Principal Component Analysis for Image Analysis

Alex Moini, Tyler Clough, Dani Lisle
University of Colorado, Boulder

Abstract

Principal component analysis (PCA) is method of dimensionality reduction with applications in a wide range of fields including epidemiology[1] and diagnoses from ECG data[2]. We apply PCA to the problem of reducing the dimensions of images while preserving visual discernability. In our method, the dimension of the images are are compressed for storage and then decompressed to produce reconstructed images, albeit at a lower resolution. A convolutional neural network (CNN) is then used to quantify whether the decompressed images remain discernible and thus viable for use in applications of image analysis, such as epidemiology.

Author roles: Alex Moini designed and performed testing with a convolutional neural network. Tyler Clough and Dani Lisle implemented principal component analysis and analyzed results. Everyone was involved in writing and editing the paper.

Introduction

Image compression is a fundamental issue in image analysis. As such, our motivation is to investigate the utility of Principal Component Analysis as a method for image compression in the context of image recognition applications. Specifically, we seek to understand the extent to which an image can be compressed while retaining visual discernibility when decompressed and reconstructed. This utility is measured using a CNN in order to mimic the standard of using deep learning models for image analysis/recognition[3]. Furthermore, we are interested in this line of inquiry because - as will be discussed in the following sections - this is a direct application of a *Matrix Methods* topic, namely Singular Value Decomposition, to an ongoing issue in industry.

In order for readers to better understand our application of PCA to the problem of image compression, we will provide a brief synopsis of why image compression is important as well as how image data is stored on a computer.

Image compression is significant because it is directly correlated to efficiency and therefore to computation time and cost. In the context of predictive medical diagnoses, image compression is useful when training a neural network to identify irregularities in ECG data because the extent to which the images can be compressed without a significant loss in the performance of the network directly impacts the cost to train such network. Furthermore, it is important to emphasize that simply reducing the size of an image is not enough, one must do so in a way that preserves the image's key features if it is to be used for image analysis.

Now, images are stored on a computer as three dimensional matrices, known as tensors. Each layer of the tensor corresponds to a color, while the value/index pairs correspond to the

[1] Nobi, A., Tuhin, KH. and Lee, JW., 2021
[2] Zheng, 2021
[3] Ewan, 2019

color's brightness and position, respectively. Image compression is then finding solutions to the problem of reducing the size of the image tensors while retaining as much of the image's original resolution as possible, i.e. using reversible operations to encode more information in a smaller matrix. In our case, PCA involves a modified process of Singular Value Decomposition, which is a well known method for reducing the size of a given matrix.

For our purposes, we will be using CIFAR-10 as our test dataset of images. CIFAR-10 is a collection of 60,0000 32x32 RGB images that span 10 different "classes", each a different animal or object: Airplanes, Automobiles, Birds, Cats, Deer, Dogs, Frogs, Horses, Ships and Trucks. The dataset is open source and was created by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton[4]. It was compiled as a tool for training machine learning models in the context of image compression, storage, and recognition, making it a perfect choice for our experiment.

In summation, our experiment is an investigation of Principal Component Analysis' viability as a method for compressing images in the context of image analysis applications. Specifically, we hope to determine the limiting size that a set of images can be compressed using PCA before the uncompressed images can no longer be accurately recognized, i.e. classified as the correct class by the CNN.

Lastly, please note that throughout the paper vectors are represented by boldface, lowercase letters, matrices are indicated by boldface, uppercase letters and entries of each are denoted by subscript(s). Additionally, a superscript T represents the transpose operation.

Methodology

In order to determine the limiting size of image compression via PCA, we need a set of images as well as a method of measuring how much information was retained in the compression/decompression. As previously mentioned we will be using the open source dataset CIFAR-10 as our test set along with a convolutional neural network to measure how much information is retained in the compressions. We chose to use a CNN because they are the standard in industry for image recognition applications and one of our group members has experience with these networks.

Next, we provide a formulation for the specific method of PCA that was used in our experiment, followed by a brief outline of our procedure.

Mathematical Formulation

Principal Component Analysis is a method for reducing the dimensionality of a given dataset. The method accomplishes this by preserving the variance within the dataset, which is defined as the variation within each variable[5]. In other words, new variables found with PCA capture the extent to which the measurements on the original variable differ.

In general the PCA method acts on a n-by-p data matrix $\mathbf{X}$, where each column of $\mathbf{X}$ represents a different variable in the set; generally, the jth column of $\mathbf{X}$ contains n measurements on the jth variable. Equivalently if each variable has an equal number of the measurements, then the ith row in the data matrix $\mathbf{X}$ contains the ith measurements on all p variables.

Now, the variance of the dataset is defined as:

$$var(\mathbf{Xa}) = \mathbf{a}^{\mathrm{T}}\mathbf{Sa} \tag{1}$$

[4] Alex Krizhevsky & Hinton, 2009
[5] Janakeiv, Nikolai, 2018

where, $\mathbf{X}$ is the data matrix, $\mathbf{a} = [a_1 \dots a_p]$ contains the coefficients from the linear combination:

$$\sum_{j=1}^{p} a_j \mathbf{x}_j = \mathbf{X}\mathbf{a}$$

And $\mathbf{S}$ is the covariance matrix[6], which is a pxp square matrix whose $\mathbf{S}_{ij}$ entry is the calculated covariance between the ith and jth variables from $\mathbf{X}$, $\sigma(\mathbf{x}_i, \mathbf{x}_j)$:

$$\mathbf{S} = \begin{bmatrix} \sigma(\mathbf{x}_1, \mathbf{x}_1) & \dots & \sigma(\mathbf{x}_1, \mathbf{x}_p) \\ \vdots & \ddots & \vdots \\ \sigma(\mathbf{x}_p, \mathbf{x}_1) & \dots & \sigma(\mathbf{x}_p, \mathbf{x}_p) \end{bmatrix}$$

where,

$$\sigma(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{n-1} \sum_{k=1}^{n} (\mathbf{x}_{i,k} - \bar{\mathbf{x}}_{i,k})(\mathbf{x}_{j,k} - \bar{\mathbf{x}}_{j,k}) = \frac{1}{n-1} \sum_{k=1}^{n} (\mathbf{x}_{j,k} - \bar{\mathbf{x}}_{j,k})(\mathbf{x}_{i,k} - \bar{\mathbf{x}}_{i,k})$$ [7]

Thus, $\mathbf{S}$ is symmetric by the commutativity of multiplication and so it may be written as the sum:

$$\mathbf{S} = \frac{1}{n-1} \sum_{k=1}^{n} (\mathbf{X}_k - \bar{\mathbf{X}}_k)(\mathbf{X}_k - \bar{\mathbf{X}}_k)^{\mathrm{T}} = \frac{1}{n-1} \sum_{k=1}^{n} (\mathbf{X}_k - \bar{\mathbf{X}}_k)^{\mathrm{T}} (\mathbf{X}_k - \bar{\mathbf{X}}_k) \tag{2}$$

Since we want to preserve the variability in the data, we seek a linear combination of the columns of $\mathbf{X}$ that will maximize $\mathbf{a}^{\mathrm{T}}\mathbf{S}\mathbf{a}$ according to (1). It can be shown[8] that this process is equivalent to solving:

$$\mathbf{S}\hat{\mathbf{a}} = \lambda\hat{\mathbf{a}}$$

with the added restriction that $\hat{\mathbf{a}}$ is a unit-norm eigenvector ($\hat{\mathbf{a}}^{\mathrm{T}}\hat{\mathbf{a}} = 1$).

