Implementation of Convolutional Autoencoder and Learnings on Image Compression

Xuanyu Chen

Australian National University, College of Engineering and Computer Science, Canberra ACT 0200
u5745613@anu.edu.au

Abstract. Autoencoders are a neural network family and being well-known for its advantage on compressing data and denoising the corrupted data. Among various techniques of implementing autoencoders, the convolutional autoencoders are efficient as it is capable of combining the benefits of convolutional architectures with the traditional autoencoders. By building a convolutional autoencoder and a convolutional neural network, we have found that how much information the image data will lose during the compression is depending on the rate of compression. The CIFAR-10 dataset is used for our training and testing due to the fact that an image data is better for the visualization of information compression. Besides, an adequate normalization technique is also implemented in order to fasten the calculation in training.

Keywords: Convolutional Autoencoder(CAE), Image Compression, Convolutional Neural Network(CNN), Data Normalization

1 Introduction

In neural network, autoencoders are a specific type in which the input is exactly the same as the output. They are well-known in the field of image compression, as they are capable of compressing the input images into a latent-space representation and then reconstructing the representation back to images (Chablani, 2017). This dimension reduction technique helps reducing the memory usage, increasing the efficiency of computation, and providing a visualization for the high dimensional data. On the other hand, due to the ability on feature selections and extractions, autoencoders are commonly used for denoising corrupted version of data as well (Monn, 2017).

Among various autoencoders, Convolutional Autoencoder(CAE) is an autoencoder based on the architectures of Convolutional Neural Network(CNN). CAE uses the convolution operator to filter the input image and extract features from the content. Implementing a CAE will provide us for a further understanding on image compression because the traditional autoencoders do not consider the fact that a high-level feature can be identified as a sum of low-level features (Galeone, 2016).

In this paper, both CNN and CAE are implemented and trained by the CIFAR-10 dataset. CAE is used to conduct the image compression and CNN is used to evaluate the difference between the original images and the corresponding autoencoder images (**Sec 2**). By analyzing the result, we are able to prove that higher compression rate causes more loss of information. Therefore, a reconstructed image from such latent-space representation will also be harder for machine to classify. (**Sec 3**)

2 Method

2.1 Dataset

CIFAR-10 (Krizhevsky, 2009) is a dataset which is widely used in many deep learning researches. The authors are Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. CIFAR-10 is made up of 32x32 color images collected from 10 classes. There are 60000 images in total, including 50000 train images and 10000 test images. We use CIFAR-10 dataset to

train both CNN and CAE in this paper since convolution is good at extracting features from image data and the image data can also offer people a visualization of the information compression.

2.2 Preprocess data by normalization

Normalizing the input data prior to the training is a useful technique in backpropagation networks. According to relevant researches, it indicates that an adequate normalization is able to achieve good results and fasten significantly the calculation in training (Sola & Sevilla, 1997). For a color image data, the value of each data point is ranged from 0 to 255. In order to normalize the data, we need to divide the data value by 255. The output will be ranged from 0 to 1.

2.3 Construct a CNN

A CNN is constructed following the structure of LeNet-5 (LeCun & Bengio, 1995). In general, there are two major sections in a CNN, the feature extraction part and the result derivation part. For the first part, it consists of several patterns. Each pattern has a convolution layer, activation layer and pooling layer. The pattern begins by extracting features in convolution layer, processing the information in activation layer later, and summing up the features in pooling layer at the end. The use of the first part is to compress high dimensionality data into low dimensionality data which inherits most of information in a concise form. For the second part, the output from the previous layers will be flattened and passed to a fully-connected network to calculate the possibilities of the input image in each class. In another word, from the machine's vision, the image is belonging to the classes whose possibility is the highest. The actual architecture of our CNN is shown as below:

