Continual Learning using auto-encoders (revised Feb 2018)

Paul Spedding

1 Abstract

Autoencoders have played a fundamental part of unsupervised machine learning. One of the many roles of an autoencoder is to classify patterns within the data. Autoencoders tend to perform well when trained on a data set of car photos, but then if we train it again on pictures of trucks and the try to classify an image of a car it forgets what it has 'learned' on its first task and it's accuracy drops. So to put autoencoders to the test, I will be discovering what the results will be if the patterns within the data keep evolving. And how quickly will the autoencoder adapt to the new changes? How will it affect the accuracy of the classification? Here I plan to use a convolution neural network to manipulate two commonly used handwritten digit data sets CFIAR100 and MNIST. To accomplish this, I will manipulate the pixels of the two data sets and compare the accuracy on this task of continual learning.

2 Introduction

Artificial intelligence is an odd phrase for describing machine learning algorithms since no one really knows what intelligence is. It's a concept which we can infer intuitively yet asked when to define it formally people can have several different ideas on what it is and it becomes a much harder and noticeable problem[5]. A Neural Network is an algorithm for both classification and regression problems is arguably one of the most successful algorithms around for solving a verity of tasks, and it's interest in both academic and commercial enterprises has risen dramatically over the past 20 years making it an interesting field of study.

A big problem for neural networks is forgetting data which they where once trained on, this type of amnesia has been well documented and have thought to become an inevitable feature of neural networks [1]. Neural networks can learn both linear and non-linear systems which is what makes them suitable for a wide variety of application. Unfortunately one draw back to using neural networks is once they are trained to do a task the next following task they learn forgets the original task which has been termed "catastrophic forgetting"[6].

For these experiments I will be focusing on the Auto encoder which is a type of neural network which input (x) is also it target (y) doing this can make the machine learn embeddings with in the input data rather than just predicting an output Y given some input X. "Autoencoders are simple learning circuits which aim to transform inputs into outputs with the least possible amount of distortion. While conceptually simple, they play an important role in machine learning. Autoencoders were first introduced in the 1980s by Hinton and the PDP group"[4]. The purpose of this paper is to test the autoencoders ability to adapt to learning concepts incrementally and to perform a number of different tests. Being able to teach neural networks incrementally would have more real world application because "in real-world settings: The sequence of tasks may not be explicitly labelled, tasks may switch unpredictably, and any individual task may not recur for long time intervals. Critically, therefore, intelligent agents must demonstrate a capacity for continual learning"[2].

3 Background

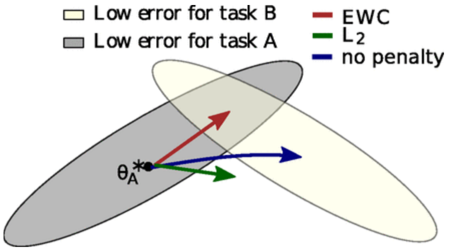
Over coming the problem of catastrophic forgetting could be an important milestone in make neural networks more intelligent and will help us to progress to a more generalised AI solutions. A number of different attempts have been attempted to solve the problem of catastrophic forgetting in neural networks first noticed in the early 1990's.

The reasons for catastrophic forgetting is a well understood one. The way a neural network is trained is you provide input data and you want to get the neural network to predicts the output given the input data. By using the data to train the network multiple times the network attempts to learn the a correlation between the input and the target it does this by updating the weights of the network to fit a representation of the model it has learnt from the data. The problem comes when given a new input the weights then update to fit the new data set which means forgetting the old internal model and replacing it with the new one, the problem is that of optimisation we need an optimal solution in which important weights are kept but not so important ones are pushed to the back or forgotten when learning new tasks.

To create real AI the learning process should be technically better or similar to the human brains way of recalling information that was once stored some form of similarity between tasks make things easier like 'transfer learning'[11] which is here you train the network on similar tasks so it is easier for the Stochastic Gradient Decent algorithm to find a function to fit the output given the input. Other attempts on tasks which share little similarity between tasks have been created in order to counter-act this 'catastrophic forgetting' one is to interlace 2 tasks in one dataset this type of parallel learning sometimes referred to as

