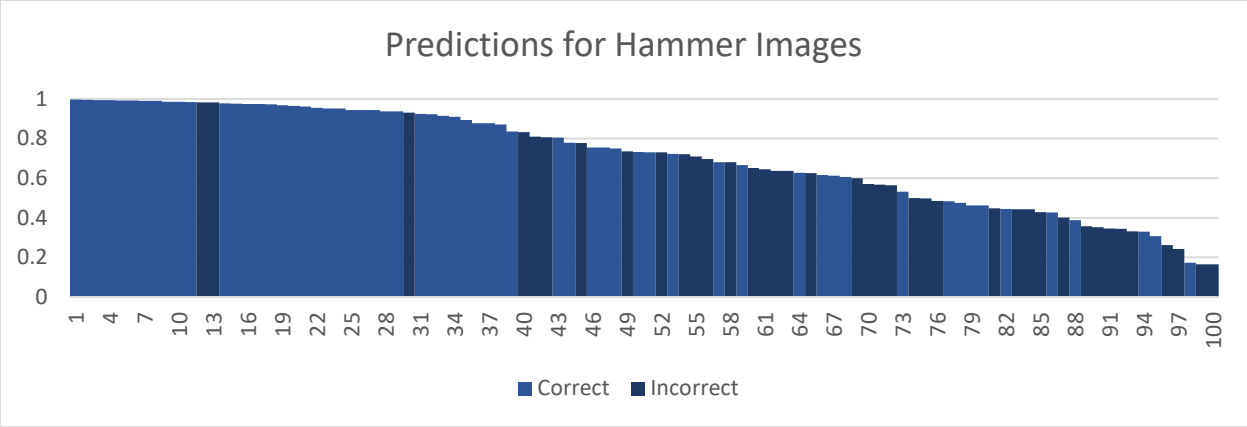

Figure 11. Results for the hammer images.

Our classification results for “hammer” images are shown above. The hammer dataset has a few different types of misclassifications. One of these incorrect predictions was “maraca” on an image of a wooden mallet. The mallet surely looks more like a wooden maraca than a claw hammer, and yet a human would likely still classify the image as a hammer. This is an example of how subtle features of the image can differentiate two very different objects, and these features are not always detected by the ResNet image classifier.

Figure 12. Wooden mallet classified as “maraca”.

There was also an instance of Google Image Search returning an image that matches the result based on textual clues, but not necessarily features of the image. Among the images of hammers, there was an image of a hammer drill. The network classifies this image as a power drill, which I believe is what a human would likely do as well. This is an example of the search results showing a picture that is only partly described by the search term.

Figure 13. Hammer drill classified as “power drill”.

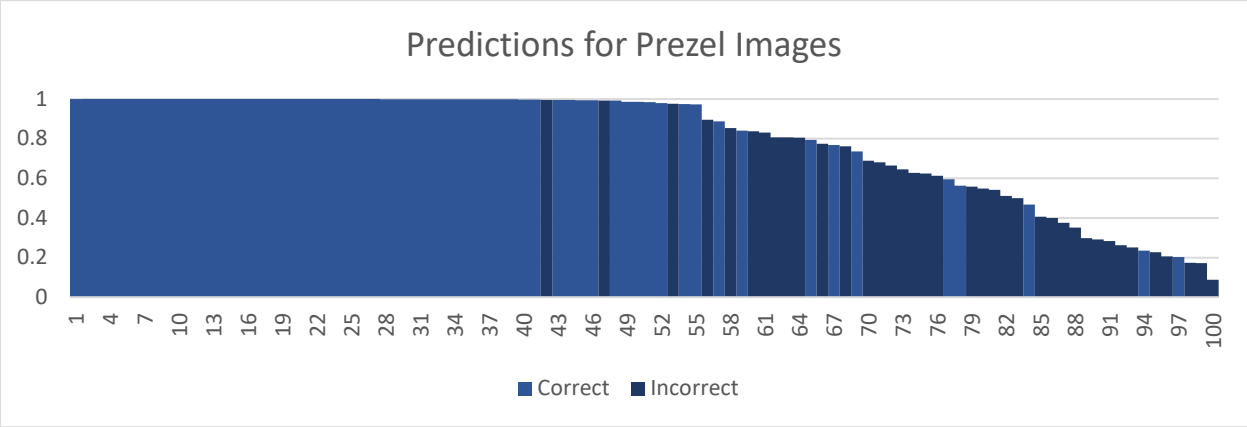

Figure 14. Results for the pretzel images.

