Feedback-Gated Rectified Linear Units

Marco Kemmerling [1]

Abstract

Feedback connections play a prominent role in the human brain but have not received much attention in artificial neural network research. Here, a biologically inspired feedback mechanism which gates rectified linear units is proposed. On the MNIST dataset, autoencoders with feedback show faster convergence, better performance, and more robustness to noise compared to their counterparts without feedback. Some benefits, although less pronounced and less consistent, can be observed when feedback is applied on the CIFAR-10 dataset.

1. Introduction

The brain has served as inspiration for practical models called artificial neural networks (ANNs) for decades. While these models are usually heavily simplified compared to the brain, they have seen massive success in areas such as computer vision, speech recognition, (...), in recent times.

Despite successes, it is clear that the average human brain is vastly more powerful than any model used in practice today, and as such it may be useful to investigate how and where exactly the brain and ANNs differ.

While there is clear evidence of prominent feedback connections in the brain, ANNs have overwhelmingly been designed based on solely the feed forward paradigm. Thus, it is of interested if and how the incorporation of feedback may help artificial models. Further, ANNs with feedback are naturally easier to investigate and manipulate than the real brain and could potentially offer insights into exactly what role this mechanism plays in the brain.

(maybe tell that ornithology story)

In the remainder of this paper, some neuroscientific background is explored, a specific model of a feedback mechanism is examined and in the following section simplified to be incorporated into artificial models. Then experiments results discussion...

2. Neuroscientific Background

2.1. Neocortex

The neocortex, part of the cerebral cortex, is a part of the brain that evolved in mammals comparatively recently. It comprises around 80% of the human brain (Markram et al., 2004) and is therefore often speculated to be responsible for the emergence of higher intelligence.

The most abundant type of neuron in the neocortex are pyramidal neurons, constituting between 70-85% of cells. In contrast to the remaining neurons in the neocortex, so called interneurons, which are mostly inhibitory, pyramidal neurons are excitatory (DeFelipe & Fariñas, 1992).

As the name suggests, pyramidal neurons have a cell body roughly shaped like a pyramid, with a base at the bottom and an apex at the top. Pyramidal neurons have two types of dendrites: basal dendrites, originating at the base, and one apical dendrite, originating at the apex. This apical dendrite terminates in what is called the apical tuft, where heavy branching of apical dendrite occurs. (DeFelipe & Fariñas, 1992).

These apical and basal dendrites are not just differently located, but also serve different functions. Basal dendrites receive regular feedforward input, while the apical tuft dendrites receive feedback input.

The neocortex appears to have a distinct structure which is characterised by its organisation into layers as well as columns. The columnar organisation is based on the observation that neurons stacked on top of each other tend to be connected and have similar response properties, while only few connections exist between columns. Columns are hence hypothesised to be a basic functional unit in the cortex, although this is somewhat debated in the neuroscience community (Goodhill & Carreira-Perpiñán, 2002).

The further organisation into six layers was proposed by Brodman in 1909 [citation]. Layers 1 and 6 are of particular interest here. Layer 1 consists of almost no cell bodies, but mostly connections between axons and the apical dendrites of pyramidal neurons (Shipp, 2007), i.e. is serves as a

[1] University of Maastricht, Maastricht, The Netherlands. Correspondence to: Marco Kemmerling <m.kemmerling@student.maastrichtuniversity.nl>.

connection hub for feedback signals. Layer 6 sends signals to neurons in the thalamus which then in turn sends signals to layer 1 neurons in the same column (Shipp, 2007), i.e. layers 1 and 6 create a loop where feedback is sent from layer 6 and received by layer 1.

2.2. Distal Input to Pyramidal Neurons

As described above, apical tuft dendrites receive feedback input which appears to modulate the gain of the corresponding neuron (Larkum, 2004). It is hypothesised that this is a way for the cortex to combine an internal representation of the world with external input, i.e. feedback to a neuron may predict whether this particular should be firing and even small feedforward input may lead the neuron to fire as long as the feedback signal is strong (Larkum, 2013).

Taking both feedforward and feedback input into account, the firing rate of a neuron can be modelled as follows (Larkum, 2004):

$$f = g(\mu_S + \alpha\mu_D + \sigma + f\beta(\mu_D) - \theta) \tag{1}$$

where f is the firing rate of the neuron, g the gain, μ_S the average somatic current (i.e. feedforward input), μ_D the average distal current (i.e. feedback input), α is an attenuation factor, σ represents fluctuations in the current, θ is the firing threshold, and $\beta(\mu_D)$ is an increasing function of the dendritic mean current which saturates for values above some current threshold.

3. Feedback-Gated Rectified Linear Units

The model described in the previous section serves as a basis to derive an activation function which can replace the common rectified linear unit (ReLU) (Nair & Hinton, 2010), i.e. $f(x) = max(0, x)$.