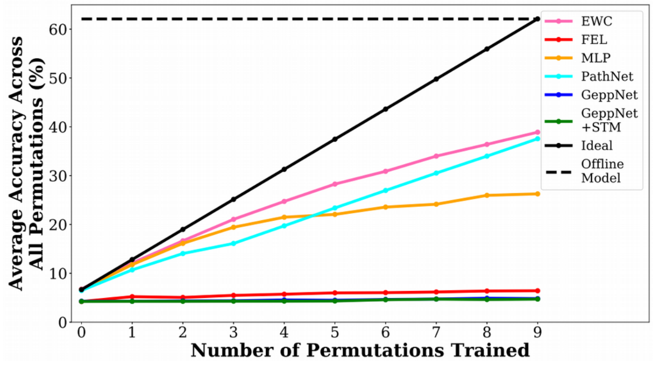
'Multitask Learning Paradigm'. This method is of training is not very usable as training two tasks concurrently can make the network not fully discover important weights for both tasks. Unfortunately this approach is impractical for learning a large number of tasks[2] has shown to be effective in certain scenarios but is impracticable and takes time to prepare the training data. In a real world setting data inst interlaced normally and this creates a problem with this style of training.

A newer method has also been developed called Elastic Weight Consolidation (EWC) which as a technique developed by DeepMind in which the neurons have a dense connection to the next layer its only some of the weights are used for task 'A' and some for task 'B' where as some are shared that way it doesn't need to forget important weight needed for either task, as for the shared weights they are updated in a way that minimises the error on both tasks as seen below.



AS YOU CAN SEE ABOVE EWC FIND WEIGHTS IN WHICH THE ERROR IS MINIMIZED FOR BOTH TASKS[2]

Even though elastic weight consolidation is a great boost compared to existing methods in the battle against catastrophic forgetting. But unfortunately does not fully solve the problem as it can still get out performed on certain datasets using different models like PathNet[12]. As for our task which deals with permutations of data EWC still out performs a lot of current solutions.



AS CAN BE SEEN THE MORE PERMUTATIONS OF THE DATA THE LESS IT IS TO THE IDEAL SOLUTION[12].

4 METHODOLOGY

The analysis of results will be focused on the scale in which catastrophic forgetting takes place on the MNIST and CIFAR data sets. Having a good understanding of the amount of forgetting that takes place inside a neural network and analyzing the results may gives us a good rounded benchmark that can be compared against future attempts to combat forgetting. By measuring the loss and accuracy of these future algorithms can tell us a metric by how much this new method combats catastrophic forgetting.

For future improvements to to combat against catastrophic forgetting. For these experiments I will be using the MNIST and CIFAR data sets. The MNIST consists of hand written digits from 0-9 and CIFAR consists of 10 classes of pictures. The MNIST data set has been made from mixing 2 databases together. "This database is created by "re-mixing" the original samples of NIST's database. The database has two parts: training samples that was taken from American Census Bureau employees and the test samples that was taken from American high school students. "[7]. The CIFAR data set was created as a joint effort between NYU and MIT, they used Google, Flickr and Altavista and only collected the first 3000 results for each term they then removed duplicates and pictures that had mostly white pixels. The search terms that they typed in was treated as a label for each image and each image was down scaled to 32 x 32 pixels[8].

I will be using Python 3.5 as it has a range of tool available for machine learning. To manipulate late the data I will be using Numpy and for the Classifiers and for the auto encoders I will be using Keras with a Theano back end.

Due to memory constraints I will have to save the data sets after each swap to do this I have decided to serialize the data using Pickle so result can be replicated. To also ensure that my results aren't just by chance I have also used the seed feature of Numpy (np.seed) and set it to 3557.

Both sets can be found in the keras.dataset import under 'MNIST' and 'cifar10'. I will keep each of the original sets untouched and then derive 6 new sets, 3 for each data set. Each newly created data set will be a permutation of the pixels from the original. The 3 sets will have the following traits:

1. 20% probability of swap
2. 40% probability of swap
3. 60% probability of swap

The next step is creating the auto encoder there are several different types of auto encoder which have different purposes such a denoisers and compressors these do not seem applicable for this experiment instead I will use a Variation Auto Encoder (VAE), This type of auto encoder trains on a set of images which in our case may be the number '3' from MNIST and will find the variation between all the images to create a generalized version of the number '3'. Once created I will get a class of images for instance the '6' from the MNIST data set and train an auto encoder on batches of the number ‘6’ until it reaches the lowest error rate, If the error rate is too high I will reevaluate my hyper parameters, this will be disused in section 5. When at the lowest error rate I will start to feed in the generated 20% probability of swap data set, from here I will measure the accuracy and error until it converges, Then I will train on the 40% with the same step as above until I get to the 60% probability of pixel swap. Knowing that the auto encoders first training set was the non-modified one if we present an image from that set to measure the accuracy and loss and compare to the accuracy and loss from when it was first trained.

The classifier will be trained and tested using stratified cross fold validation

5 Experiments

Each of the experiments will have a different activation functions they will be Sigmoid, Maxout and ReLU.

