Deep Convolutional AutoEncoder-based Lossy Image Compression

Patel Vishwa*, Desai Jaimin*, Solani Kunal, Chinathambi Sneha and Bangar Siddhi

Machine Learning Internship, SKNCOE, Pune

Abstract—**Big enterprises and organisations store a vast amount of images. The current technologies of image compression use similar characteristics within an image to compress the image. This document builds on to demonstrate how Deep Auto encoder neural network can be used to train on and compress large sets of images. MNIST handwritten digits dataset is used as training and testing set. 28 x 28 image is converted to a vector of size 784 x 1 which is then fed to the network. The 784 x 1 vector is then gradually compressed in each iteration of the training phase of the network. Once the vector is compressed, the middle hidden layer of the network will hold a compressed feature vector which can be stored or transmitted. This compressed feature vector later can be converted back to obtain a near approximation of the original image.**

Introduction

Image compression has been a fundamental and significant research topic in the field of image processing for several decades. Traditional image compression algorithms, such as JPEG [1] and JPEG2000 , rely on the hand-crafted encoder/decoder (codec) block diagram. They use the fixed transform matrixes, i.e. Discrete cosine transform (DCT) and wavelet transform, together with quantisation and entropy coder to compress the image. However, they are not expected to be an optimal and flexible image coding solution for all types of image content and image formats.

Deep learning has been successfully applied in various computer vision tasks and has the potential to enhance the performance of image compression. Especially, the autoencoder has been applied in dimensionality reduction, compact representations of images, and generative models learning . Thus, auto encoders are able to extract more compressed codes from images with a minimised loss function, and are expected to achieve better compression performance than existing image compression standards including JPEG and JPEG2000. Another advantage of deep learning is that although the development and standardisation of a conventional codec has historically taken years, a deep learning based image compression approach can be much quicker with new media contents and new media formats, such as 360-degree image and virtual reality (VR) . Therefore, deep learning based image compression is expected to be more general and more efficient.

PREVIOUS WORKS

Deep Auto encoders perform nonlinear dimensionality reduction on input images to reduce them to a compressed representation. It has been used a number of times in the past to compress images. The results obtained from these implementations and the insights gained from their work are mentioned below.

Huffman Coding is a lossless compression algorithm which assigns codes to the pixels based on the frequency of occurrence. Pixels occurring more frequently will have shorter codes whereas pixels that occur less frequently will have the same length codes. The result is variable length codes that contain integral number of bits. The codes end up having unique prefix property. The codes are stored on a binary tree. The binary tree is built using the codes, starting from the leaves and moving up to the root of the tree.

An experiment was performed to compress a grey scale image using Huffman coding algorithm. Huffman coding was used to reduce the number of bits required to represent each pixel. Experiment results showed that up to a 0.8456 compression ratio was obtained on the image.

COMPARISON WITH PCA & JPEG

Anand Atreya and Daniel O'Shea have presented their use of Deep Auto encoders. Their first step was to train the auto encoder. First the images were fed into the nonlinear encoding layer. This encoding layer gradually compressed the input image pixels into a compressed feature vector. After the encoding was complete, the compressed feature vector was again fed into the decoding layer which then reversed the process of the encoding layer to obtain the reconstruction of input images. The difference between the reconstruction and input images was regarded as error to further minimise the difference. Also, a sparsity parameter was used to ensure sparsity of the activation vectors in each layer except the output layer.

The next step was to find an ideal set of quantisation values using k-means clustering to encode the activation values. Finally, they compared this compression algorithm with other common algorithms like JPEG and PCA. They found that Deep Auto encoder image compression algorithm outperformed JPEG in high compression and reduced quality scheme. Also, Deep Auto encoder showed better non-linear representation of the input image than that of PCA and hence Deep Auto encoder had better reconstruction quality. They observed that Deep Auto encoder was able to find out statistical regularities in a specific domain of images which was not possible by JPEG.

REPRESENTING AND GENERALIZING NONLINEAR STRUCTURE IN DATA

G. E. Hinton and R. R. Salakhutdinov have also presented the use of auto encoder to reduce the dimensionality of data. They trained a stack of Restricted Boltzmann machines (RBMs) to pre-train the neural network with each RBM consisting a layer of feature detectors. After the pre training, the RBMs were unrolled to get full layers of the neural network which at last were fine-tuned using Back propagation algorithm to reduce the data reconstruction error.

RBM is a two layer network in which the input binary pixels are connected to the hidden layer using symmetrical weights. The binary pixels are the nodes in the visible layer and the hidden layer activations are updated using sigmoid function on the sum of all the products of inputs and their respective weights. Again a sigmoid function is used on the sum of all the products of hidden values and their respective weights. The hidden layer here acts as a feature detector. The number of nodes in the hidden layer is always kept smaller than the number of pixels in the input image so that the hidden layer does not end up learning an identity function. With gradual decrease in number of pixels in the hidden layers, more RBMs are added onto the network with first layer of feature detectors as visible layer until a desired encoding state is reached. It is mentioned in the paper that each layer of the detectors captures strong, high-order correlations and this is an efficient way to progressively reveal low-dimensional, nonlinear structure. After pre training the network, the encoder part of the neural network is then unrolled to get the decoder part where same weights of auto encoder are used. The weights are later adjusted in the fine tuning stage to get optimal reconstruction. They in general concludes that back propagation together with Deep Auto encoder is a brute force towards efficient representing and generalising nonlinear structure in data.

