Hybrid Auto-encoder Self-Organizing Maps for Image Compression

Nathan Elazar
Australian National University, College of Engineering and Computer Sciene
u5826185@anu.edu.au

Abstract. In this paper we propose a new model for unsupervised learning which uses both an auto-encoder and a self-organizing map. We evaluate our model at image compression using the CIFAR-10 dataset and find that our proposed hybrid model achieves a better reconstruction error than an auto-encoder by itself.

Keywords: Neural Network, Auto-encoder, Self-organizing Map, Unsupervised Learning, Compression

1 Introduction

While supervised neural networks have shown remarkable success across a wide range of applications, they still suffer from considerable draw-backs, such as a need for large amounts of labelled data. Unsupervised learning methods offer to alleviate this, and other flaws of popular neural network models [1]. In this paper we implement a popular unsupervised type of neural network called an autoencoder, as well as a hybrid autoencoder and self-organizing map model. Our goal is to determine whether the combination of these two types of unsupervised learning can lead to better compressed representations of image data. We evaluate both an autoencoder baseline and our hybrid model's unsupervised learning capabilities by measuring how well they can perform image compression on data from CIFAR10, and how well the compressed representations they produce can be used for image classification. We choose to use the CIFAR10 image dataset to evaluate our models because its images are relatively complicated (for example, CIFAR10 contains images of animals) and in colour, making it difficult for unsupervised models to compress them.

2 Models

2.1 Feed-forward Neural Network

Feed-forward neural networks (NN) are a special case of parametric regression where the function to be learned is equal to the composition of multiple layers, each layer is a function of the form $\sigma(T(x))$ [2]. Here, T is an affine transformation from the layer input dimension to the desired output dimension, and σ (known as the activation function) is a non-linear function applied component-wise [2]. During training, an optimization algorithm is used to find parameters of T that minimize some *loss* function. The loss function takes as input a transformation T and returns a real number, representing the difference between the models' predictions and the actual target values in the training set [2].

2.1.2 Convolutional Layers

Convolutional layers are a specialized neural network layer, designed to work specifically on image input [3]. In a convolutional layer, instead of applying an affine transformation to the entire input, an affine transformation is applied to many different *patches* of the input image. How an input image is split up into patches is determined by the *patch size*, which controls how big each patch is, and the s*tride,* which controls the spacing between two consecutive patches. For example, with a patch size of 3x3 and a stride of 1 every possible 3x3 grid of pixels from the input image would be treated as a separate patch. Each of these patches would have the same affine transformation applied to it, and the outputs from each of the affine transformation applications would be arranged into a grid, or a *feature image*, which is then passed through an activation function to produce the output of the convolutional layer [3]. Networks using convolutional layers have been shown to significantly outperform regular feed-forward neural networks in image processing tasks [3].

2.1 Auto-encoder

Auto-encoders are a form of unsupervised learning algorithm that attempt to construct a condensed representation of their input [4]. They are functionally equivalent to a feed-forward neural network with target values which are precisely

its inputs, so that its loss function outputs the difference between the model output and the original input [4]. By forcing the input to flow through a layer with a much smaller dimensionality than the input data, the activations of that layer can be considered a compressed representation of the original data.

2.3 Self-Organizing Map

Self-organizing Maps (SOM) are another form of unsupervised algorithm that attempts to perform dimensionality reduction of the input data. A SOM consists of multiple units, each unit has a vector of the same dimension as the input and a coordinate in the map grid which is of much lower dimensionality, usually only 1 or 2 [5]. To train the SOM, all units' vectors are initialized to small random values, then an input vector is sampled at random from the training dataset and presented to the units. The unit with the closest vector to the input is deemed the best matching unit (BMU). The BMU and units with grid-coordinates near the BMU then have their vectors moved closer to the input sample. How 'near' two units are is determined by a *neighborhood* function, which takes the coordinates of two units and returns a real number indicating their distance. The amount that each units' vector is updated is given by the product of the neighborhood function with the BMU, the learning rate, and the distance between the BMU's vector and the input vector. This process is repeated many times, with the learning rate reduced over time [5].

After training, the SOM can be used to *discretize* input samples. A given input vector can be replaced by the vector of that inputs' BMU. Although the replaced vector still has the same dimensionality as the original input, it can be stored with one number – the ID of the corresponding BMU. This means that the replaced input can be used in a neural network and still maintain the benefits of a high dimensional vector, while being able to be stored with one number, effectively compressing it.

2.4 Hybrid AE-SOM

Inspired by [6], we propose a hybrid AE-SOM model, which uses an AE to learn a dense representation of the input, then uses the activations of the AE hidden layer as training input into a SOM. It is hoped that this model will allow better compression, by combining the benefits of dense and discrete representations. In [6], a similar model was used for the purpose of learning an invertible function, however in this work we emphasize the techniques compression ability.

