A Biologically Inspired Visual Working Memory for Deep Networks

Ethan Harris

Dr Jonathon Hare Professor Mahesan Niranjan

Vision Learning and Control Group
Department of Electronics and Computer Science
University of Southampton

https://arxiv.org/abs/1901.03665
https://github.com/ethanwharris/STAWM

March 20, 2019

Neural Networks and the Brain

- Neural networks don't look like the brain
- Or do they?
- We need to be clear with how we characterise the brain and neural networks
- Should they be similar?

The Scope of Neuroscience

- Neuroscience can be used to mean many things
- Best described as a spectrum, from cellular dynamics, to neuroanatomy, to psychology
- What are we interested in? Psychophysics, applies across the spectrum, concerned with understanding our perception of the world around us

In This Lecture

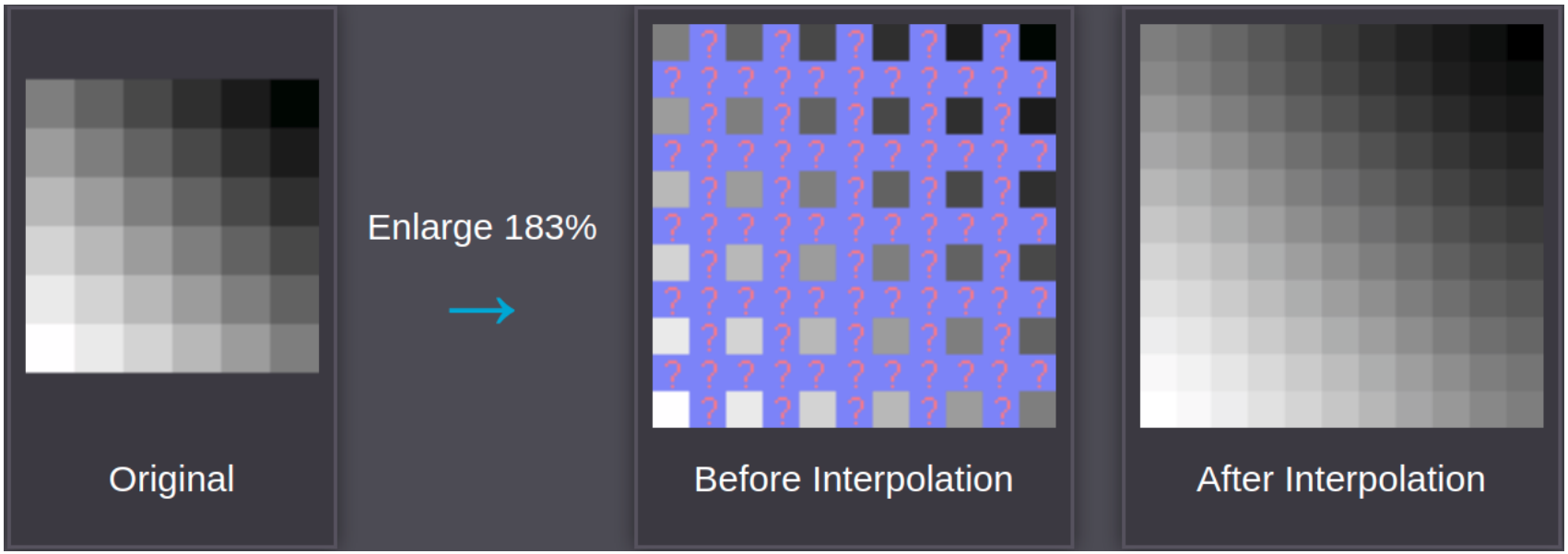
- We will look at a particular class of deep network which tries to model visual attention
- We will consider relevant concepts from the science of perception to bring these models closer to visual processing in nature
- Through experimentation we will demonstrate the viability of this alternative approach to deep learning research

An Attention Primer

- In a deep network we can construct layers which, rather than process a particular signal, alter or warp it in some fashion to focus on the relevant information
- We call this attention
- Our function needs to be differentiable (almost everywhere, non-zero at least somewhere) for this to work

Interpolation (an aside)

- Estimate value by considering its location among known points
- Can take the nearest neighbour or use a linear estimate (bilinear for 2D data like images)

