ON IMAGE DATA AUGMENTATION: TRAINING CNNs WITH LOW-RANK APPROXIMATION OF IMAGES USING THE SINGULAR VALUE DECOMPOSITION

Kennette James Basco[1]
kmaddela@nyit.edu
New York Institute of Technology

Huanying (Helen) Gu, Ph.D.[1, 2]
hgu03@nyit.edu
New York Institute of Technology

ABSTRACT

This study investigates enhancing the generalization of Convolutional Neural Networks (CNNs) for low-resolution images using Singular Value Decomposition (SVD) augmented images. It provides a theoretical overview of SVD and a method to control compression through hyperparameters: energy factor and skip threshold. Methods involve preprocessing, training models, and benchmarking validation and testing accuracy on various datasets (grayscale and RGB). Experiments using SVD augmentation show degraded validation and testing accuracy. In summary, while SVD augmented data shows theoretical promise, it does not alleviate the training-production data skew and even increases the signal-noise ratio. Rigorous hyperparameter tuning of the energy factors and skip threshold may regularize CNNs, but the computational resources required outweigh the marginal gains in regularization, if at all. This study highlights the limitations of SVD based augmentation and underscores the need for alternative regularization techniques in deep learning.

Keywords: convolutional neural networks, singular value decomposition, regularization, adversarial learning

1 INTRODUCTION

Data augmentation, particularly in image processing/computer vision, is a useful regularization method in training deep convolutional neural networks (CNN). Regularization is any method that decreases the optimization performance (some metric on the training data) that improves generalization performance (some metric on the testing data). Since the goal of machine learning is to predict data that has not been seen yet, data augmentation of the training dataset is an essential and established method of regularization. Conventional image augmentations include: random alterations in orientation, rotation, scale, addition of pixel value noise, etc.

The problem. Insufficient volume of data is a bottleneck of successfully training neural networks (Halevy, Norvig, & Pereira, 2009), even more so in CNNs given that images are high-dimensional in nature with varying resolution, sizes, signal-noise ratio, etc. In addition, when CNNs that are trained using high resolution images are deployed into production and are used for inference of low resolution images, training-production skew is introduced.

A potential remedy. This study aimed to determine if artificially increasing the training dataset with low-rank approximation of images with the singular value decomposition (SVD)

[1] College of Engineering & Computing Sciences
[2] Project Supervisor

results in regularized CNNs. Fitting the models with high-resolution images and applying image compression through SVD may allow the CNN to generalize more real-world samples, specifically those that are captured with low-resolution cameras – solving the training-production skew.

2 THEORY

2.1 SINGULAR VALUE DECOMPOSITION

SVD is a matrix factorization method defined for general matrices that decomposes a matrix $\boldsymbol{A}$ into 3 components:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\mathbf{V}^{T} \tag{1}$$

, i.e., $\boldsymbol{A}$ is a product of its left singular vectors, diagonal singular values, and the transpose of its right singular vectors (Strang, 2016). These singular matrices diagonalize $\boldsymbol{A}$ into the singular value matrix $\boldsymbol{\Sigma}$ – this conventionally contains singular values arranged in non-increasing order:

$$\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_{\min(m,\,n)}) \quad \text{where: } \sigma_1 \geq \sigma_2 \geq \ldots \sigma_{\min(m,\,n)} \tag{2}$$

2.2 LOW-RANK APPROXIMATION

The Eckart-Young theorem states that a rank k matrix produced by the multiplication of $\boldsymbol{A}$'s first k singular values and its corresponding left and right singular vectors is the closest approximation to $\boldsymbol{A}$. Succinctly, the rank k approximation of $\boldsymbol{A}$ is denoted as:

$$\boldsymbol{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^{T} \qquad \boldsymbol{A}_k = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^{T} + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^{T} + \ldots \sigma_k \mathbf{u}_k \mathbf{v}_k^{T} \tag{3}$$

, and given another matrix approximation $\boldsymbol{B}$ of rank k, the closest approximation of $\boldsymbol{A}$ is $\boldsymbol{A}_k$ (Eckart & Young, 1936):

$$||A - A_k||_{\mathbf{F}} \leq ||A - B||_{\mathbf{F}} \tag{4}$$

where the matrix norm is specifically the Frobenius norm. I.e, matrices or rank-2 tensors can be compressed into equation (1) and decompressed using a low-rank approximation via a truncated matrix product (3) that results in the closest approximation, and preserves the largest variance of the original matrix $\boldsymbol{A}$.

