Self-Learning Neural Network Architectures for Variable Length Encoding Schemes

Nicholas Nordlund[1], Heesung Kwon[2], Geeth de Mel[3], and Leandros Tassiulas[1]

[1]Yale University
[2]U.S. Army Research Laboratory
[3]IBM Research - UK

***Abstract*—Data analytics problems require real-time processing of complex data gathered by a large network of sensors. As networks grow larger and the analytics more complex, centralized solutions are becoming increasingly infeasible. Analytics problems leverage the density of sensor networks to make predictions based on spatio-temporal correlations which are difficult to compress and turn into a more easily communicable form. Our previous work uses neural networks to learn a fixed-length encoding scheme to capture these correlations and solve analytics problems using distributed inference. In this paper, we present a novel neural network architecture that allows for variable size outputs from fully connected neural network layers.**

I. Introduction

The growth of sensor networks in tactical environments is making data available faster than we can extract meaningful insights from it. Latency-sensitive tasks such as real-time threat detection and tracking can suffer due to the communication and computational costs associated with a centralized approach. In our previous works, we leverage the fact that data gathered from dense sensor networks is often correlated across collocated sensors. [1], [2]. We design a network of distributed learning agents that learn fixed-length encodings for their video frames and communicate encoded frames to other agents. Each agent then uses the global information contained within the encoded frames it receives to augment its own local knowledge when processing a video analytics query.

While this system allows learning agents to handle correlated inputs naturally and cooperate with neighboring sensors, the use of a fixed-length encoding limits its performance. Different inputs contain different amounts of information, yet a fixed length encoding schemes always encodes all inputs at the same rate. Our previous approach to the distributed inference problem relies on using neural networks to learn encoding schemes. This approach is based on the concept of neural networks as autoencoders. Here, a neural network processes an input into a smaller dimensional space. The network then reconstructs the image from this compressed form. It is this compressed form that we broadcast to other agents in our distributed inference problem.

Neural networks are limited by their static architectures. For example if the hidden layer that generates an encoding contains ten neurons, then the encoding will be a sequence of ten 32-bit numbers. Recurrent neural networks (RNN) do allow for variable length outputs, but training them to learn the lowest possible length depending on each input would be difficult [3]. Learning the number of iterations before a RNN should stop generating outputs is impossible to represent in a differentiable loss function.

In this short paper, we introduce the concept of a hidden layer that has "volume". In an autoencoder with "volume", we learn a scalar volume parameter $\theta \in [0, 1]$ along with the weights for the hidden layers. This parameter determines the fraction of neurons in the encoding layer that contribute to the final encoded input state. For example, if there are ten total neurons in the hidden layer and its volume θ is $.5$ for a given input, then only the outputs of the first five neurons make up the final encoding. We show that autoencoders that use this method of forming variable length encoding achieve similar reconstruction accuracy to vanilla autoencoders on the MNIST data set of handwritten digits while reducing the size of the encoded state as much as 20%.

Future works will generalize the techniques presented here to more complex applications of distributed inference in tactical networks. For example, a coalition of video cameras from different groups who cannot share centralized computational resources may cooperate to perform distributed real-time threat detection. This short paper serves merely as a proof of concept for our novel method of self-learning neural network architectures.

II. Autoencoders with Volume

We train two autoencoders - one with our new volume mechanism and another vanilla autoencoder with a fixed size encoding. Both autoencoders are trained over ten epochs on the training set of 60,000 handwritten digits from the MNIST data set. The performance of both autoencoders is validated on a set of 10,000 digits. From the results shown in Table I, we can see that both autoencoders achieve similar reconstruction fidelity, but the Volume Autoencoder uses nearly 20% fewer neurons.

We employ a simple architecture with three layers. The input layer is a fully-connected layer of 784 neurons that take the pixel values from the images as their inputs. The next fully-connected layer has 32 neurons. We refer to the outputs of this layer as the encoding of the input. The final fully-connected layer of 784 neurons reconstructs the original image

TABLE I
VARIABLE LENGTH VS. FIXED LENGTH RECONSTRUCTION ERROR

	Error	Avg. Encoding Size (Neurons)
Volume Autoencoder	0.016	26.1
Vanilla Autoencoder	0.011	32

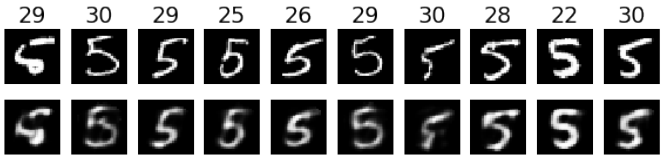

Fig. 1. Ten examples of the original digits from the validation set (top row) and their reconstructions (bottom row). Each column is labeled with the number of neurons used for reconstruction as determined by the volume parameter.

from the encoded space. We use a ReLU activation function for the encoding layer and a sigmoid function for the output layer. The network is trained using an Adam optimizer with a learning rate of .001.

