Incorporating Feedback in Convolutional Neural Networks

Christian Jarvers (christian.jarvers@uni-ulm.de)
Institute of Neural Information Processing, Ulm University
James-Franck-Ring, 89081 Ulm, Germany

Heiko Neumann (heiko.neumann@uni-ulm.de)
Institute of Neural Information Processing, Ulm University
James-Franck-Ring, 89081 Ulm, Germany

Abstract

Convolutional neural networks (CNNs) are a state-of-the-art machine learning method, partially inspired by the hierarchical structure of cortex. They typically process information from input to output in a feedforward manner. It has been shown that incorporating feedback pathways can improve their performance and robustness. However, little is known about why feedback helps and how feedforward and feedback signals are best combined.

Here, we compare feedforward and feedback networks using a multi-digit classification task, quantifying performance as well as robustness against image noise. We show that the advantage of feedback networks which add the feedback to the feedforward signal is largely due to the increased receptive field size of their neurons. In addition, we show that networks which use modulating or subtractive feedback (inspired by theories of feedback processing in cortex) outperform additive architectures and have increased robustness against noise.

These results provide a first step towards using feedback in convolutional neural networks more effectively.

Keywords: deep learning; convolutional neural networks; feedback; biased competition; predictive coding

Introduction

Deep networks are currently the most successful method for a large number of computer vision tasks and are increasingly seen as a possible model for the human visual system (Kriegeskorte, 2015). They are partially inspired by the hierarchical arrangement of processing areas in visual cortex and similarly process images via a sequence of trainable layers. However, one crucial difference between biological and deep vision is that the latter is not a one-directional hierarchy but has abundant feedback connections (Kravitz, Saleem, Baker, Ungerleider, & Mishkin, 2013).

Incorporating feedback connections in deep networks has several possible advantages. Zamir et al. (2016) argued that feedback networks can structure their predictions according to a taxonomy and make rough predictions early on. They proposed a generic way to implement feedback in CNNs using convolutional long short-term memory (LSTM) layers and showed that they outperform comparable feedforward networks on several tasks. Similar results were demonstrated with a feedback architecture based on residual networks (Liao & Poggio, 2016). Spoerer, McClure, and Kriegeskorte (2017) showed that networks with feedback outperformed networks without when controlling for the number of trainable parameters and that this was especially the case in challenging conditions with visual occlusions or noise.

While these studies show that feedback improves network performance, it is not clear how feedforward and feedback signals should be combined. Some studies added forward and feedback signals (Liao & Poggio, 2016; Spoerer et al., 2017), while others used schemes based on LSTMs (Zamir et al., 2016). Lotter, Kreiman, and Cox (2016) used a more elaborate architecture based on predictive coding, a scheme that may also be used in cortex (Spratling, 2008).

Here, we evaluated several possible schemes for combining feedforward and feedback signals with respect to their performance on a multi-digit classification task as well as their robustness to noise. We compared schemes using addition (as previous works), gating (inspired by LSTMs), modulation (inspired by biased competition (Spratling, 2008)), subtraction and division (both inspired by predictive coding (Spratling, 2008)). The modulating and subtractive feedback architectures offered the best tradeoff of performance and robustness.

Methods

Datasets

All networks were trained on multiMNIST images (Sabour, Frosst, & Hinton, 2017). Each image was generated by sampling two digits from the MNIST training set (Lecun, Bottou, Bengio, & Haffner, 1998), applying a random shift of up to 4 pixels in x- and y-direction to each and superimposing them by a pixel-wise maximum operation. Example images are shown in Figure 1. The two digits in each image typically had a large overlap, such that the network had to disambiguate local information, similar to the digit clutter condition in (Spoerer et al., 2017) but on handwritten instead of computer generated digits.

Test images were generated in the same way as the training images, but from the MNIST test dataset. In addition, noisy versions of the test dataset were generated by adding varying salt-and-pepper noise or white noise to the images (see Figure 1 c and d).