This implies:

$$var(\mathbf{X}\hat{\mathbf{a}}) = \hat{\mathbf{a}}^{\mathrm{T}}\mathbf{S}\hat{\mathbf{a}} = \hat{\mathbf{a}}^{\mathrm{T}}\lambda\hat{\mathbf{a}} = \lambda\hat{\mathbf{a}}^{\mathrm{T}}\hat{\mathbf{a}} = \lambda$$

meaning maximizing the variance corresponds to finding the largest eigenvalue of $\mathbf{S}\hat{\mathbf{a}}$.
Now,

We can simplify this process[9] by first centering $\mathbf{X}$ to obtain a new matrix $\mathbf{X}^*$. Here, "centering" means subtracting the mean of each column from each entry in that column, i.e.

$$\mathbf{X}_j^* = \mathbf{X}_j - \bar{\mathbf{X}}_j$$

Centering $\mathbf{X}$ thus simplifies (2) to:

$$\mathbf{S} = \frac{1}{n-1} \sum_{k=1}^{n} (\mathbf{X}_k - \bar{\mathbf{X}}_k)^{\mathrm{T}} (\mathbf{X}_k - \bar{\mathbf{X}}_k) = \frac{1}{n-1} (\mathbf{X}^*)^{\mathrm{T}} \mathbf{X}^*$$

Equivalently,

$$(1-n)\mathbf{S} = (\mathbf{X}^*)^{\mathrm{T}} \mathbf{X}^* \tag{3}$$

[6] Janakeiv, Nikolai, 2018
[7] this rearranging of the product terms is necessary for the equation (3) on the following page
[8] Ian T. Jolliffe and Jorge Cadima, 2016
[9] Ian T. Jolliffe and Jorge Cadima, 2016

Therefore, equation (3) links the spectral decomposition of $\mathbf{S}$ to the singular value decomposition of $\mathbf{X}^*$, as it can be shown[10] that:

$$(1-n)\mathbf{S} = \mathbf{A}\mathbf{L}^2\mathbf{A}^{\mathrm{T}} \tag{4}$$

where, the columns of $\mathbf{A}$ correspond to the non-zero eignevectors of $(\mathbf{X}^*)^{\mathrm{T}}\mathbf{X}^*$
and $\mathbf{L}$ is a diagonal matrix with the singular values of (1-n)$\mathbf{S}$ as its entries.

And thus, the method of PCA is equivalent to solving the SVD of $\mathbf{X}^*$ because maximizing the variance corresponds to finding the largest eigenvalues of $\mathbf{S}$, which are the entries of $\frac{1}{1-n}\mathbf{L}^2$.

Finally, PCA gets its name from the linear combination of the data centered matrix $\mathbf{X}^*$ using the coefficients from the unit-normed eigenvector corresponding to the kth eigenvalue, $\hat{\mathbf{a}}_k$, which is known as a "principal component" (PC). These PCs are then used to reduce the dimensionality of a given dataset as enough PCs are generated to capture a sufficient amount of the variance within the original set.

Procedure

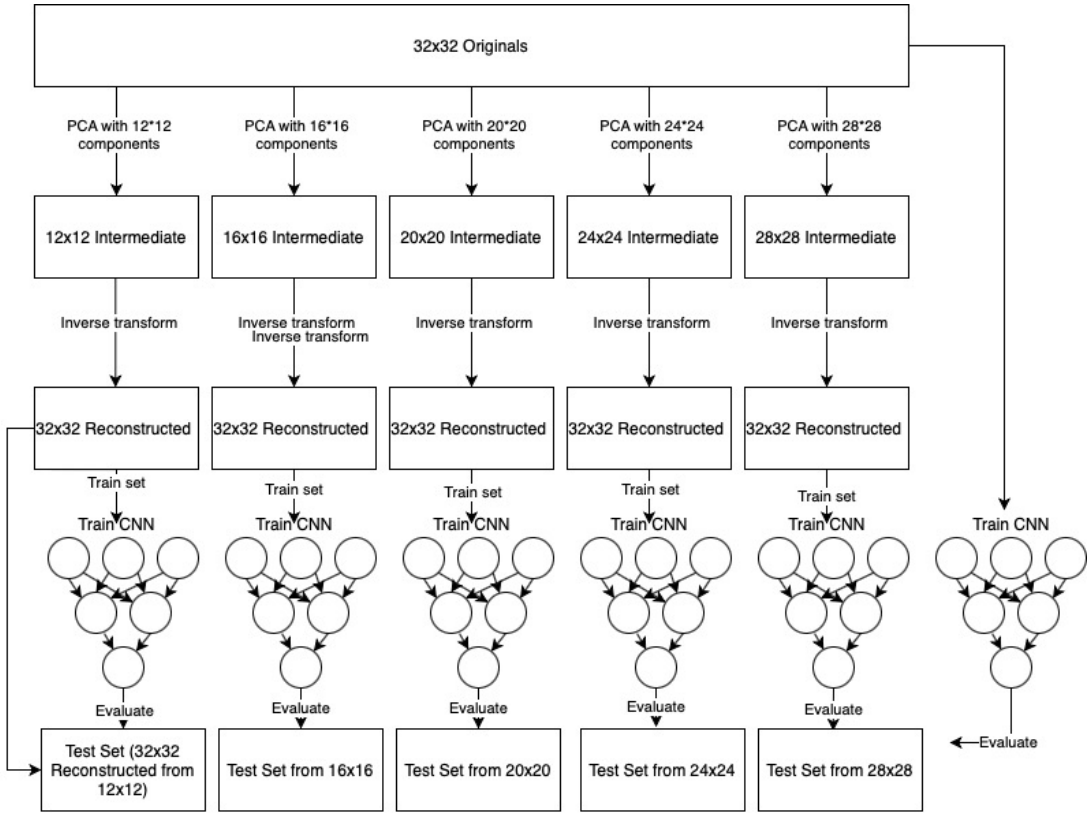


Figure 1: Procedure

[10] Ian T. Jolliffe and Jorge Cadima, 2016

Our project follows the procedure below, as illustrated in Figure 1 on the previous page:

1. A baseline was established by training a CNN using the original 32x32 images from CIFAR-10 and then measuring its accuracy score on a test set[11].