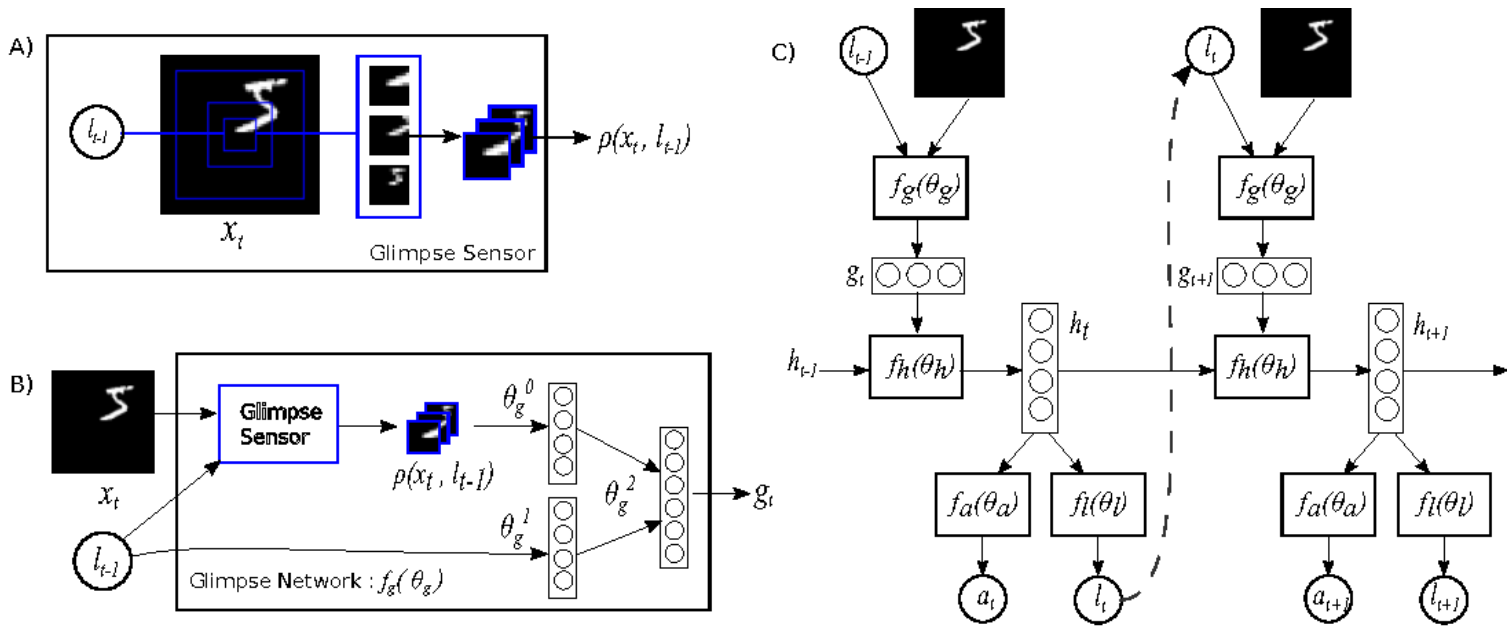


Spatial Transformer Networks (Jaderberg et al., 2015)

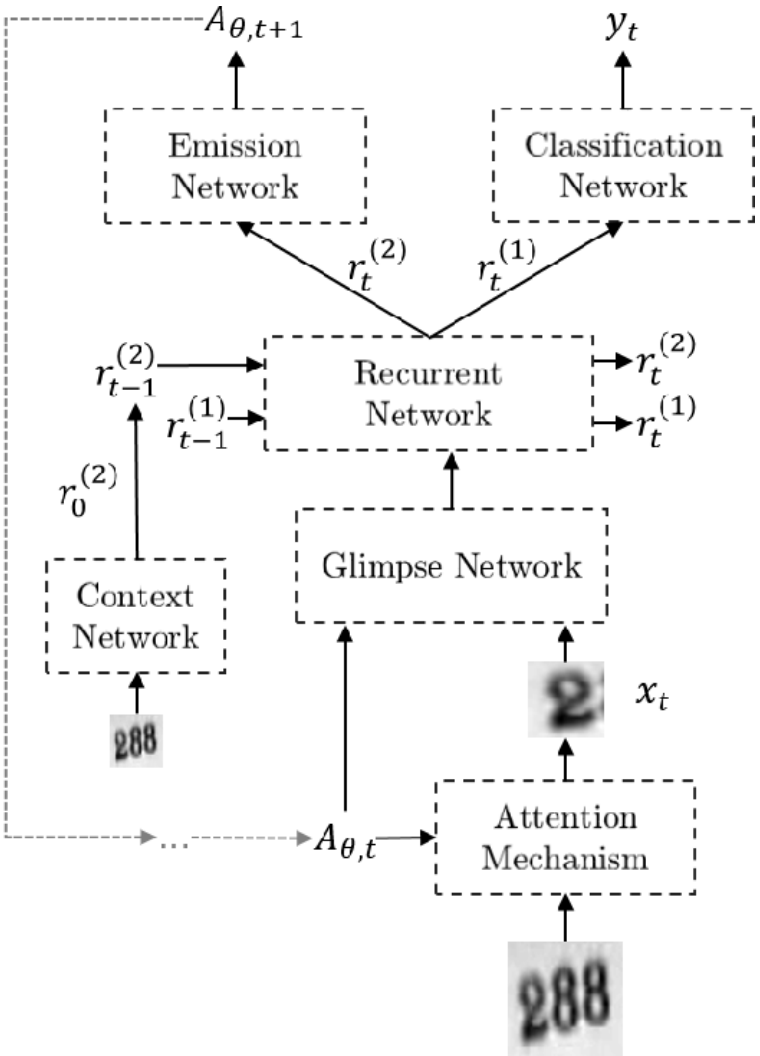
- Interpolation schemes are differentiable
- Bilinear is just a linear combination of stuff, Nearest Neighbour is piece-wise differentiable (the gradient is 1 for each pixel I choose and 0 elsewhere; think dropout)
- We can differentiably make a flow field from 6 affine co-ordinates (other such transforms are available), giving a network the ability to learn to spatially transform an image based on some observations
- No explicit supervision is required to learn the attention policy, just backprop through the whole network