Networks

We evaluated nine network architectures. Five of these were feedback networks with different coupling mechanisms.

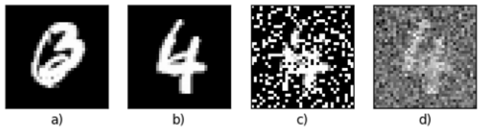

Figure 1: a) and b) Example images from the MultiMNIST dataset. c) Example image with 50% salt-and-pepper noise. d) Example image with white noise where $\sigma = 0.5$.

Table 1: Comparison of network architectures.

Net	Layers	Kernel	Parameters	Top RF
`b`	2	3x3x32	9,898	5x5
`bf`	2	3x3x45	19,234	5x5
`bk`	2	5x5x32	26,794	9x9
`slim`	8	3x3x17	18,676	17x17
`fb`	2(x4)	3x3x32	19,114	17x17

The remaining four networks were feedforward networks and served as baselines.

Feedforward networks Three of the feedforward networks were adapted from Spoerer et al. (2017). The bottom-up network (denoted `b`) was equivalent to a single forward pass through a feedback network without any recurrence. It consisted of two convolutional layers with 32 kernels of size 3x3 and rectified linear unit non-linearity, followed by local response normalization (lrn).

Since feedback networks had additional connections from the second to the first layer, their number of free parameters was higher. Spoerer et al. (2017) proposed to control for this with two additional architectures: one with additional feature maps per layer (denoted `bf`) and one with larger kernels (denoted `bk`). For a comparison of the network architectures and their number of free parameters, see Table 1.

These baselines control for the number of free parameters, but not for the processing depth and the size of the highest receptive field. Feedback networks are unrolled over time (see Figure 2), resulting in more convolutions between the first input and the final readout. Each convolution increases the effective size of the receptive field of the top-most neurons, such that neurons in a feedback network could integrate information from a far larger spatial extent (17x17 pixels, in contrast to 5x5 to 9x9 pixels in feedforward architectures). To control for this aspect, we also included a narrower but deeper feedforward network (denoted `slim`) with 8 layers, resulting in a 17x17 receptive field at the top level as well.

Feedback networks All feedback networks were based on network `b`, with an additional convolutional feedback connection from the upper to the lower layer. This feedback introduced a cyclic dependency, which was implemented by unrolling the network over time (four time steps, see Figure 2). They differed in the way the forward and backward signal were combined (see Table 2).

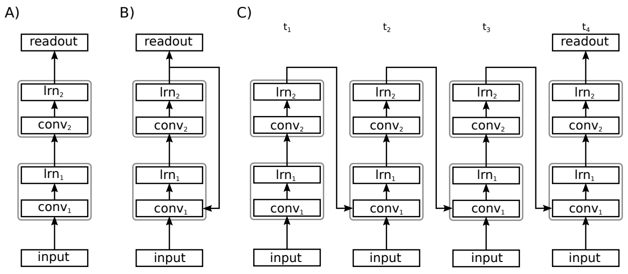

Figure 2: Network architectures. **A)** Bottom-up network with two blocks of convolutions with local response normalization (lrn), followed by readout (global pooling and dense layer). **B)** Feedback architectures with an additional recurrent connection. **C)** Unrolled feedback network over timesteps to resolve cyclic dependency introduced by feedback.

Table 2: Feedback schemes.[1]

Network	$conv(x,y) =$
`add`	$r(w_{ff} * x + w_{fb} * y + b)$
`gate`	$r(w_{ff} * x \odot \sigma(w_{fb} * y) + b)$
`mod`	$r(w_{ff} * x \odot (1 + w_{fb} * y) + b)$
`sub`	$r(w_{ff} * x + b) - w_{fb} * y$
`div`	$r(w_{ff} * x + b) \oslash \left[w_{fb} * y\right]_\varepsilon$

Previous work on feedback in CNNs simply added forward and feedback signals, after convolving each with a different kernel, and applied a non-linearity (Liao & Poggio, 2016; Spoerer et al., 2017). We call this the additive architecture (`add`).