2. Next, in our 5 trials a PCA[12] was used to compress the set of 32x32 originals into sets of 28x28, 24x24, 20x20, 16x16, and 12x12; however, as images are stored as 3D tensors, the 3 layers were first split and the PCA transform was then run on each of the Red, Green and Blue matrices. We refer to the matrices resulting from this transformations as "intermediates". It is important to note that information is lost during the compression of the original images into the intermediates, which thus affects the reconstructed images.

3. An inverse PCA transformation was then applied to reconstruct the images from the intermediates and a CNN was trained on each of the reconstructed sets (28x28, 24x24, ...) using the same method as for the originals, i.e. 50k training set and 10k testing set. An illustration of this process is provided in Figure 2 below.

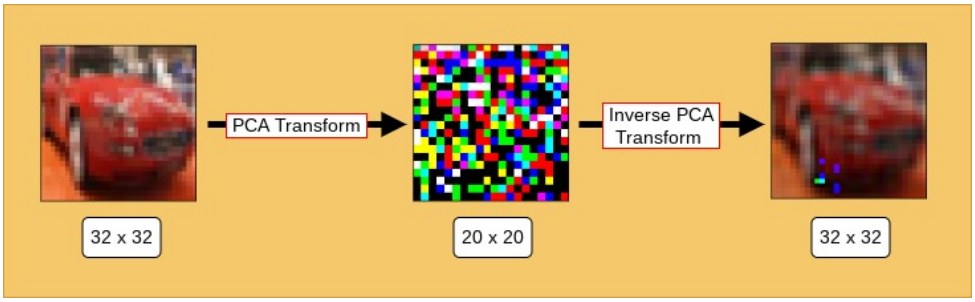


Figure 2: Illustration of the PCA compression/decompression process. This intermediate is 20x20.

4. Lastly CNNs were trained on each of reconstructed sets and tested. Finally, the resulting metrics were compared in order to determine the extent to which the images could be compressed without compromising the CNNs ability to accurately classify the images.

[11] Specifically, out of the 60k images in CIFAR-10, 50k were used to train the CNN and 10k were reserved for the test run

[12] We used scikit-learn's PCA package for our experiment. The parameters were adjusted so that the PCA performed was done using a data centered matrix and thus mimics the method discussed in the Mathematical Formulation section above.

Numerical Results

Accuracy

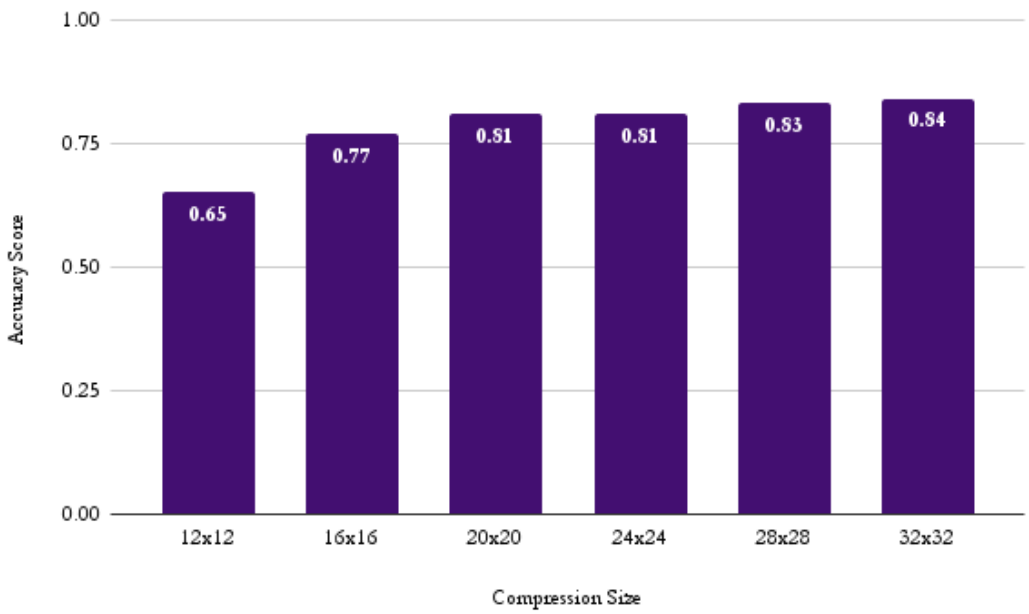


Figure 3: Accuracy of CNN vs PCA Compression

Here we define accuracy as the ratio of the number of correct guesses over the number of total images passed to the CNN. This metric is useful because it is a direct measurement of the CNN's ability to correctly classify the images and thus is also an indirect measurement of information retained during the PCA compression.

As a baseline, the CNN was able to identify uncompressed images with 84% accuracy and, as you can see, a proportional accuracy score can be achieved by compressing the images down to 20x20 intermediates, which corresponds to a $\sim 61\%$ reduction in the image's size. Interestingly, the 24x24 and 20x20 compression scored the same accuracy score.

Precision and Recall

The next metrics we consider are precision and recall, which form a paradigm commonly used in machine learning. Both are calculated for each classifiable object (truck, cat, dog, etc.) individually. Specifically,

$$\begin{aligned}\text{Precision} &= \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \\ &= \frac{\text{Correctly Identified as Truck}}{\text{Correctly Identified as Truck} + \text{Incorrectly Identified as Truck}}\end{aligned}$$

&

$$\begin{aligned}\text{Recall} &= \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \\ &= \frac{\text{Correctly Identified as Truck}}{\text{Correctly Identified as Truck} + \text{Missed Trucks}}\end{aligned}$$

Both quantities, although not particularly significant when considered individually, are useful when used in combination. For example, if the CNN guessed that every image was a truck, then it would have a 100% Recall score but its Precision score would indicate that it was not functioning properly. Similarly, if the CNN only guessed that one image was a truck and got it right, then it would have a 100% Precision score and the Recall score would highlight the reality. As such, it is necessary to have high scores in both metrics as an indicator for success.

The subsequent tables display precision and recall (as well as accuracy) scores for all classifiable objects, over all compression rates applied in this project.