2.3 IMAGE COMPRESSION VIA THE SVD

The representation of images in computers is usually arranged in rank-3 tensors *of height x width x channels* (channels last format): Numbers of image channels differ such as RGB, RGBa, and grayscale. Since the SVD operation is only defined for rank-2 tensors such as grayscale images, individual SVD operations (see figure 2.1) are done for the channel dimension of rank-3 tensors (RGB and RGBa images). This lossy compression aims to retain maximal information for every channel of the image, and resulting in the closest approximation of the pre-image.

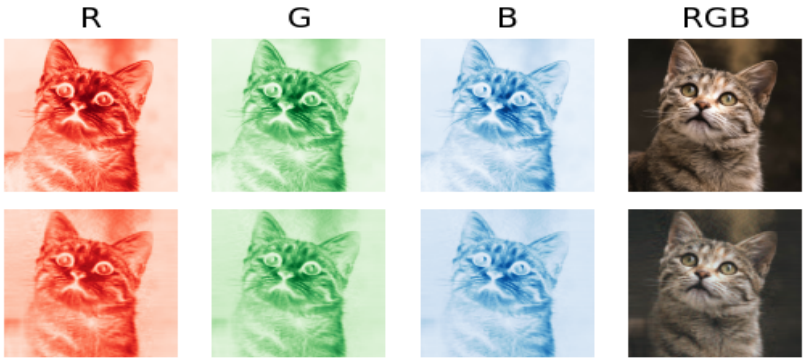

Figure 2.1: (Top) Decomposition of image channels. (Bottom) Channel-wise SVD of the image

In addition to this, the quality (resolution) of the compressed images appear to be similar to images that are captured with low-quality cameras (see figure 2.2). But, this conjecture may not hold true for various reasons such as subjectivity, the assumption of independence of the channel dimensions (see section 2.4 – *performance implications*), etc.

Figure 2.2: Cumulative low-rank approximations of an image

2.4 PERFORMANCE IMPLICATIONS:

Computational overhead. The SVD is performed by the underlying implementation of the LAPACK routine _gesdd, that uses a divide and conquer approach with a time complexity of $O(n^3)$ (Gu & Ren, n.d.), where n = min(height, width). Implementing an SVD layer makes training of CNNs become intractable in time (even more so when computations are done per channel of each image). To overcome this issue, augmented images are pre-computed (see section 3.2 – *Data Preprocessing*).

Assumption of feature channel independence. Channel-Wise SVD operations on images rely on the assumption of channel (RGB) independence. However, this decouples the correlation of colored images which can result in non-representation of the augmented training data since "[natural image channels] are actually highly correlated" (Chollet, Kalinowski, & Allaire, 2022). This addition of noise warrants hyperparameter tuning of the degree of compression and model training as it is very likely that highly compressed images will be adversarial examples (Goodfellow et al., 2014).

3 METHODOLOGY

3.1 BENCHMARKING DATASETS

Dataset Name	Samples	Classes	Class Distribution	Description	Model Used (see section A.2)
MNIST DIGITS	70,000	10	Balanced	Grayscale digits	Simple
MNIST FASHION	70,000	10	Balanced	Grayscale clothing items	Simple
MALARIA	27,558	2	Balanced	RGB thin blood smear images of segmented cells	ResNet25
CATS VS DOGS	21,524	2	Balanced	RGB images of cats or dogs	ResNet25

Table 1: Datasets Overview (see A.1 – *Datasets & Models*)

3.2 DATA PREPROCESSING

Image sizes are resized into defined arbitrary shapes. The datatypes of the pixel values are casted to float32 precision as a space-time vs. accuracy trade-off: float16 is too imprecise for gradient-based learning, while float64 has a larger constant time computational overhead per floating point number. Finally, pixel values are min-max normalized to the range of [-1, 1].