In contrast, Zamir et al. (2016) used LSTMs to combine feedforward and feedback inputs. LSTM cells use several gates to select which parts of the input and recurrent state should be processed. We designed one feedback architecture (denoted `gate`) that used a simplified version of this gating principle, such that feedback (transformed by a sigmoid non-linearity) selected whether feedforward signals were passed on or suppressed.

Another possible role of feedback is to amplify parts of the forward signal. This has been proposed as a canonical cortical computation (Brosch & Neumann, 2014). Since each convolution was followed by local response normalization, this scheme resembles biased competition (Spratling, 2008), with feedback acting as a bias and normalization putting the cells into competition. We denote this architecture as `mod` (modulating).

Finally, feedback could have a suppressive effect, filtering out parts of the forward signal that are not informative. This is akin to predictive coding, which can be implemented by subtractive or divisive feedback (Spratling, 2008). We implemented both versions, denoted `sub` and `div`, respectively.

[1]With forward input x, feedback input y, bias b, weights w_{ff} and

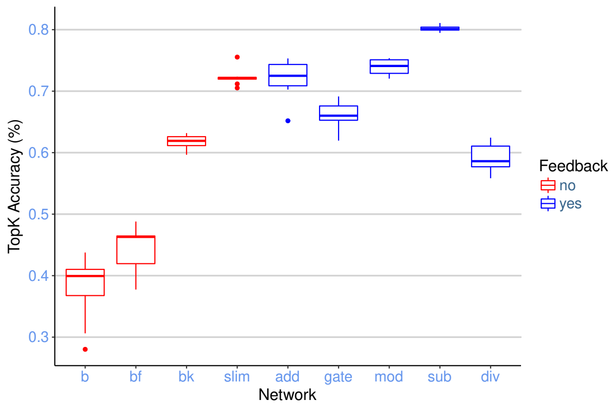

Figure 3: Performance on the test set.

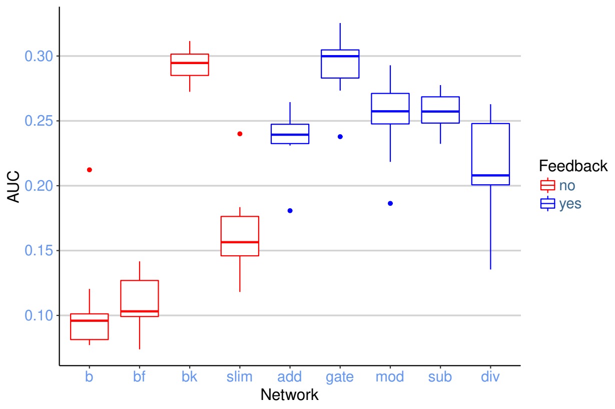

Figure 4: Robustness against white noise.

Training

The training was carried out in two steps. First, several short training runs were carried out for each architecture to optimize the learning rate and the strength of the local response normalization layers. Subsequently, each network architecture was trained ten times with different random initializations.

All kernels were initialized using the Glorot initialization scheme (Glorot & Bengio, 2010) and subject to L2 regularization with a coefficient $\lambda = 0.0005$ throughout the training, as in (Spoerer et al., 2017). Input images were normalized to have zero mean and unit variance. Optimization was carried out with stochastic gradient descent and a momentum term of $\mu = 0.9$. We used TensorFlow 1.8.0, Python 3.6.5 and NumPy 1.14.3 on Nvidia V100 GPUs.

Results

We measured performance as multiclass top-k accuracy, where k is the number of digit classes in an image (i.e., an image with k digit classes was classified correctly if the k strongest responses of the readout layer matched these k classes). The bottom-up network (`b`) performed worst with about 40% while the subtractive feedback network (`sub`) performed best with about 80% (see Figure 3).