Above, we have the results for the “pretzel” images. There were no misclassifications for the first forty-one images. However, as the confidence of the classifier dropped, the number of misclassifications rose. Only ten of the forty-five least confident predictions were correct predictions.

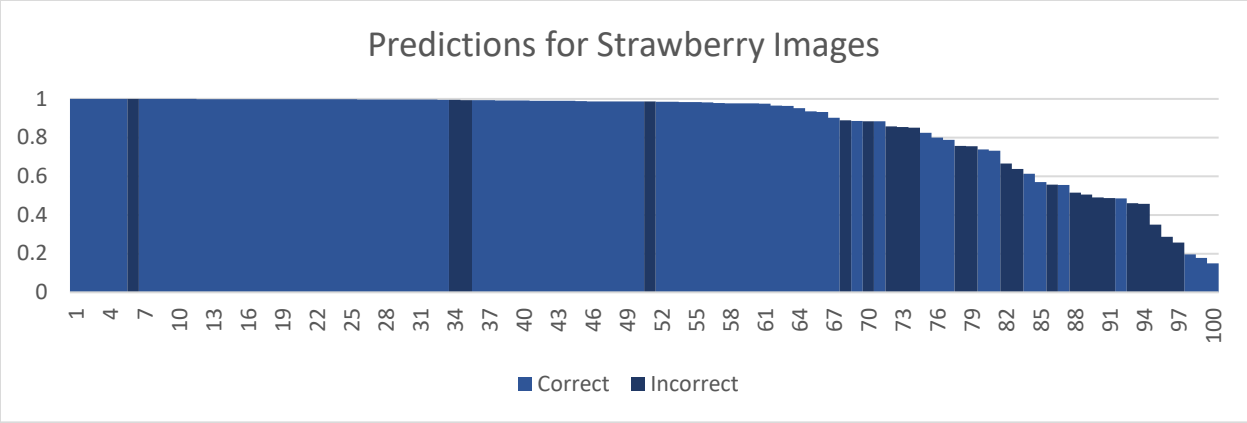

Figure 15. Results for the strawberry images.

Just like in the “hammer” dataset, there were a few images in the “strawberry” set that were given incorrect classifiers and yet it can be argued that the ResNet classifier did not make an error. There were simply results in the dataset that more closely resembled objects from other categories. For example, there were several images of strawberry trifles, and trifle was another category that the network could choose.

Figure 16. Strawberry search results returned trifle.

4. Approach for Similarity Analysis

Image Similarity Analysis Based on ResNet Classification

One of the limitations of the ResNet classifier, and many classifiers in general, is that there are only a select number of categories from which to choose. For ResNet, there are 1,000 possible predictions that the network might make. If we input an image that is not a member of any of the categories, then the network cannot possibly make the correct prediction. However, the network will still give confidence values for the existing categories. We can leverage these predictions to get more information about the image or set of images that do not fit into the predetermined categories.

We created a new version of the classifier that takes in a set of images. This classifier calculates the distance in feature space between each input image with each other input image. When we pass a single image into the classifier, it returns a vector with 1,000 dimensions. Each dimension has a unique label from the set of categories, and the value that the network assigned the possibility for that label. The sum of the values for all 1,000 dimensions is one. We could find the Euclidean distance between each of these 1000-dimensional vectors, but nearly all of the dimensions have miniscule values. Only the top few labels contain meaningful values. Because of this, we implemented a custom distance function that calculates the distance between two vectors based on shared labels in the top fifty highest prediction categories. We wrote this custom distance function in Lua, and call it from within the classifier code. It finds the distance between each pair of images and outputs it in a format that we can read with our visualization program.

Running the Experiment

We used a modified version of our previous batch script to classify all the images in a category. In this experiment, we gather the top fifty predictions made by the classifier instead of only the top five. Once we have these top fifty categories and confidences for each image, we can calculate the dot product of the two feature vectors we are comparing using our custom distance procedure. This dot product will only multiply together values of dimensions with the same label. This creates a distance function which returns a higher value when two vectors have similar labels with higher prediction values. When two vectors have no matching labels in their

top predictions, we return a value of zero, signifying that these images are not close together in the feature space.