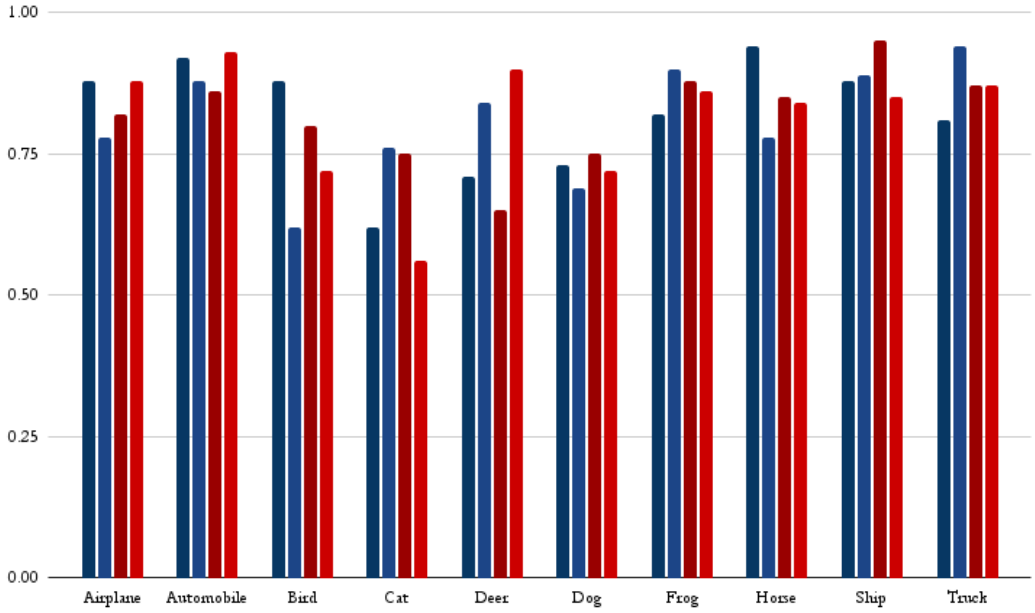


Figure 4: Precision and Recall for 24x24 and 20x20 Compressions

Note that Precision scores are on the right and Recall scores are on the left

As you can see, the precision and recall scores are very similar between the 24x24 and 20x20 compressions. Interestingly, the CNN had a harder time correctly classifying the animals over the vehicles, but overall the scores are respectable. Also of note is that the 20x20 compressions (red) had fewer categories in which the precision and recall scores differed by a large amount, but in those categories in which there was a large variation (Cat and Deer) the difference was very large.

Compression Time

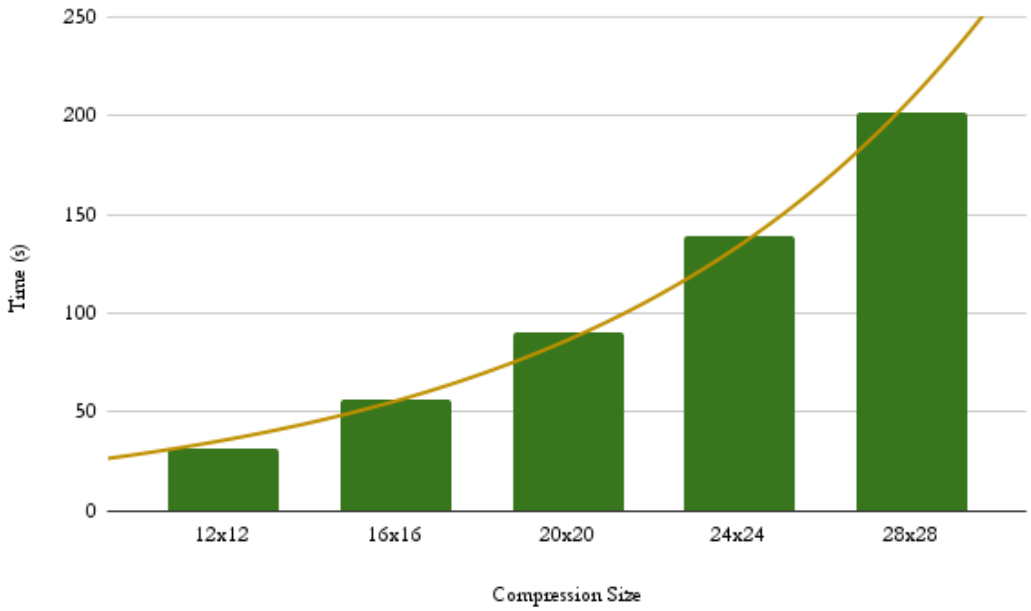


Figure 5: Time for Full Compression/Decompression for each Compression Size

Here we have measured the total time it took for the compression/decompression process, which is another relevant metric as this is directly correlated to the cost to train the CNN. As one might expect with computer processing, the time grows exponentially with the image size. This exponential growth is largely driven by the fact that the PCA method requires finding eigenvalues and corresponding eigenvectors, which is a computationally expensive task.

Discussion

Results

Considering the whole of the accuracy, precision and recall scores, we determine that out of our set of compressions, 20x20 would be the optimal size. We have come to this conclusion based on the fact that the 24x24 and 20x20 compressions have the same accuracy score and that there is no clear distinction between their precision and recall scores across all the classes. As such, choosing a 20x20 compression saves both storage capacity as well as time both of which translate directly to cost and efficiency. Additionally, in applications where very high accuracy scores are needed, one may consider using a 28x28 as doing so reduces the storage size by a little over 24% an minimal cost to accuracy.

Additional Insights

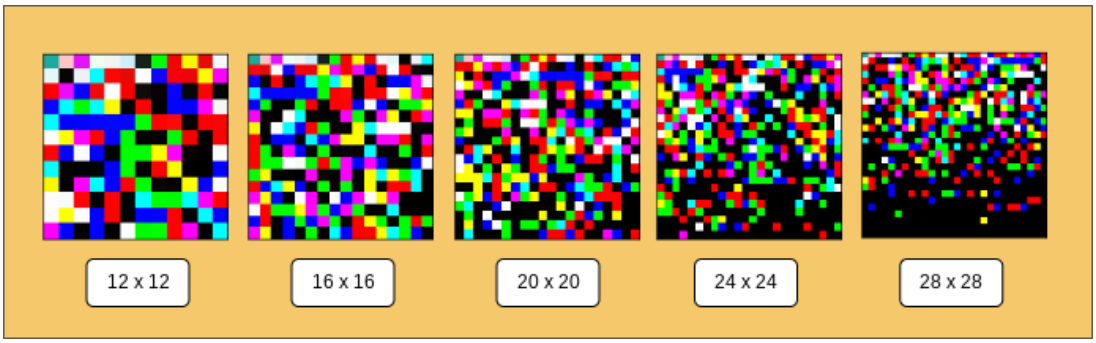


Figure 6: Comparison of Intermediate Matrices for various compression sizes

Figure 6 illustrates an interesting phenomenon that occured during our experiment: decreasing the compression size results in a clustering of black pixels at bottom of the intermediate matrices. The RGB code for black is 0 and so it appears that the PCA has encoded no information in these pixels; however, as these images are the overlays of the Red, Green and Blue intermediate matrices, any pixel that is purely red, green or blue is 0 in the same position for the other colors. What is then unique about these black pixels is that the pixel has a 0 value in all three intermediate matrices[13].

Now, before an hypothesis of what is causing this clustering, we must first clarify pixels in the intermediate images represent. To begin, the variance score for our compressed images did not reach 95%+ until the compression size was around 12x12[14], which corresponds to $12 * 12 = 144$ principal components. Interestingly, this is a much larger set of PCs needed to retain such high variance than is noted in the example application from the review by Jolliffe and Cadima, in which only 2 PCs were needed to retain 93.7% of the variance in their original dataset[15]. As was discussed in the Mathematical Formulation section, each of these PCs is really the linear combination resulting from multiplying the original data matrix $\mathbf{X}^*$ (the R, G and B matrices) by a unit normed eigenvector $\hat{\mathbf{a}}$.

