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 networks with feedback are applied on the CIFAR-10 dataset.

1. Introduction

The brain has served as inspiration for artificial neural networks (ANNs) for decades. While these models are usually heavily simplified compared to the brain, they have seen significant successes in areas such as image recognition (Krizhevsky et al., 2012), speech recognition (Hinton et al., 2012), and machine translation (Sutskever et al., 2014) in recent times.

Despite successes, it is clear that the average human brain is vastly more powerful and versatile 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.

One such discrepancy between ANNs and the brain is the existence of feedback, or top-down connections. While there is clear evidence of prominent feedback connections in the brain, ANNs have overwhelmingly been designed based on the feedforward paradigm, although networks that do not work solely on the feedforward principle exist and are called recurrent neural networks (RNNs). Most RNNs used in practice today focus on recurrent connections from one layer to itself (e.g. LSTM networks (Hochreiter & Schmidhuber, 1997)), which, while recurrent, arguably do not constitute top-down connections. These networks are typically applied on problems where the input consists of sequence data, where the recurrence allows for memory of previously seen elements of the sequence.

However, the usefulness of recurrent connections or feedback is not necessarily restricted to sequence data. If the input is image data, a first look, or pass, at an image could be used to construct a rough idea of what the image contains, as well as to identify areas of interest, which can then be further examined on a second pass.

While the network architectures considered in this paper feature real top-down connections, the focus is not on the network topology itself, but on how these top-down connections influence the behaviour of single neurons, i.e. a mechanism for incorporating feedback.

This feedback mechanism is derived from neuroscience literature and examined from two broad angles: (1) Whether the feedback mechanism can in any way improve on standard methods. Relevant metrics include convergence speed and performance quality of the trained network. (2) If examining the feedback's properties and how it behaves under certain conditions (e.g. noisy signals) can offer any insights into what role the feedback might fulfil in the brain. Needless to say, care has to be taken when trying to infer functionality of mechanisms in the brain from simplified artificial networks. Nevertheless, experimentation on artificial models offers an intriguing opportunity, as they are naturally easier to investigate and manipulate than the real brain.

In the remainder of this paper, some neuroscientific background is explored in section 2 to serve as context for the feedback mechanism, followed by a description of the feedback mechanism itself as it occurs in the brain (section 2.1). In section 3 the mechanism is adapted for use in ANNs and some practical considerations on its use are given in section 3.1. The following sections describe a range of experiments with the intention to provide answers to the research questions posed above.

2. Neuroscientific Background

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.,

---

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

2004) and is therefore often speculated to be responsible for the emergence of higher intelligence.

The most abundant type of neuron in the neocortex is the pyramidal neuron, 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 the 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 (Larkum, 2013).

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 (Brodmann, 1909). 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. it serves as a 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.

2.1. 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 neuron 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):

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

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:

$$f = \mu_S + f\beta(\mu_D) \qquad (2)$$

Removing f from the right hand side:

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

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:

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

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:

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

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

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

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 priorly 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 then be utilised in the second pass. Although more than two timesteps are not required, it is possible to use an arbitrary number of timesteps, which is examined in section 4.1.1.

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.

The resulting networks can then be unrolled to create a feedforward network, so that for t timesteps, each layer occurs t times, while using the same weights at each timestep (see figure 1). Since the unrolled network is purely feedforward, the standard backpropagation is a suitable learning rule.

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

Dropout (Srivastava et al., 2014) should be used by dropping out the same units in all passes. Otherwise, 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, which defeats the purpose of dropout.

4. Experimental Results

The preceding sections describe a feedback mechanism and how it can be implemented in practice. Here, a range of ex-
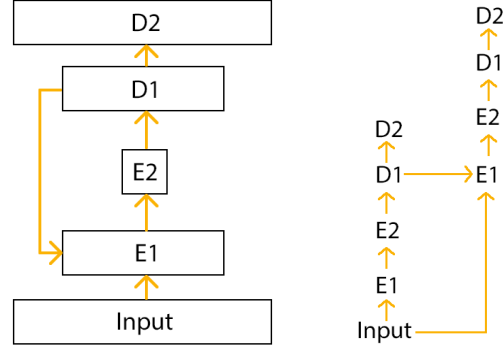

Figure 1. Left: autoencoder with (partial) feedback. Right: Unrolled autoencoder.

periments is performed to observe how this feedback mechanism changes the behaviour of ANNs. Several networks are applied on two datasets, MNIST (LeCun et al., 2010) and CIFAR-10 (Krizhevsky et al., 2014). Specifically, the experiments are designed to answer the research questions posed in the introduction: (1) whether feedback can improve the performance of ANNs, (2) whether observing how the feedback works in artificial models can reveal any clues on what function feedback has in the brain. Sections 4.1.3, 4.1.4, and 4.2.2 serve to answer the latter question, where section 4.1.3 is more of a general analysis of feedback, while sections 4.1.4 and 4.2.2 test whether feedback might increase the networks robustness to noise. The remaining sections are concerned primarily with question (1) in that they test convergence speed and performance quality in various configurations.