Given the performance implications (see section 2.4) of the SVD operation, experiments that use the SVD augmentation will pre-compute the augmentations of the training data. These augmentations are then concatenated to the training dataset – and the entire training dataset is shuffled. Pre-computation separates the preprocessing and training step of the model so that the performance of mini-batch stochastic gradient descent is unaffected by the time complexity overhead per image of the SVD operation, i.e., the forward and backward pass scales only linearly to the amount of the number of samples.

3.3 RANDOM MEASURE OF DISTORTION:

The energy factor is the proportion of the cumulative sum of the singular values of an image's singular value matrix. The SVD distortion rate uses the energy proportion disregarded by excluding the lowest singular values and their corresponding left and right singular vectors.

Energy factors are randomly generated using a normal distribution with a default mean = 0.975 and std = 0.025 that determines the amount of energy in the decompressed image after the SVD. Excessive augmentation of the training data increases the noise, and the training of CNNs

will fail or diverge. Another limiting factor was to implement a skip threshold = 80.00%. This random variable determines whether an image will be augmented or skipped. See figure 3.1 for a sample of SVD randomized augmentation colored and grayscale datasets.

There was an inverse correlation (via trial and error of model training) between these hyperparameters and the signal-noise ratio. The energy factors and skip thresholds are adjusted accordingly to enable proper training of the CNNs.

Figure 3.1: SVD energy factors simulated from a gaussian distribution with mean = 0.90 & std = 0.10 with skip threshold of 80%. (Left) Original samples. (Right) Augmented samples highlighted with borders.

3.4 MODEL TRAINING & EVALUATION

Holdout validation was used to split the training and testing data (80-20 split). Furthermore, K-fold (5) cross validation was used to split the training data to ensure validation accuracy was not sensitive to the partial train-validate split. Each fold performance was recorded and the mean and standard deviation of losses and metrics were calculated and visualized as a function of the number of epochs.

Given the balanced datasets, accuracy was a perfectly valid metric. The epoch that has the highest validation accuracy from each fold is recorded using model checkpointing (see section 3.5 – *Other Regularization Techniques*). The models are then re-trained with the entire training dataset using an epoch $\mathbf{E}^*$calculated with equation 5:

$$\mathbf{E}^* = round(mean(\sum_{f=1}^{folds} argmax(\mathbf{ModelEpochs}_{\text{Validation Accuracy}})) * 1.2) \quad (5)$$

since there is 20% more data in the regular fitting of the model. Finally the generalization performance was evaluated using the testing set. Model & dataset configurations that utilized SVD augmentation have twice the epoch of their non-SVD counterparts.

3.5 OTHER REGULARIZATION TECHNIQUES

Model Checkpointing. In K-fold cross validation, a separate validation data was evaluated by the model after every epoch. The state of the model with the highest validation accuracy was saved to checkpoint the epoch of the robust fit.

Dropout. A dropout layer was attached before the model's (not included in figure A.2) softmax layer to "reduce conspiracies by introducing noise in the output values" which mitigates overfitting of the model (Srivastava et al., 2012).

3.6 HYPOTHESIS TEST

Confidence interval. The mean and standard deviation of the converged K-fold validation accuracies were computed. If the lower bound of the alternative hypothesis' (H_a) confidence interval is greater than the upper bound of the null hypothesis (H_o), we reject the null.

