SCIENTIFIC RESEARCH CENTER

International Journal of Research in Science and Engineering
Vol. 6, No. 4, 2018, pp. 69-83.

ISSN 2347-9353

International Journal of Research in Science and Engineering

www.scientificrc.com

Use deep reinforcement learning in Nuclear classification in images

Fariba Nofeli[1], Mohammad Reza MahdaviZadeh[2]

PhD in General Engineering, University: Mississippi State University
Master of science in Civil Engineering, University: Mississippi State University

Abstract

The main purpose of this study is to investigate and evaluate the possibility of using deep reinforcement learning in core classification in images. For this purpose, deep reinforcement learning was introduced and then the classification of the core in the images was examined. Finally, it was concluded that changing the scale in the image will transfer the coefficients to other scales. Therefore, in order to create scale independence, we need features that have been extracted from the combination of subscales of different scales, and the use of deep reinforcement learning in core classification in images is very useful.

Keywords: Reinforcement learning, core classification in images, discrete transformations

Introduction

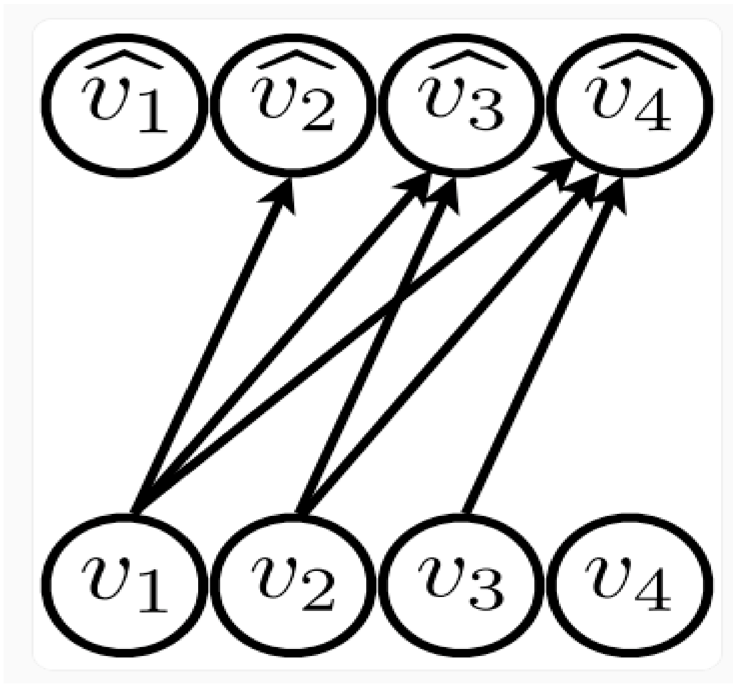
Recent advances in the relative modeling of neurotransmitters have led to significant results in image and audio modeling, as well as language modeling and machine translation.This post appears to be in a relatively older model than the Neurosurgery Model-Family Model of Predicting the Distribution of Neural Self-Regression.One of the limitations of neural self-regression is that uniform expansion to a deep version, with several layers of hidden units, is significantly more expensive.Deep neural networks are at the core of advanced models for supervised tasks such as image recognition and speech recognition (Dahl 2013).

A self-regression model is based on the fact that any subsequent D distribution can be factorized into a conditional distribution product in any order:

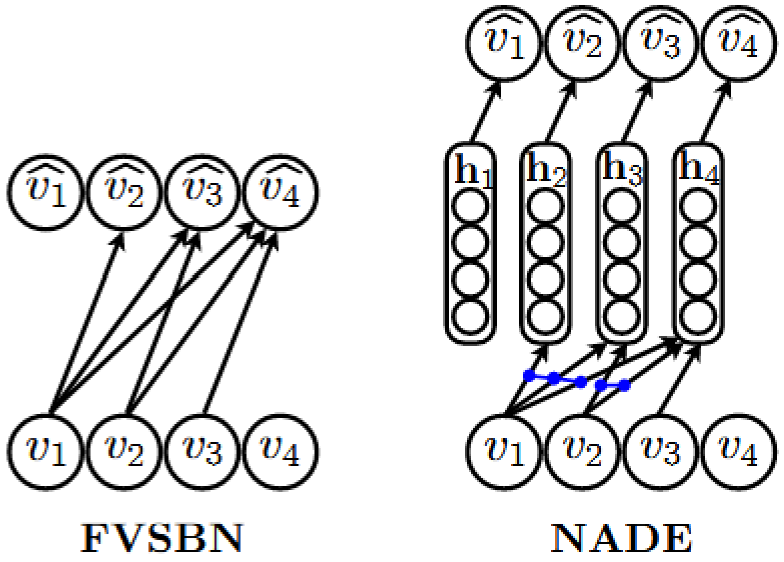
$$p(\mathrm{x}) = \prod_{d=1}^{D} p(x_d|\mathrm{x}_{<d})$$

