Image Reduction & Classification

David Anderton and Ray Dedhia

***Abstract*— Google Cloud Vision classifies images into categories and detects objects and faces in images using "powerful machine learning models."[1] In this paper, we examine how well Google Cloud Vision understands the content of compressed images. In addition, by compressing each of our images by several different amounts, we sought to determine how much one can compress an image until Google Cloud Vision cannot determine its the content. We also used different methods of compression to see if there is a specific image compression method or methods that maintain(s) the legibility of images for Google Cloud Vision.**

I. INTRODUCTION

We used 22 unique images in our study, from four major categories: six animals, six famous landmarks, five landscapes, five street signs. In our study, we used both the color and grayscale versions of our images, resulting in a total of 44 images.

To compress our images, we used five main methods: (1) posterization through k-means clustering, (2) low-rank approximation using PCA, (3) the "mean pooling" operationcommonly used in neural networks, (4) the command-line program mogrify and (5) dropout (randomly removing a specific percentage of the pixels).

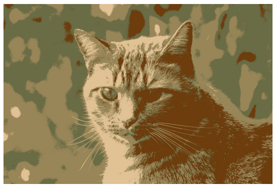
We used Google Cloud Vision to classify our images before and after compression.

II. COMPRESSION METHODS

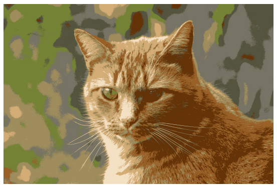
A. Posterization

We used k-means clustering to implement a "posterization" effect. K-means clustering is an algorithm that groups a set of data points into k groups by mapping each data point to one of k centroids such that the distance between each data point and its centroid is minimized. It does this by randomly generating k centroids, (1) assigning each data point to the nearest centroid, (2) updating the centroid values be setting them equal to the arithmetic mean of the data points assigned to them in step 1, and repeating steps 1 and 2 until some stopping criteria has been met.

We stopped our k-means clustering algorithm after 10 iterations to maximize efficiency, and used $k = 4$ and $k = 8$. Below are the results of putting an image of a cat through 4-means and 8-means clustering.



$k = 4$



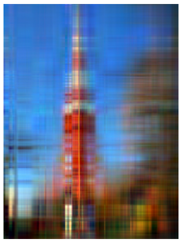
$k = 8$

B. PCA

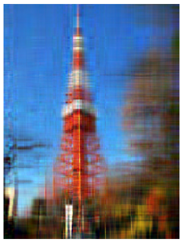
The principle components of some matrix M are its left and right singular vectors (u_i and v_i respectively) and its singular values ($sigma$). The decomposition of a matrix into these components is referred to as the Singular Value Decomposition. Principle Component Analysis (PCA) is a form of data analysis that uses the largest principle components of a matrix of data in order to analyze the data.

By the Eckart-Young theorem, the rank k matrix M_k closest to M is the sum of the product of the k largest singular values of M with their corresponding left and right singular vectors. That is, $M_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$.

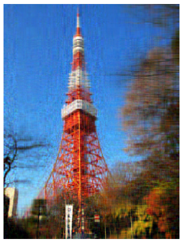
Thus, PCA can be used to find the best low-rank approximation for a matrix. We used PCA to reduce our image matrices to rank-5, rank-10 and rank-25 matrices. Below are the results of calculating the best rank-5, rank-10 and rank-25 approximations of an image of the Tokyo Tower.



rank-5



rank-10



rank-25

C. Mean Pooling

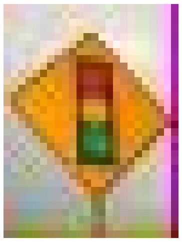
"Mean pooling" refers to the operation of dividing an image into regions of size $m \times n$, calculating the arithmetic mean of those regions, and outputting an image consisting of just those calculating averages.[2]

We used regions of size 16×16, 32×32 and 64×64 filters Below are the results of putting an image of a lights

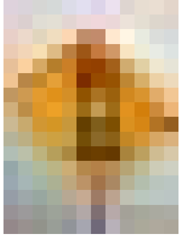
[1] https://cloud.google.com/vision/

[2] http://ufldl.stanford.edu/tutorial/supervised/Pooling/

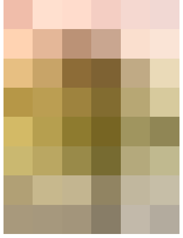
sign through 16×16, 32×32 and 64×64 mean pooling.