4.1. MNIST

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).
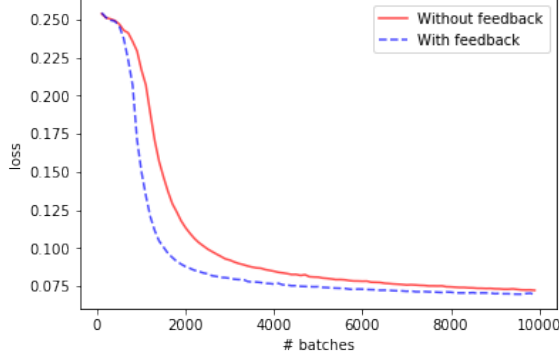
Figure 2. Test set loss of autoencoders with and without feedback. The dimension of the second encoding layer is 196.
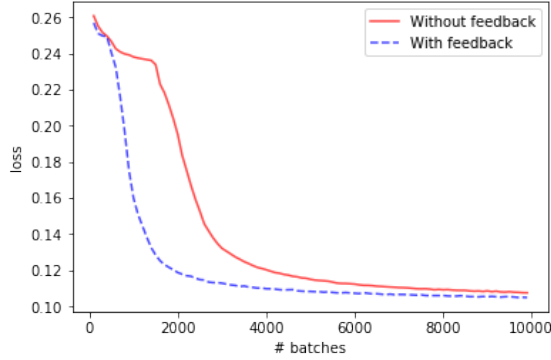

Figure 4. Autoencoder performance with varying numbers of timesteps. Each configuration was trained and evaluated 10 times. The curves shown are the averaged losses on the test set.

4.1.1. 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 4, 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.

4.1.2. COMPREHENSIVE FEEDBACK

In the previous experiments, feedback is only sent from one decoding layer to one encoding layer. Naturally, there are many more possible configurations that incorporate further feedback connections. In the following experiment, each layer receives feedback from every layer above it, i.e. every possible top-down connection is present in the network. This will be referred to as *comprehensive* feedback, whereas the previous approach will be referred to as *partial* feedback.

As shown in figure 5, the configuration explained above does not only converge faster than a standard autoencoder, but also settles to a smaller loss value, which was not the case when only partial feedback was applied.

4.1.3. FEEDBACK VS CONSTANT GAIN

In an effort to gain some understanding on how exactly feedback helps to improve performance, the frequency of different feedback values is examined. A distinction is made between feedback and gain, where feedback refers

Figure 3. Test set loss of autoencoders with and without feedback. The dimension of the second encoding layer is 10.

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

Figure 2 shows the loss curves for the autoencoder with and without feedback. While the autoencoder with feedback converges noticeably faster, the difference is relatively small. It is conceivable that feedback might have a greater effect if the difficulty of the task is increased. While difficulty is not a well defined term, reducing the dimension of the second encoding layer (i.e. the bottleneck) can arguably be seen as an increase in difficulty.

The dimension of the second encoding layer is thus reduced to 10 (this modification will persist in all subsequent experiments) and the experiment is repeated. Indeed, figure 3 shows a much larger gap between the autoencoder with feedback and the one without it, supporting the hypothesis that feedback may be more beneficial on more difficult tasks.
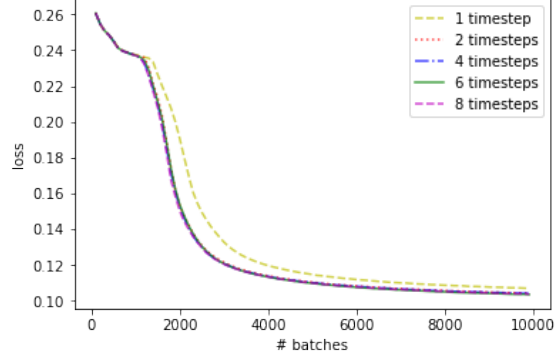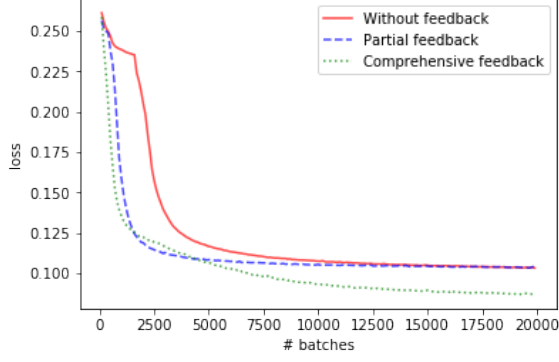
Figure 5. Loss on the test set of autoencoders without feedback, partial feedback, and comprehensive feedback. Note that the horizontal axis is different from previous figures, i.e. the training time is longer.

to μ_D and gain refers to $\frac{1}{1-min(\frac{\beta_{max}}{\eta}\ (\mu_D),\beta_{max})}$.

Figure 6 shows the data as collected in a 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 7, 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, i.e. for an autoencoder with comprehensive feedback, yields results as illustrated in figure 8.

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 performance.