Recurrent Models of Visual Attention (Mnih et al., 2014)

- RNNs are a thing (and sometimes they work!)
- Combine them with an attention mechanism and you have a model that can process an image through a series of glimpses
- Called a Recurrent Attention Model (RAM)



Enriched DRAM (Ablavatski et al., 2017)



- Original RAMs used non-differentiable hard attention to pixels and had to be trained with reinforcement learning
- Enriched DRAM replaces this with a spatial transformer, now we don't need RL!

Attention in Nature

- When Humans attend to a visual scene we rely on our memory (**not** an RNN)
- Every time we glimpse at something, this triggers a short term change to the strength (efficacy) of connections (synapses) in the brain
- These changes help to alter our perception of future glimpses for a short period of time

Plasticity

- In 'The Organization of Behavior' Hebb (1949) described an observable learning procedure in neurons which became known as Hebb's postulate:
- 'When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.'
- Unsupervised learning from inputs based on associations between them
- Held to be the basis of learning and memory within the brain

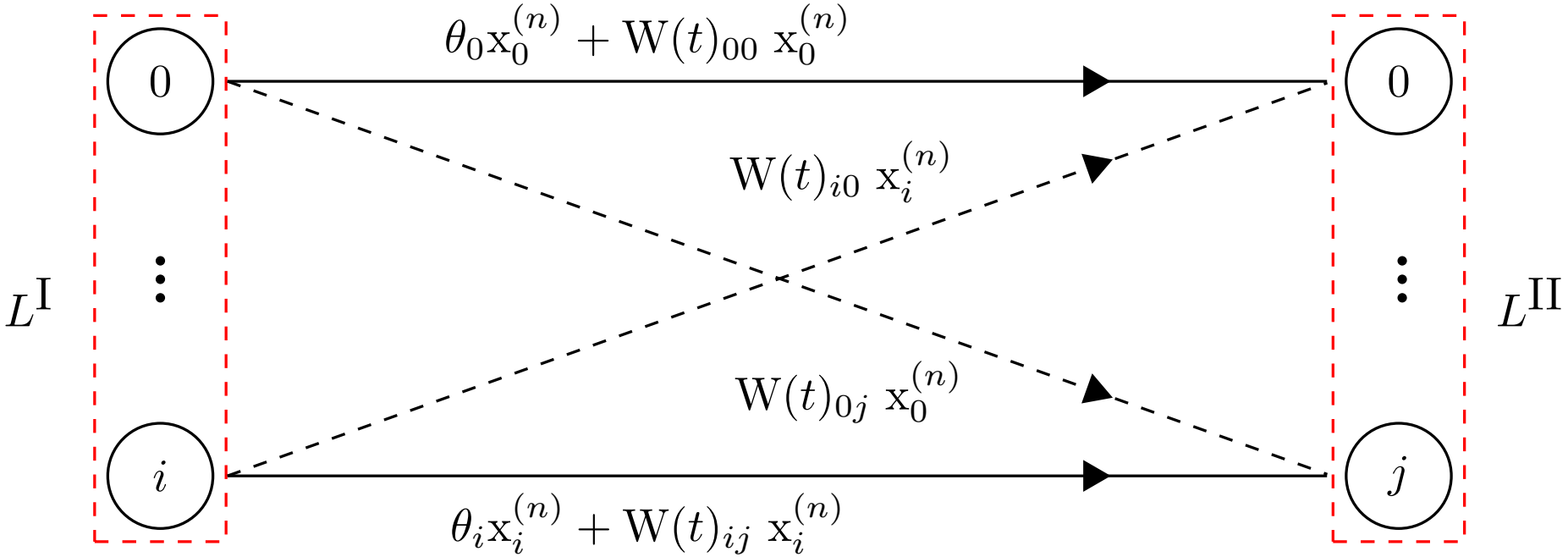
A Differentiable Memory Mechanism

- Suppose we want to model plasticity in the brain
- We can use Hebbs rule 'Neurons that fire together wire together'
- Rosenblatt had some (continuous time, discrete activation) stuff that did this (Rosenblatt, 1962)
- Can we make a discrete time, continuous activation relaxation of it?

The Hebb-Rosenblatt Memory

- Learning rule based on Rosenblatts perceptron

$$\mathbf{W}(t+\Delta t) = \mathbf{W}(t) + \Delta t\, \mathrm{diag}(\boldsymbol{\eta})(\mathbf{y}^{(n)}(t)\mathbf{x}^{(n)\top}) - \Delta t\, \mathrm{diag}(\boldsymbol{\delta})\mathbf{W}(t) \quad (1)$$

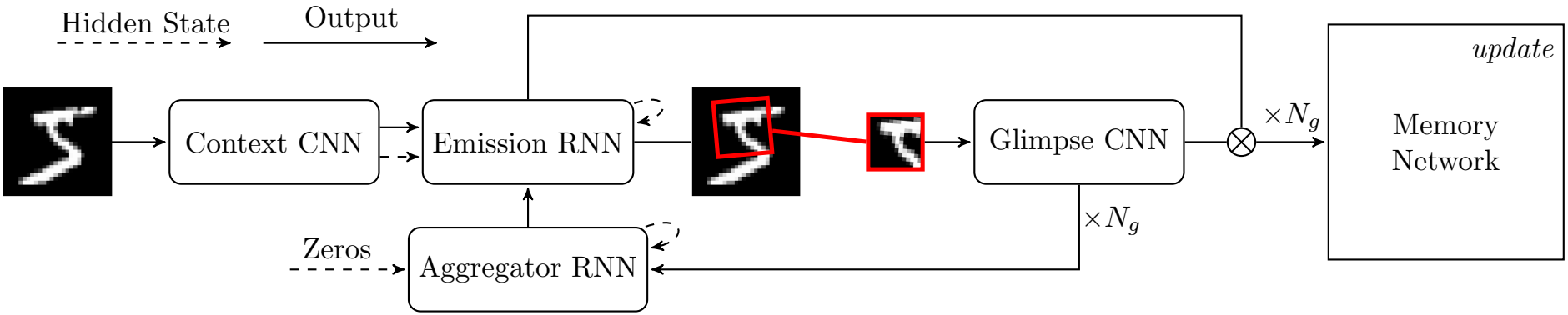


Hebbian Learning vs Backprop (an aside)

- Backpropagation is used to update weights on a backward pass according to the expected outputs
- With Hebbian learning we only make observations about our inputs, looking for structure in the data from the ground up
- This has a strong connection to component analysis (e.g. Oja's rule (Oja, 1982))

STAWM: Short-Term Attentive Working Memory

- Combine our Hebb-Rosenblatt memory with the EDRAM from before
- Minor changes to the 'what', 'where' pathways

