Image Recognition and Labeling in Reduced Dimensions

David Anderton and Ray Dedhia

***Abstract*— As the use of artificial intelligence continues to expand into the realm of image compression[1][2], questions are being raised on how well the original artefacts and nature of the source image are preserved. The consequences of being able to utilize smaller output images that maintain the majority of pertinent information from their often significantly larger source images are huge. With smaller inputs, applications from self driving cars that rely on camera input, autonomous flight, through to training neural networks could require significantly less processing time and resources.**

An interactive exploration of the results is available at the following URI: `https://mit18065-dimensions.herokuapp.com/`

I. INTRODUCTION

In this paper, we briefly examine how the Google Cloud Vision API responds to the content of a selection of images pre- and post-compression. We also consider a variety of compression techniques and extents. The goal: produce a broad estimate of how much an image can be compressed until the Google Cloud Vision API, as a proxy for other classification algorithms, can no longer determine useful insights about the constitution of the image.

II. METHODOLOGY

Our methodology was as follows:

Within our study we utilised 22 unique images, from four major categories: six animals, six famous landmarks, five landscapes, five street signs. Expanding this to both the color and grayscale versions of these images, we had a resulting set of 44 source images.

After collating the source images we sent them through the Google Cloud Vision API[3] and get a control set of labels for the 44 images. Then compress the images, using the five following methods: (1) Posterization through k-means clustering, (2) :ow-rank approximation using Principle Component Analysis (PCA), (3) "Mean pooling" operation commonly used in neural networks, (4) Mogrify (a command-line utility) and (5) Dropout (randomly removing a specific percentage of the pixels).

Once the images were compressed in all of the above methods we then re-queried the Google Cloud Vision API to re-classify the images post-compression and considered the results.

III. COMPRESSION METHODS

A. Posterization

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. The resulting k-means clustering implemented a posterization effect.

We arbitrarily stopped our k-means clustering algorithm after 10 iterations as an intuitive stopping point for efficiency, and used $k = 4$ and $k = 8$. In Fig. 1 below are the results of putting an image of a cat through 4-means and 8-means clustering.

[1]See: `https://ai.googleblog.com/2016/09/image-compression-with-neural-networks.html`

[2]See: `http://www.wave.one/icml2017/`

[3]The Google Cloud Vision API, `https://cloud.google.com/vision/`, classifies images into categories and detects objects and faces in images utilising non-public machine learning models.

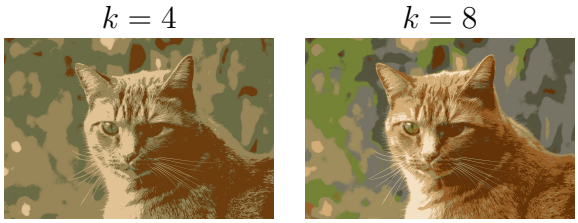

Fig. 1: Image of a cat compressed using 4-means and 8-means clustering.

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 most commonly as the Singular Value Decomposition, or SVD. PCA is a form of matrix analysis that utilizes the largest principle components of a matrix of data in order to create a summarization or approximate the underlying data.

By utilising the Eckart-Young theorem[4], 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 utilised to find the best low-rank approximation for M. Within our study we used PCA to reduce our image matrices to rank-5, rank-10 and rank-25 matrices. Fig. 2 below shows the results of calculating the best rank-5, rank-10, and rank-25 approximations of an image of the Tokyo Tower.

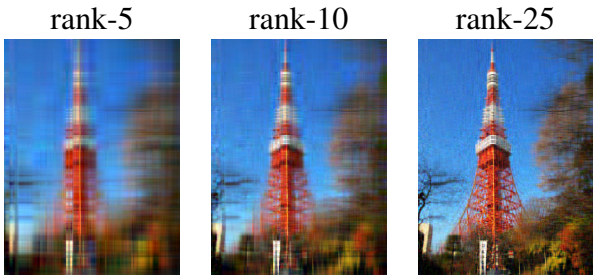

Fig. 2: Best rank-5, rank-10, and rank-25 matrix approximations of an image of the Tokyo Tower.

