Convolutional Autoencoders and the Distinctiveness Measure

Jonathon Martin

Research School of Computer Science, Australian National University
{u5582907@anu.edu.au}

Abstract. Using a convolutional autoencoder with general architecture, a measure of distinctiveness is used to progressively prune the produced data encoding. It is found that the decoder component was able to tolerate a small amount of pruning to the encoded state, but excessive pruning introduced noise in the decompressed samples.

Keywords: neural network, convolutional, compression, distinctiveness

1 Introduction

Image compression is a pervasive technology in the digital age; enabling the fast transfer of data in quantities not achievable otherwise. Common compression schemes include both lossy formats e.g. JPEG, and lossless formats e.g. PNG, TIFF [1]. These formats and many others rely on the underlying strategies of pixel transformations, quantization and entropy encoding [2].

Compression itself is always a compromise between the quality of a decompressed artifact (i.e. similarity to the original) and the degree of compression achieved. Neural networks have an advantage over common algorithmic compression schemes, as they can easily be tweaked to achieve the desired degree of compression for a given application. The tradeoff for neural networks is that decompression is rarely optimal, as the procedure for doing so is learned from a finite set. A particular example of this is the method used by [3]. In this study Gedeon & Harris progressively reduce the size of the hidden layer in a neural network using *distinctiveness* as a measure, in order to achieve an appropriate balance between compression ratio and decompressed quality.

Convolutional neural networks (CNNs) are a popular variation of deep learning networks, which specialize in spatial data. Their operation of applying the same function over many 'receptive fields' of the input data is inspired by the operation of the human visual system. CNNs can be thought of as feature extractors, as they excel at generating high level features of inputs from lower level features or the raw input. CNNs have a long history in machine learning but have only recently been properly acknowledged for their feature extracting prowess. This is due to the large number of parameters required to achieve quality results, which required increases in computing power and the application of GPUs to neural network training. The exceptional performance of [4] on the ImageNet classification problem brought CNNs into the spotlight and they have since been applied in some the most advanced artificial intelligence projects to date.

Autoencoders are neural networks that have the same input and output space, generally with a smaller number of hidden neurons in the centre of the network. This structure causes the network to compress it's input and then decompress it back to the original. Autoencoders have been successful in a number of dimensionality and feature extraction tasks. For this reason, convolutional autoencoders are particularly interesting, as both CNNs and autoencoders do some amount of feature extraction. We can conceptualize this affinity by imagining that convolutional autoencoders do not create an arbitrary encoding of the input data, but rather extract its primary features which can then be pieced back together to form the original.

The *distinctiveness* of a hidden neuron in a network is a measure of its angle relative to its sibling neurons [5]. This angle is measured between the activation vectors over the pattern presentation set. This vector is determined by presenting the network with each pattern in the training set and recording the activations of each neuron, which is ultimately a representation of the functionality of that neuron. If two neurons have the same, or more realistically, similar activation vectors then they are said to be non-distinct and are interpreted as computing the same function. The intuition for reducing network size is that if two neurons are doing the same computation, then one of them is necessarily redundant.

Pruning is used in artificial neural networks for a number of reasons. One such is that researchers are not able to tell *a priori* the optimal network size for a given dataset. In this case, a researcher will construct a network that they are confident is larger than required (but hopefully will not overfit) and prune away neurons as needed until the measured drop in performance is no longer tolerable. A second reason is training time: a network with fewer weights to train will necessarily require less operations to perform backpropagation and will consequently take less time to train.

The goal of this paper is to assess the efficacy of applying pruning via distinctiveness to convolutional autoencoders. A small, general architecture has been chosen so that the results might be applicable to any application. The the UCI Faces dataset [8] was used for this implementation.

2 Method

The UCI faces dataset [8] consists of consists of 640 black and white face images. These are taken with varying poses (straight, left, right, up) and expressions (neutral, happy, sad, angry) with some of the faces are wearing sunglasses. The faces are available in 128x128, 64x64, 32x32 resolutions, here we use the 32x32 variation of [8] to achieve reasonable training times on the available hardware. This dataset is generally used for classification, but here we take the images for compression and ignore the labels. For ease of use with the designated machine learning library [6], it was preferable to convert the chosen images to PNG format, and pad the vertical dimension to make the images square. Figure 1 shows an example of one of the final images. This dataset was divided into a training set of 588 images, and a validation set of 52 images.

Fig. 1. UCI Faces example, preprocessed.