To arrive at a more practical activation function, g and θ are dropped from equation 1, since the threshold is modelled through the bias unit and the gain (i.e. slope) of a ReLU is by definition 1 and can thus be safely dropped. Dropping the summands $\alpha\mu_D$ and σ is less justifiable, but since they do not contribute to the core property of gain increase, they will be disregarded here, arriving at the following simplified relationship:

$$f = \mu_S + f\beta(\mu_D) \tag{2}$$

Removing f from the right hand side:

$$f = \frac{1}{1 - \beta(\mu_D)}\mu_S \tag{3}$$

What remains is an exact definition of $\beta(\mu_D)$, which, according to (Larkum, 2004), is "an increasing function of the dendritic mean current μ which saturates for values above 1000pA“. In other words, the function is bounded, i.e. the gain cannot be increased to arbitrarily high values. Accordingly, some maximum value β_{max} the function can produce and a threshold value η which describes when this maximum is reached need to be defined. Assuming a piecewise linear model, $\beta(\mu_D)$ is thus defined as follows:

$$\beta(\mu_D) = min\left(\frac{\beta_{max}}{\eta}\mu_D, \beta_{max}\right) \tag{4}$$

As there are no obvious values to assign to β_{max} and η, they are treated as hyperparameters. Since setting β_{max} to 1 results in a division by 0 and a value of $\beta_{max} > 1$ causes a negative slope, β_{max} should be smaller than 1.

Plugging equation 4 into equation 3 yields:

$$f = \frac{1}{1 - min(\frac{\beta_{max}}{\eta}\ \mu_D, \beta_{max})}\ \mu_S \tag{5}$$

Since negative values for μ_S are not taken account, it is replaced with $max(0, \mu_S)$, i.e. the classic ReLU function:

$$f = \frac{max(0, \mu_S)}{1 - min(\frac{\beta_{max}}{\eta}\ \mu_D, \beta_{max})} \tag{6}$$

3.1. Feedback-Gated ReLUs in Practice

The feedback path attempts to mimic the top-down path in the brain. As such, the origin of feedback terminating in a layer should be a layer that is higher in the (feedforward) hierarchy.

Since feedback from higher layers can only be computed if these higher layers have received feedforward input, at least two time steps are needed to incorporate the modified ReLUs into a network. Concretely, some data, e.g. an image is fed into the network twice, where the first pass enables the computation of feedback which can be utilised in the second pass. Although more than two timesteps are not necessary, it is possible to use an arbitrary number of timesteps, which is examined in section 4.1.

Any layer that receives feedback requires an additional set of weights to compute μ_D. Specifically each layer h_i with size n receiving feedback from layer h_j with size m introduces $n \times m$ additional parameters.

Dropout (Srivastava et al., 2014) should be used by dropping out the same units in all passes. If e.g. dropout is only applied on the last pass, the remaining units will still receive signals from dropped out units in previous passes.

In convolutional neural networks (LeCun, 1989), feedback is implemented on a filter-wise basis, i.e. each neuron does

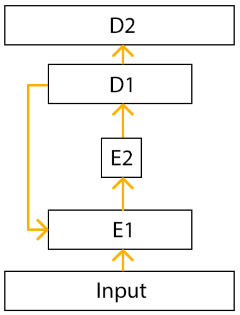

Figure 1. Autoencoder with (partial) feedback.

not receive its own unique feedback signal, but rather every filter receives a unique feedback signal that is shared between all units belonging to that filter.

4. MNIST

GIVE SOME INTRODUCTION TO EXPERIMENT SECTION HERE (SHOULD TIE BACK INTO THE RESEARCH QUESTIONS POSED IN THE INTRODUCTION)

The MNIST dataset is composed of 28×28 pixel binary images of handwritten digits, split into 60000 training and 10000 test instances (LeCun et al., 2010). Each image is associated with one of ten classes representing the digits between 0 and 9.

The models used in the following experiments are based on a (non-convolutional) autoencoder with two encoding and two decoding layers. The input layer has dimension (1×784), the first encoding layer (E1) outputs data of dimension (1×392), the second (E2) of dimension (1×196), the first decoding layer (D1) of dimension (1×392) and the second decoding layer (D2) restores the data back to its original dimension. Except for the final layer, each layer is followed by a ReLU activation. The final layer makes use of a sigmoid activation function.

First experiments were performed with only a single feedback connection between the first decoder and the first encoder (see figure 1).

Optimal values for η and β_{max} were determined by a grid search ($\beta_{max} = 0.95, \eta = 5$).

To increase the difficulty of the task, the dimension of the second encoding layer is reduced to 10 and the experiment is repeated (this modification will persist in all subsequent experiments).

