Sample Title for a Paper Published in the SMU Data Science Review Journal

First Author[1], Second Author[1,2,3], and Third Author[3]

Graduate Student, Master of Science in Data Science, Southern Methodist University, Dallas TX 75275 USA {byrned,jsanterre}@smu.edu

Abstract. This paper presents a new semi-supervised learning technique which combines the dimensionality reduction technique of Parametric t-SNE to gate a collection of sparse interconnected collection of clustered Feed Forward Sub-Networks.

Parametric t-SNE creates a low dimensional map of training data which will then be used to gate a layer of feed forward "Expert" networks whose output is a sparse n-dimensional vector. The output layer is trained using a cross entropy loss function and classes are identified with logistic regression classifiers. The benefits of this novel approach allows for encoding similar classification characteristics in physically close proximity. Implementing a sparse versus fully connected layer limits the number of calculable parameters, and allowing for weak interactions across experts allows for relevant information to be shared but not to dominate or infiltrate distant experts with noise.

Deep learning's explosive growth in learning over the past few years has been field by the technique's success at embedding high dimensional data into multilevel encoded representations of increasing abstraction to solve complex functions and or to model natural processes such as Natural Language Processing (NLP) and Computer Vision, and Speech Recognition.

Deep Neural Networks DNNs learned embeddings are typically trained from labeled data by implementing a learning procedure utilizing a variant of gradient descent modulated by a learning rate to iteratively descend a generally convex polynomial order objective function to a learned representation.

In common feed forward DNNs, neurons from the input to the output layers are typically fully connected. That is every node at any one layer is connected to every other node in adjacent layers. This fully connected nature supplies each successive layer with any and all embeddings of its preceding layer in any possible combination. It is then the task of the training procedure to effectively reduce this over abundance of connections by iteratively applying the delta rule to adjust the weights on the connections. These learned weights then. act as a gating threshold which activates or deactivates the neurons they are connected. The net result is a collection of weights and neurons that when applied an input stimulus effectively recreates an alternative representation or classification of the input data at the output of the network.

There has been great successes achieved by researchers utilizing DNN architectures by increasing of both the amount of labeled training data

and the number of layers and thus parameters in the model. In general it has been shown that with sufficiently large datasets, increasing the number of parameters in DNNs can give much better prediction accuracy.

However, training such models is time consuming compute resource intensive. The difficulty in training deep neural networks begins and ends with the fact that the design principle on which many are founded, full connectivity, contributes to a quadratic increase in calculable parameters as the number of input dimensions and parameters increases. In a typical DNN, there may be millions of parameters and similarly large set of labelled training examples.

In a multi-classification problem, competing classifications with similar and dissimilar features will inject noise relative to each competing target classification. For example the theoretical ideal set of weights that consistently identifies cats in images is most likely in conflict with set of all possible weights of connections tuned to correctly classify trees. There will be some overlap, and that overlapping region is nonlinear objective function the neural network is trying to optimize. However, arriving at one of the many possible number of near optimal solutions to this equation is not trivial.

There have been a number of solutions to address this problem in the literature. The techniques vary, but the end results are typically lower pass filtering noise across the network.

Dropout is a method by which a tunable parameter p, often 50, of random hidden units and their connections are eliminated from the network with each training case. During test all available neurons are included, but the trained weights on their connections are multiplied by the dropout probability p. Dropout gives major improvements in training time and accuracy by encouraging each individual neurons to learn a feature by degrading or filtering the impact other hidden units have on its connection weights.

DropConnect is similar to Dropout in that it introduces dynamic sparsity within the model, but instead of dropping neurons and their connections, it drops only connections based again on a settable probability parameter.

Embedding auto-encoders have been successful in image noise reduction and modeling the time series relationships of stock trading data. Auto encoders work by compressing input data of n dimensions into a smaller p dimensional hidden layer which is then re-encoded into an n dimensional output layer through gradient descent training.

In Convolutional Neural Networks, pooling layers are often utilized to reduce the size of the feature maps coming out of a filter bank before feeding the resultant filtered feature map to successive layers. This has the effect of reducing the number of interfering connections, reducing the computational complexity, and low pass filtering noisy features.

While Dropout and DropConnect reducing the number of parameters during training, but reinstate them during testing, a true reduction in the network connections has not been achieved. Song Han et al. chose to learn both the weights on connections and the existence of connections during training. Their method first trains the network, prunes low weight

connections, and then retrains the network to fine tune the remaining connections.

Deep Compression has achieved compression rations of 98% with no loss of accuracy following a similar process, but in the final step they used Huffman encoding to further reduce the size of the network.

Inspired by research on Rat Brain Tissue which found that synaptic formation for specific functional brain regions can be modeled as a random formation, StochasticNets modified this above approach by starting with a sparse randomly connected network and ending with a sparse network. So the pruning is inherent in the design. Decebal Constantin Mocanu et al. argue that DNN should never have fully connected layers contrary general practice, and demonstrate the effectiveness of sparse representations on several diverse neural network architectures (RBMs, MLPs, and CNNs).

While reducing the number of interfering connections has been shown to be an effective compression and regularization technique, the problem with DNNs tasked with classifying numerous items is really a function of graph theory, local specializations, and near and far influences.

Frankle and Carbin (2019) found out that fine-tuning the weights after training was not required for pruned networks. They suggest that the structure of the pruned network is the determining factor.

There is a biological corollary to this approach as well. Certain pathways in the brain have repeating sparse connectivity subnetworks, motifs, that define their architecture. These repeating structures were also found in other real world networks include statistically significant subnetworks. Research has shown many complex biological, social and technical networks naturally form networks which tend to follow a logical, locally spatial dense connectivity with sparse regional connectivity between motifs. This local preference with weak connections with distant modules is repeated at many different macro levels appears to be a powerful modeling technique.