Each 32 by 32 image consists of 1024 values, which are scaled to the interval [0, 1] to be fed into the input layer of the network. These values are compressed into the 128 hidden layer values, which are then decompressed into the 1024 output values. The output values are then scaled back to an 8-bit range and converted into images. With all hidden neurons intact, this reduction in size is equivalent to a compression ratio of 8:1.

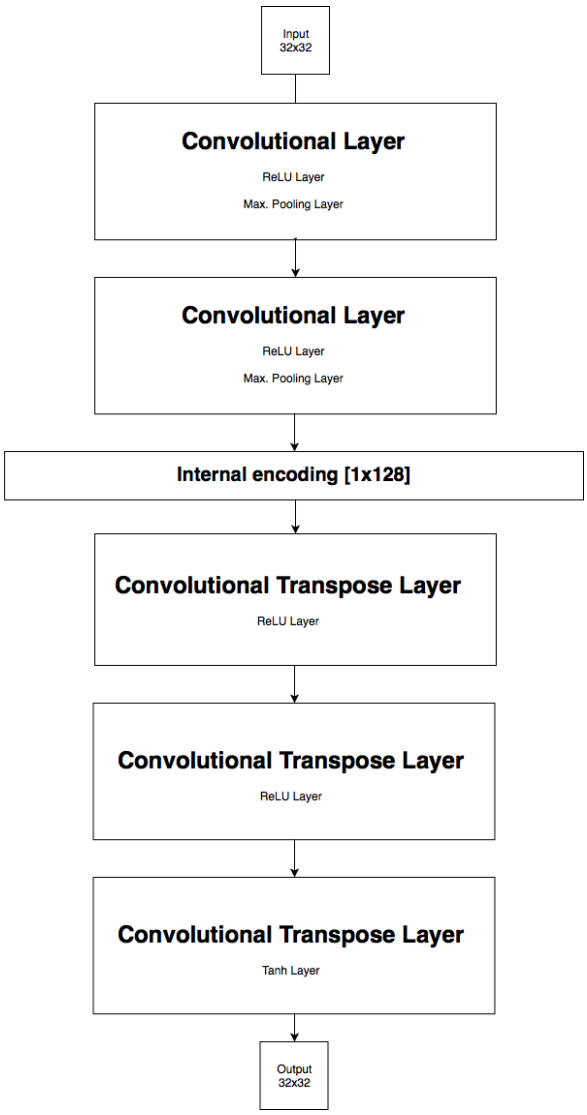

Fig. 2. Convolutional autoencoder architecture

An autoencoder consists of both an encoder and decoder as explained above, so the network architecture is defined in those two parts. The encoder consists of 2 convolutional layers, each followed by a Rectified Linear Unit (ReLU) layer, and then by a Maximum Pooling layer. A ReLU layer is simply the application of the ReLU activation function. This is a non-linear activation function which is frequently used in CNNs to combat vanishing gradients. The described function is as follows:

$$relu(x) = \max(0, x)$$

Max. pooling layers operate as a form of downsampling, reducing the size of the output of each convolutional layer, while preserving a reasonable amount of information. The first convolutional layer expands the single input channel to 256 channels, and the second reduces this to 32 channels. All layers together reduce the input space of 1024 to 128 values for each input sample. The decoder consists of 3 convolutional transpose layers, each followed by a ReLU layer. These together attempt to reproduce the input space. This architecture was modeled on [7] and modified for use with the distinctiveness measure and [8]. Figure 2 shows this architecture.

Each time the network training was performed, it was for 200 epochs, meaning 200 presentations of each image in the training set described previously. As each training batch was processed, the mean image of that batch was removed from each individual to reduce noise. This mean image was added back to the result before loss was computed. At the end of each 200 epochs, the activation for each item in the training set was recorded for each hidden neuron. The angle between each of these activation vectors was then computed. The pair was hidden neurons with the smallest angle between them was determined to be non-distinct, and one of the pair was removed from the network. Training would then begin again, until the desired number of hidden neurons was

reached. Since a convolutional autoencoder does not produce a 1-dimensional encoding, it was necessary to convert the encoding to 1-dimension for pruning, and then convert back for decoding. The data was not processed by any layers during this step. As a control, a second network was trained with the same variables, but without pruning. From here on, the un-pruned network will be referred to as network A, and the pruned network as network B.