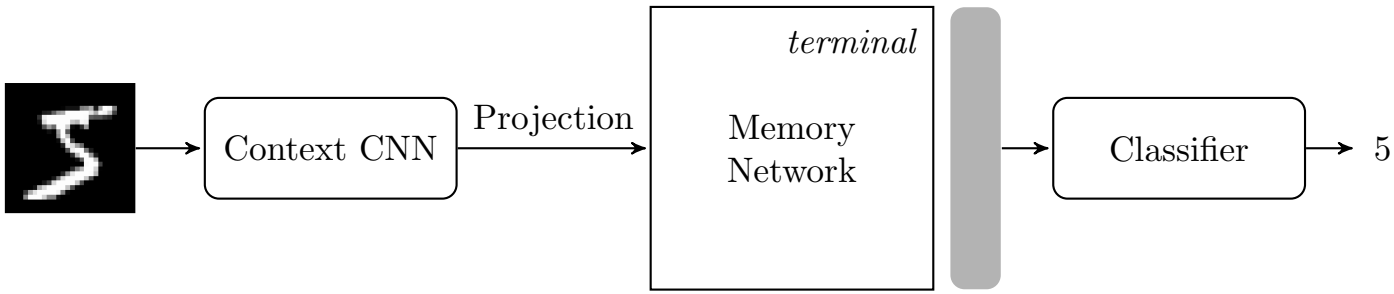


Using the Memory

- Now we have a memory, how do we use it?
- Need some kind of query signal to project through the weights
- For this we use the output of the context CNN followed by a single linear layer
- Once we have this representation we can attach networks that use it in different ways

Classification

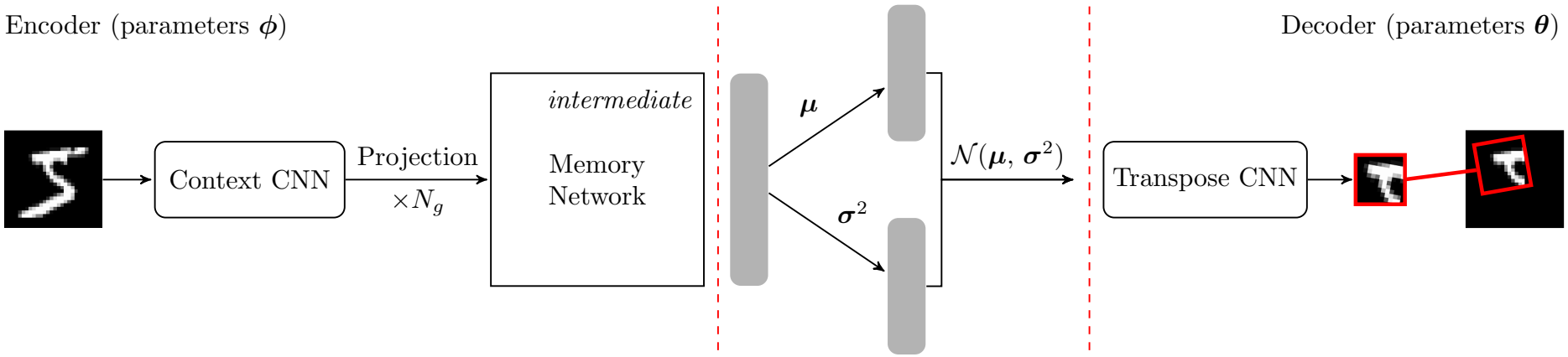
- Classification is easy, just add some linear layers



- What about something more complicated? Maybe even recurrent?

Drawing

- For drawing we can use transpose convolutions to output a sketch after each glimpse



- We just need a way to combine the sketches

Alpha Compositing *over* operator (an aside)

- We can use some old school computer graphics
- Sequentially (and differentiably) combine images using an alpha mask

$$\begin{aligned} \mathbf{C}_N =& \mathbf{C}_{N-1} \odot (1 - \mathbf{B}) + t_{\mathbf{A}^{-1}}(\sigma(\mathbf{U}_N)) \odot \mathbf{B} \ , \\ & \mathbf{B} \sim \mathit{Concrete}(t_{\mathbf{A}^{-1}}(\sigma(\mathbf{P})), \tau) \end{aligned} \tag{2}$$

- $\mathbf{C}_0 \in \{0\}^{H_i \times W_i}$, $t_{\mathbf{A}^{-1}}$ is an inverse glimpse transform, σ is the sigmoid function, $\mathbf{U}_N$ is a sketch or update and $\mathbf{P}$ are the alpha probabilities

Classification Experiments

- Supervised classification performance on MNIST (LeCun, 1998)

Model	Error
RAM, $S_g = 8$, $N_g = 5$	1.34%
RAM, $S_g = 8$, $N_g = 6$	1.12%
RAM, $S_g = 8$, $N_g = 7$	1.07%
STAWM, $S_g = 8$, $N_g = 8$	$0.41\%_{\pm 0.03}$
STAWM, $S_g = 28$, $N_g = 10$	$0.35\%_{\pm 0.02}$

Classification Experiments

- Self-supervised classification performance on MNIST

Model	Error
DBM, Dropout (Srivastava et al., 2014)	0.79%
Adversarial (Goodfellow et al., 2014)	0.78%
Virtual Adversarial (Miyato et al., 2015)	0.64%
Ladder (Rasmus et al., 2015)	0.57%
STAWM, $S_g = 6$, $N_g = 12$	0.77%