(results)

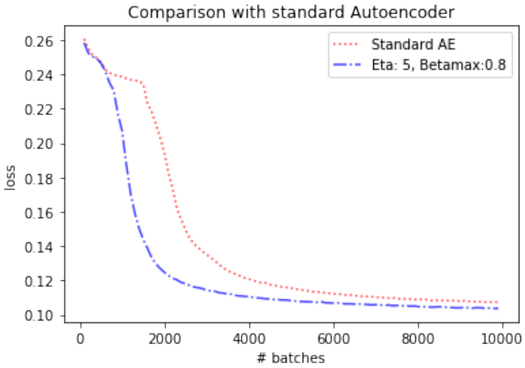

Figure 2. Placeholder, make sure to update this figure

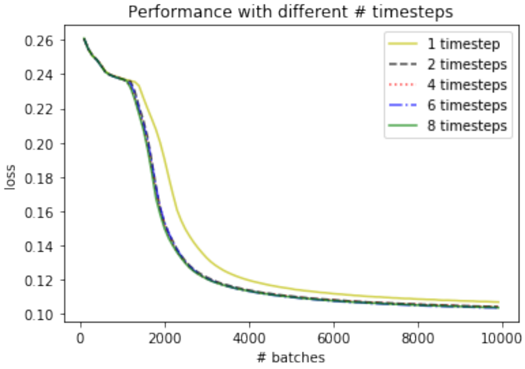

Figure 3. Performance with varying numbers of timesteps. Each configuration was trained and evaluated 10 times. The curves shown are the averaged losses on the test set.

4.1. More Than Two Timesteps

While at least two timesteps are required to incorporate feedback, it is not clear whether exactly two timesteps should be used or whether > 2 timesteps can be beneficial. To examine this, autoencoders with 1, 2, 4, 6, and 8 timesteps are trained.

The results, depicted in figure 3, show that more than two timesteps yield no or negligible improvement. This may of course be data and/or task dependent. Since MNIST is a fairly simple dataset (binary images, clear separation of background and foreground, etc.), it is not inconceivable that tasks on other datasets may benefit from more than two timesteps.

4.2. Comprehensive Feedback

In previous experiments, feedback was only sent from one decoding layer to one encoding layer. Naturally, there are many more opportunities to incorporate feedback. Specifically, in the following, each layer receives feedback from every layer above it.

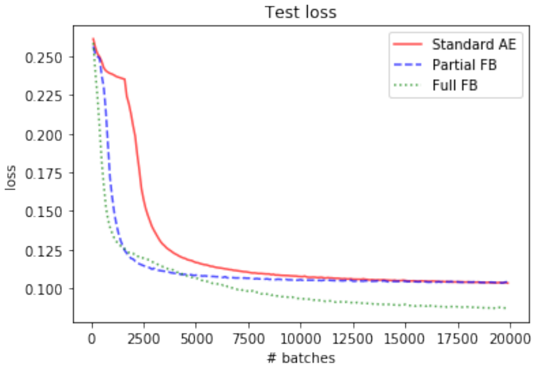

Figure 4. Placeholder, make sure to update this figure

As shown in figure 4, the configuration explained above does not only converge faster than a standard autoencoder, but also settles to a better performance than the model with only partial feedback.

4.3. Feedback vs Constant Gain

In an effort to get some understanding how exactly incorporation helps to improve performance, the feedback values computed by a network for all instances of the test set are visualised in a histogram. A distinction is made between feedback and gain, where feedback refers to μ_D and gain refers to $\frac{1}{1-min(\frac{\beta_{max}}{\eta}(\mu_D),\beta_{max})}$.

Figure 5 shows the data as collected in the network with a single feedback connection.

While there are some smaller gain values, the overwhelming majority of values are the maximum gain the network can produce. This raises the question whether there is much benefit to learning feedback or whether it might be similarly beneficial to simply multiply all activation values by a constant.

This is easily tested by setting the gain of every ReLU in the affected layer to a constant value of 10.

As can be seen in figure 6, this does lead to a steeper loss curve than the standard autoencoder, although not quite as steep as that of the autoencoder with actual learned feedback. Further, the performance after training is completed is worse than that of the standard autoencoder.

Repeating this same experiment for more than one feedback connection yields to results as illustrated in figure 7.

In this setup, the simple multiplication by a constant initially converges even faster than the autoencoder with learned feedback. While it does not achieve the same performance as the feedback autoencoder in later stages of training, it is on par with the standard autoencoder's per-

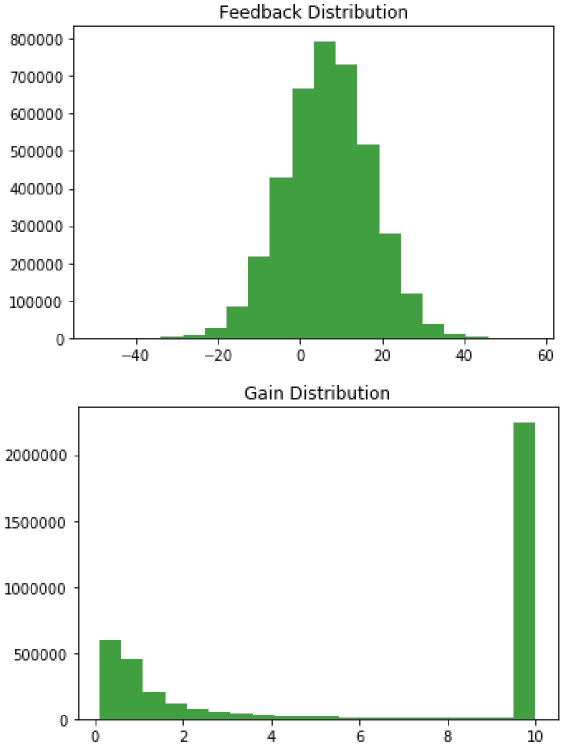

Figure 5. Distribution of feedback (top) and gain (bottom) values collected in a network with partial feedback over the complete MNIST test set.