A learning rate of e^{-4}, a batch size of 48 and L1 loss were used. While mean-squared error loss is more commonly used for autoencoder tasks, it was found that the L1 loss function, which minimizes the absolute differences between data points, produced more faithful reconstructions of the sampled images.

3 Results & Discussion

Images decompressed by the constructed networks definitely do not compare to modern compression schemes, but we do find an interesting effect in the resulting data.

Figure 3 shows both an original, non-compressed image and its decompressed equivalent produced by network A. Figure 4 shows a similar image pair produced by network B. Both of these sample results are from the validation set and have the relevant loss and number of pruned hidden neurons listed beneath them. We can see that both network A & B were able to reproduce the high-level structure of the input images, including markings in the background, the shape of the head and shoulders, and a rough hairline. However, with this level of pruning applied, network B produces images that are noticeably 'noisier' than network A.

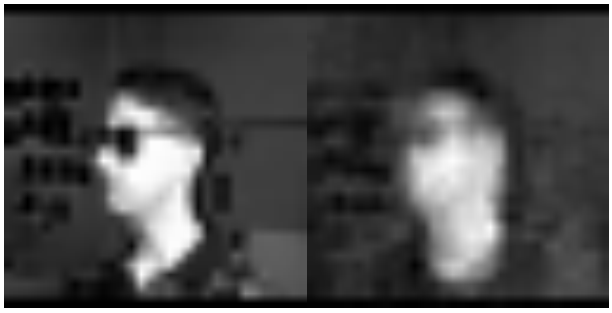

Fig. 3. Original (left), decompressed (right). Network A. Loss=0.0273, Pruned=0

Fig. 4. Original (left), decompressed (right). Network B. Loss=0.0287, Pruned=12

Fig. 5. Original (left), decompressed (right). Network A. Loss=0.0272, Pruned=0

It is worth noting that these effects are not consistent across the training of both networks. Take Figure 5 and Figure 6, which show image pairs taken from network A & B respectively after 400 epochs of training time. At this stage network B has had 4 hidden neurons removed. We can see that the quality of the produced images is similar, indicating some level of robustness exists in the convolutional architecture. These results are similar to those shown in [9], which found decompression quality of a linear autoencoder could be improved by pruning via distinctiveness, but only to a point, after which the quality quickly decreased again.

These results are consistent with our understanding of CNNs. If we return to the feature extractor analogy, we can imagine that removing particular neurons in the centre hidden layer interferes with the subsequent layers ability to detect nearby features. This introduces 'uncertainty' in these layers which manifest as noise in the resulting output.

This analysis of decompressed image quality and its divergence over the course of training is visualized in Figure 7 which shows the loss of each network over the 1200 total epochs that each was trained for.

It is clear that both networks presented were not able to reproduce the finer details present in the source images. Most prominent is the lack of obvious facial features in the decompressed samples. There are two possible explanations for this:

1. The constructed networks were not wide or deep enough to extract the high-level features needed in the internal encoding that would allow the decoder to reconstruct these fine details.
2. We are biased in assessing the quality of these reconstructions because we are human (presumably) and are especially equipped for detecting the presence of faces.

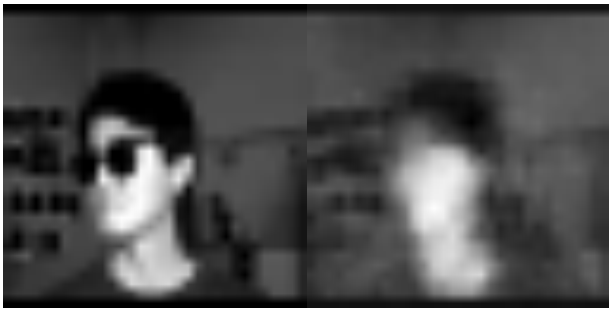

Fig. 6. Original (left), decompressed (right). Network B. Loss=0.0322, Pruned=4

Apart from this limitation in the reconstruction of faces (which may or may not exist) the training of both proposed networks was also limited by the size of the chosen dataset. Many successful applications of CNNs can be partly attributed to the immense data used in their training. For this reason, our application is at risk of overfitting – although the similar results found in both the final stages of training and the validation set provide some argument against this.