4 RESULTS

Outline of Experiments:

- Experiment I: Determining if SVD has a regularizing effect on the (converged) trained model
- Experiment II: Determining if SVD has an additional regularizing effect along with default/conventional augmentations (see remark on section 4.2).
- Experiment III: Determining if SVD has an additional regularizing effect of a feature-extracted and fine-tuned pre-trained model (see *appendix B.1 for more details*).

4.1 EXPERIMENT I:

- H_o: SVD augmented data does not offer any regularizing effect.
- H_a: SVD augmented data does offer some regularizing effect.

Datasets	Augmentation Method/s	Energy Factor	Skip Threshold	Epochs	Mean K-fold Max Val-Accuracy	Retrained Testing Accuracy
MNIST DIGITS	None	N/A	N/A	10	[98.90%, 99.14%]	99.19%
	SVD	97.50%	80.00%	20	[98.60%, 98.86%]	98.70%
MNIST FASHION	None	N/A	N/A	30	[90.15%, 90.57%]	90.96%
	SVD	97.50%	80.00%	60	[88.58%, 88.90%]	90.39%
MALARIA	None	N/A	N/A	50	[95.63%, 95.89%]	95.61%
	SVD	99.99%	80.00%	100	[88.07%, 88.39%]	95.94%
CATS VS DOGS	None	N/A	N/A	100	[91.33%, 92.49%]	92.28%
	SVD	99.99%	80.00%	200	[85.40%, 86.70%]	93.98%

Table 1: Results of using SVD augmented data

4.2 EXPERIMENT II:

- H_o: SVD augmented data has no regularizing effect along with the default augmentation methods.
- H_a: SVD augmented data has some regularizing effect along with the default augmentation methods.
    - *Remark: The default augmentation methods are defined as follows:*
        - *RandomHorizontalFlip(0.5): 50% of the images are flipped horizontally*
        - *RandomRotation(0.1): The image is rotated using some* $\mathbb{R} \in [-1, 1] \equiv [-36°, 36°]$
        - *RandomZoom(0.2): The image is zoomed in/out using some* $\mathbb{R} \in [-.2, .2]$

Datasets	Augmentation Method/s	Energy Factor	Skip Threshold	Epochs	Mean K-fold Max Val-Accuracy	Retrained Testing Accuracy
MALARIA	Default	N/A	N/A	50	[96.84%, 97.14%]	97.12%
	Default + SVD	99.99%	80.00%	100	[92.71%, 93.03%]	97.44%
CATS VS DOGS	Default	N/A	N/A	100	[96.27%, 97.05%]	99.48%
	Default + SVD	99.99%	80.00%	200	[93.95%, 95.15%]	94.89%

Table 2: Results of SVD augmented data along with a default augmentation layer

4.3 EXPERIMENT III:

- H_o: SVD augmented data has no regularizing effect on a feature-extracted and fine-tuned pre-trained model (see appendix B – *Transfer Learning*).
- H_a: SVD augmented data has a regularizing effect on a feature-extracted and fine-tuned pre-trained model.
    - *Remark: The results below concern the fine-tuning step of transfer learning.*

Datasets	Augmentation Method/s	Energy Factor	Skip Threshold	Epochs	Mean K-fold Max Val-Accuracy	Retrained Testing Accuracy
CATS VS DOGS	Default	N/A	N/A	50	[98.46%, 98.88%]	99.79%
	Default + SVD	97.50%	90.00%	100	[98.97%, 99.29%]	98.54%