[4]See Golub et al. `https://www.ime.usp.br/~jstern/miscellanea/seminario/Golub87.pdf`

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 calculated averages.[5]

For our study we considered regions of size 16×16, 32×32 and 64×64. In Fig. 3 below are the results of putting an image of a US traffic light sign through a 16×16, 32×32, and 64×64 mean pooling filter.

Fig. 3: Image of a US traffic put through 16×16, 32×32, and 64×64 mean pooling filters.

D. Mogrify

The mogrify command line utility, part of the Image Magick suite of tools[6], can be used to perform a number of operations on images, such as blurring, cropping, and compression. We used mogrify to compress our images using the option `-quality [num]`[7]. For JPEG image compression the expected value of `[num]` is an integer from 1 to 100. We used the value of 1, that applies the most extreme JPEG compression from mogrify. In Fig. 4 below, an image of the ocean before and after it has been compressed by the command

[5]See the UFLDL Tutorial on Pooling: `http://ufldl.stanford.edu/tutorial/supervised/Pooling/`

[6]See ImageMagick's documentation of the mogrify command: `https://www.imagemagick.org/script/mogrify.php`

[7]See ImageMagick's documentation of the quality option: `https://www.imagemagick.org/script/command-line-options.php\#quality`

`mogrify -quality 1` can be considered.

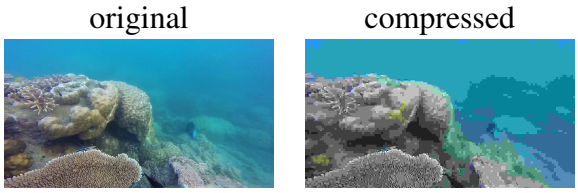

Fig. 4: Image of the ocean that has been compressed with the command `mogrify -quality 1`.

E. Dropout

Inspired by the concept of dropout layers in neural networks[8], which randomly remove nodes in the neural network to prevent overfitting, we implemented an algorithm that randomly removes percentage p of the pixels in an image. It does this by removing some percentage p of the columns and the same percentage p of the pixels from each of the remaining columns.

We implemented this algorithm with $p = 0.2, 0.5$, where p is a value between 0 and 1. In Fig. 5 the results of putting an image of a tiger through dropout with $p = 0.2$ and $p = 0.5$ can be considered.

Fig. 5: Image of a tiger put through dropout with $p = 0.2, 0.4$.

IV. RESULTS

A. Pre-Compression

Once the collection of images in the various categories had been located through Creative Commons Image Search[9] the images were then sent via http request to the Google Cloud Vision API and returned with an assigned set of labels and respective confidence score. All of the labels assigned to the color images were found to be intuitively accurate and relevant by the authors, and for each color image, at least one label closely identified the image (e.g. “tiger” for the image of a tiger and “tower” for the image of the Tokyo Tower).

However, not all the grayscale versions of images were accurately classified according to the authors intuition. Generic labels such as “black and white”, “black”, “white”, “art”, and “monochrome photography” were prevalent throughout grayscale images - a behavior noted later in this paper where the artefacts of compression were noted as an classifiable part of the image. According to the authors judgement 40.9% of the grayscale images were not closely identified, and 18.2% of them were given solely inaccurate labels.

B. Post-Compression

Table 1 and 2 below display our analysis of how accurately images are labeled by the Google Cloud Vision API after they are compressed using each the methods detailed earlier in the paper. We define accuracy as the coincidence of labels between pre- and post-compression images. We ignored more general labels such as “photography,” “black,” and “sky,” instead focusing on labels that either correctly or falsely identify specific objects,

[8]See Srivastava et al.: `http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf`

[9]See: `https://search.creativecommons.org/`

landforms, etc. in the images.