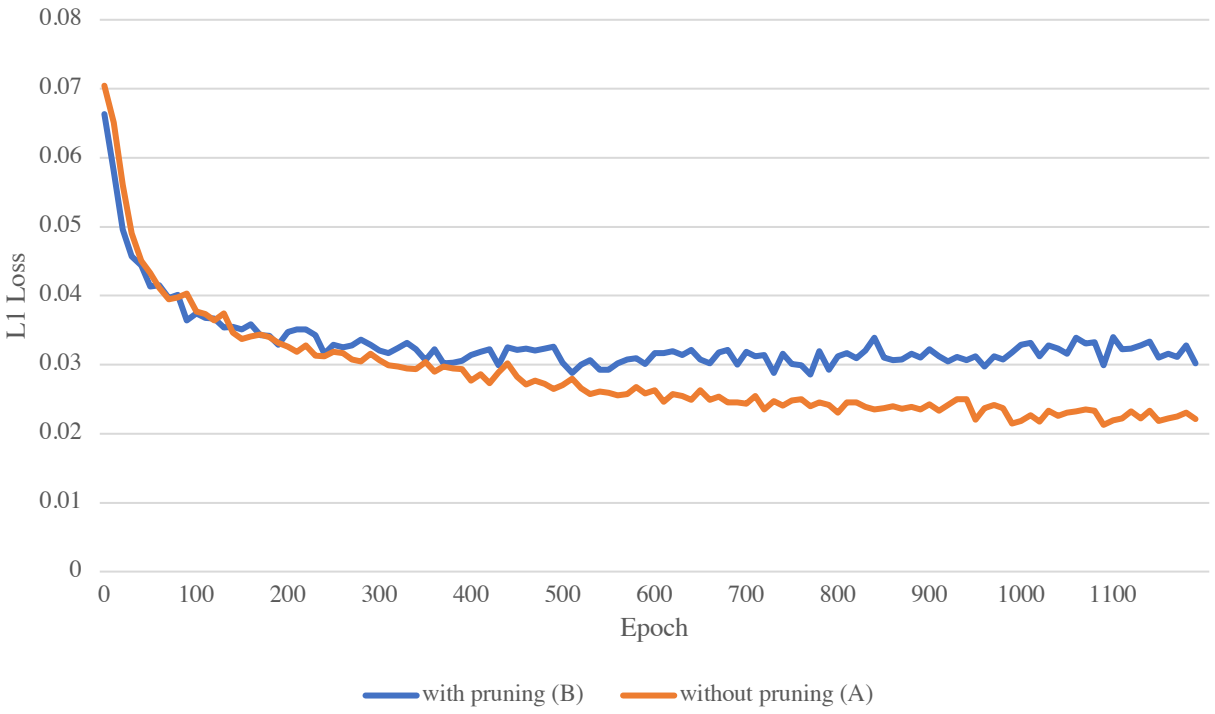

Fig. 7. Loss of network A & B during training

4 Conclusion & Future Work

The use of pruning via distinctiveness appears to have some value in the realm of convolutional autoencoders, but unfortunately must be monitored carefully to avoid over pruning. It may be that this technique is more valuable in processes that require less precision or that tolerate noise more effectively - possibly convolutional classifiers.

It would be useful to attempt replication of these findings with a larger, more varied dataset which would allow for a better generalization of the compression process. Performing a similar experiment with a larger network may also be valuable, this would allow one to assess the full potential of CNNs to reconstruct data from a minimal collection of primary features.

References

1. Murray, J., vanRyper, W.: Encyclopedia of Graphics File Formats, 2nd Edition. O'Reilly Media. (1996)
2. Jiang, J.: Image compression with neural networks – A survey. Signal Processing: Image Communication. vol. 14, pp. 737--760. (1999)
3. Gedeon, T.D., Harris, D.: Progressive Image Compression. IJCNN. (1992)
4. Krizhevsky, A., Sutskever, I., Hinton, G. E., ImageNet Classification with Deep Convolutional Neural Networks. NIPS. (2014)
5. Gedeon, T.D.: Data mining of inputs: analysing magnitude and functional measures. International Journal of Neural Systems. vol. 8, pp. 209--218.
6. PyTorch. http://pytorch.org/
7. PyTorch. beginner-autoencoder. https://github.com/L1aoXingyu/pytorch-beginner/blob/master/08-AutoEncoder/conv_autoencoder.py
8. Dua, D., Karra Taniskidou, E. UCI Machine Learning Repository. (2017). School of Information and Computer Science, University of California. http://archive.ics.uci.edu/ml.
9. Martin, J., Compressing Large Images using the Distinctiveness Measure. Research School of Computer Science, Australian National University.