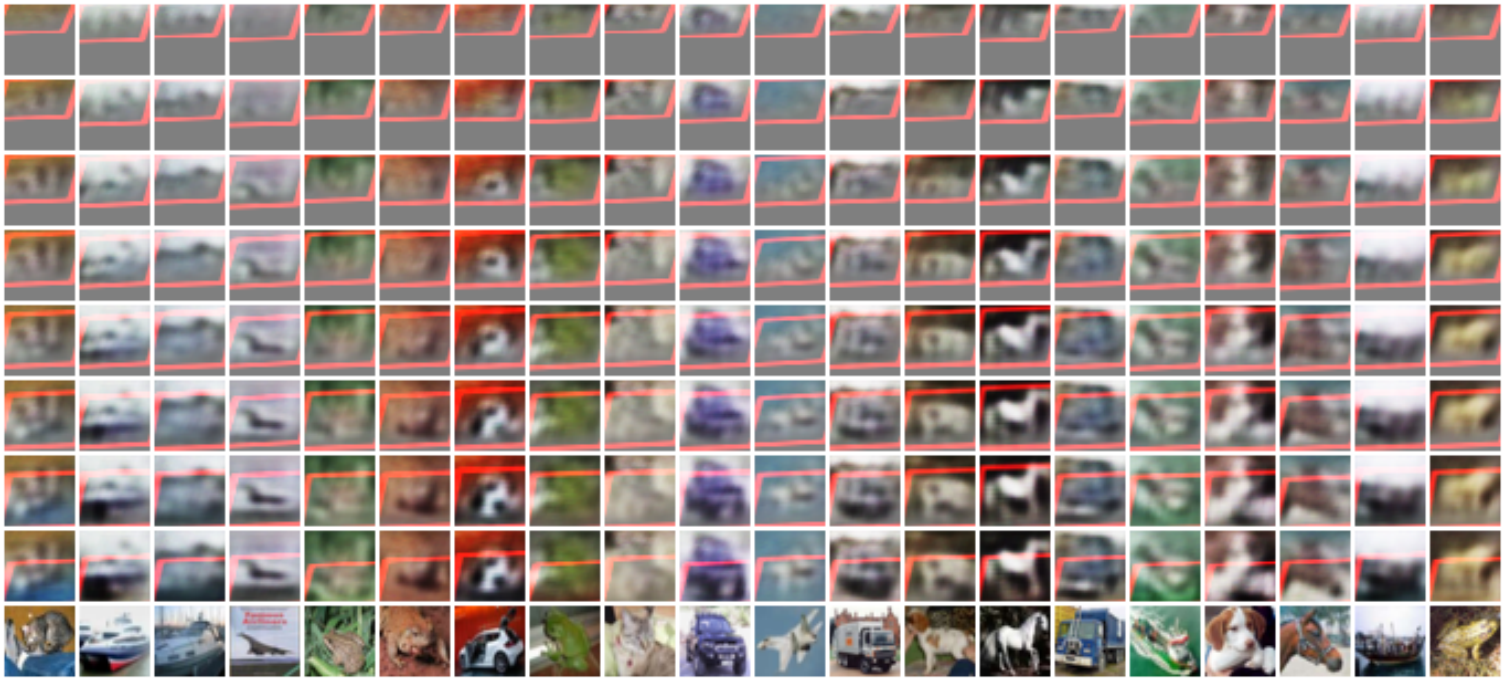
Classification Experiments

- Self-supervised classification performance on CIFAR-10 (Krizhevsky and Hinton, 2009)

Model	Error
Baseline β-VAE (Higgins et al., 2016)	$63.44\%_{\pm 0.31}$
STAWM, $S_g = 16$, $N_g = 8$	$55.40\%_{\pm 0.63}$