In a vanilla autoencoder, the output of the encoding layer will be of dimension $N \times 32$ for a batch size of N. In our architecture, we repeat this tensor thirty-two times along the second axis to form a new tensor of dimension $N \times 32 \times 32$. Finally, we perform element-wise multiplication between this tensor and a binary lower triangular matrix with ones along the diagonal. This masking allows us to train autoencoders with hidden layers of sizes between one and thirty-two that use the same weights.

To achieve a variable length encoding, we must train our network for all possible encoding sizes. While this may seem like a lot of extra processing, the computation of all thirty-two reconstructions only takes place in the training stage. Once the network learns to estimate the proper volume parameter θ for a given input, it can choose the size of the encoding accordingly. During this inference stage, the volume network is no more complex than a vanilla autoencoder.

We reason that at some point for each input, increasing the size of the encoded space has diminishing returns. Thus, we introduce a volume parameter to manage the trade-off between reconstruction accuracy and communication costs. The goal of this parameter is to determine the point at which the marginal gains in reconstruction accuracy of adding an additional neuron does not offset the increased cost of a larger encoding. The loss function used to train the volume parameter is therefore related to the reconstruction loss functions of each of the thirty-two autoencoders.

In the training stage, every input image results in thirty-two reconstructed images. We calculate the reconstruction loss as the mean square error between the predicted image and the original input image. This set of reconstruction losses is not differentiable, so the gradient descent methods used to train neural networks will not work. To make this function differentiable, we interpolate between the points using a linear least-squares regression, $\hat{\beta} = (X^T X)^{-1} X^T y$. In this expression, y denotes the vector of observed reconstruction losses for each length of the encoding, and each column i of X is counting set from 0 to 31 inclusive raised to the i^{th} power.

Our experiments suggest that a quadratic function is sufficient to approximate the reconstruction losses as a function of the volume parameter $\theta \in [0, 1]$. Therefore we define the penalty function $f(\theta) = \hat{\beta}_0 + \hat{\beta}_1 \theta + \hat{\beta}_2 \theta^2$. The total loss $L = MSE + \alpha f(\theta)$ is the weighted sum of the mean square error across reconstructions from encodings of all lengths and the penalty function $f(\theta)$. We train the neural network by taking the gradient of this loss with respect to all the neural network weights. We set the α parameter to .0001.

III. DISCUSSION

Upon visual inspection of input and reconstructed image pairs, images of digits that deviate most from their average have higher values for θ. Some examples of reconstructed images are shown in Figure 1. This intuitive result verifies that our autoencoder does not need as much complexity to encode digits that closely resemble the average digit, but when the input most deviates from the norm, the network can use additional neurons to describe it better. While vanilla autoencoders have no way of limiting their own complexity for easy-to-describe inputs, our method is capable of generating variable length codes.

Much more exploration remains before this new idea of layer "volume" can be applied to more complex applications. The networks we trained were very sensitive to the magnitude of the α parameter. Setting it too high drives the volume to zero, while setting it too low drives the volume to one. In the future, we may also experiment with different interpolation methods besides linear least-squares regression.

The neural networks used today have rigid architectures. Designing deep neural network (DNN) architectures for any given task requires a lot of intuition and experimentation with hyperparameters. Residual Networks were the most recent disruption to the field of DNNs because they allowed for some structural modifications to DNNs by allowing them to bypass some of their own layers [4]. The method we introduce here allows for a much more powerful form of learned structural modifications to neural network architectures.

REFERENCES

[1] N. Nordlund, H. Kwon, G. R. De Mel, and L. Tassiulas, "Image classification on the edge for fast multi-camera object tracking," in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, pp. 1–5, Oct 2018.

[2] N. Nordlund, H. Kwon, and L. Tassiulas, "Cooperative learning for multi-perspective image classification," in *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, June 2019.

[3] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 3104–3112, Curran Associates, Inc., 2014.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016.