Where x <d represents the first dimension of d-1 x in the current order. Therefore, we can create a regression self-generating model with only the parameter of all the individual conditions in this equation (Bangu 2009).

One of the easiest ways to do this is to have a sequence of binary values and assume that the output at each step is only a linear combination of the previous values. (Hugo et al. 2011).We can then transfer this weighted set through a sigmoid to get the probability of output for each step. This type of model is called a complete Sigmoid Belief (FVSBN) network:



Here we have binary inputs v and generate binary outputs v.. v3 ^ is generated from inputs v1 and v2.The chart below compares the complete belief system of sigmoid and nerve regression itself:



Comparison of two estimators

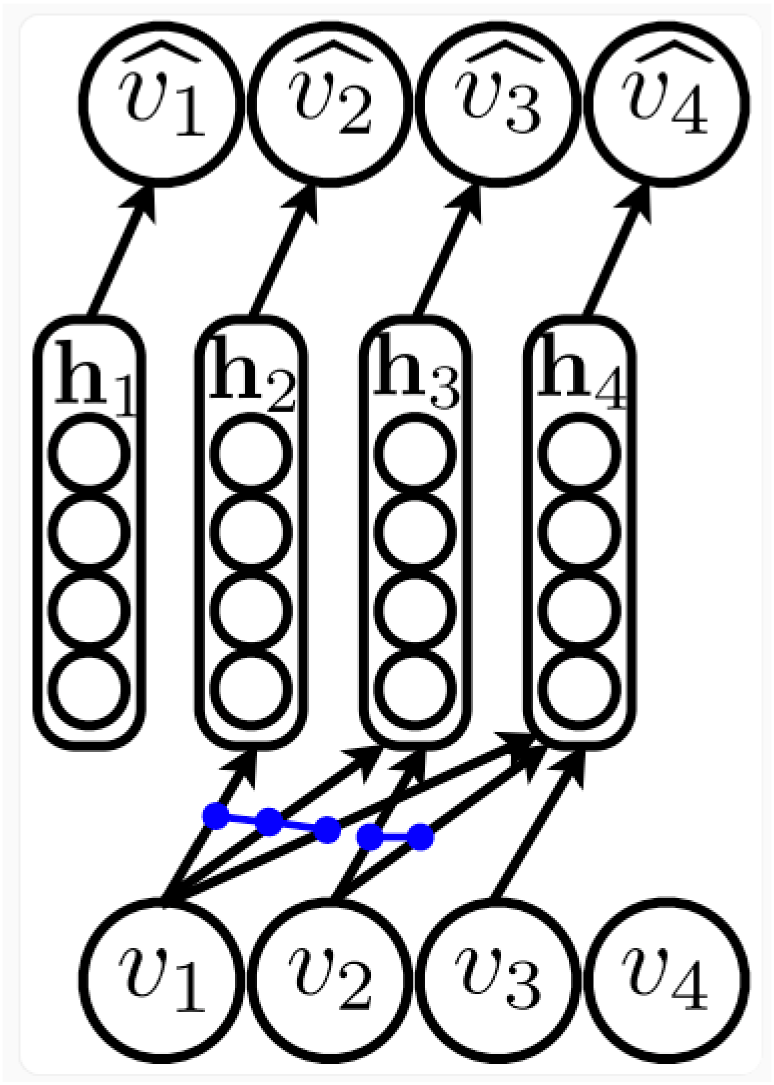
How to estimateFirst, we calculate the value of the hidden unit h_0, and this is just the sigmoid of the fanatic value of that c (the value of the bias is hidden for the whole unit, anyway (therefore, h_0 = sigm) c)

Then we try the probability of a visible unit of v_0 to clarify the condition in our parents (there is no first parent for each unit), and this is another sigmoid: p (v_0 = 1 | without parents) = sigm (b_0 + V_0 * h_0).