Among the architectures compared in Spoerer et al. (2017), we observed the same ordering of performance with `b` outperformed by `bf` (Wilcoxon signed-rank test, $Z = 2.87$, $p < 0.005$), which is outperformed by `bk` ($Z = 3.78$, $p < 0.001$), which in turn is outperformed by `add` ($Z = 3.78$, $p < 0.001$).[2] Thus, we replicated part of the findings of Spoerer et al. (2017) on a different dataset. However, the additional baseline `slim`, which controlled for network depth and receptive field size, did not perform significantly worse than the additive recurrent network ($Z = 0.15$, $p = 0.880$). It therefore seems that the improved performance of feedback networks reported in Spoerer et al. (2017) is largely due to the increased depth from temporal unrolling and the resulting larger effective receptive fields.

As `slim` performed best among the feedforward architectures, we used it as baseline to compare against the feedback networks. The architectures `mod` and `sub` performed better than baseline ($Z = 2.69$, $p < 0.001$ and $Z = 3.78$, $p < 0.001$, respectively), whereas `gate` and `div` performed significantly worse ($Z = 3.78$, $p < 0.001$ and $Z = 3.78$, $p < 0.001$, respectively).

In addition to performance, we also investigated the networks' robustness to noise. We did this by adding varying levels of white noise (variance $\sigma \in [0.0, 1.0]$ in steps of 0.1) or salt-and-pepper noise (probability p of corrupting a pixel $p \in [0.0, 1.0]$ in steps of 0.1). We tested each network for each noise level and normalized results (normalized accuracy $acc_{norm}(p) = (acc(p) - acc(0))/(acc(1) - acc(0))$) and calculated the area under the curve (AOC) as an indicator for robustness. A smaller AUC indicates that performance degraded more rapidly with increasing noise.

The bottom-up architecture with increased kernel size (`bk`) and the network with gating feedback (`gate`) were most robust against noise (see Figures 4 and 5). The difference in performance between these networks was not significant ($Z = 0.605$, $p = 0.545$ for white noise, $Z = 0.756$, $p = 0.450$ in salt-and-pepper noise). The good robustness of `bk` is likely due to the fact that larger kernels have a stronger smoothing effect, averaging out part of the noise.

Among the networks with 3-by-3 kernels, the feedback architectures were generally more robust than the feedforward baselines. All feedback networks were more robust than the best baseline (`slim`) on white noise and `gate`, `mod` and `sub` were also more robust on salt-and-pepper noise (see Table 3).

Thus, feedback seems to increase the robustness of convolutional networks against noise, especially if it is gating, modulating or subtractive. Similarly, larger kernels also increase robustness.

w_{fb}, convolution $*$, element-wise multiplication $\odot$, ReLU nonlinearity r, sigmoid nonlinearity σ and thresholding $[]_\varepsilon$.

[2] All p-values were adjusted to control the false-discovery rate.

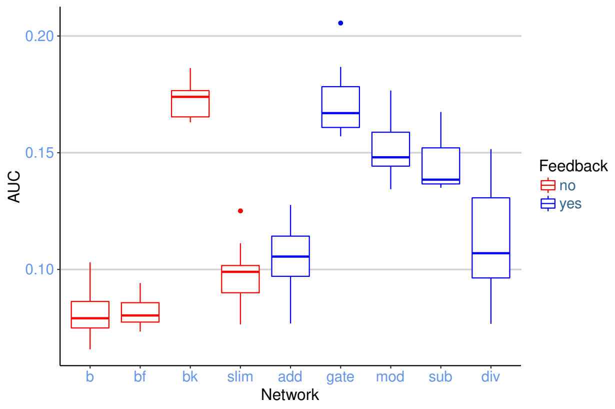

Figure 5: Robustness against salt-and-pepper noise.