16×16



32×32

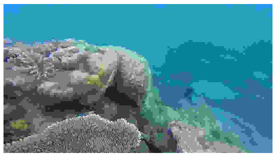


64×64

D. Mogrify

The mogrify program can be used to perform a number of operations on images, such as blurring, cropping, and compression.[3] We used mogrify to compress our images using the option `-quality [num]`[4]. For JPEG and MPEG images, the quality goes from 1 to 100. We used 1, the lowest image quality. Below is an image of the ocean before and after it has been compressed by the command `mogrify -quality 1`.

before

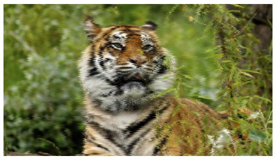


after

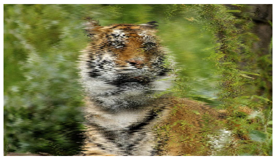
E. Dropout

Inspired by dropout layers in neural networks, which randomly remove nodes in the neural network to prevent overfitting, we implemented an algorithm that randomly removes some percentage p of the pixels in an image. It does this by removing some percentage p of the columns and some percentage p of the pixels from each of the remaining columns.

We implemented this algorithm with $p = 0.2, 0.5$. Below are the results of putting an image of a tiger through dropout with $p = 0.2$ and $p = 0.5$.



$p = 0.2$



$p = 0.5$

III. Classification

A. Before Compression

Google Cloud Vision assigned each image a set of labels. All of the labels assigned to the color images were accurate or close to accurate, and for each color image, at least one label closely identified the image (e.g. “tiger” for the iamge of a tiger and “tower” for the image of the Tokyo Tower).

However, all of the 22 grayscale images were mis-labeled as “monochrome photography,” 9 of them were not closely identified, and 4 of them were only given incorrect labels.

B. After Compression

method of compression	**fraction of images given a subset of correct labels**	**fraction of images given the most accurate label given to their un-compressed versions**
mogrify	18/22	15/22
dropout (20%)	17/22	15/22
dropout (50%)	13/22	8/22
pooling (16x16)	20/22	14/22
pooling (32x32)	9/22	4/22
pooling (64x64)	1/22	0/22
PCA (25)	21/22	18/22
PCA (10)	17/22	9/22
PCA (5)	6/22	0/22
clustering (8)	22/22	20/22
clustering (4)	19/22	15/22

(Note: this table only analyzes the color images.)

As expected, the less the image was compressed, the more accurate the labels it was assigned were. In addition, larger images had significantly higher accuracy rates than smaller images.

From the data, it appears that k-means clustering was the best compression method in terms of maintaining the legibility of the images for Google Cloud Vision. In a sense, this makes sense, because posterization generally simplifies images while retaining the overall shapes in the images.

In future analyses, it may be helpful to estalish a more specific definition of accuracy and a way to compare the accuracy of Google Cloud Vision when classifying images before and after compression, as the table above does not consider a number of factors, such as the certainty of different labels, and the number of completely incorrect labels assigned to images.

IV. Conclusions

This information is useful because data reduction has been used as a method to fight adversarial attacks, in which stickers, random noise, etc. are added to images, causing neural networks to misclassify them.[5] [6] For exmaple, putting adversarial stickers on stop signs can cause neural networks to be unable to detect stop signs.[7] The amount by which one can compress an image before it becomes illegible to neural networks, as well as the best method to use to compress an image to maintain legibility, can inform methods that use data reduction to fight adversarial attacks.

[3] https://www.imagemagick.org/script/mogrify.php

[4] https://www.imagemagick.org/script/command-line-options.php#quality

[5] https://openreview.net/pdf?id=S10qYwywf

[6] https://www.researchgate.net/publication/315890594_Dimensionality_Reduction_as_a_Defense_against_Evasion_Attacks_on_Machine_Learning_Classifiers

[7] http://bair.berkeley.edu/blog/2017/12/30/yolo-attack/

ACKNOWLEDGMENT

We would like to thank Prof. Strang for his support and interesting and informative lectures.

REFERENCES

[1] “Cloud Vision API.” Google Cloud. https://cloud.google.com/vision/

[2] “Pooling.” UFLDL Tutorial. http://ufldl.stanford.edu/tutorial/supervised/Pooling/

[3] “mogrify.” ImageMagick. https://www.imagemagick.org/script/mogrify.php

[4] “command line options.” ImageMagick. https://www.imagemagick.org/script/command-line-options.php#quality

[5] Gopalakrishnan, Soorya, et. al. “Combating Adversarial Attacks Using Sparse Representations.” ICLR 2018. https://openreview.net/pdf?id=S10qYwywf

[6] Bhagoji, Arjun, et. al. “Dimensionality Reduction as a Defense against Evasion Attacks on Machine Learning Classifiers.” https://www.researchgate.net/publication/315890594_Dimensionality_Reduction_as_a_Defense_against_Evasion_Attacks_on_Machine_Learning_Classifiers

[7] Evtimov, Ivan, et. al. “Physical Adversarial Examples Against Deep Learning Networks.” Berkeley Artificial Intelligence Research. Published 30 December 2017. http://bair.berkeley.edu/blog/2017/12/30/yolo-attack/