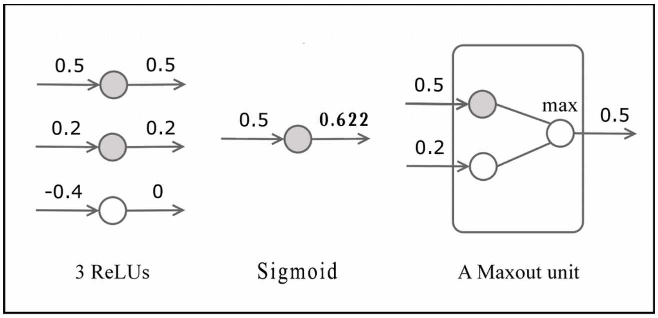


Fig 1 shows the ouput of each neuron, slightly modified from figure 1[10]

The Sigmoid transfer function is an interesting function because it what introduces non-linear behavior into the neural network helping it learn complex systems, I would find it interesting to see the effects of catastrophic forgetting on this. I chose a Rectified Linear Unit because even though its behavior is mostly linear it is still considered a non-linear transfer function due to the cut off point being at 0.

And each of the 3 activation functions will have a Dropout layer not included or included, this is to help prevent over fitting and to encourage more generalization[1] to the weights as Geoffrey Hinton described in his paper[9]. I will use stochastic gradient descent for all experiments and I will use a Dense neuron for each experiment meaning each neuron is connected to all neurons in the forward layer. In total there are 6 experiments to carry out 1 for each transfer function and them 3 experiments will both have Dropout on and off.

Reading over the paper “An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks“[1] They have conducted a similar experiment with the MNIST data set, they per mutated the data set to create a new one and trained on an old task and then a new task. In the results you can see that the combination of Dropout and Maxout drop quickly and rise slow when given the new task.

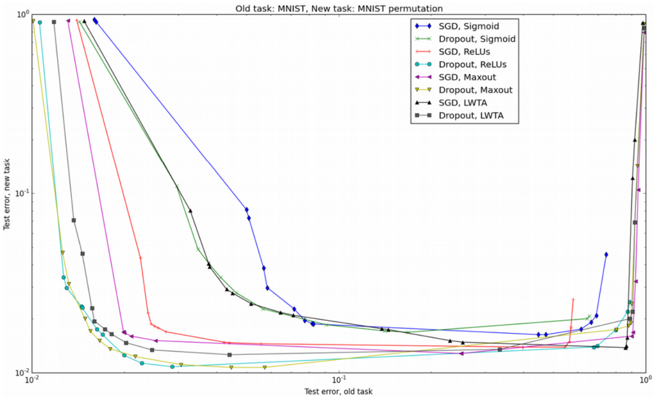


Figure 1. A comparison of attempted methods and the advantages of using the Dropout and Maxout layer on error from task ‘A’ to task ‘B’[1]

As can be seen makes an interesting combo to study especially since this was only based on classification and not on auto encoder

I will be using two different types of neural network architectures to experiment, the first being a Multi layer Perceptron and a Convolutional Neural Network. While I feel that the MLP will out perform the CNN as I’ve not seen much literature on CNN VAE it wont do us any harm to try and see what results we get.

The hyper parameter will be chosen though experimentation to get the best results, choosing hyper parameters is a hard thing to do with out testing I find. But the loss function I will be using is categorical cross entropy as it is for classifying multiple category's or classes. The number of Epochs I will set to 100 but this is subject to change also when I come to tune the hyper parameters. Validation will be done with stratified K fold cross validation as it is used for classification given an equal distribution between K folds making the validation test more accurate when averaged.

The last part of the architecture would be ho many deep layers to include and the amount of neurons, this again is a parameter that I ill test out I will start of small and increase until the accuracy fall or the loss increases.