This means that any pixel with a value of 0 corresponds to the nullspace of $\mathbf{X}^*$, or equivalently to a eigenvalue of 0[16]. Now, only singular matrices have an eigenvalue of 0 and so, in true scientific fashion, we have termed the clustering of black pixels at the bottom of the intermediate matrices to be a result of the "singularness" of the original image, which we define as the lack of color variation within the image. To clarify this metric consider Figure 7 on the following page:

[13]Note that the pixel value must be 0 in all three matrices because there are no negative RGB codes
[14]See Appendix for full set of compressed/decompressed images along with intermediates
[15]Ian T. Jolliffe and Jorge Cadima, 2016
[16]Note that this is not entirely true because a nonzero number could be treated as 0 if it is smaller than the mathematical precision of the computer.

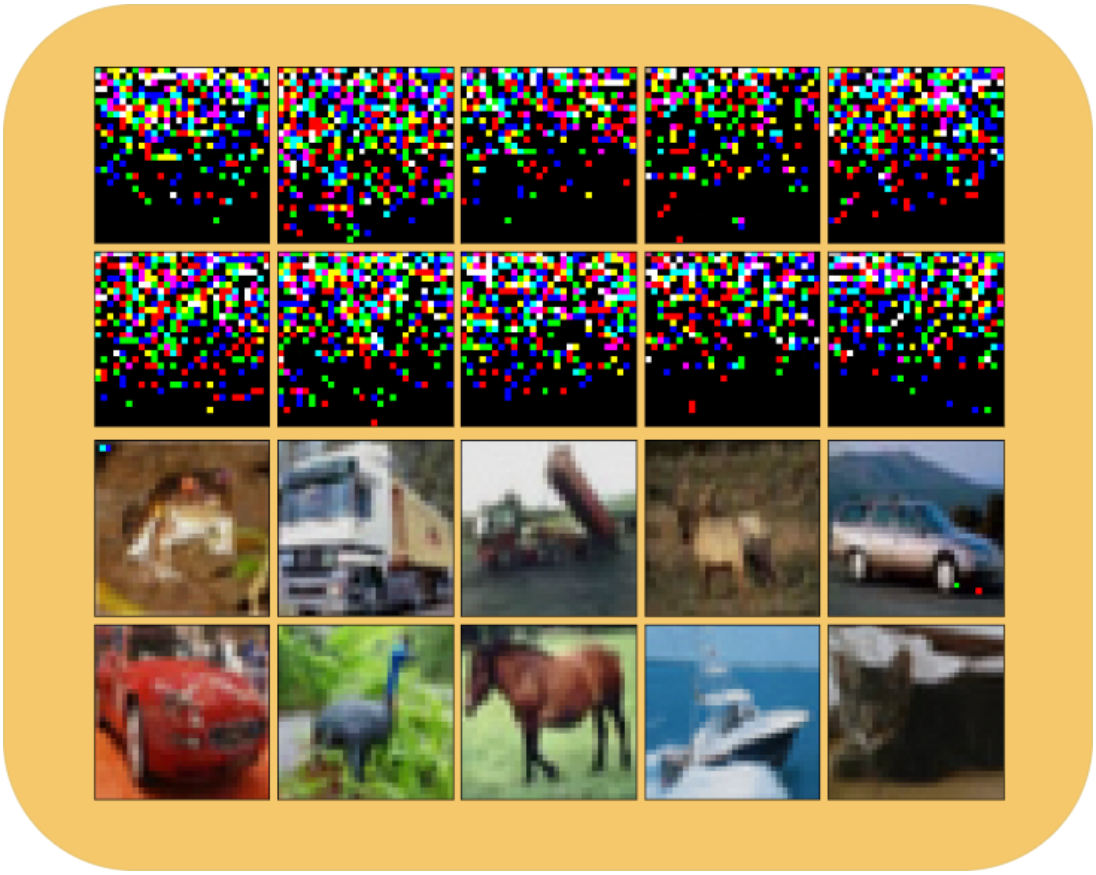


Figure 7: Side by Side Comparison of Intermediate Matrices and Reconstructed Images from the 28x28 Compression

Notice that for the pictures with the most color contrast - the red car, horse and 18 wheeler - have the least amount of clustering; whereas the colors with the least contrast - the boat, cat and elk - have the most clustering. We hypothesize that the images with the least color variation has similar color values in the same areas across the Red, Green and Blue layers; thus, the clustering of black pixels results from there being a number of eigenvalues, b, after which additional eigenvalues no longer contribute to the variance (i.e. are 0) and these are most likely to line up across the Red, Green and Blue layers when the original image does not have a lot of color variation. However, this is only a speculative hypothesis and further experiments would be needed to verify it.

Now, before concluding we believe that it is important to note a potential flaw in the design of our experiment which leads to a further line of inquiry and possibly a improved method of compression using PCA. Recall from the Mathematical Formulation section that PCA is intended to be applied to nxp data matrices, where the p columns correspond to p variables, each with n measurements. However, we applied it to 32x32 Red, Greeen and Blue image layers in which each pixel is, more or less, independent of the others. Therefore, the PCA viewed these layers as data matrices of 32 variables with 32 measurements each, but this is not the case.

As such, we hypothesize that rearranging the data matrix to better reflect the expected input for the PCA may lead to an improvement in compression. This could be accomplished by choosing the natural variables of an RGB image, namely, the Red, Green and Blue layers. In this case, the data matrix $\mathbf{X}$ would then be a $(32 * 32)$x3 matrix with the flattened Red, Green and Blue layers as its columns. One could then pass $\mathbf{X}$ through the PCA and then reconstruct the image by un-flattening the columns and overlaying each on top of one another. Furthermore, the dots of color within the clusters may be a result of limitations due to the mathematical precision of the computer used for the transform; for, if the additional variance contributed by a set of eigenvalues was nonzero but smaller than the precision of the computer, this would result

in the computer misidentifying the eigenvalue as 0, resulting in a 0 PC and thus a black pixel. However, this additional variance could build up to the point where it was within the computer's precision, at which point the PC would be nonzero, resulting in a colored pixel. The fact that no two same colors appear along the same row in the clusters supports this argument, because the pixels are ordered along the rows.

Again, we have not been able to test this method so this hypothesis is merely speculative as well. However, if it is a viable solution it would greatly reduce the amount of storage needed as a data matrix of only 3 variable would have at most 3 principal components and thus the largest intermediate could only be 3x3.

Conclusion

In this project we have explored the method of Principal Component Analysis and applied it to the problem of reducing the dimensionality of images. This problem is significant of its implications to the field of image analysis using Convolutional Neural Networks. We found that, out of our set of compressions, a 20x20 compression was optimal because it saved more storage and time without any tradeoff in the CNN's accuracy score and with little change in its precision and recall scores. Lastly, we discussed a pattern that occurred in the intermediate matrices as well addressed a limitation of our method and proposed a possible solution.