The next line of algorithm 1 completely disrupts me, because it is assumed that the multiplication (v) (which starts at zero) is expressed by ..., but isn't that just zero? And what does this phrase mean in parentheses? (It appears to be equal to the probability of v_0 to light up, if the observer v_0 = 1, or complements this probability otherwise,).

Then we add the value of a, which is then used for the second hidden node h_1 (meaning that the equations for the equation, I guess). When we do a = a + W_0 * v_0.

Nervous regression itself can be seen as a format of this, in which instead of the linear parameter of each condition, we direct the inputs through a neural network:



Estimation of random neural automatic distribution.

Specifically, each parameter condition is as follows:

$$p(x_d|\mathrm{x}_{<\mathrm{d}}) = \mathrm{sigm}(b_d + \mathrm{V}_{d,:}\mathrm{h}_d)$$

$$\mathrm{h}_d = \mathrm{sigm}(c + \mathrm{W}_{:,<d}\mathrm{x}_{<d})$$

Where W, V, B and C are the learnable parameters of the model. It can then be taught by reducing the likelihood of negative data entry.

When compared to FVSBN, there is additional weight distribution in the input layer of neural regression: each input element uses the same parameters when calculating different output elements.This parameter is inspired by the Boltzmann Restricted machine, but also has computational advantages - at each stage we only need to calculate the share of the new sequence element (we do not need to reconsider all the previous elements) (Benigno et al. 2016).

Modeling documents with neural regression

In the standard model of neural regression, input and output are double variables. In order to work with text sequence, the neural self-regression model expands neural regression by considering each element in the input sequence as multivariate observations, or in other words, one of the predefined sets of symbols (from a fixed vocabulary). . Similarly, the output must now be multi-volume, and therefore a softmax layer is used in the output instead of a sigmoid. Then the conditions of neural self-regression by:

$$p(x|\mathrm{x}_{<\mathrm{d}}) = \frac{\exp(b_{w_d} + \mathrm{V}_{w_{d},:}\mathrm{h}_d)}{\sum_w \exp(b_w + \mathrm{V}_{w,:}\mathrm{h}_d)}$$

$$\mathrm{h}_d = \mathrm{sigm}\left(c + \sum_{k<d} \mathrm{W}_{:,x_k}\right)$$

An additional type of parameter sharing is introduced in the input layer - each element has the same weight regardless of where it appears in the sequence (so if the word "cat" appears, positions 2 and 10 appear, The same weight is used each time)

There is another way to do this. There is currently a set of parameters for each word, regardless of where it is in the sequence, and the common name for this architectural pattern is word embedding.So we can use neural self-regression as a way to make magic words, but with a set of different constraints we may use models like Word2Vec.

For each input in the sequence, neural self-regression uses the sum of the pairs of previous values (passing through the non-linear sigmoid) to predict the word in the next step. The final display of a document is just the amount of hidden layer in the final step (or in other words, the total displacement of words via nonlinear).