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.
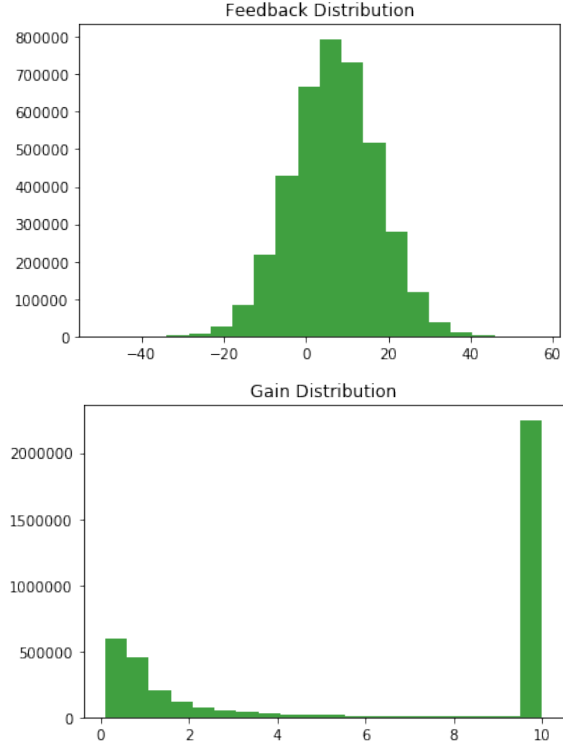

Figure 6. Distribution of feedback (top) and gain (bottom) values collected in a network with partial feedback over the complete MNIST test set.
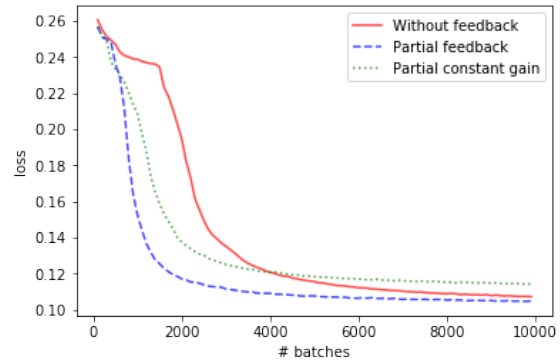

Figure 7. Comparison of a standard autoencoder, an autoencoder with partial feedback, and an autoencoder with partial constant gain (the gain of all units in the second encoding layer is set to 10)
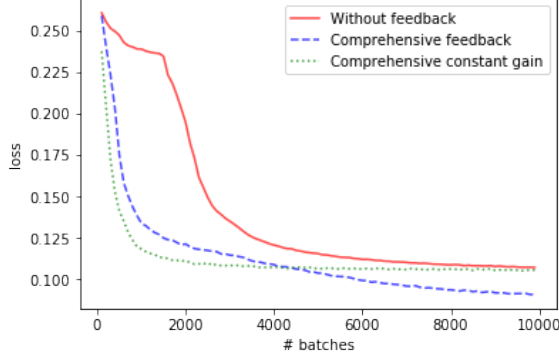
Figure 8. Comparison of a standard autoencoder, an autoencoder with comprehensive feedback, and an autoencoder with comprehensive gain (the gain of all layers is set to 10).

4.1.4. 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.

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

As figure 10 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.
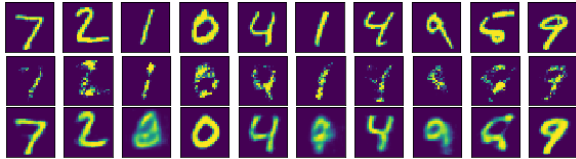

Figure 9. Gaussian noise with zero mean and standard deviation $\sigma = 2.0$ is added to networks with and without feedback. The top row shows input instances to the network, the middle and bottom row show reconstructions of the network without and with feedback (respectively).

4.2. 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 fol-
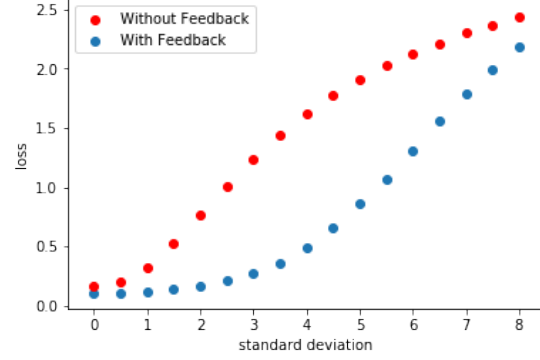

Figure 10. Gaussian noise with zero mean and varying standard deviations (horizontal) is added to networks with and without feedback. The quality of the reconstruction, as measured by the loss function (vertical axis), with respect to the magnitude of the standard deviation is shown for both networks.

lowing classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck (Krizhevsky et al., 2014).

4.2.1. 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.

As figure 11 shows, the autoencoder with feedback clearly performs better than the one without it, although 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.

4.2.2. NOISY ACTIVATIONS

The experiment from section 4.1.4 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 13, the network with feedback is much more sensitive to (even small amounts of) noise than the one without feedback.

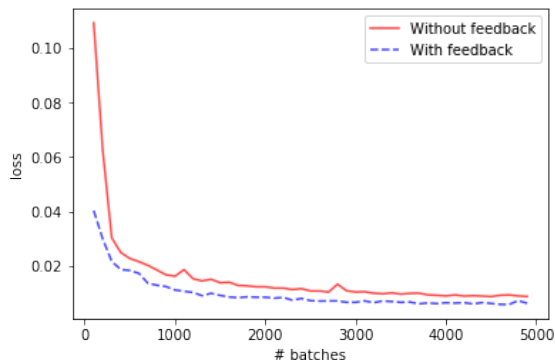This may be an indication that the feedback learned by