Table 3: Results of SVD augmented data of a pre-trained model

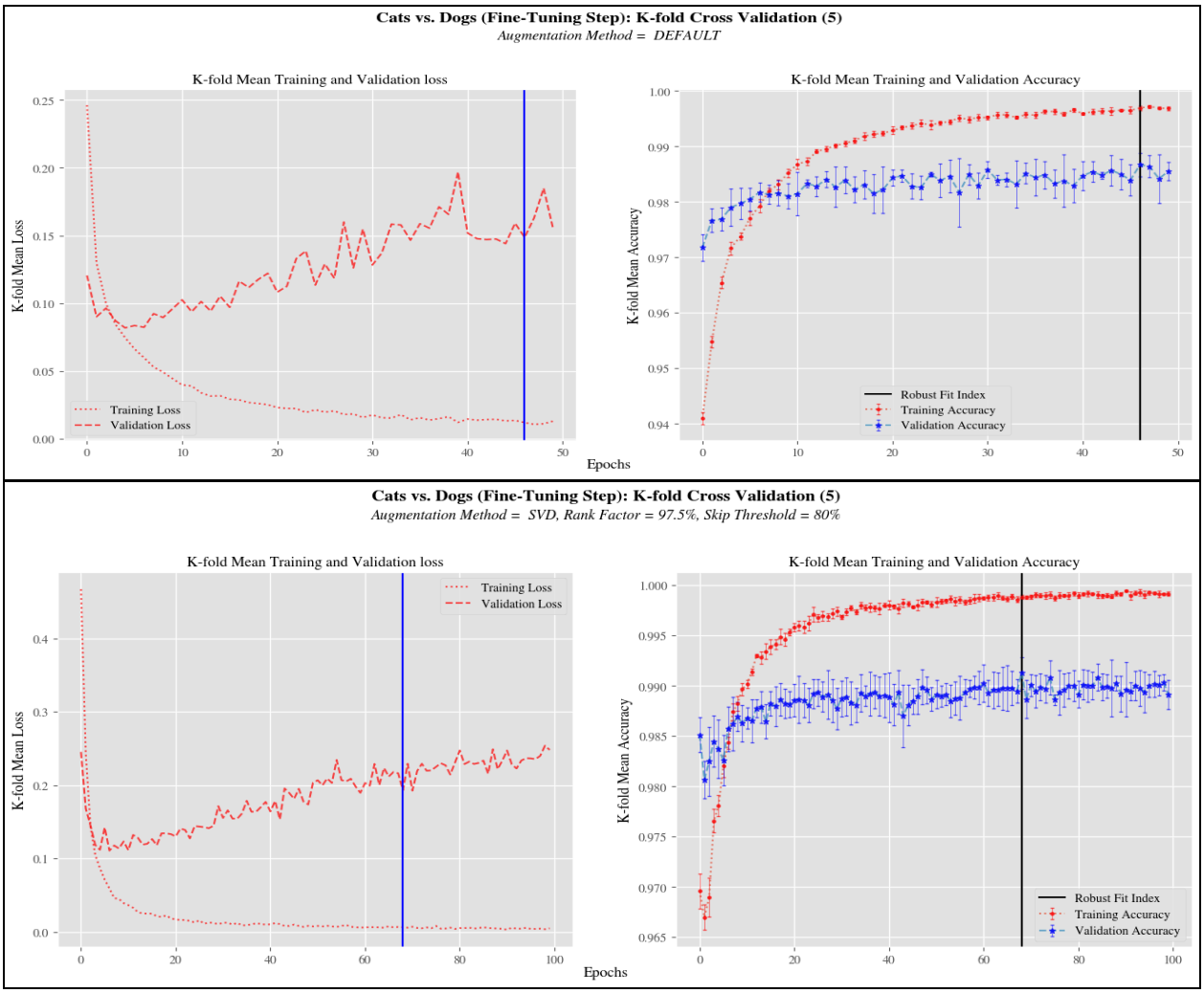

Figure 4: Training and validation, loss and accuracy plotted as a function of epochs

5 DISCUSSION

Every model configuration for the first and second experiment retained the null hypothesis. This may be due to the decrease of the signal-noise ratio of the additional augmented examples. These examples are adversaries that are non-representative of the original training data despite being in theory, the closest lower rank approximation of their pre-images (Eckart & Young, 1936).

Despite the model training of the SVD augmented data being successful, it requires at least twice the number of epochs for model convergence.