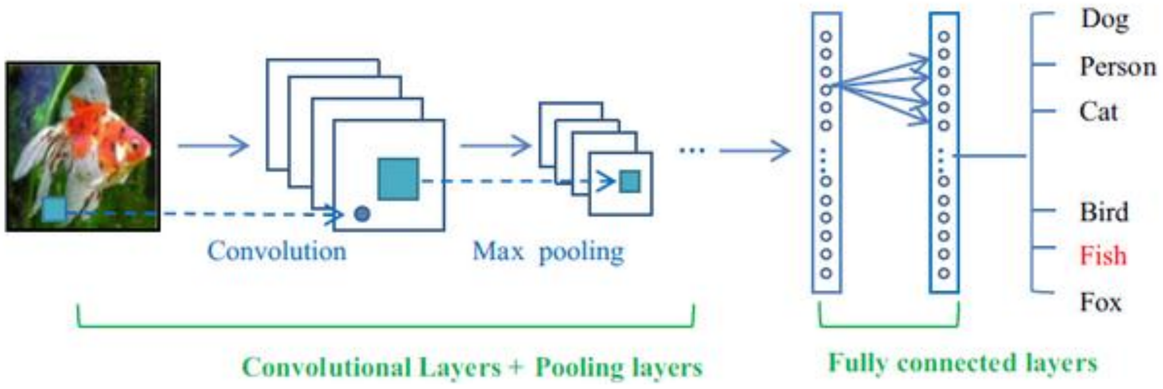
There is one limitation that has not yet been discussed - the order. Instead of teaching in a sequence of words, they appear in the document in order, as we practice when teaching a language model, for example, neural regression in the random definition of words in a document. Thus, magnets that are useful for predicting what is expected to appear together in a complete document, instead of focusing on patterns that focus on syntax and word instructions (or focus on smaller texts around each word). , useful.

Convolutional neural network

Convolutional neural networks (CNN) are one of the most important deep learning methods in which several layers are taught in a powerful way.This method is very efficient and is one of the most common methods in various applications of computer vision. The general picture of a convolutional neural network architecture is shown in Figure 2.In general, a CNN network consists of three main layers: the convolution layer, the pooling layer, and the fully connected layer. Different layers perform different tasks. In the following form.An overview of the convolutional neural network is shown for layer-by-layer image categorization.There are two steps to training in each neural network. The feed forward stage and the backpropagation stage. In the first stage, the input image to the network is fed, and this operation is nothing but a point multiplication between the input and the parameters of each neuron and finally the application of the convolution operation in each layer. Then the network output is calculated. Here, in order to adjust the network parameters or in other words the same network training, the output result is used to calculate the network error rate. To do this, compare the network output using an error function with the correct answer, and thus calculate the error rate. The next step is based on the calculated error rate of the backpropagation phase. In this step, the gradient of each parameter is calculated according to the chain rule and all the parameters change according to the effect they have on the error created in the network. After the parameters are updated, the next step of feed-forward begins. After repeating a suitable number of these steps, the network training ends.

A variety of CNN network layers

In general, a convolutional neural network is a hierarchical neural network whose canoeing layers are one-to-one with the pooling layers, followed by a number of fully connected layers.



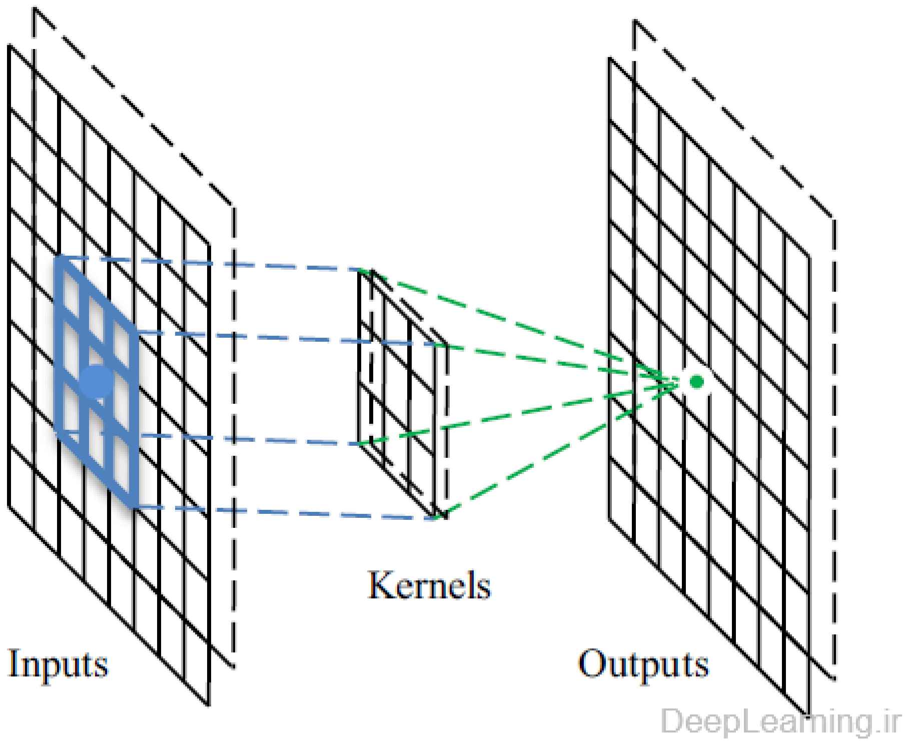
An overview of the architecture of a convolutional neural network