Figure 11. Test set loss of autoencoders with and without feedback on the CIFAR-10 dataset. Neither model makes use of batch normalisation.
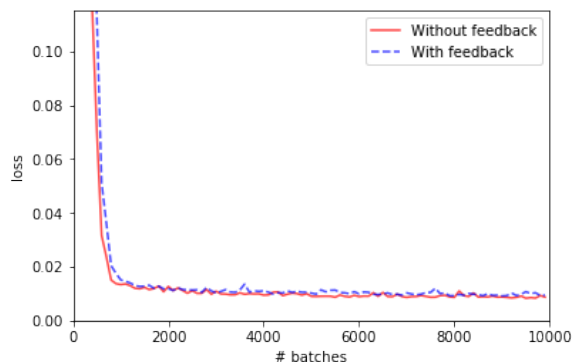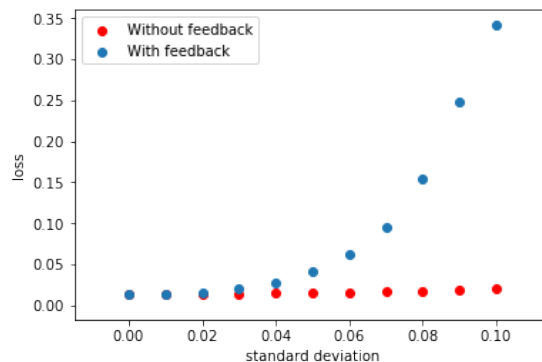

Figure 13. Gaussian noise with zero mean and varying standard deviations is added to the CIFAR-10 autoencoders with and without feedback. Although this is not apparent due to the scale of the plot, the data for the network without feedback follows a similar shape to the one with feedback.

the network is fundamentally different from the feedback learned in the MNIST experiments, such that it has a compounding effect on noise, rather than a rectifying one.

4.2.3. 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 (comprehensive) and without feedback. Figure 14 shows only a marginal performance difference between the two networks, with the feedback network being slightly better. At the end of training, the classification accuracy over the complete test set is about 0.7% higher for the network with feedback.

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 does not converge when batch normalisation is disabled (be it with or without feedback).

Figure 12. Test set loss of autoencoders with and without feedback on the CIFAR-10 dataset. Both models make use of batch normalisation.

5. Conclusion

The feedback mechanism presented here is able to improve performance of conventional networks both in terms
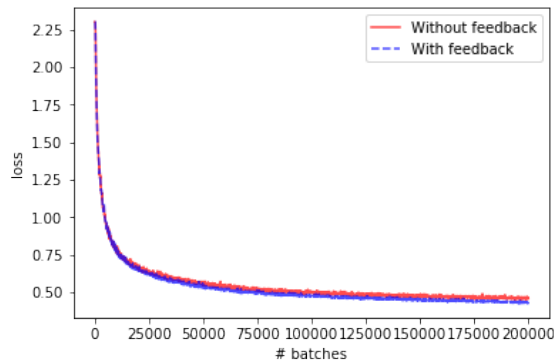
Figure 14. Classification loss on the CIFAR-10 test set. The training time of 200000 batches corresponds to 512 epochs.

of convergence speed and performance of the trained network when applied on the MNIST dataset. The benefits of feedback are less clear, however, when applied on the CIFAR-10 dataset. In principle, an autoencoder with feedback can outperform a corresponding autoencoder without feedback to a small degree, but this positive effect of feedback is negated when batch normalisation is utilised in the autoencoders. Understanding this unfavourable interaction between feedback and batch normalisation may be an opportunity to gain a deeper understanding on how feedback works and what role it fulfils.

Feedback appears to have some positive effect when performing classification on CIFAR-10, although this effect is so small that drawing any firm conclusions seems ill-advised.

When investigating the networks robustness to noise, an even larger divide between performance on MNIST and CIFAR-10 can be observed. On CIFAR-10, feedback is not only not beneficial, it actually heavily increases the network's sensitivity to noise, while the MNIST autoencoder becomes more robust when feedback is present.

A possible explanation for this difference across datasets could be that the effectiveness of the feedback mechanism is data-dependent, i.e. it may be leveraging the highly regular structure of the MNIST dataset and is thus not as useful on the less regularly structured CIFAR-10 dataset.

A further general difference between the experiments on the two different datasets is the use of convolutional layers, which were used in all of the CIFAR-10 experiments, but not in any of the MNIST experiments. It may be that providing feedback on a filter-wise basis is too simplistic, or that some other aspect related to convolution is not conducive to the feedback mechanism. Further research on the combination of feedback and convolutional networks may lead to some configuration that allows for more clear benefits of feedback.

Naturally, it might also be the case that the results on MNIST are merely an outlier, which somehow defies a more fundamental problem with the usage of feedback in current ANNs, e.g. it may be that backpropagation is not an ideal learning algorithm for feedback, or that feedback relies on more realistic models such as spiking neural networks (Ghosh-Dastidar & Adeli, 2009).

Should clear evidence arise that feedback is useful beyond MNIST, an interesting avenue of future research would be the creation of feedback based multi-modal models, where sensory inputs from multiple different sources are combined to perform e.g. a classification task. For instance, if a network receives both visual and auditory input, the barking of a dog may result (mediated by feedback) in a higher expectation to observe a dog in the visual input.

References

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

Brodmann, Korbinian. *Vergleichende Lokalisationslehre der Grosshirnrinde in ihren Prinzipien dargestellt auf Grund des Zellenbaues*. Barth, 1909.

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.

Ghosh-Dastidar, Samanwoy and Adeli, Hojjat. Spiking neural networks. *International journal of neural systems*, 19(04):295–308, 2009.

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

Hinton, Geoffrey, Deng, Li, Yu, Dong, Dahl, George E, Mohamed, Abdel-rahman, Jaitly, Navdeep, Senior, Andrew, Vanhoucke, Vincent, Nguyen, Patrick, Sainath, Tara N, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29 (6):82–97, 2012.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

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.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 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.