References

Alex Krizhevsky, V. N., & Hinton, G. (2009). Learning multiple layers of features from tiny images. *https://www.cs.toronto.edu/ kriz/cifar.html*.

Ewan. (2019). *What is image recognition.* Retrieved 2022-04-26, from https://deepomatic.com/what-is-image-recognition

Ian T. Jolliffe and Jorge Cadima. (2016). Principal component analysis: a review and recent developments. *Phil. Trans. R. Soc. A.*.

Janakeiv, Nikolai. (2018). *Understanding the covariance matrix.* Retrieved 2022-04-27, from https://datascienceplus.com/understanding-the-covariance-matrix/

Nobi, A., Tuhin, KH. and Lee, JW. (2021). Application of principal component analysis on temporal evolution of covid-19. *PLoS ONE*.

Zheng, R. C., J. (2021). On the application of principal component analysis to classification problems. *Data Science Journal*.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. (2022). A convnet for the 2020s. *arXiv*.

Appendix

Convolutional Neural Network

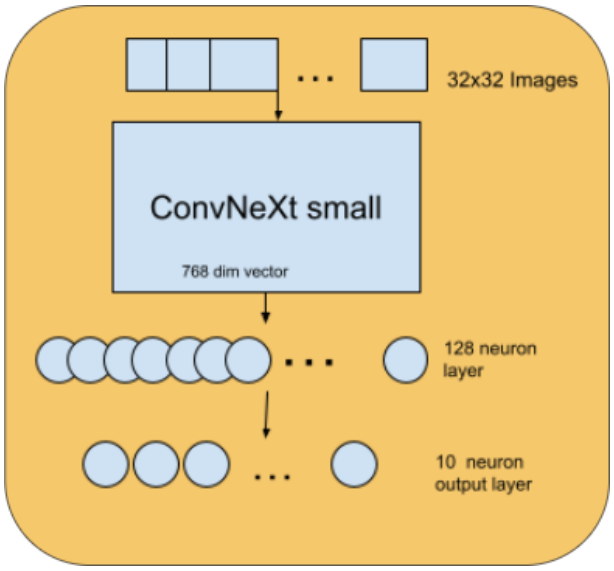


Figure 8

In this paper we use the ConvNeXt pre trained convolutional neural network as a method of judging how well the PCA is at lowering the dimensionality of an image and reconstructing it. We chose to use ConvNeXt because it is the current state of the art model in image classification and image processing tasks. It ranked above ViT (Vision Transformer) and other state of the art neural network based architectures[17].

[17] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie, 2022

ConvNeXt uses a number of convolutional, pooling, and residual blocks to build a high dimensional vector representation of an image. The vector representation can be used to classify images when paired with a dense layer of neurons as a "classification head."

In our implementation we used a ConvNeXt-small pretrained layer to produce a dense vector of size (1x768) followed by a dense layer of 128 neurons. This is then followed by an output layer with 10 neurons, one neuron for each class of the CIFAR-10 dataset. Each neuron represents a probability of the class being the true class according to the network. In order to determine the prediction we take the largest value in the output layer as the final prediction.

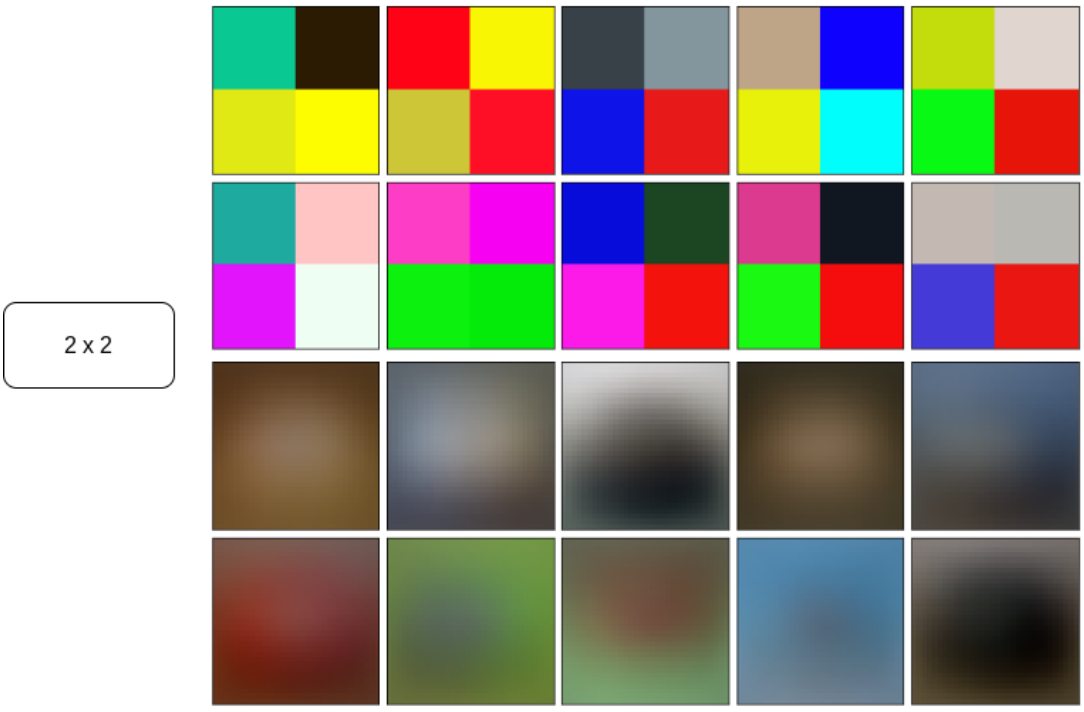
Each network was trained for 3 epochs, on 32x32 images, each with the same architectural specifications and hyperparameters.