"Turning neural networks" are a class of deep neural networks that are commonly used to perform visual or speech analysis in machine learning. To explain CNN, work continues with the most basic element of this neural network, the perceptron.

Convolution layer

The weight sharing mechanism in each feature map drastically reduces the number of parameters. The local connection learns the connection between neighboring pixels. It causes immutability and stability to the displacement of the object. Because of the benefits of convolution, some well-known research papers have used it to replace fully connected layers, thereby speeding up the learning process.

One of the most interesting ways to manage canoeing layers is the NIN method, in which the main idea is to replace the canoeing layer with a small perceptron neural network that includes several completely connected layers with nonlinear activation functions. In this way, linear filters are replaced by nonlinear neural networks. This method results in good image categorization (Krishusky et al. 2012).

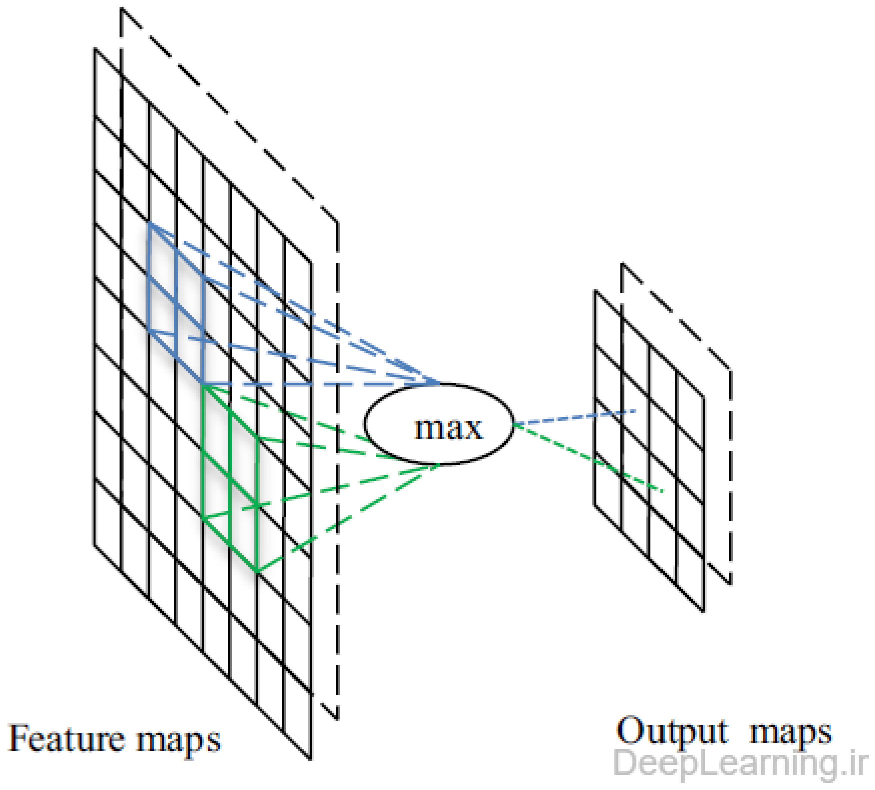


Conv olution layer operation

Perceptron

Persephone is an essential component of the neural network that can also be learned. The development of this component was therefore inspired by a biological element called neuron. Because a neutron, like a neuron, receives an input signal, processes it, and simulates an answer. The signal perceptron can be used to distinguish linear separable problems.