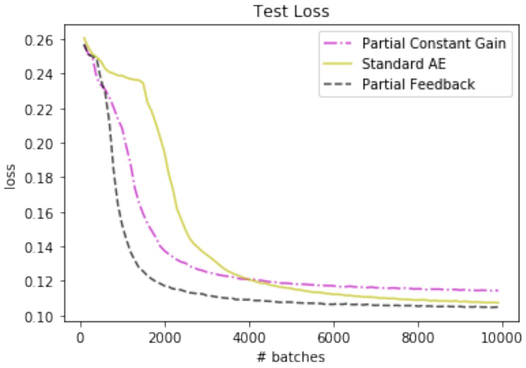

Figure 6. Placeholder, make sure to update this figure

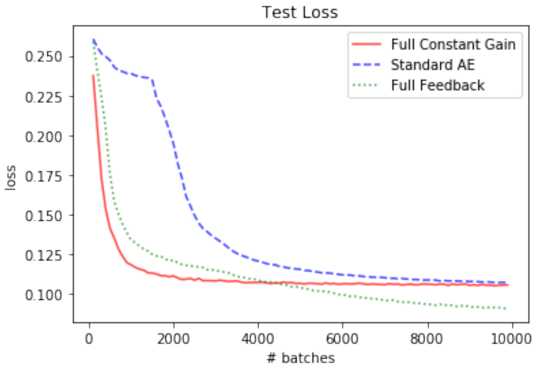

Figure 7. Placeholder, make sure to update this figure

formance.

Clearly, the effects of feedback cannot be fully explained by this constant gain, but the idea of a constant gain seems to have some merit.

4.4. Noisy Activations

While noisy signals are usually not an issue in artificial networks, noise in the brain is very prevalent (Faisal et al., 2008). To see whether feedback makes the model more robust to noise, gaussian noise with zero mean and various standard deviations is added to the (pre-)activations of both the network with feedback and the one without it. The networks are only evaluated with added noise, training is performed without noise. Note that in the network with feedback, noise is added to the activations in both passes.

$$h = f(W^T x + b + \mathcal{N}(0,\, \sigma^2)\,) \tag{7}$$

As figure 9 shows, the use of feedback significantly increases the network's robustness to noise. While this is not especially useful for machine learning models, it may be part of the reason why the feedback path exists in the brain.

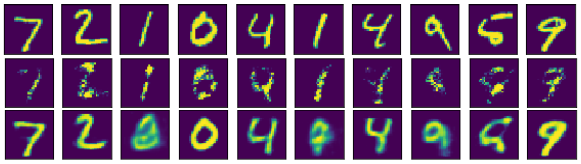

Figure 8. Gaussian noise with zero mean and standard deviation $\sigma = 2.0$ is added to networks with and without feedback. The top row shows input instances to the network, the middle and bottom row show reconstructions of the network without and with feedback (respectively).

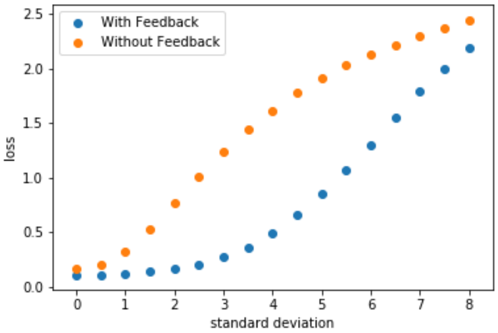

Figure 9. Gaussian noise with zero mean and varying standard deviations (horizontal) is added to networks with and without feedback. The quality of the reconstruction, as measured by the loss function (vertical axis), with respect to the magnitude of the standard deviation is shown for both networks.

5. CIFAR-10

The CIFAR-10 dataset is composed of 32×32 pixel colour images of various objects, split into 50000 training and 10000 test instances. Each image belongs to one of the following classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck (Krizhevsky et al., 2014).

5.1. Autoencoder

Similarly to the MNIST experiments, an autoencoder is trained on the CIFAR-10 dataset. Again, the architecture consists of two encoding and two decoding layers. Contrary to MNIST, the encoding/decoding layers used here are convolutional/transposed convolutional layers with 16 5×5 filters.

While the autoencoder with feedback clearly performs better than the one without it, the difference between the two is not as pronounced as it is in the MNIST experiments.