Table 3: Differences in robustness between feedback networks and `slim`.

	white noise	salt-and-pepper noise
`add`	$Z = 3.18,\ p < 0.003$	$Z = 1.134,\ p = 0.285$
`gate`	$Z = 3.704,\ p < 0.001$	$Z = 3.780,\ p < 0.001$
`mod`	$Z = 3.629,\ p < 0.001$	$Z = 3.780,\ p < 0.001$
`sub`	$Z = 3.704,\ p < 0.001$	$Z = 3.780,\ p < 0.001$
`div`	$Z = 2.721,\ p < 0.008$	$Z = 1.361,\ p = 0.217$

Conclusion

We show that feedback improves the performance of convolutional networks as well as their robustness to noise on multiMNIST. For additive feedback, the performance advantage vanishes when controlling for the depth of the unrolled network and accordingly for the effective receptive field size. This suggests that the ability of feedback networks to integrate information from a larger context over time is an important factor in their performance. It also shows that carefully designed control architectures are crucial to assess whether feedback does in fact improve performance.

We compared different schemes for integrating feedback and feedforward signals and showed that the choice of feedback scheme can have significant impacts on performance and robustness. In this study, modulating feedback inspired by biased competition and subtractive feedback inspired by predictive coding offered the best performance and robustness. These results can guide future investigations into neural networks with feedback.

We designed the architectures to make them as comparable to each other as possible. For example, the subtractive and divisive networks included local response normalization (just like all others), even though this may be seen as a departure from the underlying principle of predictive coding. Testing such variants, as well as more elaborate variants of the other feedback schemes, goes beyond the scope of this paper and is left for future work.

In addition, the comparison of feedback architectures (both in performance and robustness) was exploratory, as we did not have initial hypotheses as to which architectures would perform best. Accordingly, the results presented here should be interpreted with care: while we conclude that the type of feedback plays an important role, further work is necessary, for example, replication of these results on more datasets. Furthermore, analysing the representations and computations in different feedback networks may help to understand why some forms of feedback perform better than others.

Incorporating feedback in deep networks and understanding how to best combine forward and feedback signals promises to help build better and more robust deep networks.

References

Brosch, T., & Neumann, H. (2014). Computing with a canonical neural circuits model with pool normalization and modulating feedback. *Neural Computation*, *26*, 2735–2789.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (pp. 249–256).

Kravitz, D. J., Saleem, K. S., Baker, C. I., Ungerleider, L. G., & Mishkin, M. (2013). The ventral visual pathway: an expanded neural framework for the processing of object quality. *Trends in Cognitive Sciences*, *17*(1), 26–49.

Kriegeskorte, N. (2015). Deep Neural Networks: A New Framework for Modeling Biological Vision and Brain Information Processing. *Annual Review of Vision Science*, *1*(1), 417–446.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Liao, Q., & Poggio, T. (2016). Bridging the Gaps Between Residual Learning, Recurrent Neural Networks and Visual Cortex. *arXiv:1604.03640 [cs]*.

Lotter, W., Kreiman, G., & Cox, D. (2016). Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning. *arXiv:1605.08104 [cs, q-bio]*.

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic Routing Between Capsules. In I. Guyon et al. (Eds.), *Advances in Neural Information Processing Systems 30* (pp. 3856–3866). Curran Associates, Inc.

Spoerer, C. J., McClure, P., & Kriegeskorte, N. (2017). Recurrent Convolutional Neural Networks: A Better Model of Biological Object Recognition. *Frontiers in Psychology*, *8*.

Spratling, M. W. (2008). Predictive coding as a model of biased competition in visual attention. *Vision Research*, *48*(12), 1391–1408.

Zamir, A. R., Wu, T.-L., Sun, L., Shen, W., Malik, J., & Savarese, S. (2016). Feedback Networks. *arXiv:1612.09508 [cs]*.