Persephone is an essential component of the neural network that can also be learned. The development of this component was therefore inspired by a biological element called neuron. Because a neutron, like a neuron, receives an input signal, processes it, and simulates an answer. The signal perceptron can be used to distinguish linear separable problems.



max pooling

Feature vector

Machine learning tasks are usually defined by how a machine learning system should process a sample.These tasks are usually a set of features that are quantified from some objects or events and we ask the machine learning system to process them for us.We usually display this example as the x ∈Rn vector, where each component of xi itself is a feature vector. For example, the properties of an image are usually the values of the pixels in that image.

Convolutional neural network architecture

With the recent advances in the use of CNN in computer vision, well-known models of convolutional neural networks have emerged.. In this section, we study the most common models and then summarize the characteristics of each in their applications (Lin and Chen 2013).The configuration and achievements of some common CNN models are given in the table.

AlexNet is a notable architecture for the Convolutional Neural Network, which includes five layers of canola and three fully connected layers. The architecture receives images in the size of 224x224x3 as input, and then processes the input image by performing sequence and pooling operations, and finally sending the results to the fully connected layers. The network is trained on the ImageNet data set and includes various regularization techniques such as data augmentation, Dropout and…. Used.

But there are two major problems with this architecture:

1- This architecture needs images with constant size.

2- There is no understanding of why this architecture works so well.

In 2013, Ziller et al. Introduced a new visual representation method that could be used to observe activities within the layers of a convolutional neural network.

Method	**Year**	**Rank**	**configuration**	**result**
AlexNet	2012	**First**	5layers of convolution + 3 layers completely connected	**An important architecture that attracted the interest of many researchers in the field of computer vision**
Clarifai	2013	**First**	5layers of convolution + 3 layers completely connected	**It made what was happening inside the network visible**
SPP	2014	**Third**	5layers of convolution + 3 layers completely connected	**By offering spatial pyramid pooling, the image size limit was removed**
VGG	2014	**Second**	15-13 Convolution layer + 3 layers all connected	**Complete evaluation of the network with increasing depth**
GoogLeNet	2014	**First**	21layers of convolution + 1 layer completely connected	**Increase network depth and width without increasing computational requirements**
ResNet	2015	**First**	152 Connection Layers + 1 Layer all connected	**Increase network depth and provide a way to prevent gradient saturation**

1. Conclusions

Just as neural self-regression calculates the probability of an input sequence, we can generalize its capability by calculating the probability of a long-term test set.In the article, the actual matrix they use is the average disorder per word for time t, input x, and size of the N test set by:

$$\exp\left(-\frac{1}{N}\sum_t \frac{1}{|x_t|}\log p(x_t)\right)$$

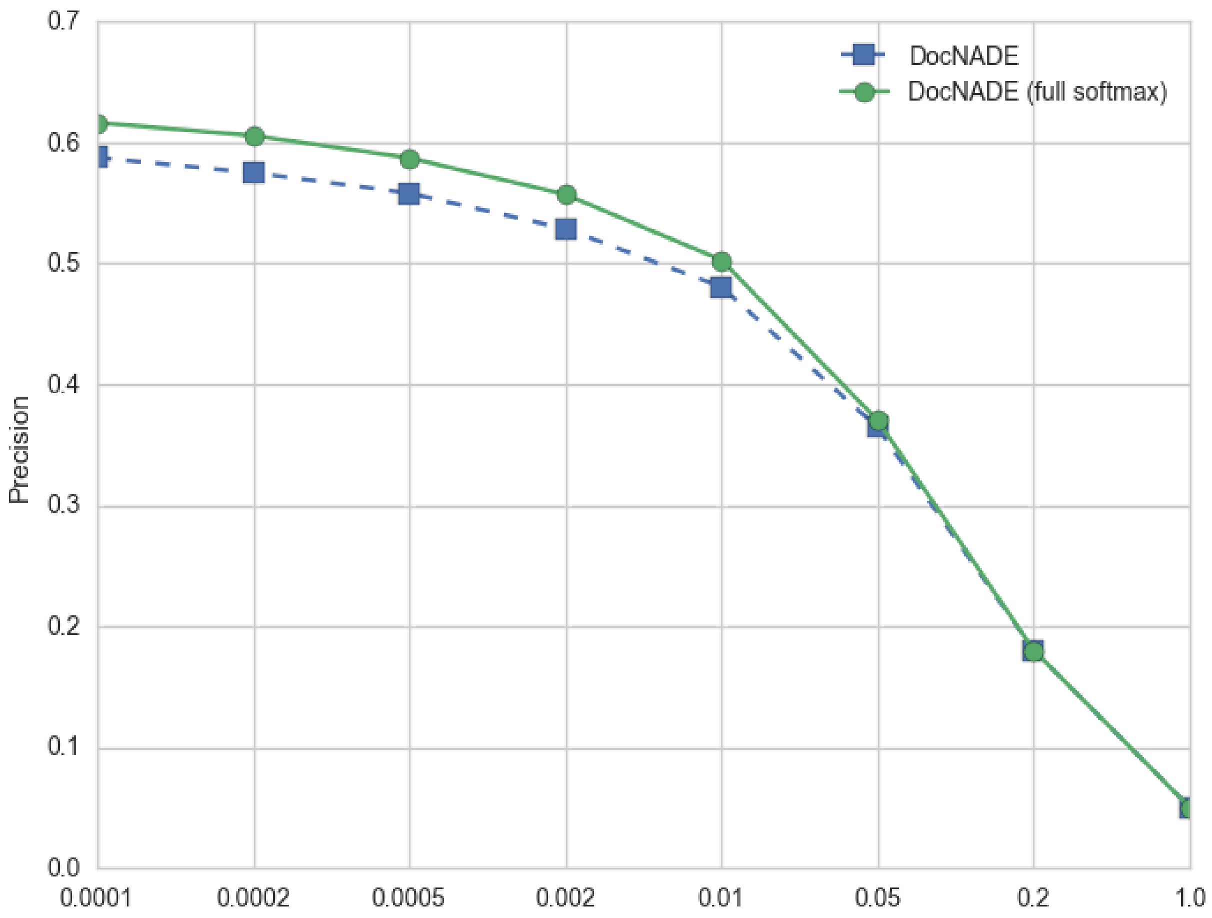
As in the article, we evaluate neural self-regression in a set (small group) of 20 newsgroups used in our previous post, which includes a set of 19,000 posts in 20 different newsgroups.

The published version of Nervous Self-Regression uses hierarchical software in this data set, despite the fact that they use a small vocabulary of 2000.

There's almost no need for a soft size in this size, when learning relatively small models in modern GPUs, just use full softmax. This makes a big difference in the reported instability numbers - the published performance shows an ambiguity in the 896 test, but with the full software we can get this to 579. Given that this is a major improvement, the table below shows this for recently released models:

Model/paper	**Perplexity**
DocNADE (original)	896
Neural Variational Inference	836
DeepDocNADE	835
DocNADE with full softmax	579

The following diagram can be provided as software output:



To evaluate recovery, first create vectors for each document in the data set. Then we use the vectors of the tested test set as "questions" and find the closest N documents in the training set (similar to the casein) for each query.

References

BenignoUria, Marc-AlexandreCôté, Karol Gregor, Iain Murray, Hugo Larochelle, Neural Autoregressive Distribution Estimation , Journal of Machine Learning Research 17 (2016) 1-37

George E. Dahl, Tara N. Sainath, and Geo_rey E. Hinton. Improving deep neural networksfor LVCSR using recti_ed linear units and dropout. IEEE International Conference onAcoustics, Speech and Signal Processing, pages 8609{8613, 2013

Hugo Larochelle, Iain Murray (2011), The Neural Autoregressive Distribution Estimator, The Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS), JMLR W&CP 15:29–37, 2011.

Krizhevsky A, Sutskever I, Hinton GE, editors. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems; 2012

Lin M, Chen Q, Yan S. Network In Network. CoRR. 2013;abs/1312.4400.

YoshuaBengio. Learning deep architectures for AI. Foundations and Trends in Machine Learning, 2(1):1{127, 2009.

Zeiler MD. Hierarchical convolutional deep learning in computer vision: NEW YORK UNIVERSITY; 2013.