The only experiment that rejected the null hypothesis was the feature-extracted and fine-tuned pre-trained model that used SVD augmented data. This was perhaps due to the rich features already learned by the convolutional base being fine-tuned, and enabled the accommodation of small gradient updates by the lower rank images. On the other hand, its generalization accuracy significantly decreased by about 69.52% while there is only a 32% reduction in the validation error rate making this experiment non-scientifically significant – this deemed the model invalid to be pushed to production.

6 CONCLUSION

- Hyperparameter tuning of the energy factor and the skip threshold of the SVD implementation may have some regularizing effect on CNNs, as seen in experiment III, but is not worth the time and resources to tune and explore due to the computational overhead of this operation.
- Training CNNs with SVD augmented images despite having a high energy factor and high skip threshold merely increased the noisiness of the data leading to decreased performance.
- Despite SVD augmented images being in theory (Eckart-Young) the closest lower-rank approximation of their pre-images, as measured in Frobenius Norm, CNNs were still tricked by these adversarial examples.

7 ACKNOWLEDGEMENTS

I am extremely grateful to my project supervisor, Dr. Helen Gu, for allowing me to explore a bold topic in this rapidly developing field. With her support, I learned, created, and acquired the tools needed to conduct this project. She serves as one of the few who are both my foundation and catalyst for pursuing deep learning.

APPENDIX A: Datasets and Models

This section details the references of the dataset and the objective. In addition, the model used for a particular dataset is based upon its complexity such as the number of samples, dimensionality, number of channels, number of classes, etc.

A.1 Dataset References

Dataset Name	Reference
MNIST DIGITS	http://yann.lecun.com/exdb/mnist/
MNIST FASHION	https://github.com/zalandoresearch/fashion-mnist
MALARIA	https://lhncbc.nlm.nih.gov/LHC-downloads/downloads.html#malaria-datasets
CATS VS DOGS	https://www.microsoft.com/en-us/download/details.aspx?id=54765

Table 4: References to benchmark datasets. These are loaded using the tensorflow-dataset library and with a batch size of 128 (see section C.1 – *Hardware & Software*)

A.2 Model Architectures

Residual Network (ResNet25). Inputs are passed to 2-D convolution layers, and then into a max pooling layer to downsample the feature maps. Intermediate layers of ResNet25 are made up of residual convolution blocks (see figure A.2). A residual/skip connection is used to address the *vanishing gradients problem* which is inherent in very deep and sequential models (such as CNNs) – this is essentially the transmission of features in the earlier layers of the model becoming small at the final output layer. This makes it difficult to train deep models.

Residual Convolution Blocks (RCB). The residual convolution blocks of ResNet are composed of 2 2-D strided convolution layers without any bias applied immediately followed by batch normalization. After the first batch normalization layer the rectified linear unit (ReLU) activation function is used as a non-linearity – the application of batch normalization before this is intentional to "fully utilize the ReLU" (Ioffe & Szegedy, 2015). These building blocks are seen in figure A.2.