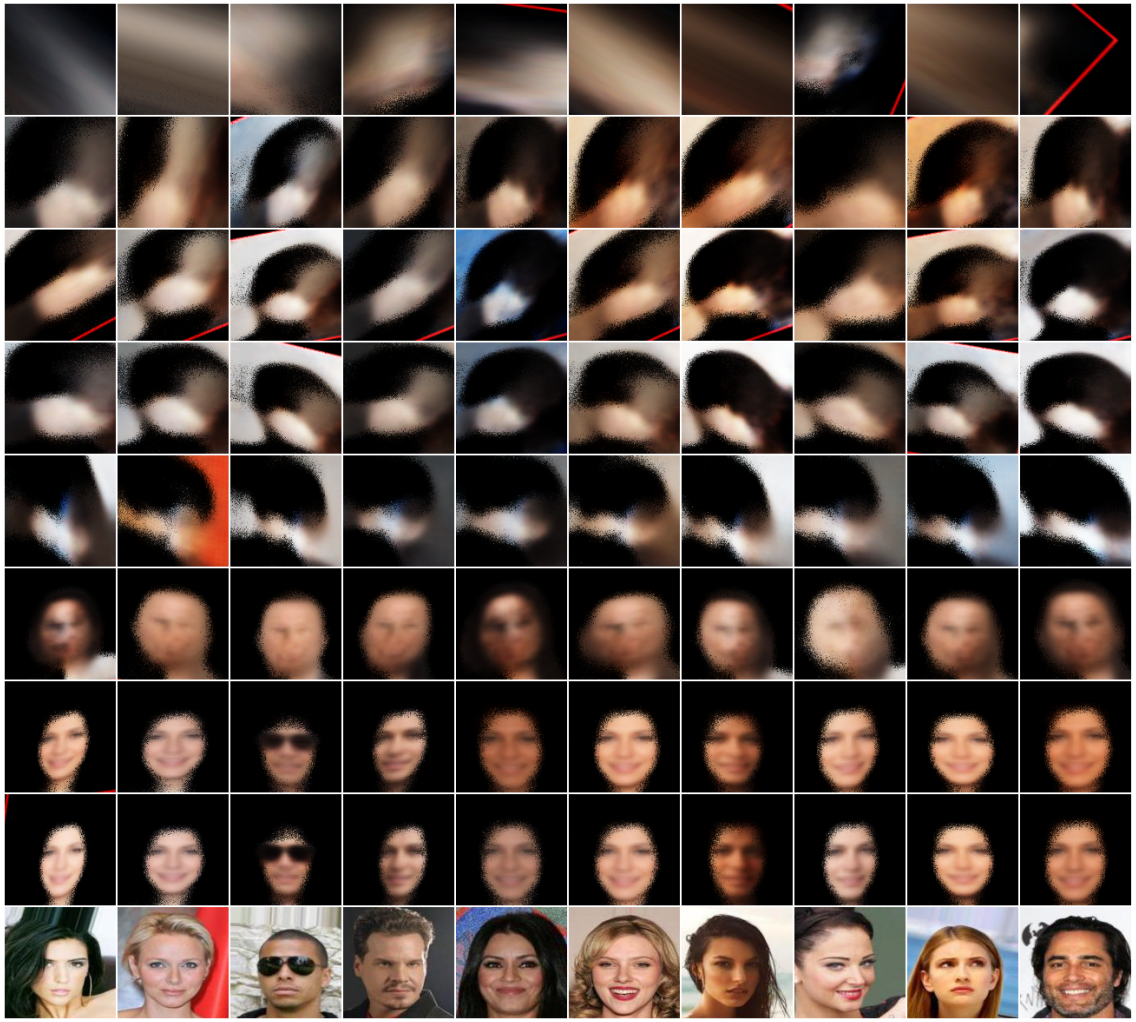
Painting CIFAR-10 (Krizhevsky and Hinton, 2009)