To visualize the distance between all one hundred images in each of our experiments, we created a 100 by 100 matrix where the value at each location corresponds to the distance in feature space for the image in that row and the image in that column. With this information, we can find the sum of each row to determine which images are closer, on average, to each other image in the set. We then sorted this matrix to create an ordered list of overall similarity of each of the images. The images at the top of this sorted list are going to be the images that share the most features in common with each other image.

We created a visualization of this data to make it easier to interpret the results. This visualization was created by transforming the output of the distance analysis classifier. We took the row, column, and float value output by the classifier and multiplied each float by 255, and then put those values into a matrix that contained values between 0 and 255. We then output this matrix into a PGM file, which is a simple, portable grayscale file that we can view as an image.

5. Similarity Analysis Evaluation

Visualizations

Below, we have the visualization for the strawberry similarity experiment we ran on the one hundred strawberry images. On the left, the rows are sorted by image order. On the right, the rows are sorted by most similar to least similar, as detailed above.

White represents values closer to 255, which means the dot product between those two images returned a value close to one. Darker values represent images that were not as similar. The diagonal, which represents when an image is compared with itself, is black. On the sorted visualization, we see black vertical bars when we compare the base image to an image that is not similar. We can see from this visualization that there are not any images that are similar to all other images in the set.

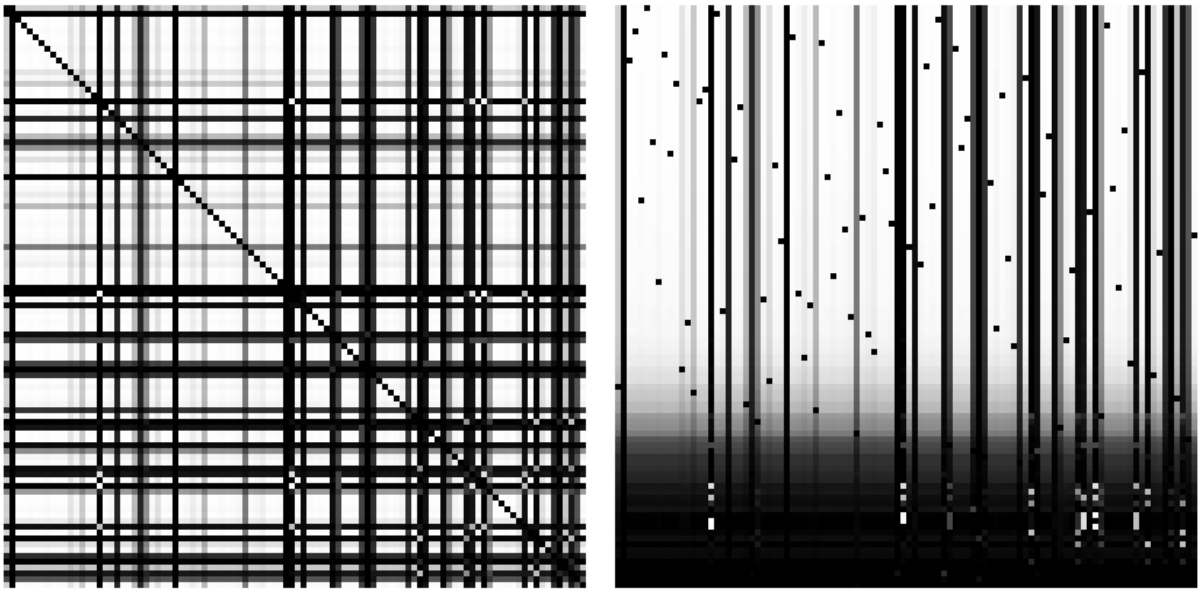

Figure 17. Visualization of distances between strawberry images, unsorted vs sorted.

The visualization of the strawberry distances is mostly white. This is because all of the images are supposed to be strawberries, and so a large percentage of the predictions made by the network have a high confidence value for the strawberry label. On the sorted visualization, we notice that the first approximately seventy rows have very similar column values. The column value is black when the two images share no common features, or dark gray when any shared features have lower confidence values.

Figure 18. Top four most similar strawberry images.