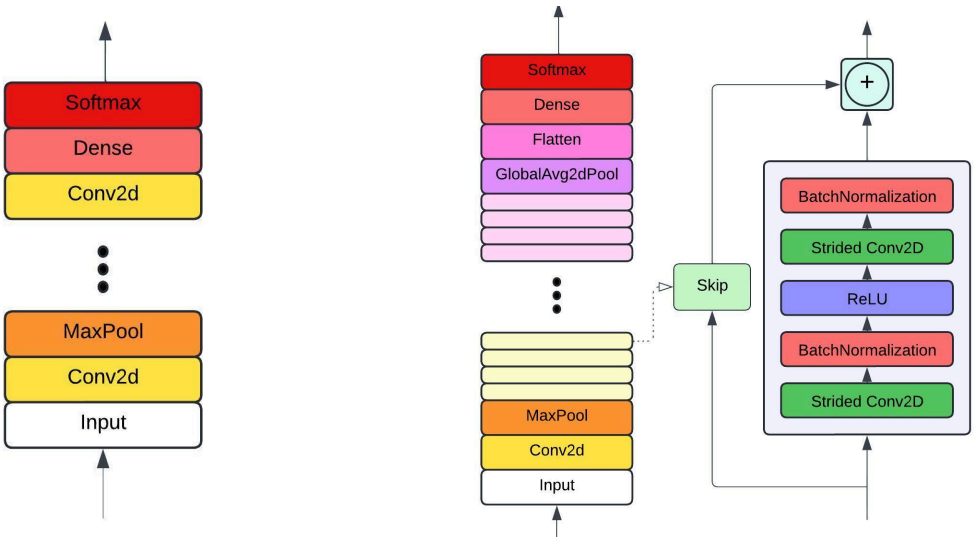

Figure A.2: (Left) Simple model of repeated 2d convolutions then max pool. (Right) ResNet25 and its residual conv blocks with increasing number of filters as the depth increases.

Model Name & Status	Depth	Trainable Parameters
Simple CNN Model	9 (without dropout)	100K (28x28x1 images)
ResNet25	25 (an RCB as a single layer and incl. dropout)	5M (100x100x3 images)

Table 5: Depth and number of trainable weights of models using inputs of arbitrary shape and dimension.

Appendix B: Transfer Learning

This section details the steps done in leveraging a pre-trained model.

B.1 VGG16 Feature Extraction and Fine-Tuning

The VGG16 (Simonyan & Zisserman, 2015) CNN model pre-trained on the image-net dataset (Deng, et al., 2009) was used as the convolutional base. The convolutional base with frozen weights was feature extracted for 50 epochs, then the 4 top-most layers of the convolutional base were unfrozen and fine-tuned with a learning rate of 0.00001 using RMSprop as the optimizer (Hinton, 2012).

Appendix C: Resources

This section details the steps the hardware and software resources used to conduct this study. Replicating the results is vital for the validity and reliability of the findings – the future peer reviewers are greatly appreciated.

C.1 Hardware & Software

The models are GPU-trained using a rented NVIDIA TESLA A100 GPU with 40GB of RAM. The experiments are implemented in python using the Keras deep learning library with TensorFlow as the backend. Loading of data is done using the TensorFlow-Datasets library.

C.2 Determinism

Global determinism is set with a seed of 42 to the python libraries that have randomness implemented such as NumPy, TensorFlow, and Keras.

- Colab Python Notebook: https://colab.research.google.com/drive/1ZDKDssQWhSfKsjO2sm3LHz3Pe2OiDq0i?usp=sharing
- GitHub Repository: https://github.com/kankenny/da-svd
- Compilation of Results and Trained Models: https://drive.google.com/drive/folders/1ik9E65xrGfsW2sKBTFQOI0zAZ7A15JGz?usp=sharing

References

Halevy, A., Norvig, P., & Pereira, F. (2009). The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems, 24*(2), 8-12. https://doi.org/10.1109/MIS.2009.36

Strang, G. (2016). Introduction to linear algebra. Cambridge Press. p .364

Eckart, C., & Young, G. (1936). The approximation of one matrix by another of lower rank. Psychometrika, 1(3), 211–218. https://doi.org/10.1007/bf02288367

Gu, M. & Ren, H. (n.d.). gesdd LAPACK implementation. Computer Science Division, University of California at Berkeley, USA

Chollet, F., Kalinowski, T., & Allaire, J. J. (2022). Deep Learning with R, Second Edition. Simon and Schuster.

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and Harnessing Adversarial Examples. Retrieved from arXiv preprint arXiv:1412.6572.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research, 15*(1), 1929–1958. https://doi.org/10.5555/2627435.2670313

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR*, abs/1502.03167. Retrieved from http://arxiv.org/abs/1502.03167

Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556.

J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

Hinton, G. E. (2012). Neural Networks for Machine Learning. Lecture 6, slide 27