6. Appendix

6.1. Hyperparameter Tuning

As mention in section 4.1, 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 15. 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 16. Autoencoder performance when β_{max} is fixed at 0.8 and η is varied.

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.

6.2. 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:

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

Figure 15. Autoencoder performance with varying hyperparameters. Top: η is fixed at 5 and β_{max} is varied, bottom: η is fixed at 50 and β_{max} is varied.

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.

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 16.
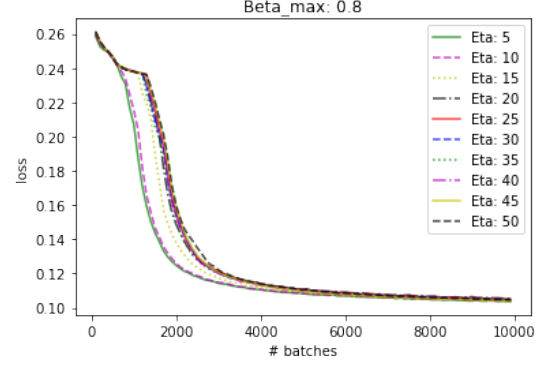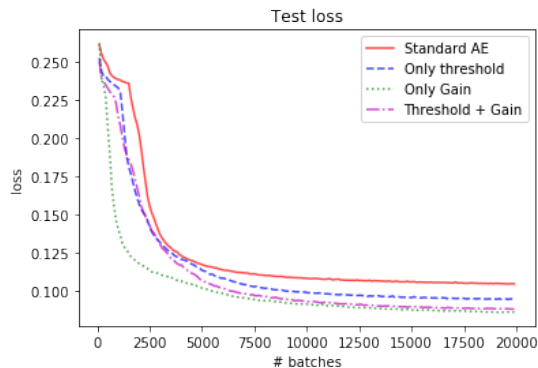
Figure 17. Performance of the standard autoencoder, an autoencoder with feedback-controlled threshold, an autoencoder with feedback-controlled gain, and an autoencoder with both feedback-controlled threshold and gain on the MNIST test set.

Figure 19. 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 $[0.0, 1.0]$

6.3. 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 19 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.
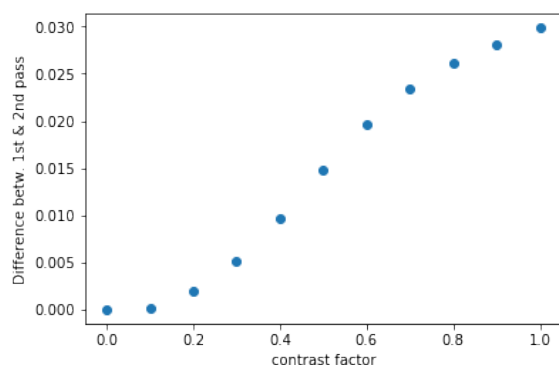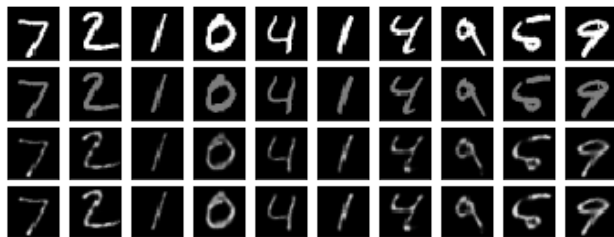
6.4. Additional Figures

The following figures contain additional data that was collected as part of the experiments in section 4.

