Analysis of Classifier Model Performance with Compressed Training Dataset

Agness Lungu
Department of Intelligent Systems Engineering
Indiana University
Bloomington, Indiana, United States

***Abstract*—Singular value decomposition (SVD) is a popular data compression technique used to decrease computational stress and increase data transfer rates. Because of its use, this approach involved training a CNN with compressed images from the CIFAR-10 data set. The results show that a reduced data size through compression affects the accuracy of the model, but not significantly until a critical rank is passed. Specifically with CIFAR-10, I found that it is possible to decrease rank by 75% while marginally decreasing accuracy.**

***Index Terms*—Singular Value Decomposition, Compression, Classification, Neural Network**

I. Introduction

Convolution Neural Networks are computationally intensive for computers. The addition of any extra complexity within a model can increase training times factors of 10. In a world of data sets which constantly grow in size and model layer designs become complex, a solution is needed to decrease the computation stress. Promising approaches involve the reduction of data somewhere within the network. According to Dong Yu at Microsoft Research Redmond, the majority of features in a network, do not contribute to the final algorithm [1]. Yu's team was able reduce the parameters of a model significantly without introducing any noticeable error. A promising solution for decrease in parameters or data size is singular value decomposition (SVD). SVD is a traditional technique used to compress images or data to save storage in computing. The technique is performed through breaking down of an array into three components as seen in equation 1 from the article [2].

$$I = U * \Sigma * V^T \tag{1}$$

Where I is the input array of $m * n$ dimensions, U is of $m * m$ dimensions, Σ is a diagonal array storing the Eigen values of I, and V^T is of $n * n$ dimensions. The broken down matrix can be further pulled apart assuming it is the summation of each index within the respective matrix. This is shown in equation 2. See [3] for more information.

$$I = U * \Sigma * V^T = \sum \sigma_1 * u_1 * v_1^T + ... + \sigma_k * u_k * v_k^T \tag{2}$$

Each value of the sum with increasing k, contain information relating to the construction of the array I. Through the summation, the importance of each term decreases as the index k increases, where the maximum summation, is the rank of the matrix I. Because of the difference in term importance, it is possible to use only some of the terms to recreate a similar version of I. For example in color images, the original image is split into 3 matrices, which correspond to each RGB layer of that image, and SVD is performed on each. An interesting prospect is to use SVD's unique properties to enhance CNN performance. A team in China found a solution for applying SVD to the weights at each layer to the parameters of the total model [4]. Despite the research above, the community struggles with a solution to the CNN computation problem. In this work, SVD is applied on the training set of a CNN classifier. A data set of images is used to train models with SVD applied of a lower ranks for each. See Figure 1, an example of three classes of the CIFAR-10 data set applied with SVD compression of decreasing rank.

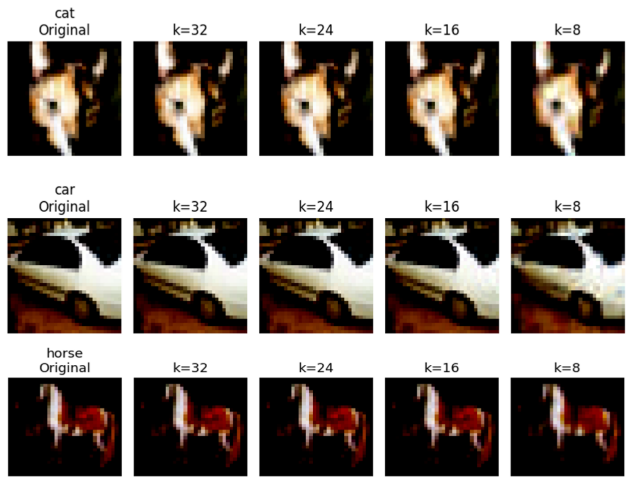

Fig. 1. Example of SVD on three CIFAR-10 Classes

II. Methodology

A. SVD Pseudo Code

To setup the experiment, an SVD function was created to disassemble the matrices at every rank, and compress the photos for input into the CNN. The process is as follows.