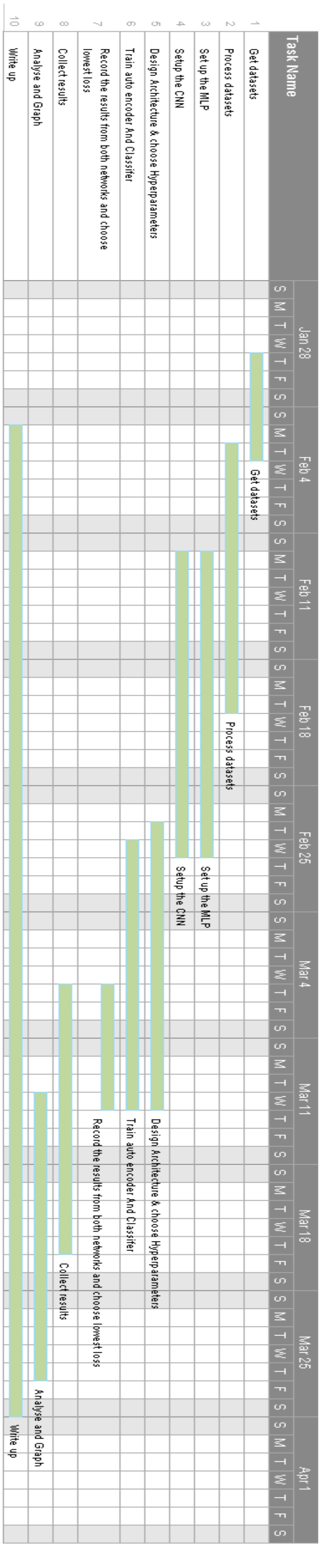
6 Discussion

We are going to be training multiple variation auto encoders on a single MNIST digit and a CIFAR picture to begin with. The one with the best accuracy will then be kept. To measure the amount of loss will be done by comparing the original trained class to the most permuted class and the result will be measured in a few ways. The accuracy of the original image to the 3rd permutated set of images also the loss while training and also the results of the classifier. I will compare and contrast the results from each experiment and the results will be plotted on a graph for easy reading this way we can see visually which setup where more accurate, had less loss than others.

7 CONCLUSION

After the short period of reading a few papers in regards to catastrophic forgetting with my limited knowledge I did not see any papers suggest a big neural network which has a minimization function to only use the bare necessity of neurons that are required to get a good loss and accuracy. To also add to that is an idea I had to include dynamically moving weights (dendrites) in which an optimization function has a hash table of all the available groupings of neurons and the curve that a cluster has given an input and use a Monte Carlo search to find an approximation to a desired input. One of the problems to this method is it has to be a huge network and multidimensional, sometimes a group of neurons find a better fit if they skip a layer and connect to a different grouping closer to the output etc..

8 PLAN



References

[1] Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, Yoshua Bengio, “An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks”, (https://www.arxiv.org/pdf/1312.6211.pdf 2015)

[2] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran and Raia Hadsell, “Overcoming catastrophic forgetting in neural networks, Pnas, (http://www.pnas.org/content/114/13/3521.full 2017)

[3] DAVID E. RUMELHART, GEOFFREY E. HINTON, and RONALD J. WILLIAMS ," *Learning Internal Representations by Error Propagation*, *PDP Group*. (http://www.dtic.mil/dtic/tr/fulltext/u2/a164453.pdf 1986)

[4] Pierre Baldi, “Autoencoders, Unsupervised Learning, and Deep Architectures”, Proceedings of ICML Workshop, (http://proceedings.mlr.press/v27/baldi12a/baldi12a.pdf 2012)

[5] Legg, Hutter, “Universal Intelligence: A Definition of Machine Intelligence “, IDSIA, (https://arxiv.org/pdf/0712.3329.pdf 2007)

[6] French, R. M , “Catastrophic Forgetting in Connectionist Networks: Causes, Consequences and Solutions”, Trends in Cognitive Sciences, (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.480.7627&rep=rep1&type=pdf 1999)

[7] Mazdak Fatahi , “MNIST handwritten digits ”, ResearchGate, (https://www.researchgate.net/profile/Mazdak_Fatahi/publication/273124795_MNIST_handwritten_digits_Description_and_using/links/54f7e82f0cf28d6dec9f6d2d/MNIST-handwritten-digits-Description-and-using.pdf 2014)

[8] Alex Krizhevsky , “Learning Multiple Layers of Features from Tiny Images ”, toronto.edu, (http://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf 2009)

[9] G. E. Hinton* , N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov , “Improving neural networks by preventing co-adaptation of feature detectors “, (https://arxiv.org/pdf/1207.0580.pdf 2012)

[10] Rupesh Kumar Srivastava, Jonathan Masci, Faustino Gomez & Jurgen Schmidhuber, “UNDERSTANDING LOCALLY COMPETITIVE NETWORKS”, (https://arxiv.org/pdf/1410.1165.pdf 2015)

[11] BERNARD ANS and STÉPHANE ROUSSET, “Neural networks with a self-refreshing memory: knowledge transfer in sequential learning tasks without catastrophic forgetting”, Connection Science, Vol. 12, No. 1, 2000, 1–19, (https://pdfs.semanticscholar.org/0742/d8530beb8d9fa006c2c8d15237cb8ebf0f22.pdf 2000)

[12] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, Christopher Kanan, “Measuring Catastrophic Forgetting in Neural Networks ”, Chester F. Carlson Center for Imaging Science (https://arxiv.org/pdf/1708.02072.pdf 2017)