Figure 18. 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.
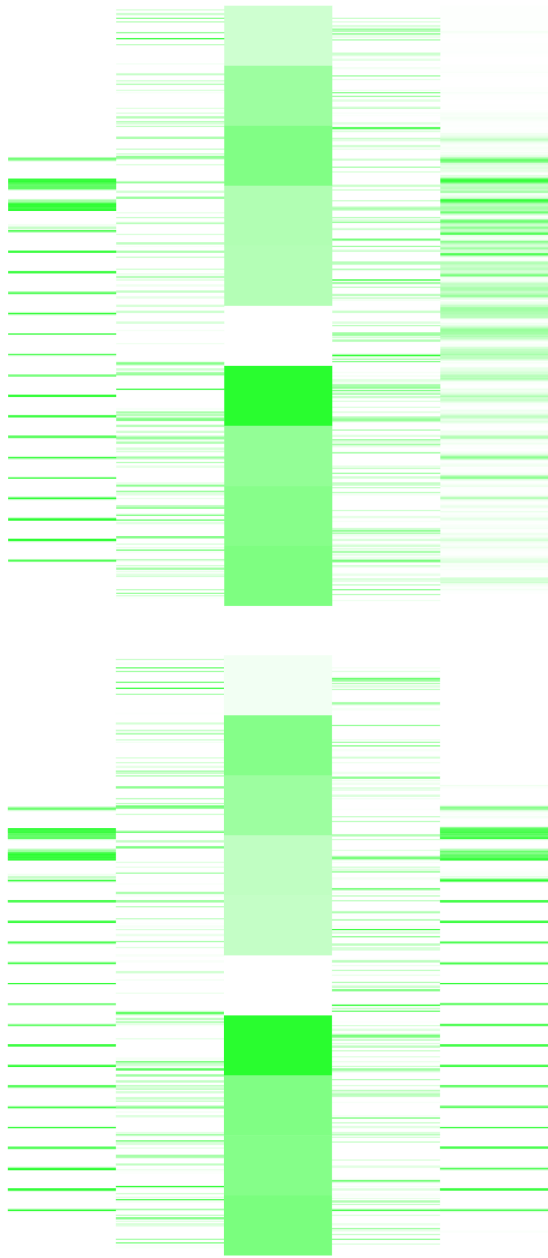
Figure 20. Visualisation of activations in the MNIST autoencoder for one particular test instance. The leftmost column corresponds to the input layer and the remaining columns correspond to the first encoding layer, the second encoding layer, the first decoding layer, and the second decoding layer, respectively. The number of rectangles in each column corresponds to the number of units in that layer. Larger values are represented by green coloured rectangles, and smaller values by white ones. Top: first pass, bottom: second pass.
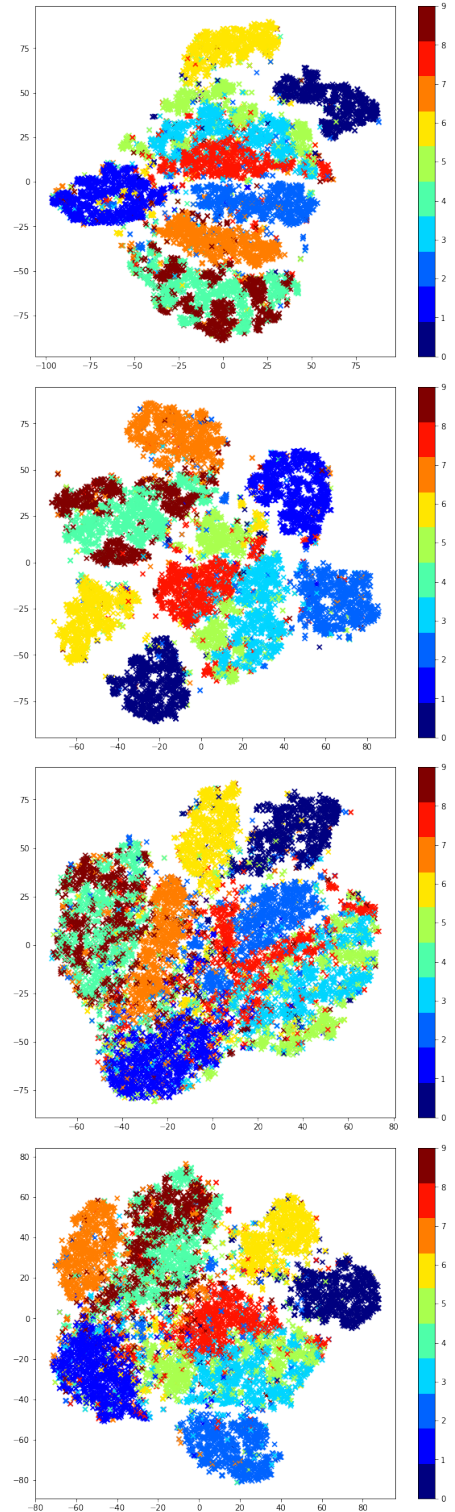