Color Images

method of compression	**% images given at least one correct label**	**% images given ”best” label**
Mogrify	81.8%	68.2%
Dropout (20%)	77.3%	68.2%
Dropout (50%)	59.1%	36.4%
Pooling (16x16)	90.9%	63.6%
Pooling (32x32)	40.9%	18.2%
Pooling (64x64)	4.5%	0%
PCA (25)	95.5%	81.8%
PCA (10)	77.3%	40.9%
PCA (5)	27.3%	0%
Clustering (8)	100%	90.9%
Clustering (4)	86.4%	68.2%

Table 1: Success rate for color images labeled by Google Cloud Vision API after image compression.

Grayscale Images

method of compression	**% images given at least one correct label**	**% images given best label**
Mogrify	59.1%	40.9%
Dropout (20%)	63.6%	27.3%
Dropout (50%)	22.7%	0%
Pooling (16x16)	31.8%	22.7%
Pooling (32x32)	4.5%	4.5%
Pooling (64x64)	0%	0%
PCA (25)	68.2%	36.4%
PCA (10)	31.8%	22.7%
PCA (5)	18.2%	4.5%
Clustering (8)	90.9%	50%
Clustering (4)	59.1%	45.5%

Table 2: Success rate for grayscale images labeled by Google Cloud Vision API after image compression.

V. ANALYSIS

As expected, the less the image was compressed, the more accurate the labels it was assigned were. However, with PCA compression often the intuitive most important entity of the image would gain a higher score. Grayscale images were also given significantly less accurate labels than the color images and, as mentioned previously, the artefacts of grayscale compression dominated the assigned labels - such as “black and white” and “monochrome photography”.

Of the four categories, the Animal dataset had the best performance, most likely because the animal images had only one dominant subject, making

them easier for classification to occur and return an intuitively accurate label. In addition, each form of compression had its own set of behaviors that the authors noted.

The smoothening effect of posterization, using the k-means clustering algorithm, was noted to occasionally removed textures that Google Cloud Vision API appeared to utilize to correctly classify parts of an image. For example, the image of the waterfall, after it was put through the 4-means and 8-means clustering algorithm, was assigned the label "snow" by the Google Cloud Vision API seemingly because the smoothened water color became white and looked more like snow (see Fig. 6 below).

Fig. 6: Potential over-smoothening of an image of a waterfall as a result of the 8-means clustering algorithm.

Low-rank image matrix approximation using PCA often resulted in image distortion in a plaid-like pattern, especially in smaller images (see Fig. 7 below). This resulted in misclassifications where again the artefacts of the compression began to dominate the labelling. Examples such as images in all four categories false labels such as "cloth," "pattern," and "line". One of the more interesting instances of this was where the image of Piccadilly Circus in London became labeled as "tartan" at rank 5.

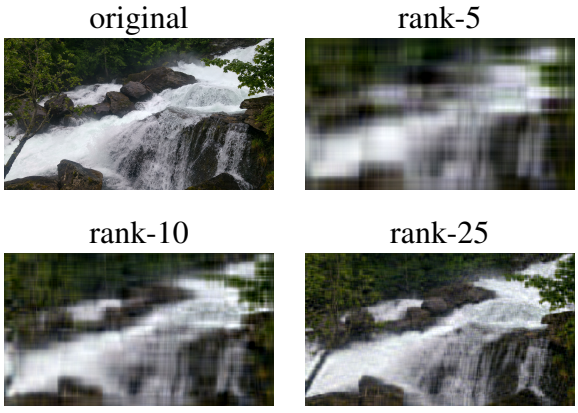

Fig. 7: Distortions in the low-rank matrix approximations of an image of a waterfall.

The "mean pooling" operation with a 64×64 filter, and, in some cases, the 32×32 filter as well, blurred to the extent that they became unrecognizable and Google Cloud Vision API could no longer give them labels that accurately represented the original source image's content (see Fig. 3). When the filter being used with the "mean pooling" operation is too large proportional to the size of the image, it seems to remove all useful information to derive insight from in the image.