Thus that leads to the assumption that while sparse networks tend to out perform dense ones, perhaps the networks we should be trying to develop are composite networks comprised of a mixture of local experts combining to solve a larger problem or set of problems.

Such technique have also beeb implemented by Hinton et al. in his Adaptive Mixtures of Local Experts and sparsely gated mixture of experts model. These models using a separate gating network which permits access to segregated subnetworks in a feed forward neural network by selectively activating or deactivating access gates. These gates and the feedforward networks in them are trained to engage a Mixture of Expert (MoE) to solve the objective function in question.

In this paper we propose a modification of this MOE model in which the gating neurons are trained using a Parametric T-SNE loss function. Parametric T-Stochastic Neighbor Embedding (T-SNE) is a method by which the Mutual Information component of similar Gaussian distributions are converted into a Euclidian distances to reduce a high dimensional signal into a lower dimensional space while preserving the structure of the data. Parametric T-SNE is used to gate the MoE FF networks so that spatial clusters can form around similar structures. Furthermore, allowing t-SNE

to drive the gates allows us embed similarities in training data samples on top of the labels already attached to the data thus effectively applying multiple classifications to our data without any added effort.
Subnetworks are sparsely initialized using methods devised in StochasticNet, and owing to the the potential influence of far flung experts on local problems, minimal connections in-between expert subnetworks are permitted albeit at a smaller randomly assigned rate.
The MoE FF networks are then trained with gradient descent independent of the parametric T-SNE Gate Training.
The result is a mixture of experts model which is spatially aware, embeds classifications beyond that which can be calculated with the and which allows for contributions from far flung expert networks through minimal influence connections.

1 Introduction

2 A Second Section

2.1 A Subsection Sample

Please note that the first paragraph of a section or subsection is not indented. The first paragraph that follows a table, figure, equation etc. does not need an indent, either.

Subsequent paragraphs, however, are indented.

Table 1 gives a summary of all heading levels. There should be zero reason to have either a 3rd-level heading or a 4th-level heading. If you feel the need to have such headings, then you should restructure your document such that no 3rd-level or 4th-level headings are needed.

Table 1. Table captions should be placed above the tables.

Heading level	Example	Font size and style
Title (centered)	**Lecture Notes**	14 point, bold
1st-level heading	**1 Introduction**	12 point, bold
2nd-level heading	**2.1 Printing Area**	10 point, bold
3rd-level heading	**Run-in Heading in Bold.** Text follows	10 point, bold
4th-level heading	*Lowest Level Heading.* Text follows	10 point, italic

Displayed and numbered equations such as Equation 1 are centered and set on a separate line.

$$x + y = z \tag{1}$$

Please try to avoid rasterized images for line-art diagrams and schemas. Whenever possible, use vector graphics instead (see Fig. 1).

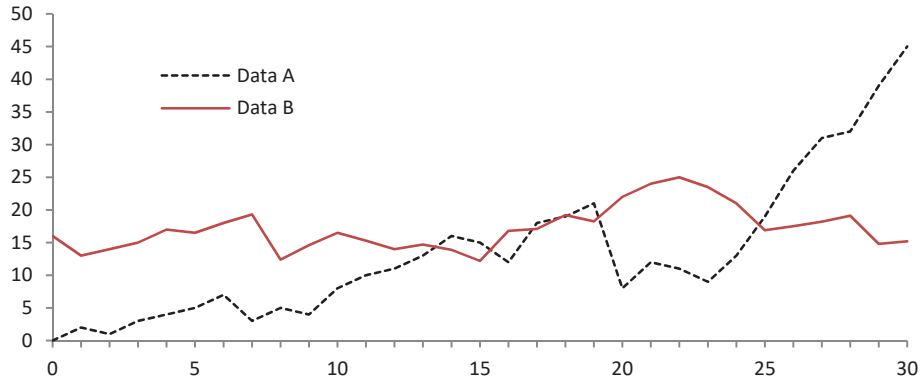

Fig. 1. A figure caption is always placed below the illustration. Please note that short captions are centered, while long ones are justified by the macro package automatically.

Theorem 1. *This is a sample theorem. The run-in heading is set in bold, while the following text appears in italics. Definitions, lemmas, propositions, and corollaries are styled the same way.*

Proof. Proofs, examples, and remarks have the initial word in italics, while the following text appears in normal font.

For citations of references, we prefer the use of square brackets and consecutive numbers where the references are numbered according to the ascending alphabet (i.e., a to z) of the first author's last name. Citations using labels or the author/year convention are not acceptable. The following reference provides a sample reference list with entries for journal articles [4, 5], a chapter [3], a book [1], and a conference proceedings without editors [2]. Multiple citations are grouped [4, 5, 3, 1, 2].

Note that URLs should be placed in Footnotes[1] and not in the references. Only documents that will not change over time should be placed in the references and cited.

References

1. Babington, P., Zimm, J.: The Title of the Book. The name of the publisher, The address of the publisher, 3 edn. (7 1993), an optional note
2. Draper, P., Flemming, I., Einstein, A., Turing, A.: The title of the conference paper. In: editor if there is one, T. (ed.) The Title of the Conference Proceedings Book. p. 213. The organization, The publisher, The address of the publisher (7 1993), an optional note
3. Eston, P.: The title of the work, 5, vol. 4, chap. 8, pp. 201–213. The name of the publisher, The address of the publisher, 3 edn. (7 1993), an optional note

[1] More information may be found at http://www.smu.edu. Last accessed 31 Dec 2018.

4. Freely, I.: Yellow river. The Journal of Small Papers **-1** (1997), to appear
5. Jass, H.: Phases of the moon. The Journal of Big Papers **4**(2), 201–213 (7 2001), an optional note