1) Unpack data set.

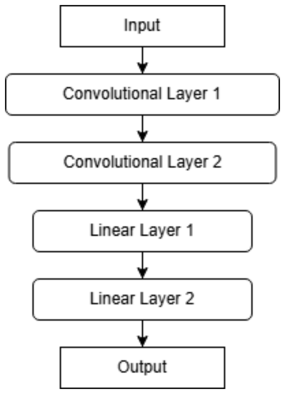

Fig. 2. Layers of CNN Model

2) Divide each image into three channels based on RGB.
3) Calculate U, Σ and V^T for the input rank.
4) Reassemble the images of each channel, excluding all vectors outside the desired rank.
5) Compile the channels together to create the color image.

The SVD function was then applied in training in a loop which stepped down a list of the ranks.

B. Model Design

A 4-layer model was created for classification for simplicity and faster train times. The model has two fully connected convolutional layers and two linear layers as seen in Figure 2. The convolutional layers each have pooling added to prevent over fitting. Additionally, RELU is performed in the forward propagation of every layer.

C. Creating a Control Test

I trained the model from Figure 2 to test the models performance with no data set modifications. This is shown in Figures 3, and 4 respectively.
The training loss in Figure 3, is calculated over the number of steps, or mini batches and lowers from a loss of about 2.25 to 0.75 at the lowest.
Figure 4, represents the test accuracy during training. Accuracy rises from about 10%, to about 65% To note, the default percent accuracy of a guess with 10 classes is 10%.

The layout of the experiment, was to apply SVD compression of rank decreasing in steps of two. For each rank a new model was trained and the information in regard to the model was recorded.

III. Experimental Setup

To run the experiment, I used Google Colab, and a personal machine. With Colab, I used NVIDIA T4 GPUs with the default memory. The second machine used for most of the training, was a personal desktop with VSCode as the IDE and an NVIDIA 3080ti for hardware. The second setup was preferred as there were no memory or runtime restrictions. A jupyter cell block configuration w as u sed to