Curiously, if batch normalisation (Ioffe & Szegedy, 2015) is used after the activation functions, feedback cannot improve on the performance of the standard autoencoder. This may suggest that somehow feedback and batch normalisation are interacting in such a way that the feedback is rendered ineffective.

5.2. Noisy Activations

The experiment from section 4.4 is repeated on the CIFAR-10 dataset. The network employed is the autoencoder without batch normalisation from the previous experiment.

Since feedback increased the robustness to noise in the MNIST autoencoder, the same behaviour would be expected here. However, as apparent in figure 12, the net-

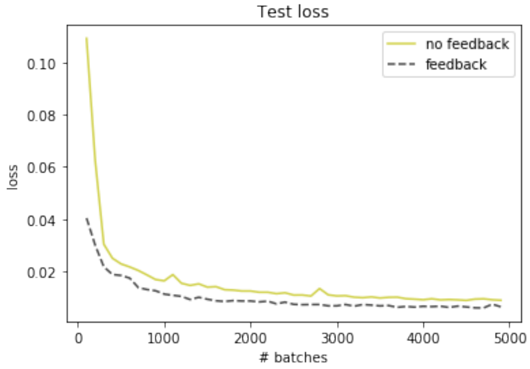

Figure 10. Test set loss of autoencoders with and without feedback on the CIFAR-10 dataset. Neither model makes use of batch normalisation.

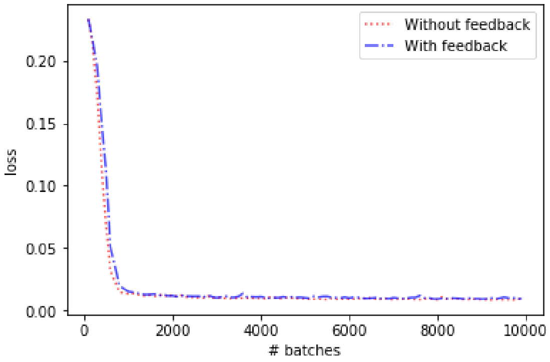

Figure 11. Test set loss of autoencoders with and without feedback on the CIFAR-10 dataset. Both models make use of batch normalisation.

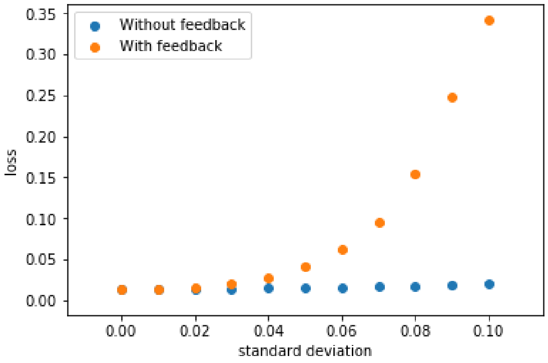

Figure 12. Gaussian noise with zero mean and varying standard deviations is added to the CIFAR-10 autoencoders with and without feedback. Although this is not apparent due to the scale of the plot, the data for the network without feedback follows a similar shape to the one with feedback.

work with feedback is much more sensitive to (even small amounts of) noise than the one without feedback.

compounding effect

5.3. Classification

Classification on the CIFAR-10 dataset is performed using a convolutional neural network. The network consists of two convolutional layers with 64 filters of size 5×5, each followed by a max pooling (Zhou & Chellappa, 1988) layer with a 2×2 window and a stride of 2. The convolution and pooling layers are followed by a fully connected layer (200 units) and a softmax (Bridle, 1990) layer. Batch normalisation is applied after the pooling layers and dropout with a rate of 0.5 is applied after the pooling and the fully connected layers.

To test whether feedback can improve classification performance, the network is trained with and without feedback. Figure 13 shows only a marginal performance difference between the two networks, with the feedback network being slightly better.

Note that the network employed here makes use of batch normalisation, which, as shown in the previous section, may be problematic in combination with feedback. Whether this is the case here is not clear, since this particular network will not converge when batch normalisation is disabled (be it with or without feedback).

6. Conclusion

tell them what you told them

The feedback mechanism presented here is able to improve performance of conventional networks both in terms of

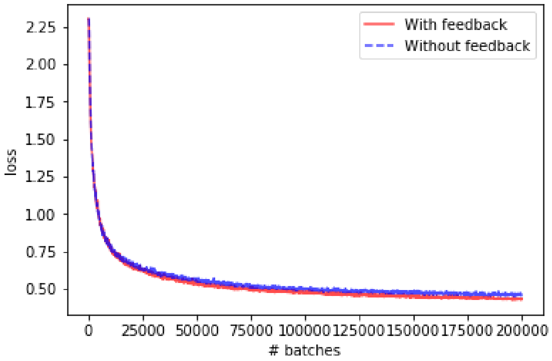

Figure 13. Classification loss on the CIFAR-10 test set. The training time of 200000 batches corresponds to 512 epochs.

convergence speed and convergence value in certain cases. The effectiveness of the mechanism is inconsistent across different datasets, with ... This allows two conclusions: (1) the effectiveness of the mechanism is data-dependent, i.e. it may be leveraging the highly regular structure of the MNIST dataset, ... , or (2) the implementation of the mechanism in the CIFAR-10 experiments is not .... Notably, in convolutional networks, feedback is given on a filter-wise basis, which may be too simplistic.