Here we have the four images that correspond to the top four most similar strawberry images in our distance analysis. In this case, the network was extremely confident that the

images were strawberries. They have confidence values ranging between 0.99996 and 0.99984. The four images are obviously quite similar, with three of them containing a single strawberry, and the other image containing a pair.

Below, we have the four images that were marked as least similar by the distance analysis. The predictions for these images were website, pinwheel, hair slide, and lipstick. These images shared very few prominent features with the data set, and therefore ranked poorly in the distance analysis.

Figure 19. Bottom four least similar strawberry images.

6. Extending Similarity Analysis

Extending Similarity Analysis Beyond ResNet Categories

In our final experiment, we wanted to analyze the performance of the ResNet classifier on a collection of images that were not one of the 1,000 ResNet categories. Naturally, we do not expect the ResNet classifier to correctly identify the image category, but we do expect the category labels to contain information that will allow us to find image similarity within the one hundred Google images.

We ran the same experiment on a set of 100 images from the search term “Pompeii”. Running that experiment produces these visualizations. On the left, we have the unsorted version of the data, which has scattered gray and white data points throughout the image. When we sort this data based on the sums of values in the rows, we get a more meaningful image.

In the image on the right, we can see white vertical bands formed at the top of the image and slowly getting darker as they go down the column. These lighter colored bands represent images that are similar to each of the top approximately twenty images in the data set.

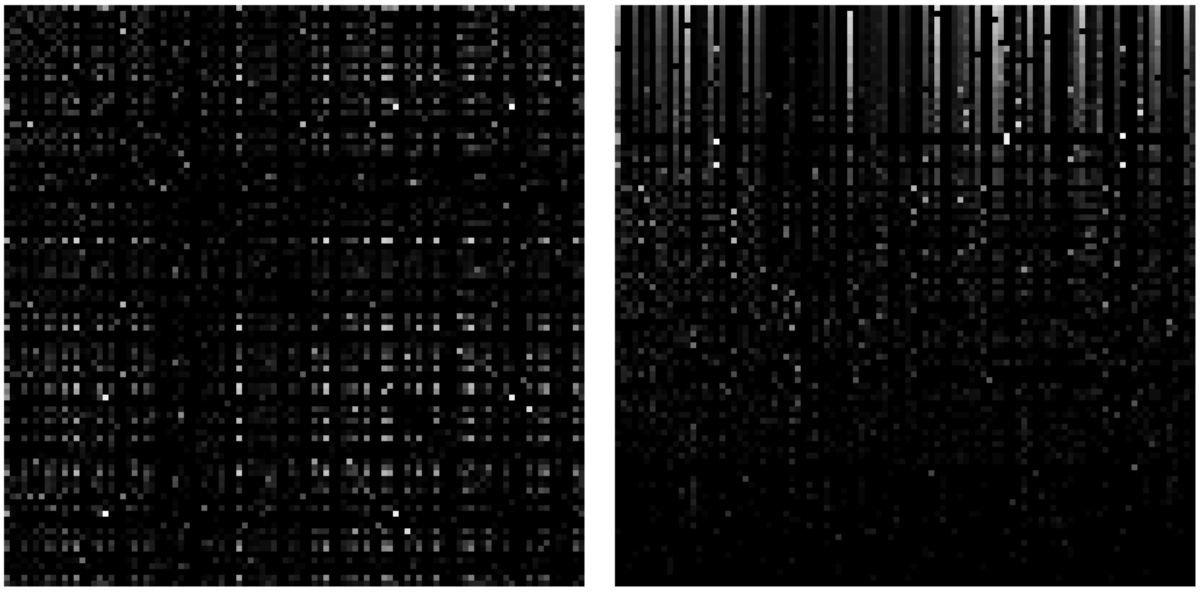

Figure 20. Visualization of distances between Pompeii images, unsorted vs sorted.

These four images are the pictures that ranked highest in the Pompeii data set. The most prominent prediction shared between them is “monument”. Also, three of the four feature vectors produced by the network contained monument, castle, palace, and mosque in the top five values. The fourth feature vector contained monument, castle, and palace in the top five. These features must make up the description of buildings depicted. It is interesting that each image is composed of a set of buildings in the foreground, with a blue sky in the background.

Figure 21. Top four most similar Pompeii images.