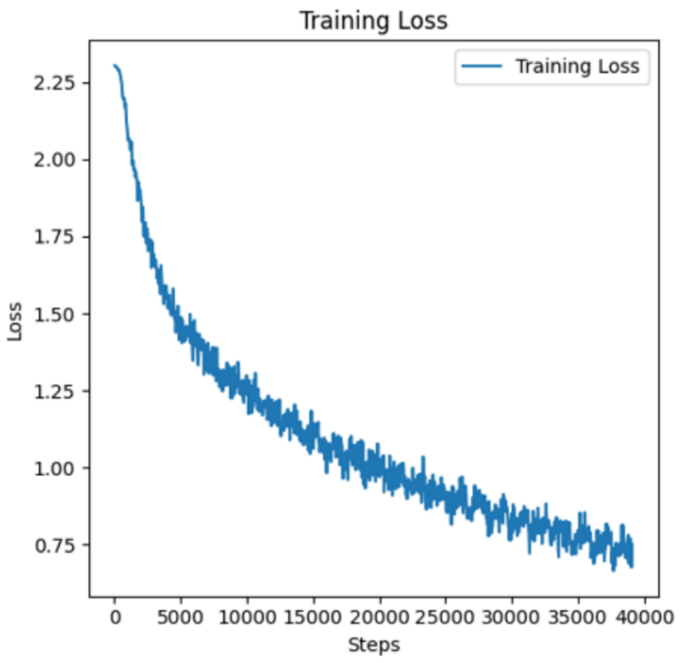

Fig. 3. Training Loss of the Model with No Compression

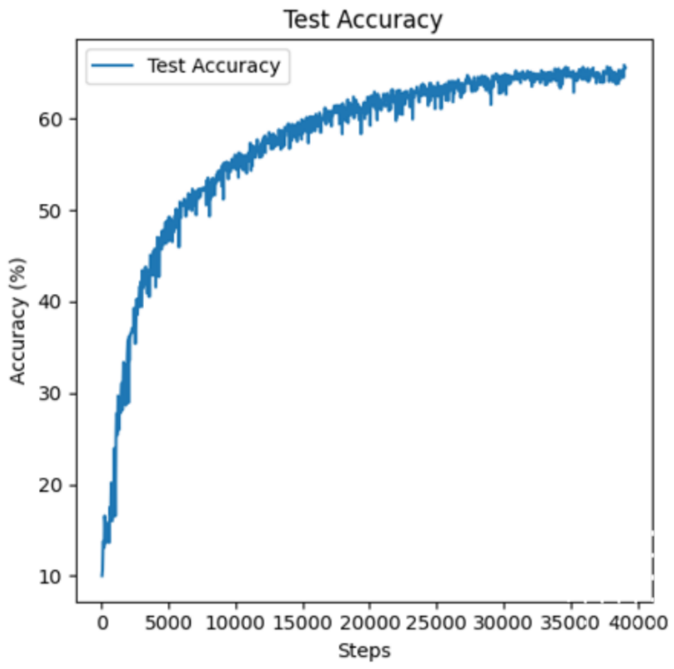

Fig. 4. Test Accuracy of the Model with No Compression

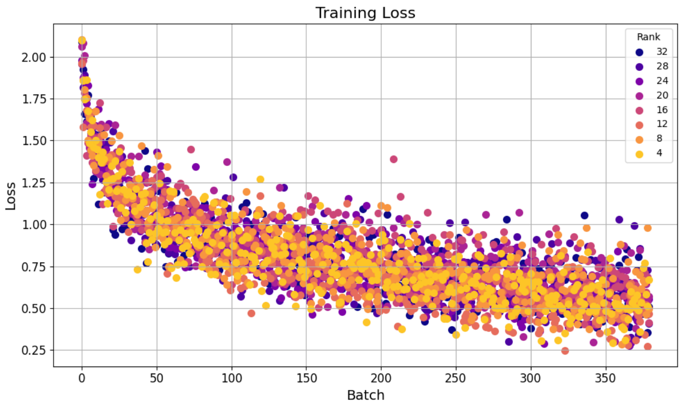

Fig. 5. Training Loss Versus to Rank

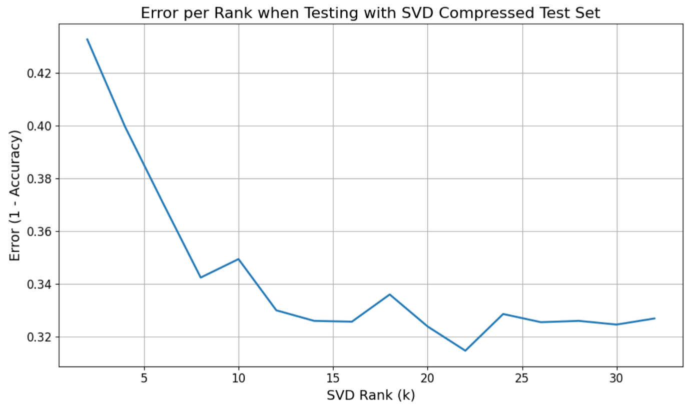

Fig. 7. Error with Compressed Test Set Versus Training Rank

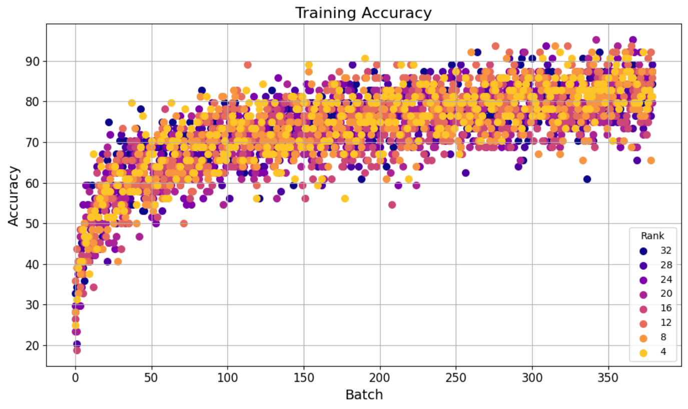

Fig. 6. Training Accuracy Versus to Rank

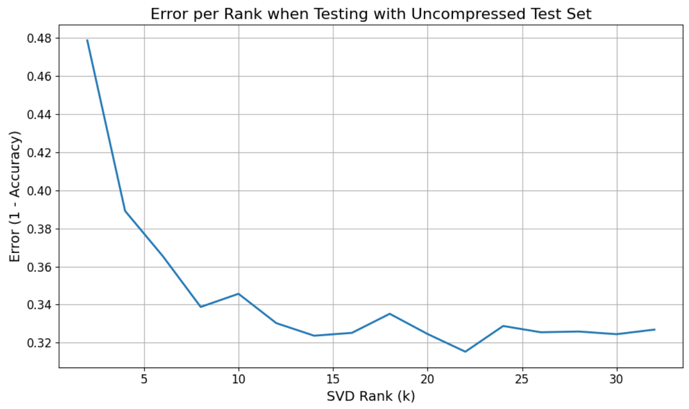

Fig. 8. Error with Stock Test Set Versus Training Rank

write the program in both cases.
The CIFAR-10 data set is commonly used for training CNN classifiers. The data set consists of 60,000 32 by 32 color images in 10 different classes. There are 6000 images per class. Additionally, the images are divided into 50000 training images and 10000 test images. The SGD optimizer was chosen for the lower computational stress, and learning rate was 0.001.

IV. Results

A. Training Results

Over an iteration that trained each model with increasingly compressed images team found that there were no significant differences in the training loss or test accuracy of the model over the training period.
Figure 5 represents the training losses at each mini batch. As claimed before there are no significant differences observed in the model's loss if it is trained with data that has been compressed. The loss of the model is observed to decrease from around 2.00, to about 0.75 on average. When focusing on mini batch 350-360 there are dips of loss to around 0.25.
In Figure 6, there is the test accuracy of the models trained over decreasing rank. Each model is tested with the stock data set and is not compressed. Interestingly, there are higher accuracies recorded than with the control model trained with no compression. This discrepancy might be due to the different hardware, as the control was run in Colab and the experiment was run on a personal computer.

B. Test Results

After the models had been trained, they were tested on two datasets, an uncompressed dataset of stock images, and a dataset of images compressed by the rank at which the respective model was trained. The error versus rank was calculated for each in comparison with rank. In Figure 7, the error against the training set of respective rank is shown.

The minimum error observed was about 0.33, which was maintained until rank decreased below 15. After ranks below 15, the error raised to a maximum of 0.43. A similar finding was seen in Figure 8, which represents the error of models tested against the uncompressed stock test set. For the stock test, a similar minimum error of 0.32 was observed, with the similar trend of error spike at ranks below 15. However, the maximum error was greater than the compressed test, with error reaching 0.48. The similarity in results of higher rank is to be expected because the higher rank SVD calculation included more summation terms, maintaining more of the Eigen values and vectors containing information about the photos. Additionally, the greater error found in testing for