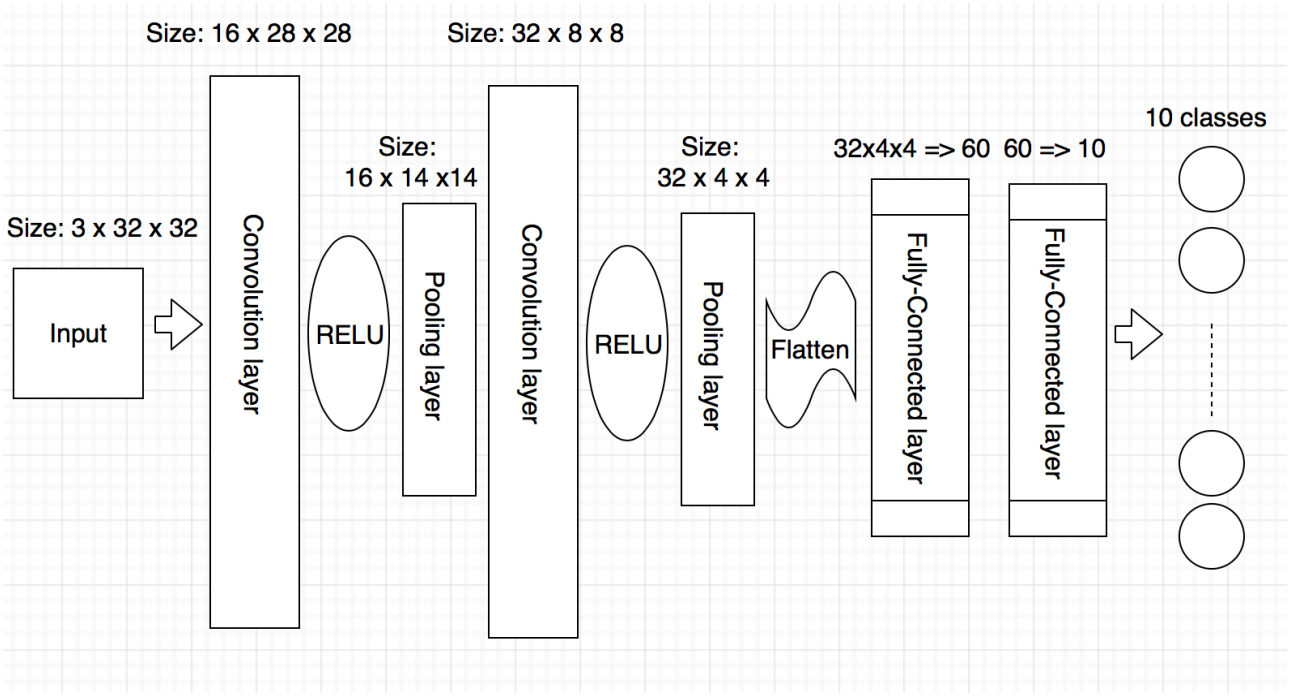

Fig. 1. CNN structure diagram

Besides, we choose RELU instead of other error gradient-based algorithms as our activation function under a consideration that RELU will try to solve the gradient vanishing problem in deep neural network.

2.4 Construct a CAE

The architecture of CAE is composed of two programs, the encoder and the decoder (Turchenko, Chalmers, & Luczak, 2017). The encoder firstly compresses the input image

into a latent-space representation using the similar structure as the feature extraction part in CNN. The decoder then tries to reconstruct an image depending on the information retrieved from representation using deconvolution and upsampling. The actual architecture of our CAE is shown as below:

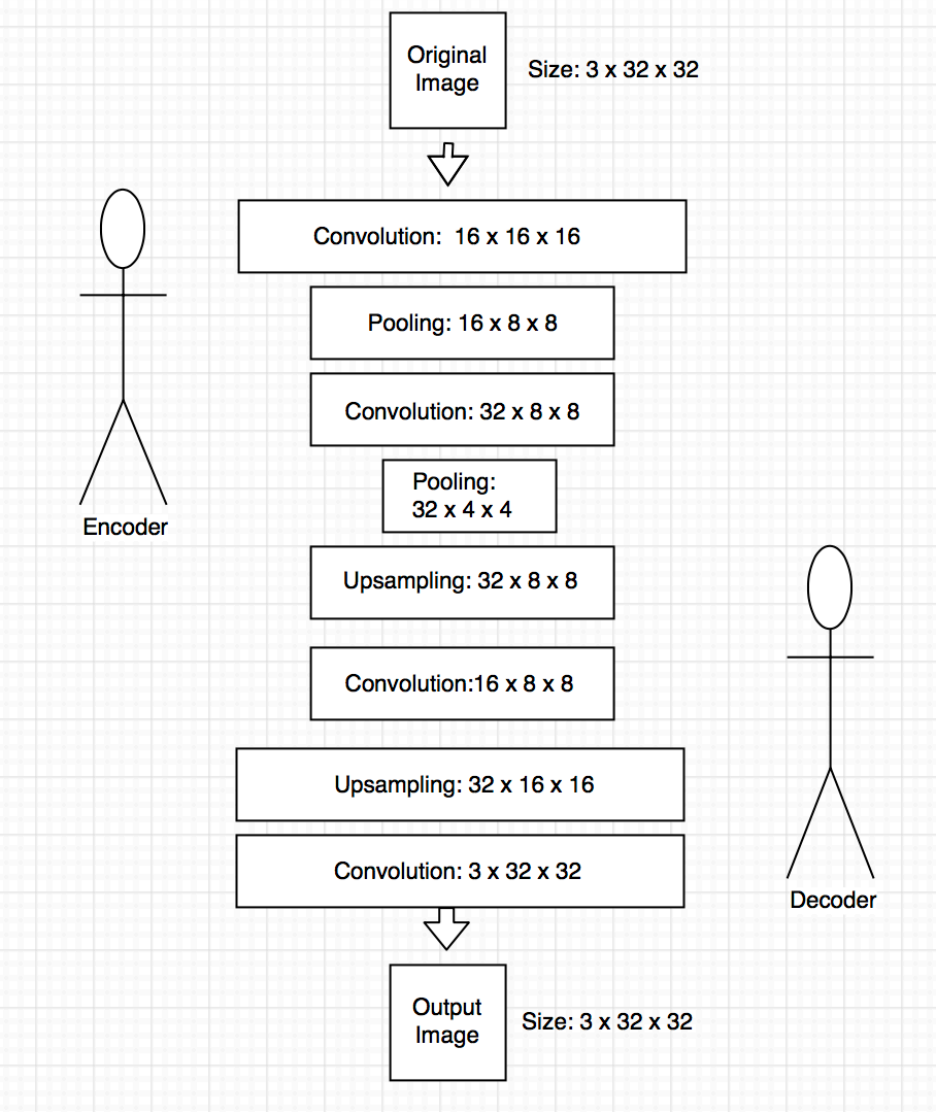

Fig. 2. CAE structure diagram

Additionally, RELU is used as the activation function throughout our CAE as well. We choose to use RELU for the same consideration as we choose it for CNN. For the upsampling techniques, the nearest neighbor upsampling is selected as this technique is straightforward to understand and the difference from selecting other upsampling techniques for the CAE is small to recognize.

2.5 Testing the performance of CAE

Training CNN for 5 epochs will give us about 65% accuracy on classifying the classes of CIFAR-10 test images. Such number of training iterations is enough to accept as the focus of this paper is not at achieving a as high as possible value of accuracy. Then, training CAE for 5 epochs as well and inputting the same input data into CAE will give us a reconstructed version of images. Comparing the accuracy of CNN on classifying the classes of original images with the accuracy on reconstructed images, the difference tells the performance of our CAE in image compression. If there is a small difference, it shows that the reconstructed images inherit most of the information from the original ones. In another word, it reflects how much information has been lost during the compression.

3 Results and Discussion

3.1 Result analysis

As the accuracy difference is able to indicate the ratio of information loss, it makes people wonder which factors can cause impact on this value. Generally guessing, the compression rate may significantly affect how much information of the original image will be lost during the compression. However, before providing a proof to this rule, it is necessary to show that there is an upper bound for improving the performance of CAE by training as we have made an assumption that CAE is not able to decide how much information will be lost. In addition, this will also answer that why it is enough to train CAE by only 5 epochs in this paper. We shall clearly confirm the assumption is true according to this loss curve:

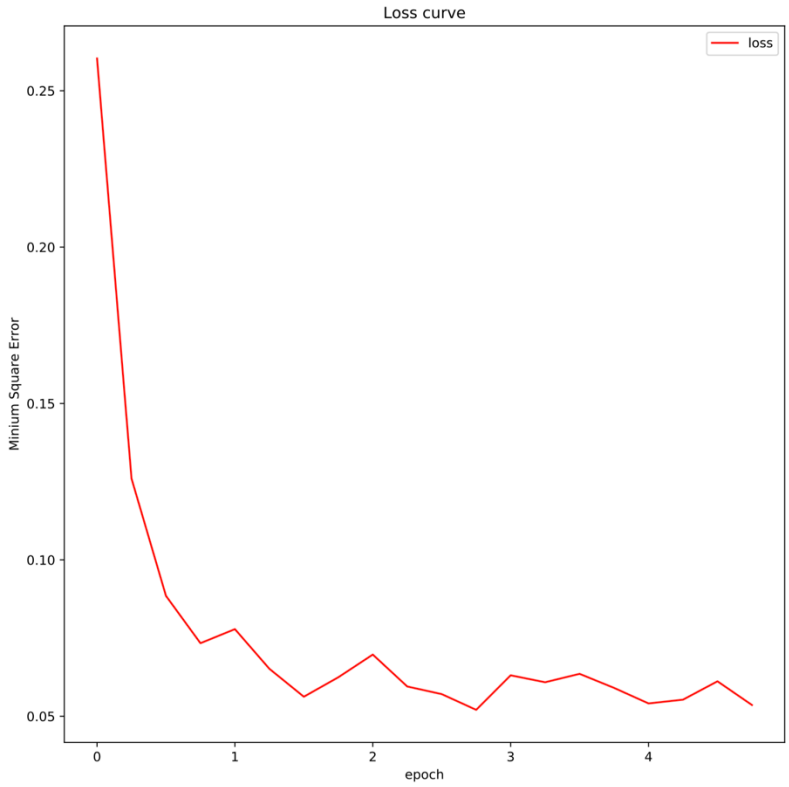

Fig. 3. CAE loss curve

From the loss curve graph, there is a global minimum of the minimum square error (MSE) at 0.05. This value reflects the loss of information in another form and cannot be decreased further through the increase of training iterations. Then, holding this agreement, we repeatedly adjust the structure of CAE to compress the input images into latent-space representations with different sizes. For instance, as the CAE structure diagram shows, our CAE currently compresses the input images from 3x32x32 to 32x4x4 representation. Both width and height are divided by 8, which is the compression rate of our CAE. By modifying the number of layers in encoder program, the CAE is able to compress the input data by some different rates, such as 4. The value of rate should be an integer which is as least 1 because a rate lower than 1 is not a compression. The following graph demonstrates this relationship between the compression rate and the loss in accuracy:

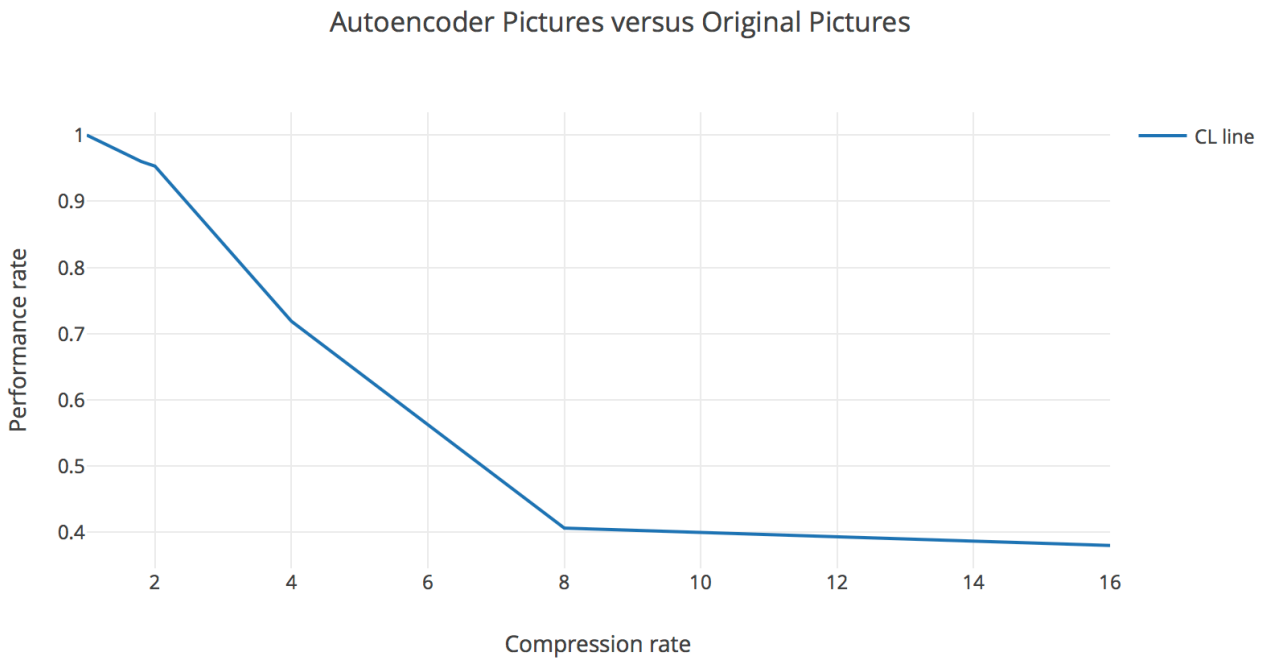

Fig. 4. Compression rate and accuracy rate

In this curve, the X coordinate represents the compression rate and the Y coordinate represents the performance ratio in CNN calculated by division of two accuracies. We can even visualize such relationship by viewing the reconstructed images from the latent-space representations in different sizes:

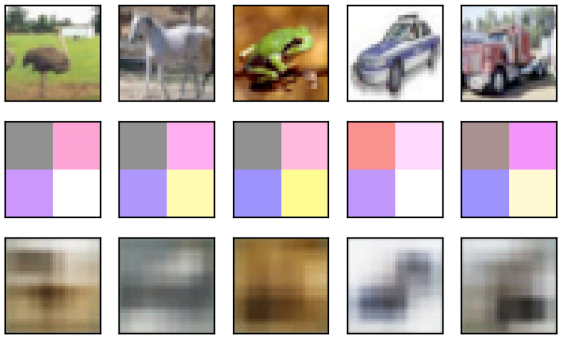

Fig. 5. compression rate = 16

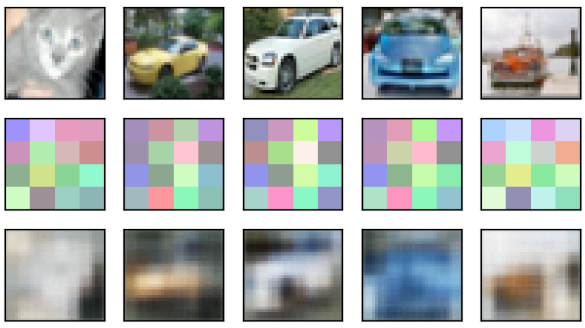

Fig. 6. compression rate = 8

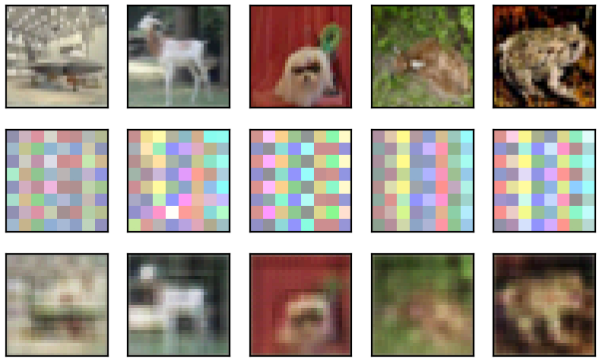

Fig. 7. compression rate = 4

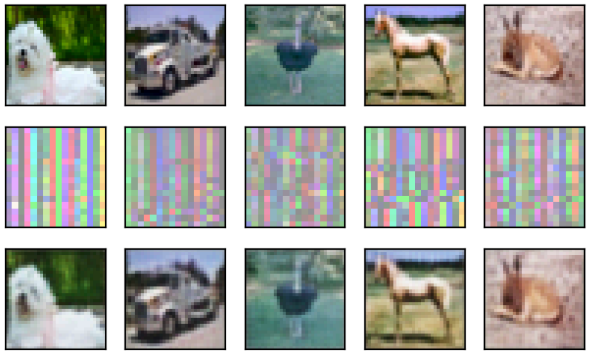

Fig. 8. compression rate = 2

The first line of the images is original images, the second line is the latent-space representation, and the third line is the reconstructed images. Therefore, according to the graph and these images, we are able to prove our assumption above that choosing a higher compression rate to compress an input image into latent-space representation will suffer more loss of information. The reconstructed images thus become harder for not only the human but also the machine to classify their classes.