The four least similar images in the Pompeii data set ranged from a picture of a museum exhibit to a blurry movie poster. These images show how Google uses image captions to determine the contents of a photo. The links to the images have titles such as “Pompeii | Sony Pictures” and “Pompeii: The Exhibition”.

Figure 22. Bottom four least similar Pompeii images.

7. Conclusions

In this research, we compared the results of ResNet image classification with the results of Google Image search. We classified 1,000 images from ten different search terms on Google Image search. Overall, the ResNet classifier had a high agreement with the Google category names, correctly predicting the category of the image 77.5% of the time.

When we analyzed the results of the ResNet classification on Google Image search results, there were several types of misclassification. In some cases, the classifier simply predicted the wrong category for an image. In the case of the wooden mallet, this image likely did not match the training data for “hammer”, and so the network made the wrong prediction. For the hammer drill, Google Image search returned an image that can be partially described by the word “hammer”. However, that image is not a hammer, and therefore the classifier did not categorize it as such. There were also images of strawberry trifles and strawberry lipstick. Again, these images can be partially described as “strawberry”, but that is not a complete description of the object in the picture.

There was also a correlation between the confidence of the ResNet classifier and the likelihood that the top prediction was correct. For each of the categories, when sorted by confidence level, the top half of the list contains a much higher percentage of correct predictions. This is especially true for the “pretzel” category. This correlation seems intuitive, because confidence is a strong factor when considering whether a human’s prediction is likely to be correct.

When we calculated the similarity between the “Pompeii” images, we noticed that the images that were most similar to the data set as a whole shared four common classifiers. These classifiers were monument, castle, palace, and mosque. This makes sense, because all four images contain ancient buildings as the main focus.

Future Work

One possible use of these calculated distances would be to create a minimum spanning tree of the feature space. Once we have created this tree, we could identify clusters of images that are similar, as well as images that create links between separate clusters. If we remove the links that weakly connect separate clusters, we could create new classifiers for the images that are contained within those clusters. This method could be used to create new categories that are not available within the 1,000 classes, but describe the images in the cluster.

One could programmatically have the classifier create new categories by automatically creating these spanning trees of datasets and separating out the clusters. These clusters could be uniquely named, and the classifier could store which set of features the images in that cluster shared. The classifier would then have a new category to choose if an image input into the network was determined to have that set of features.

8. References

[1]K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition", 2015.

[2]"ResNet Training in Torch", GitHub, 2018. [Online]. Available: https://github.com/facebook/fb.resnet.torch. [Accessed: 16- Apr- 2018].

[3]"A Quick Introduction to Neural Networks", the data science blog, 2016. [Online]. Available: https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/. [Accessed: 16- Apr- 2018].

[4]"An Intuitive Explanation of Convolutional Neural Networks", the data science blog, 2016. [Online]. Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/. [Accessed: 16- Apr- 2018].

[5]"CIFAR-10 and CIFAR-100 datasets", Alex Krizhevsky - Toronto, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html. [Accessed: 16- Apr- 2018].

[6]A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", 2009.

[7]"ImageNet", 2016. [Online]. Available: http://www.image-net.org/. [Accessed: 16- Apr- 2018].

[8]"File:Colored neural network.svg - Wikimedia Commons", Commons.wikimedia.org, 2013. [Online]. Available: https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg. [Accessed: 16- Apr- 2018].

[9]T. Dettmers, "Understanding Convolution in Deep Learning", 2015. [Online]. Available: http://timdettmers.com/2015/03/26/convolution-deep-learning/. [Accessed: 16- Apr- 2018].

[10]"Deep learning concepts", Towards Data Science, 2017. [Online]. Available: https://towardsdatascience.com/deep-learning-concepts-part-1-ea0b14b234c8. [Accessed: 16- Apr- 2018].

[11]"File:Max pooling.png - Wikimedia Commons", Commons.wikimedia.org, 2015. [Online]. Available: https://commons.wikimedia.org/w/index.php?curid=45673581. [Accessed: 16- Apr- 2018].

[12]"File:Typical cnn.png - Wikimedia Commons", Commons.wikimedia.org, 2015. [Online]. Available: https://commons.wikimedia.org/w/index.php?curid=45679374. [Accessed: 16- Apr- 2018].