Github Files

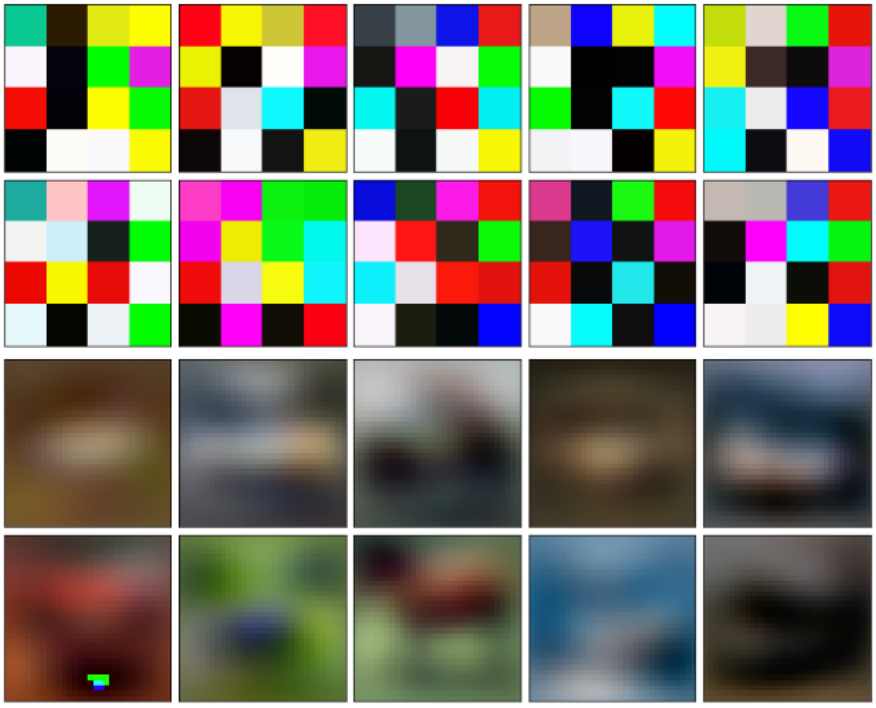
We have provided the link to the GitHub that stores all the files used for the experiment. Be sure to reference the "readme.md" for the purpose of the files in each folder.

Intermediate Matrices and Reconstructed Images

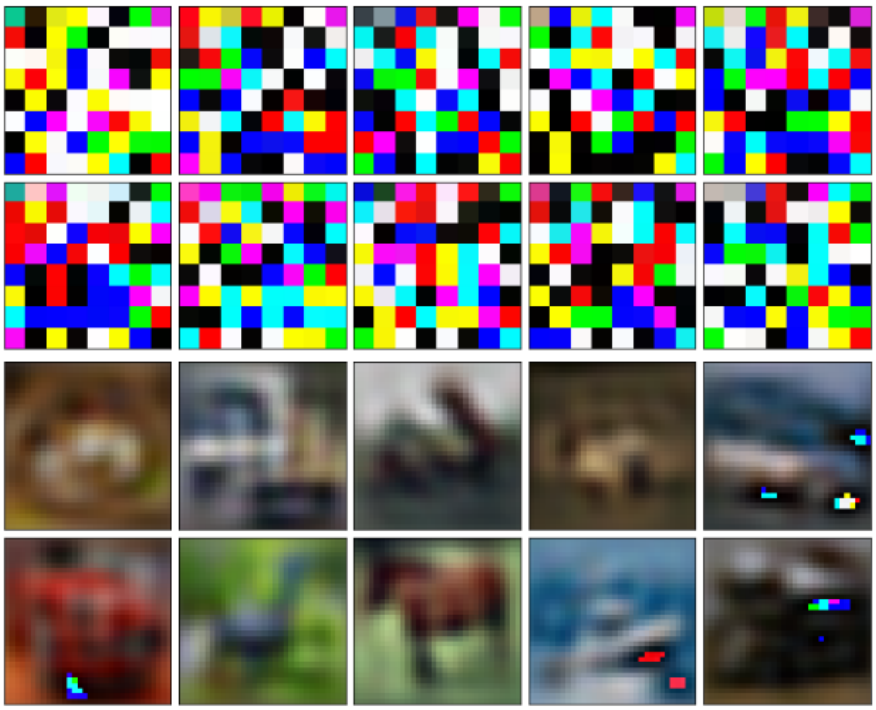
(Including Bonus Compressions not Referenced in Paper)