3.2 Comparison with other paper

In the report “Winner-Take-All Autoencoders” published by Alireza Makhzani and Brendan Frey, the authors proposed a winner-take-all (WTA) method which is able to learn hierarchical sparse representations in an unsupervised fashion and combines this method with convolutional architectures to build a CONV-WTA autoencoder (Makhzani & Frey, 2015). They believe this autoencoder will have benefits from both sides. The autoencoder is experimented with various popular datasets in order to test its performance and prove their opinions. Their result of their autoencoders with CIFAR-10 is shown as below:

	Accuracy
Shallow Convolutional Triangle k-means (64 maps) [3]	62.3%
Shallow CONV-WTA Autoencoder (64 maps)	68.9%
Shallow Convolutional Triangle k-means (256 maps) [3]	70.2%
Shallow CONV-WTA Autoencoder (256 maps)	72.3%
Shallow Convolutional Triangle k-means (4000 maps) [3]	79.6%
Deep Triangle k-means (1600, 3200, 3200 maps) [20]	82.0%
Convolutional Deep Belief Net (2 layers) [6]	78.9%
Exemplar CNN (300x Data Augmentation) [21]	82.0%
NOMP (3200,6400,6400 maps + Averaging 7 Models) [22]	82.9%
Stacked CONV-WTA (256, 1024 maps)	77.9%
Stacked CONV-WTA (256, 1024, 4096 maps)	80.1%
Supervised Maxout Network [19]	88.3%

(a) Unsupervised features + SVM (without fine-tuning)

(b) Learnt dictionary (deconv-filters) 64conv5-64conv5-64conv5-64deconv7

Fig. 9. CONV-WTA autoencoder trained on the CIFAR-10 dataset

From the table, we can read that the accuracy of the Supervised Maxout Network is 88.3% for the original CIFAR10 image data but the accuracy then drops down to 72.3% for the data compressed by a rate of 2 (256 = 16 x16 = 32 / 2) and 68.9% for the data compressed by a rate of 4 (64 = 8 x 8 = 32 / 4). Besides, the authors also mentioned in the report that both the CONV-WTA autoencoder and neural network has been trained by NVIDIA GPU for enough times. Thus, the difference of the accuracy in their table supports our statement above that choosing a higher compression rate to compress image data will cause more information loss during the compression.

4 Conclusion and Future Work

We have made a statement that how much information of image data will be lost during the compression is depending on the rate of compression. Both a CNN and a CAE are built for the purpose of proving this statement. The CIFAR-10 dataset is used for our training and testing as choosing image data can offer people a visualization of the data compression. Prior to training, we also implemented an adequate normalization technique on the input data to fasten the calculation. The result of the experiment, including both a loss curve and the images in different compression rates, proves our assumption at the beginning is correct. Additionally, the data from another report “Winner-Take-All Autoencoders” also supports the statement from another hand.

The future work of this paper will be extending the statement that the loss rule will still hold for any kinds of autoencoder techniques. In this paper, the statement is proved based on the compression result of a convolutional autoencoder. In order to confirm the statement is without loss of generality in essence, it is necessary to give further evaluations about this rule on different autoencoders, such as an auto-associative autoencoder (Gedeon & Harris, 1992) and a variational autoencoder (Doersch, 2016). About the auto-associative autoencoder, the rule has been proved to hold according to the result data of our previous paper. So, we then would like to begin a research on implementing a variational autoencoder for the proof.

5 Reference

Chablani, M. (2017, June 26). *Autoencoders—Introduction and Implementation in TF.* Retrieved from towardsdatascience.com: https://towardsdatascience.com/autoencoders-introduction-and-implementation-3f40483b0a85

Doersch, C. (2016). *Tutorial on Variational Autoencoders.* UC Berkeley.

Galeone, P. (2016, November 24). *Convolutional Autoencoders*. Retrieved from P. Galeone's blog: https://pgaleone.eu/neural-networks/2016/11/24/convolutional-autoencoders/

Gedeon, T., & Harris, D. (1992, June). PROGRESSIVE IMAGE COMPRESSION. *Neural Networks, 4*, 403-407.

Krizhevsky, A. (2009). *Learning Multiple Layers of Features from Tiny Images.* Retrieved from The CIFAR-10 dataset: http://www.cs.utoronto.ca/%7Ekriz/cifar.html

LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 1995.

Makhzani, A., & Frey, B. (2015). *Winner-Take-All Autoencoders.* University of Toronto. Neural Information Processing Systems Foundation.

Monn, D. (2017, July 17). *Denoising Autoencoders explained .* Retrieved from towardsdatascience.com: https://towardsdatascience.com/denoising-autoencoders-explained-dbb82467fc2

Sola, J., & Sevilla, J. (1997). *Importance of input data normalization for the application of neural networks to complex industrial problems.* IEEE.

Turchenko, V., Chalmers, E., & Luczak, A. (2017). *A Deep Convolutional Auto-Encoder with Pooling - Unpooling Layers in Caffe.* Canadian Centre for Behavioural Neuroscience, Department of Neuroscience.