Noise

Section 5.1 suggests that there may be an (unfavourable) interaction between the feedback mechanism and batch normalisation. Investigating why this is the case may shed further light on how the feedback works and what role it fulfils.

suggest future work: relationship feedback batchnorm

multi modal

References

Bridle, John S. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pp. 227–236. Springer, 1990.

DeFelipe, Javier and Fariñas, Isabel. The pyramidal neuron of the cerebral cortex: Morphological and chemical characteristics of the synaptic inputs. *Progress in Neurobiology*, 39(6):563–607, 1992.

Faisal, A. Aldo, Selen, Luc P. J., and Wolpert, Daniel M. Noise in the nervous system. *Nature Reviews Neuroscience*, 9(4):292–303, 2008.

Goodhill, Geoffrey J and Carreira-Perpiñán, Miguel Á. Cortical columns. *Encyclopedia of cognitive science*, 2002.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Krizhevsky, Alex, Nair, Vinod, and Hinton, Geoffrey. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

Larkum, M. E. Top-down dendritic input increases the gain of layer 5 pyramidal neurons. *Cerebral Cortex*, 14(10): 1059–1070, 2004.

Larkum, Matthew. A cellular mechanism for cortical associations: an organizing principle for the cerebral cortex. *Trends in neurosciences*, 36(3):141–151, 2013.

LeCun, Yann. Generalization and network design strategies. *Connectionism in perspective*, pp. 143–155, 1989.

LeCun, Yann, Cortes, Corinna, and Burges, Christopher JC. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

Markram, Henry, Toledo-Rodriguez, Maria, Wang, Yun, Gupta, Anirudh, Silberberg, Gilad, and Wu, Caizhi. Interneurons of the neocortical inhibitory system. *Nature Reviews Neuroscience*, 5(10):793–807, 2004.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Shipp, Stewart. Structure and function of the cerebral cortex. *Current Biology*, 17(12):R443–R449, 2007.

Srivastava, Nitish, Hinton, Geoffrey E, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

Zhou, YT and Chellappa, R. Computation of optical flow using a neural network. In *IEEE International Conference on Neural Networks*, volume 1998, pp. 71–78, 1988.

7. Appendix

7.1. Hyperparameter Tuning

As mention in section 4, optimal values for β_{max} and η are determined by a grid search. The initial grid is defined by $\eta = [5, 10, 15, \ldots, 50]$ and $\beta_{max} = [0.1, 0.2, \ldots, 0.8]$.

The highest value for β_{max} (0.8) consistently shows the best performance regardless of η's values, as exemplified by figure 14. Note that a high constant value of η with varying values of β_{max} will generally lead to less spread between the loss curves, since the activation function will be more sensitive to β_{max} when η is low.

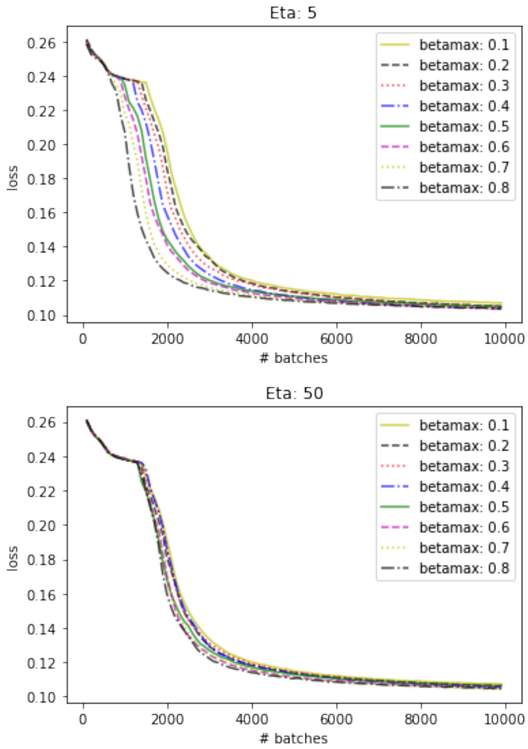

Figure 14. Top: first pass, bottom: second pass

While higher values of β_{max} lead to better performance, the inverse relationship can be seen with η, i.e. lower values of η lead to better performance. This is illustrated in figure 15.