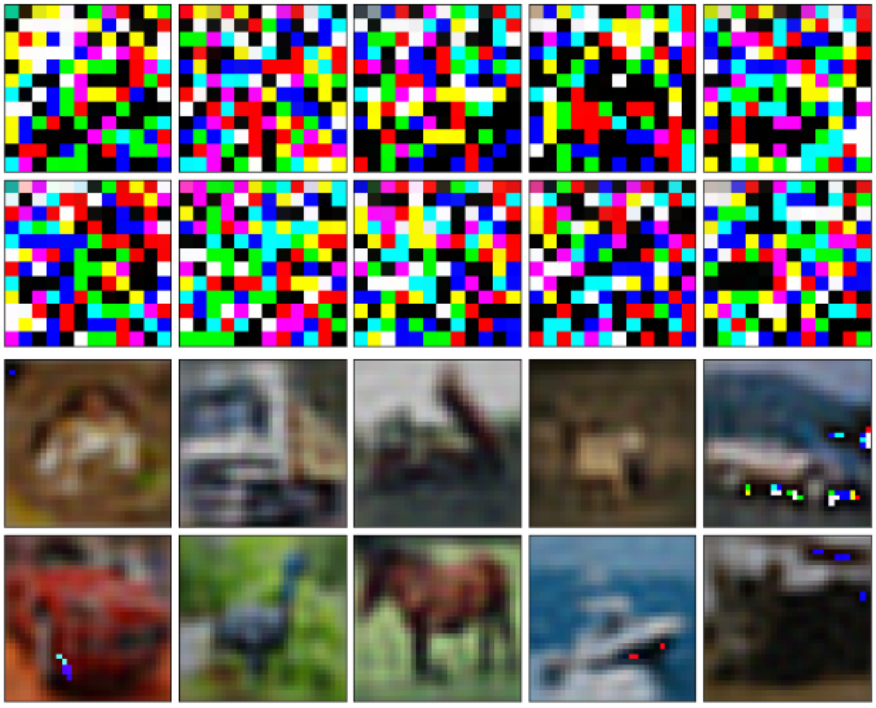
4 x 4



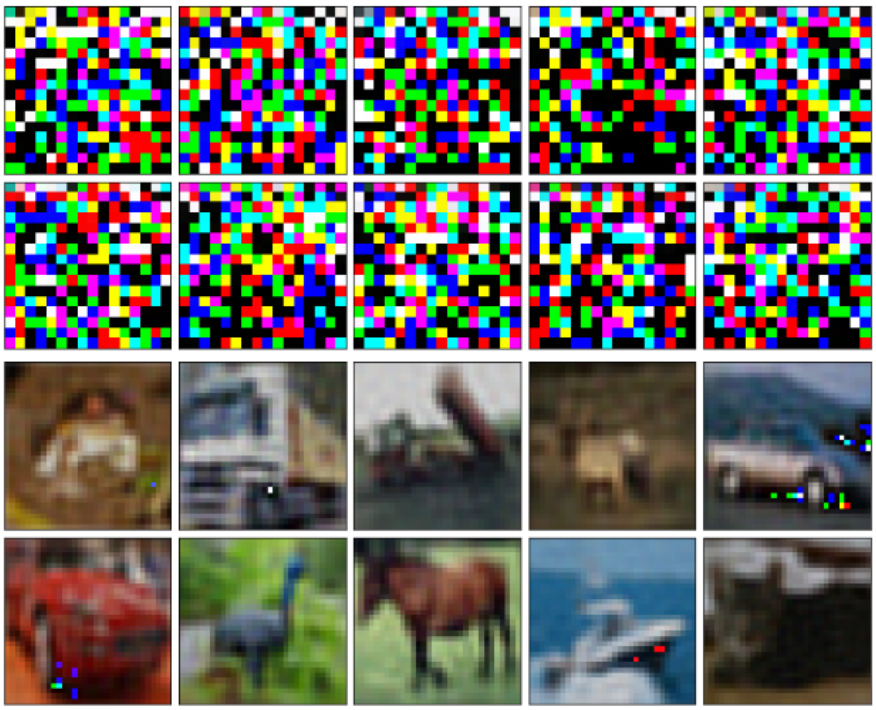
8 x 8



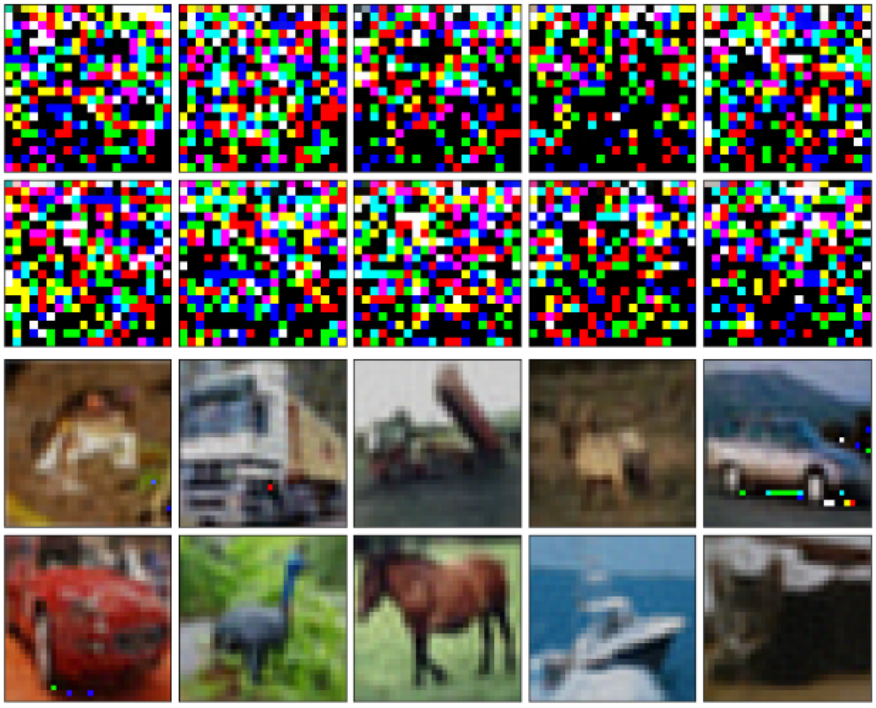
12 x 12



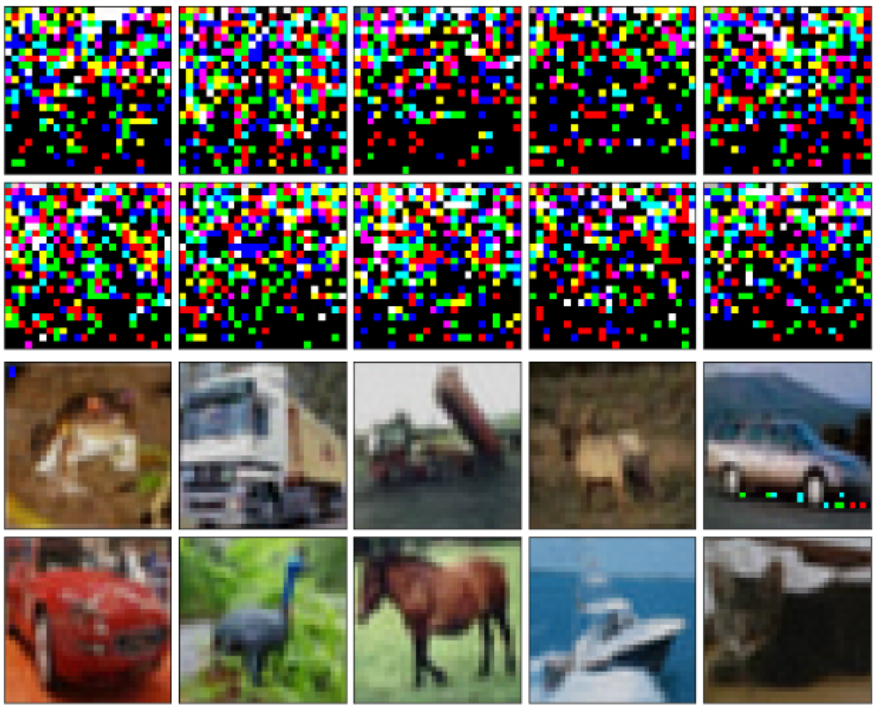
12 x 12



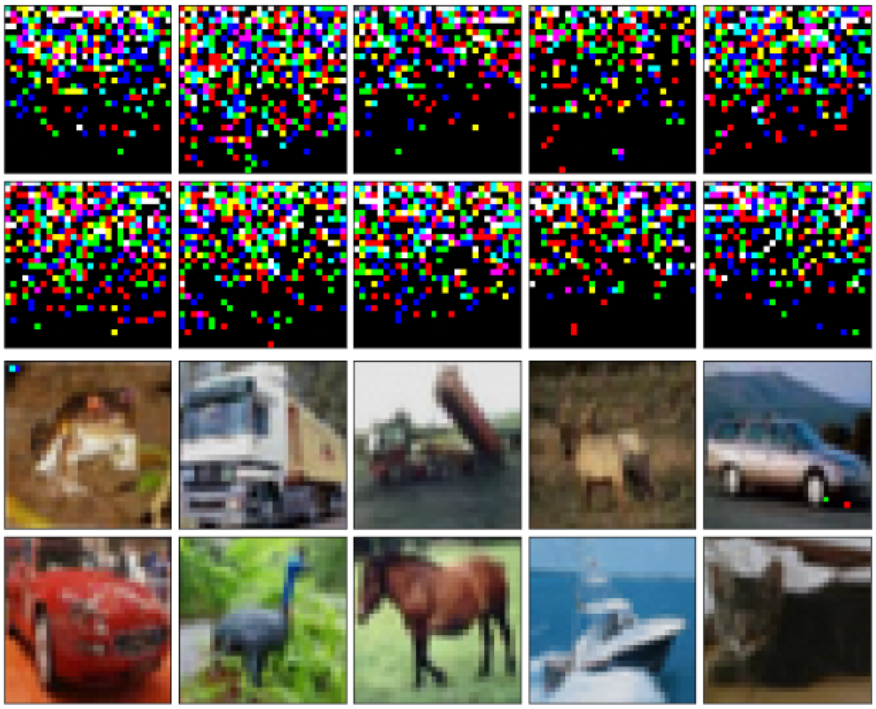
16 x 16



24 x 24



28 x 28



32 x 32
(Originals)