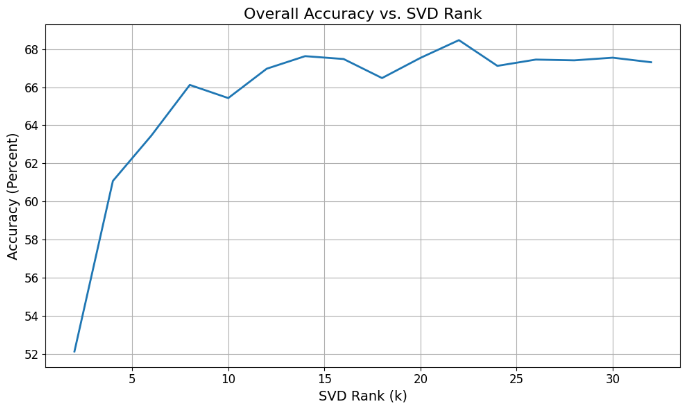

Fig. 9. Accuracy with Stock Test Set Versus Training Rank

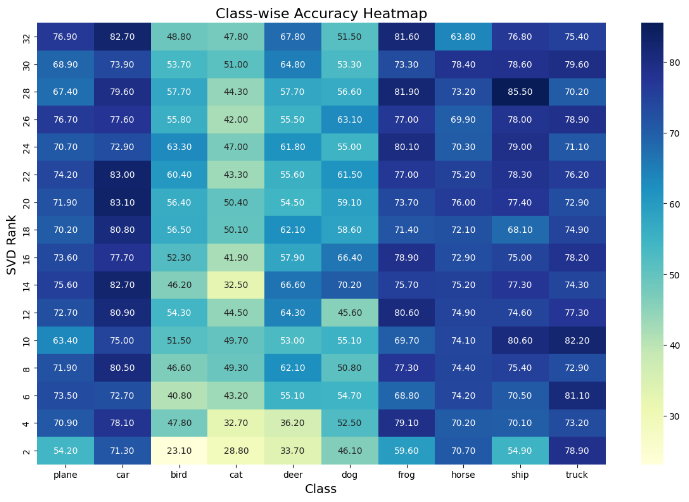

Fig. 10. Heat Map of Class Performance per Rank

the non-compressed training set at smaller ranks, is due to the larger amount of vectors ignored. In Figure 7, the model responds with greater error despite being trained with a data set of greater compression. I believe there is a critical percentage reduction of rank, which will increase error and decrease accuracy significantly. In Figure 9, the test accuracy of the trained model per rank is shown.

It is believed that certain classes have certain parameters which are more important for identification. Additionally, some classes may require more information for effective classification. Figure 10 shows a heat map of the accuracies of every class for models trained with varying levels of com-pression. While some of the lowest accuracies are seen at the lowest ranks, i.e. the lighter colors, there is also no significant difference for most classes. Additionally the decrease in rank appears to have a greater effect on some classes such as bird, cat, and deer. However, the truck, and car show no decrease in accuracy.

V. Conclusion

CNNs are an effective way to perform many tasks with machine learning. However, due to their high computational load, CNN models will run for longer times. In this work, I have shown that using SVD, to about 75% is an effective way to decrease the data size while maintaining a high accuracy. There is no significant observed effect on the compression depending class type. An interesting point was the discrepancy between the control test and the compressed tests, so I would like to revaluate this difference. In future work, it is necessary to monitor the used resources of the machines and time taken during training versus compression amount. Moreover, additional research is required to determine with certainty the varying effects of compression on model performance for different classes. If a certain class type could make error from compression greater, then this could apply to any input data with features of varying importance. Because the world contains many diverse data sets this would be crucial in understanding model functionality compared with data features. SVD compression is a technique used consistently through out the field because of its promise for increased computation rates. Understanding SVD's effects on training can help the community better understand their models.

References

[1] D. Yu, F. Seide, G. Li and L. Deng, "Exploiting sparseness in deep neural networks for large vocabulary speech recognition," 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, 2012, pp. 4409-4412, doi:10.1109/ICASSP.2012.6288897.

[2] Geek for Geeks, 2024, https://www.geeksforgeeks.org/singular-value-decomposition-svd/.

[3] J. Hui, "Machine Learning — Singular Value Decomposition (SVD) and Principal Component Analysis (PCA)," Medium.com, 2019, https://jonathan-hui.medium.com/machine-learning-singular-value-decomposition-svd-principal-component-analysis-pca-1d45e885e491.

[4] J. Wang, S. Li and W. Wang, "SVD-Based Channel Pruning for Convolutional Neural Network in Acoustic Scene Classification Model," 2019 IEEE International Conference on Multimedia and Expo Workshops (ICMEW), Shanghai, China, 2019, pp. 390-395, doi: 10.1109/ICMEW.2019.00073.