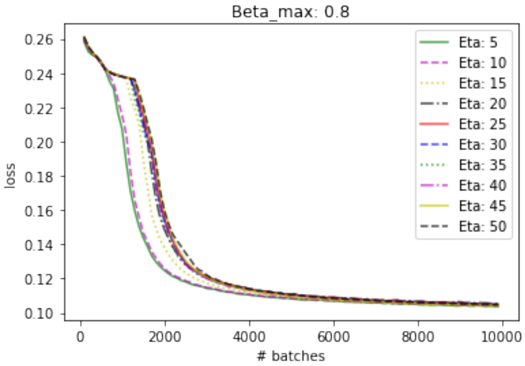

Figure 15. Top: first pass, bottom: second pass

A second grid search with $\eta = [1, 2, 3, 4, 5], \beta_{max} = [0.8, 0.85, 0.9, 0.95]$ is performed to determine whether even lower/higher values can further improve performance. Indeed, increasing β_{max} to 0.95 leads to better performance, but further decreasing η is not advantegeous.

7.2. Visualising Activations

7.3. Feedback-Controlled Threshold

Equation 1 describes not only gain modulation through feedback, but also an adjustment of the activation functions threshold, i.e. $\alpha\mu_D$ is one of the terms in the summation. While gain modulation is the main property of interest in this paper, it is conceivable that the change in threshold plays a significant part in this mechanism as well.

Incorporating this threshold mechanism into equation 6 leads to:

$$f = \frac{max(0, \mu_S + \alpha\mu_D)}{1 - min(\frac{\beta_{max}}{\eta}\ \mu_D, \beta_{max})} \tag{8}$$

where α is a parameter to be learned by the network. While α could also be set to a constant (tuned) value, prior experiments suggest that it is beneficial to let the network adjust $alpha$ during the course of training.

As can be seen in figure 17, the added threshold mechanism is not able to improve upon the network implementing the gain mechanism. Although the models with feedback-controlled threshold both perform better than the standard autoencoder, the model with only gain and no threshold mechanism still has the overall best performance.

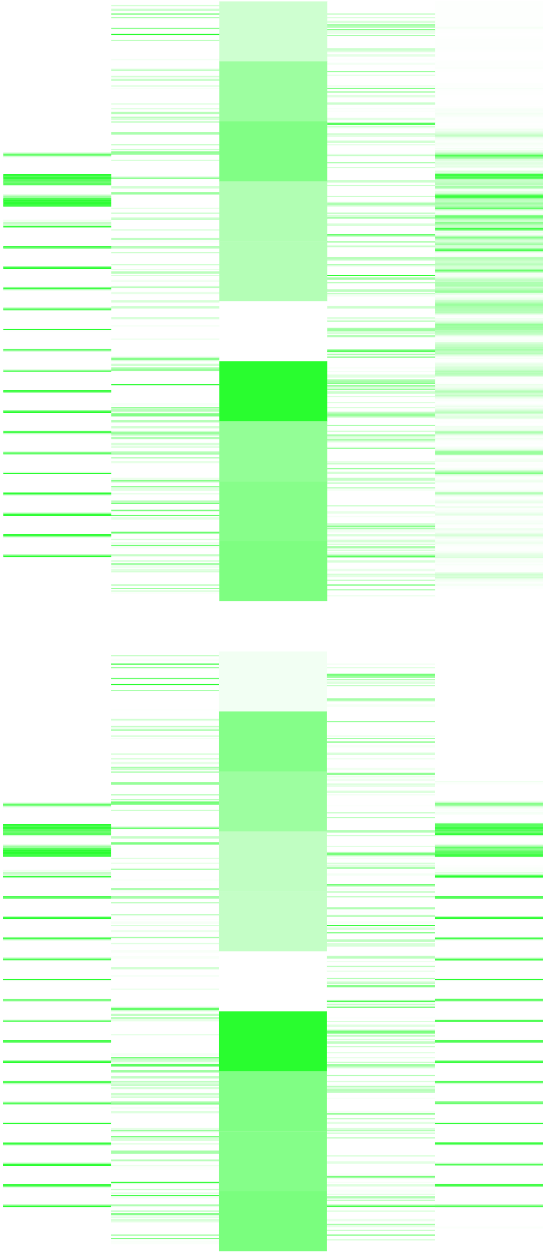

Figure 16. Top: first pass, bottom: second pass

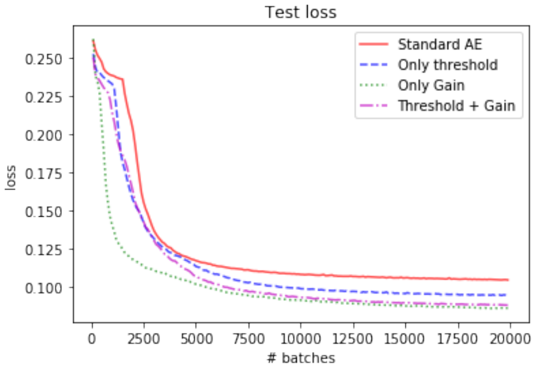

Figure 17. Performance of the standard autoencoder, an autoencoder with feedback-controlled threshold, an autoencoder with feedback-controlled gain, and an autoencoder with both feedback-controlled threshold and gain on the MNIST test set.