DEEP AUTO ENCODERS AS FEATURE DETECTORS

Andrew Y. Ng and others have demonstrated the use of large-scale unsupervised learning algorithms for building high-level features which includes the use of stack of RBMs to construct auto encoders.

They have presented the training using unlabelled data to create high-level, class-specific feature detectors. They have trained a 9-layer locally connected sparse auto encoder with pooling and local contrast normalisation on a large dataset of images (the model has 1 billion connections, the dataset has 10 million 200 x 200 pixel images). They trained the network using model parallelism and asynchronous stochastic gradient descent (SGD) on a cluster of 1,000 machines (16000 cores) for 3 days. They proved that it is possible to train a face detector without having to label images as containing a face or not.

They have used sparse Deep Auto encoder with three important constituents: local receptive fields, pooling and local contrast normalisation. The local receptive field method is used to scale the auto encoder to large images. The idea behind this method is that each feature in the auto encoder can connect only to a small region of the lower layer. Also, to achieve invariance to local deformations, they employed local L2 pooling and local contrast normalisation. L2 pooling allowed to learn invariant features. The deep auto encoder was constructed by replicating three times the same stage composed of local filtering, local pooling and local contrast normalisation. The output of one stage is input to the next one and the overall model is lastly interpreted as a nine-layered network. The first and second sublayers are known as filtering and pooling respectively. The third sublayer performs local subtractive and divisive normalisation and it is inspired by biological and computational models. The weights are not shared which allow the learning of more invariances other than translational invariances. In a nutshell, they show that Deep Auto encoders perform well as feature detectors in case of images. (Andrew Y. Ng, 2012)

THE ENHANCED COMPRESSION MODEL

The enhanced compression model compresses 8 bits per pixels to 3 bits per pixels. Also, the compression model used is lossless and perfectly recovers the image. Present compression models follow one of the basic theoretical compression models depicted in Figure 3 and Figure 4. Figure 3 presents the lossless compression model and Figure 4 presents the lossy compression model. The lossless and lossy compression differ only in the "Quantization" step as show below. Compression is generally obtained by eliminating different kind of redundancy. Pixel mapping reduces the inter pixel redundancy, Quantization reduces psychovisual redundancy and symbol encoding in final step eliminates the coding redundancy.

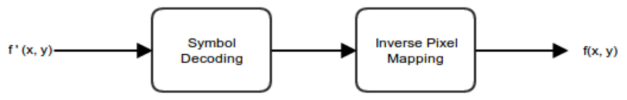


Figure 2 - Decompressing image

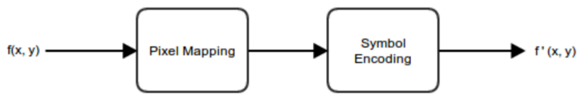


Figure 3 - Lossless compressing model

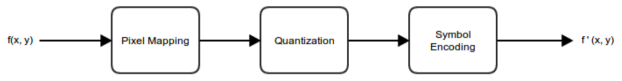


Figure 4 - Lossy compression model

f(x, y) : origin image,

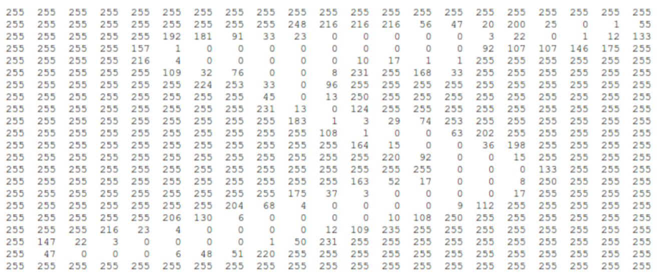
f `(x, y) : compressed image.

Above compression models are sufficient for compressing individual images. However, for a set of similar images containing set redundancy the general compression models are not sufficient. The enhanced compressed model proposes to handle set redundancy in image compression by introducing 'Set Mapping' in compression step and 'Inverse Set Mapping' in decompression step.

Deep Auto encoder learns the set mapping and inverse set mapping between multiple images automatically. In a way, the Deep Auto encoder is similar to the enhanced compression model proposed.