Image compression using the `mogrify -quality 1` command occasionally resulted in random bright colors showing up in the compressed images that did not occur in the original images (see Fig. 8 below).

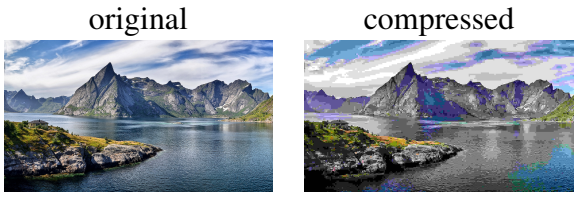

Fig. 8: Purple highlights appear in an image of mountains after compression using the command `mogrify -quality 1`.

The dropout algorithm resulted in distortion of the compressed images, almost as though the images were being reflected on moving water. Issues with classification from this could potentially be alleviated by adding a smoothening factor to our dropout algorithm, for example by using an algorithm such as seam-carving[10]

VI. CONCLUSION

Overall, 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 API. Reflecting on the details of each compression techniques this starts to become a more natural hypothesis, as posterization generally simplifies images whilst retaining the overall dominant shapes in the images.

[10]For a further exploration of seam-carving, see: `https://cs.brown.edu/courses/cs129/results/proj3/taox/`

Our initial recommendation is that using at least 8-means clustering to reduce dimensionality of image data can maintain the majority of labels from Google Cloud API. However as images become busier with a richer set of entities it is important that the value of k is increased. The exact optimum value of k could be an interesting research question to explore.

As this study was mainly based on the author's intuition in terms of defining an "appropriate" or "accurate" label future analysis should amend for this short coming - perhaps though establishing a more specific definition of accuracy using group or mechanical turk[11] rankings and labelling of images. This would provide us with a more reliable and scaleable "ground truth" for understanding the accuracy of Google Cloud Vision API's classification of images pre- and post-compression. The table above also does not consider a number of factors, such as the certainty of different labels, and the number of completely incorrect labels assigned to images. We encourage others to further this papers brief exploration and help to develop more robust understanding of how compression effects classification.

The authors also believe much can be gained from exploring the ways in which the different image compression methods interact with, and possibly combat, adversarial attacks. Adversarial attacks are where stickers, random noise, etc. are added to images or scenes, causing neural networks to misclassify them or fail to detect them appropriately [12]. Initial research has shown promise that data reduction could be used a method to fight adversarial attacks[13] and, given the consequences as neural networks are deployed more routinely into our daily lives, the authors believe further investigation here is promptly needed.

ACKNOWLEDGMENT

We would like to thank Prof. Gilbert Strang for his continued support and interesting and informative lectures, as well as our fellow students.

[11]See: https://www.mturk.com/

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

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

REFERENCES

[1] Johnston, Nick and David Minnen. "Image Compression With Neural Networks." Google AI Blog. Published 29 September 2016. https://ai.googleblog.com/2016/09/image-compression-with-neural-networks.html

[2] "Real-Time Adaptive Image Compression." WaveOne, Inc. WaveOne. Published May 2017. http://www.wave.one/icml2017/

[3] "Cloud Vision API." Google Cloud. https://cloud.google.com/vision/

[4] Golub, G. H., Alan Hoffman and G. W. Stewart. "A Generalization of the Eckart-Young-Mirsky Matrix Approximation Theorem." https://www.ime.usp.br/~jstern/miscellanea/seminario/Golub87.pdf

[5] "Pooling." UFLDL Tutorial. http://ufldl.stanford.edu/tutorial/supervised/Pooling/

[6] "mogrify." ImageMagick. https://www.imagemagick.org/script/mogrify.php

[7] "command line options." ImageMagick. https://www.imagemagick.org/script/command-line-options.php\#quality

[8] Srivastava, Nitish, et. al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research*, 15 (2014): 1929-1958. Published 14 June 2014. http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf

[9] Tao, Xiaofeng. "Seam Carving." https://cs.brown.edu/courses/cs129/results/proj3/taox/

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

[11] 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

[12] 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/