3 Experiments

3.1 Dataset

To determine these models' capacity for compression we evaluate them on CIFAR-10, a publicly available dataset of 32 by 32 RGB images. Each image is labelled with one of 10 classes, indicating the object in the image. The dataset contains 60,000 images, in our experiments we use 40,000 for training, 10,000 for validation and 10,000 for testing. In our experiments we scaled pixel colour values to be between 0 and 1.

3.2 Classification

To determine how well the compressed representations can be used for classification tasks, we use our AE and AE-SOM to create compressed representations of our dataset, then use those to perform classification of the image labels.

3.3 Compression

To evaluate raw compression ability, we use the AE and AE-SOM compress images and then reconstruct the original input image. We use the mean squared error of the original input image and the reconstructed image, also known as the reconstruction error, to measure model performance.

3.4 Implementation Details

The architecture we use in our experiments consists of 3 layers. The first layer is a convolutional layer with dimensionality 5, patch width 5, and stride 2. The second layer is a fully connected layer with dimensionality 500. The

third is the so-called *compression* layer, which is a fully connected layer with a dimensionality $d<<500$. The final layer is the output layer, which outputs either the reconstructed image or label classification, depending on the experiment.
All of our neural network models were implemented with an ELU activation function, as it has been shown to have superior performance to other popular activation functions, especially for image processing [7]. We trained our models using stochastic gradient descent with momentum=0.5 and a learning rate of 0.5. Models were trained for 200 epochs, afterwards their accuracy on the test dataset is measured. Models were implemented with a hidden dimensionality of 20, 50, 100 and 200. Since the original images are 32*32*3=3072 dimensional, these hidden sizes represent quite significant compression ratios.

The SOM used in the AE-SOM had 400 units, arranged in a 20x20 grid. The SOM used an initial learning rate of 0.01, and its neighborhood function was a Gaussian wit sigma=0.3. After the AE had been trained for 200 epochs, it was used to produce compressed representations of all images in the dataset. Compressed representations from the input data were then randomly sampled and presented to the SOM 30,000 times. The AE-SOM was then constructed by creating a new network with the same hidden weights as the AE, and a new output layer. The output of the hidden layer is concatenated with the SOM discretization of the input and fed into the output layer. The AE-SOM was then trained for a further 200 epochs.

All of the model hyper-parameters and network architectures were chosen so as to maximize validation set accuracy. The number of epochs to train for, 200, was chosen because all models' validation accuracies consistently converged within 200 epochs during our experiments.

4 Results and Discussion

4.2 Classification

Table 1. Validation classification accuracy.

D	AE	AE-SOM	NN
25	3567	35.76	53.66
50	36.97	37.40	54.96
100	38.21	38.16	53.88
200	38.89	39.02	54.59

We implemented a feed-forward NN as a baseline to compare classification accuracies against, as can be seen the auto-encoder model performed significantly worse than the baseline. The only difference in our AE and NN models is that for the AE the hidden layer was first trained by trying to predict the input, then the hidden layer was frozen while a new output layer was used to learn to predict image labels. While the NN model had both hidden and output layer trained at the same time. This might indicate that the hidden representations learned by NN models are very specific to the training goal, and not generalizable, since the AE output layer was not able to learn to use the hidden representations effectively. This phenomenon is an active area of study in transfer learning [8].
The AE-SOM model performed better than the AE by itself, although still worse than the NN baseline.

4.3 Compression

Table 2. Test set reconstruction mean squared error.

D	AE	AE-SOM
25	54.3628	47.1820
50	39.7447	33.4436
100	27.8556	22.4578
200	18.7145	14.5785

It can be seen that our hybrid AE-SOM model performs noticeably better than the baseline AE. This would indicate that the discrete compression learned by the SOM is orthogonal to the dense compression learned by the AE.

5 Conclusion and Future Work

In this paper we have demonstrated the effectiveness of the proposed hybrid AE-SOM model for unsupervised learning and image compression. We have demonstrated that self-organizing maps and auto-encoders learn different compressed representations of the input and that there is benefit in combining these approaches. In the future we would like to investigate a more cohesive model by learning SOM and AE representations of the input jointly, instead of training the SOM on the output of the AE as in this work. Furthermore, we would like to investigate the use of the SOM output map coordinates as features to be used in the model, this would likely involve using a SOM with a much higher dimensionality than 2.

References

1. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, *61*, 85-117.
2. Bebis, G., & Georgiopoulos, M. (1994). Feed-forward neural networks. *IEEE Potentials*, *13*(4), 27-31.
3. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).
4. Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504-507.
5. Kohonen, T. (1998). The self-organizing map. *Neurocomputing*, *21*(1-3), 1-6.
6. Mus, A. O. (2010, November). Inversion of many-to-one mappings using self-organising maps. In *International Conference on Neural Information Processing* (pp. 447-453). Springer, Berlin, Heidelberg.
7. Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.
8. Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1345-1359.