- We can view the update sequence for the drawing model

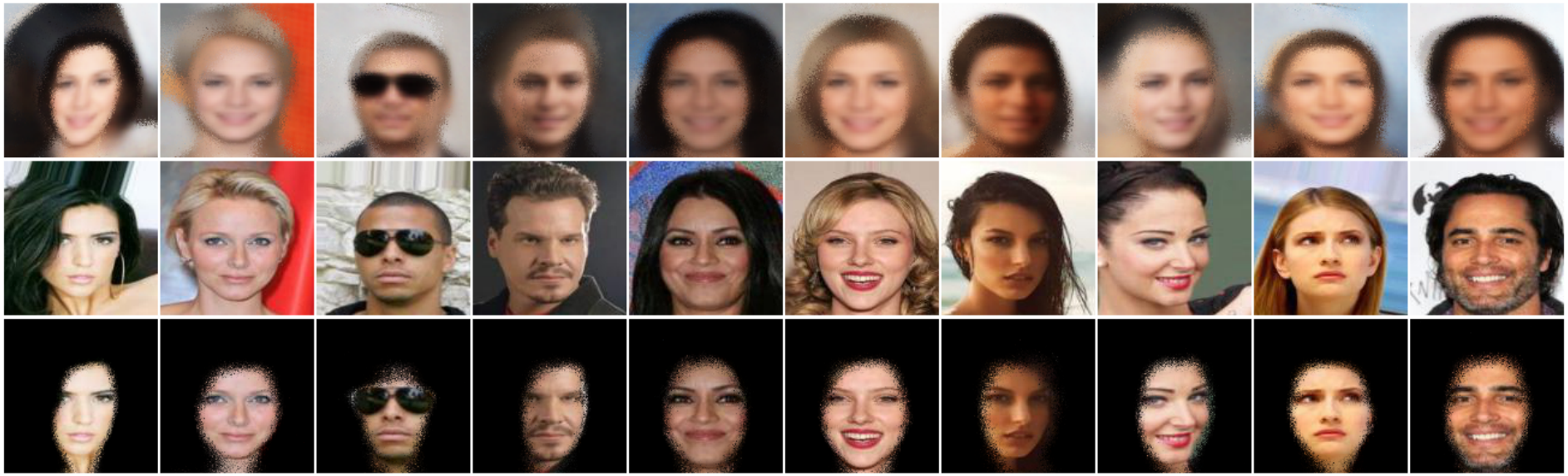


- What if the update sequence becomes interesting?

CelebA (Liu et al., 2015) Emergent Segmentation



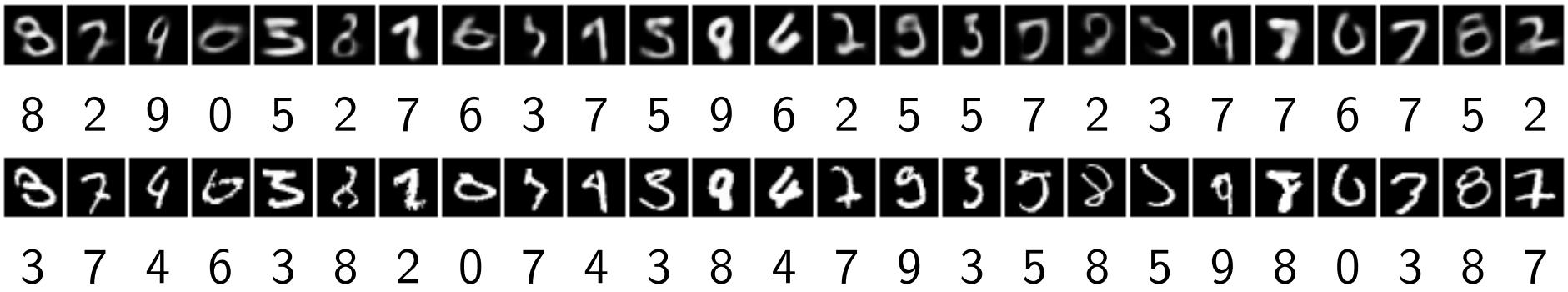
CelebA (Liu et al., 2015) Emergent Segmentation



Top: the drawn results. Middle: the target images. Bottom: the learned mask from the last glimpse, pointwise multiplied with the target image.

Interpretable Classifiers

- What if we allow STAWM to use multiple query vectors through the memory to perform a series of different tasks like classification and drawing?



Top: sketchpad results and associated predictions for a sample of misclassifications. Bottom: associated input images and target classes.

Summary / Conclusions

- We constructed a model which has a memory much more similar to that found in nature
- We used that model to perform tasks such as drawing in a way that is much more natural for humans
- Not only does this model work much better than previous attentional models it also has some emergent properties
- STAWM learns the structure in images of faces from CelebA for segmentation without supervision and learns to draw and classify in tandem so that we (humans) can view and understand its mistakes
- Should neural networks look more like the brain? We think so

References

Ablavatski, A., Lu, S., and Cai, J. (2017). Enriched deep recurrent visual attention model for multiple object recognition. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 971–978. IEEE.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Hebb, D. O. (1949). *The organization of behavior: A neuropsychological theory*. New York: Wiley.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2016). beta-vae: Learning basic visual concepts with a constrained variational framework.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images.

LeCun, Y. (1998). The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

Miyato, T., Maeda, S., Koyama, M., Nakae, K., and Ishii, S. (2015). Distributional Smoothing with Virtual Adversarial Training. *stat*, 1050:2.

Mnih, V., Heess, N., Graves, A., et al. (2014). Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212. IEEE.

Oja, E. (1982). Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273.

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. (2015). Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554.

Rosenblatt, F. (1962). *Principles of neurodynamics*. Spartan Books.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.