Figure 21. T-SNE visualisation of the second encoding layer of the autoencoder over the whole MNIST test set. From top to bottom: first pass, second pass, first pass with noise (as described in section 4.1.4), second pass with noise.
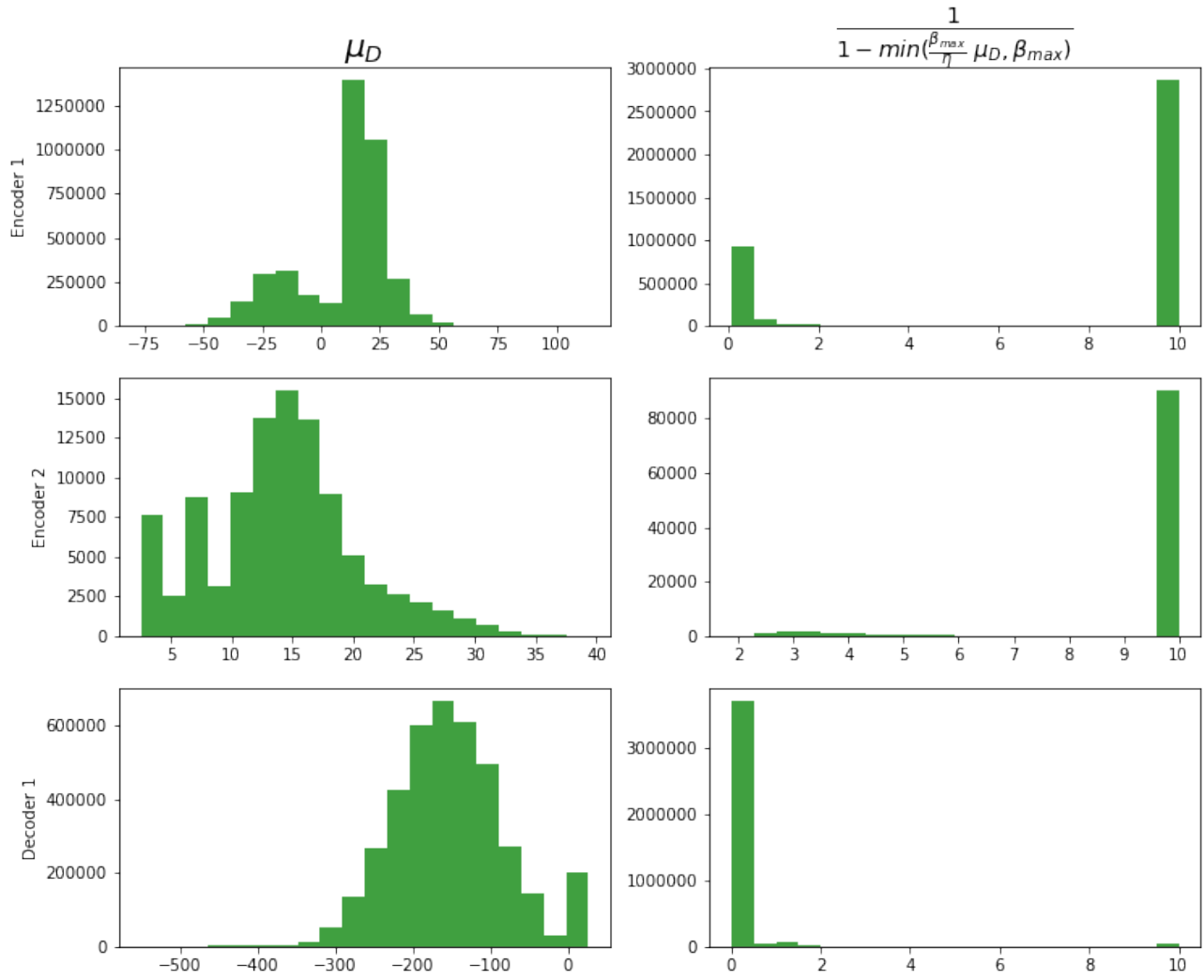
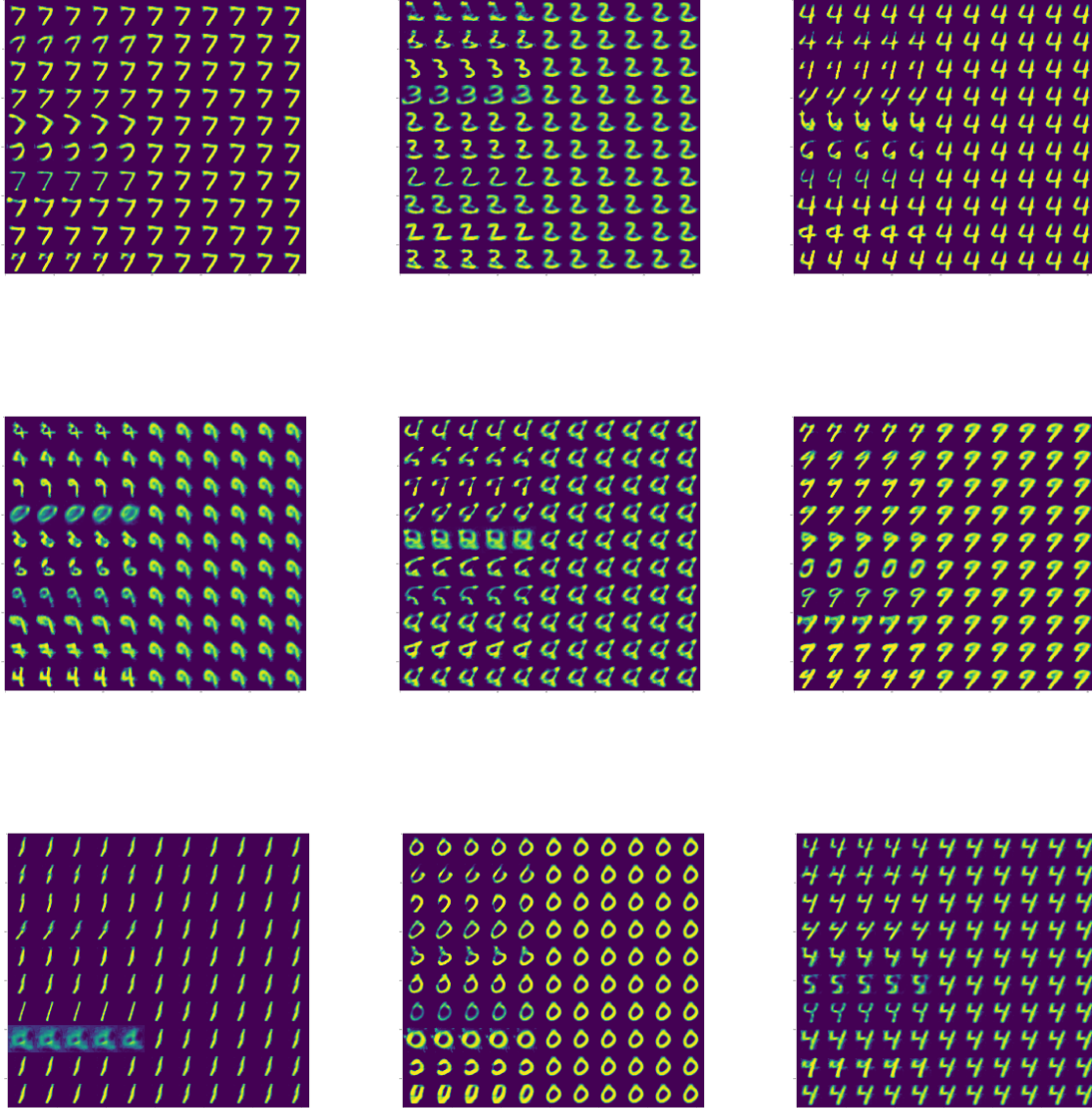*Figure 22.* Histograms as seen in section 4.1.3, but for the autoencoder with comprehensive feedback.