7.4. T-SNE

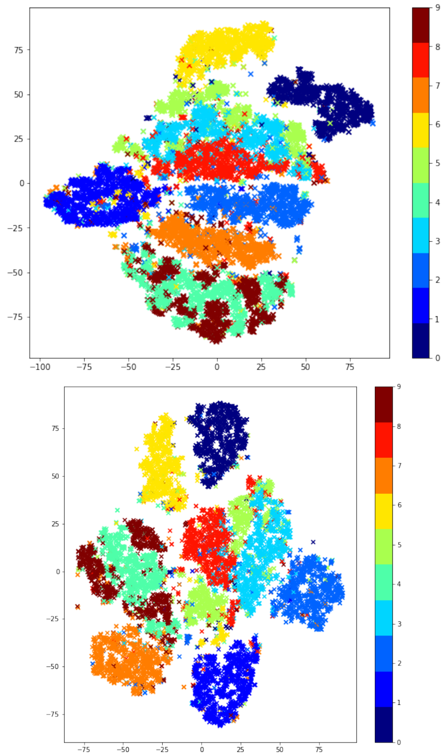

Figure 18. T-SNE visualisation of the second encoding layer of the autoencoder over the whole MNIST test set. Top: first pass, bottom: second pass

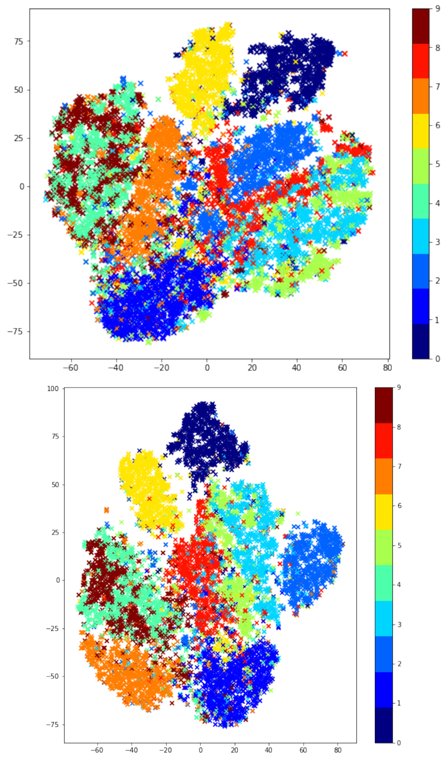

Figure 19. T-SNE visualisation of the second encoding layer of the autoencoder over the whole MNIST test set. Top: first pass, bottom: second pass

7.5. Input With Reduced Contrast

Images with reduced contrast are presented to the trained (on regular contrast images) network, to see if the second pass can reconstruct an image that is more akin to a regular contrast image. To reduce the contrast, each pixel of the image is multiplied by some contrast factor $0 \leq c \leq 1$.

Figure 20 shows the absolute difference in mean pixel value between the first and second pass reconstructions for a number of different contrast factors. A high contrast input image leads to a larger difference in mean pixel value, while a low contrast image leads to a smaller difference between first and second pass reconstructions.

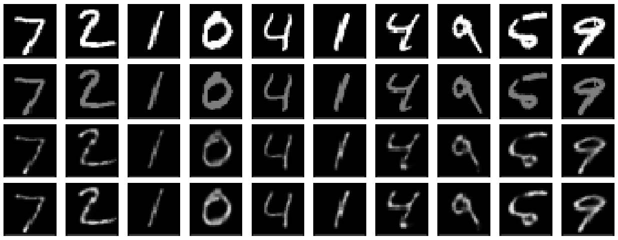

Figure 21. From top to bottom: original image, contrast reduced image, first pass reconstruction, second pass reconstruction. The contrast reduced image was produced by multiplying the original image with a contrast factor of 0.5, i.e. each pixel in the contrast reduced image has values in the range $[0.0, 0.5]$ instead of $[0.0, 1.0]$

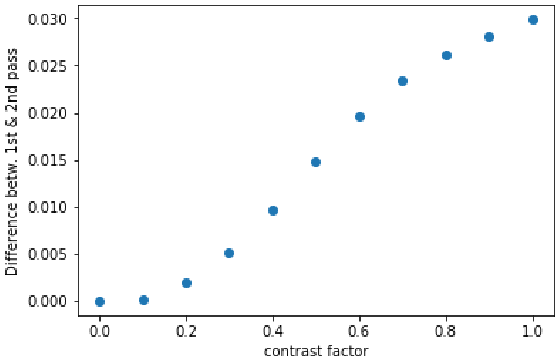

Figure 20. Absolute difference in mean pixel value between first and second pass reconstructions as a function of different contrast factors (from 0.0 to 1.0 in 0.1 increments). A contrast factor of 1.0 corresponds to no reduction in contrast, while a contrast factor of 0.0 means the input images are entirely black.

7.6. Reflections

material learned: got more comfortable with neural nets (theory - practice divide), initialisations, batchnorm, data normalisation