METHODOLGY

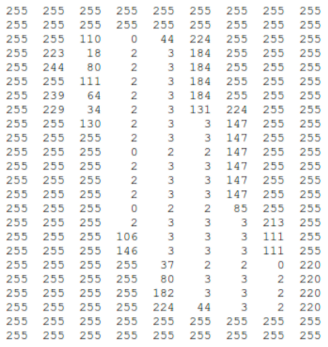
The handwritten digits images from MNIST database is



255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	248	216	216	216	56	47	20	200	25	0	1	55
255	255	255	255	255	192	181	91	33	23	0	0	0	0	0	3	22	0	1	12	133
255	255	255	255	157	1	0	0	0	0	0	0	0	0	0	92	107	107	146	175	255
255	255	255	255	216	4	0	0	0	0	0	10	17	1	1	255	255	255	255	255	255
255	255	255	255	255	109	32	76	0	0	8	231	255	168	33	255	255	255	255	255	255
255	255	255	255	255	255	224	253	33	0	96	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	45	0	13	250	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	231	13	0	124	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	183	1	3	29	74	253	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	108	1	0	0	63	202	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	164	15	0	0	36	198	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	220	92	0	0	15	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	0	0	0	133	255	255	255
255	255	255	255	255	255	255	255	255	255	255	163	52	17	0	0	8	250	255	255	255
255	255	255	255	255	255	255	255	255	175	37	3	0	0	0	0	17	255	255	255	255
255	255	255	255	255	255	255	204	68	4	0	0	0	0	9	112	255	255	255	255	255
255	255	255	255	255	206	130	6	0	0	0	0	10	108	250	255	255	255	255	255	255
255	255	255	216	23	4	0	0	0	0	12	109	235	255	255	255	255	255	255	255	255
255	147	22	3	0	0	0	0	1	50	231	255	255	255	255	255	255	255	255	255	255
255	47	0	0	0	6	48	51	220	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255

Figure 7 - Data sample showing matrix of digit 5

used as the training and testing dataset. The images are of size 28 x 28 (784 total pixels in each image) and hence the size of input layer is 784. There are altogether 60000 images in the MNIST database. First 20000 images are used as training data and 1000 images are used as testing data. The data samples are shown in Figure 7 and Figure 8.



255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255
255	255	110	0	44	224	255	255	255
255	223	18	2	3	184	255	255	255
255	244	80	2	3	184	255	255	255
255	255	111	2	3	184	255	255	255
255	239	64	2	3	184	255	255	255
255	229	34	2	3	131	224	255	255
255	255	130	2	3	3	147	255	255
255	255	255	2	3	3	147	255	255
255	255	255	0	2	2	147	255	255
255	255	255	2	3	3	147	255	255
255	255	255	2	3	3	147	255	255
255	255	255	2	3	3	147	255	255
255	255	255	0	2	2	85	255	255
255	255	255	2	3	3	3	213	255
255	255	255	106	3	3	3	111	255
255	255	255	146	3	3	3	111	255
255	255	255	255	37	2	2	0	220
255	255	255	255	80	3	3	2	220
255	255	255	255	182	3	3	2	220
255	255	255	255	224	44	3	2	220
255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255

Figure 8 - Data sample showing digit 1

NETWORK ARCHITECTURE SETUP

The neural network used here is a Deep Auto encoder which consists of two symmetrical deep belief networks that typically have two to four layers of encoding hidden layers and another two to four layers of decoding hidden layers. Each pair of layers in the deep belief network is part of a

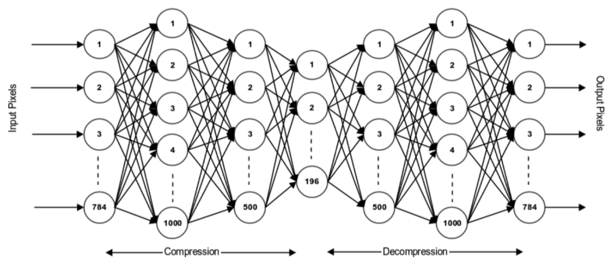


Figure 9 - Deep Autoencoder architecture

stack of Restricted Boltzmann Machines (RBMs). Each pair consists of two layers: a visible layer and a hidden layer (feature detecting layer).

The encoding layer consists of three hidden layers with each of size 1000, 500 and 196 respectively. The first hidden layer is sized 1000 nodes because expanding the size in this way compensates the incomplete representation provided by the sigmoid belief units. Hence, the encoding layer compresses the 784 pixels into 196 values. *list*)

The encoding layer is first pre-trained by forming 3 RBMs: input layer and first hidden layer, first hidden layer and second hidden layer and second hidden layer and third hidden layer. Each RBMs are pre-trained at first to produce a 196 size compressed feature vector. Then, the RBMs are unrolled to form the decoding layer of the deep belief network. Hence, the decoding layer consists of two hidden layers and one output layer of size 500, 1000 and 784 respectively. The decoding layer then transforms the compressed feature vector by feed

EVALUATION & TESTING

Evaluating the final predictions made by the Neural Network and comparing the accuracy and loss with the expected values and try to achieve it using Optimizer and Loss function.

Optimizer : adam
Loss Function : binary_cross_entropy

Testing the final model and deploying it on a web server with proper front end finishes the software development cycle of the proposed model.

REFERENCES

- The Hundred-Page Machine Learning Book by Andriy Burkov
themlbook.com
- Hands-On Machine Learning with Scikit-Learn, Keras and Tensor Flow: Concepts, Tools and Techniques to Build Intelligent Systems by Aurelien Geron

- Towards DataScience