Figure 23. Different gain values are manually fed into the second encoding layer of the network and the resulting reconstruction is visualised. In each of the above images, one specific input image is presented to the network, but the gain is varied. In row i of each image, every unit of the second encoding layer receives a gain of 10, except for unit i, which receives a gain between 0 and 10, depending on the column it is in. When using the MNIST autoencoder with comprehensive feedback, it can be observed that only one unit in the second encoding layer has any variation in gain (the remaining ones have a constant gain of 10 regardless of the input). This one unit corresponds to the fourth row from the bottom of each image and seems to be responsible for setting the 'intensity' of the reconstruction.

6.5. Reflections

6.5.1. NON COMPUTER SCIENCE LITERATURE

Venturing into a field of study that is not one's own is a challenging endeavour due to the sheer amount of foreign concepts and terminology that can make the literature seem very opaque. Consequently, when trying to familiarise ourselves with a new field of study, a systematic approach has to be taken, identifying the basic concepts necessary to understand the literature of interest, internalising those basic concepts, and then building up more advanced concepts over time. The emphasis is on *necessary* concepts, since there is a virtually unlimited list of things to be learned, but to achieve anything in a reasonable amount of time, the scope has to be limited.

6.5.2. NEURAL NETWORKS IN PRACTICE

While a university course is great for developing a theoretical understanding of neural networks/deep learning, there is a large amount of details that have to be considered in practice. These details include the choice and usage of programming language/framework, how to initialise weight matrices, what optimisers to use, what learning rates or other hyperparameters to set, etc. These things could be (and partially are) taught in courses, but they are often not inherently interesting and thus unlikely to be internalised well when only looked at from a theoretical perspective. Simply applying these things in practice in some moderately long project helps to build a clear picture and gain experience, which removes this barrier of entry from all similar subsequent projects.

6.5.3. ORGANISING WORK

Organising and documenting work is important to avoid unnecessarily redundant work and saves time and effort especially in the later stages of a project. It is a lesson that has to be learned many times and is ultimately a tradeoff between the time spent organising things and the time wasted due to a lack of organisation. Any self-directed project helps gain the necessary experience to find a process resulting in minimal, but still sufficient documentation and organisation.
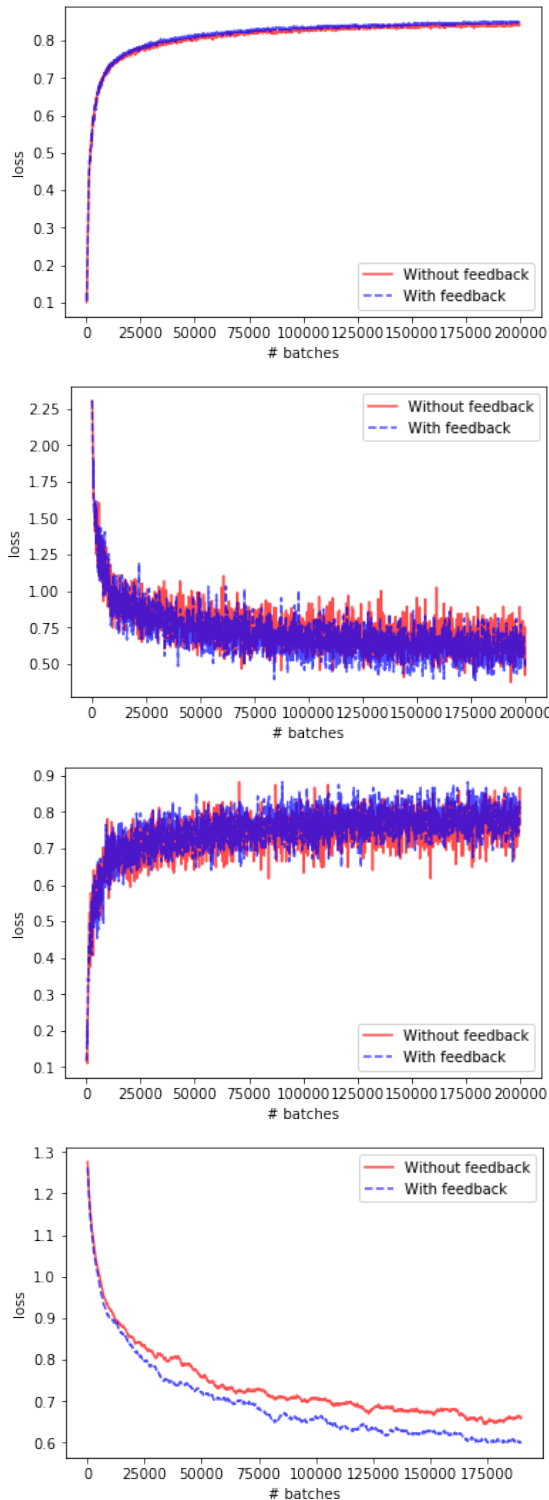
Figure 24. CIFAR-10 classification as seen in section 4.2.3. From top to bottom: test set accuracy, training set loss, training set accuracy, training set loss after applying a moving average filter (window size 100).