Towards Biologically Plausible Convolutional Networks

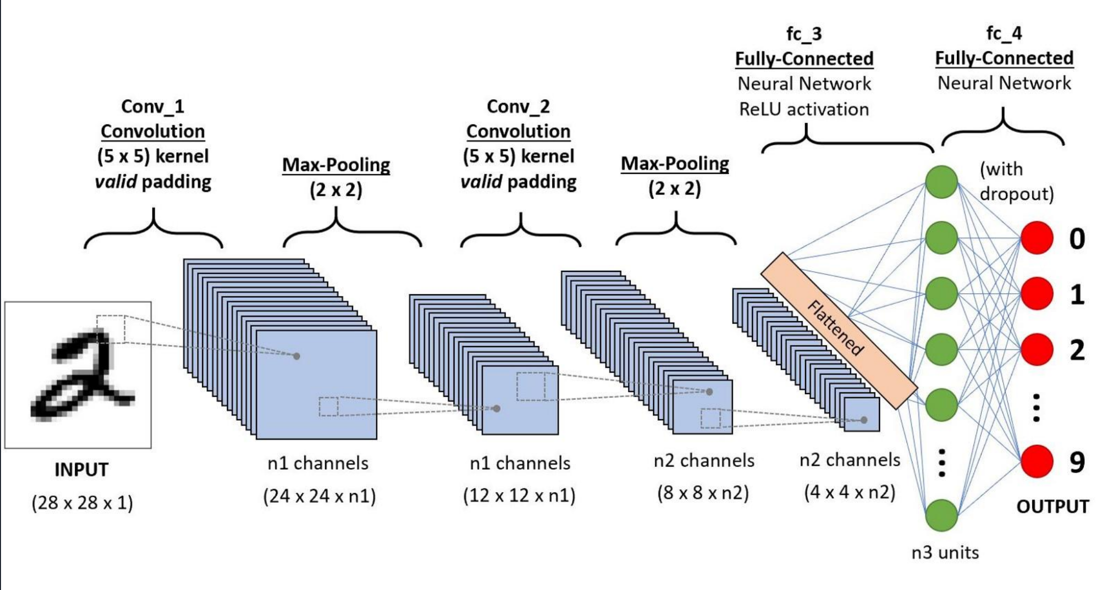
Authors:
Roman Pogodin
Yash Mehta
Timothy P. Lillicrap
Peter E. Latham – Gatsby Unit, UCL

Presentation by:
Ali Abolhassanzadeh Mahani

What are Convolutional Neural Networks?

A convolutional neural network, or CNN, is a deep learning neural network designed for processing structured arrays of data such as images. Convolutional neural networks are widely used in computer vision and have become the state of the art for many visual applications such as image classification, and have also found success in natural language processing for text classification.

– Thomas Wood, DeepAI.org



Why use CNN?

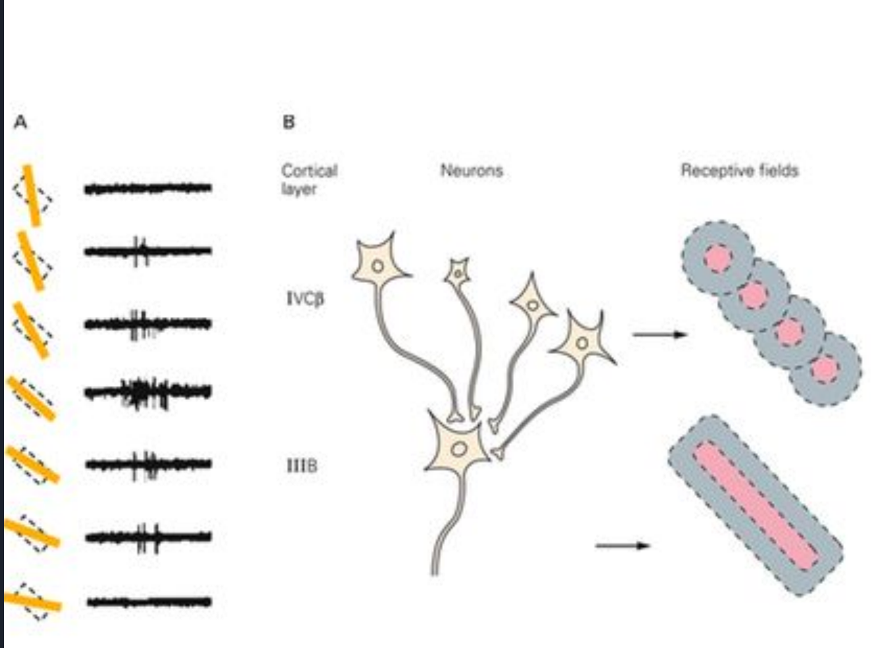
- Reduce # of parameters through “Weight Sharing”
- Reduce training time
- Increase accuracy

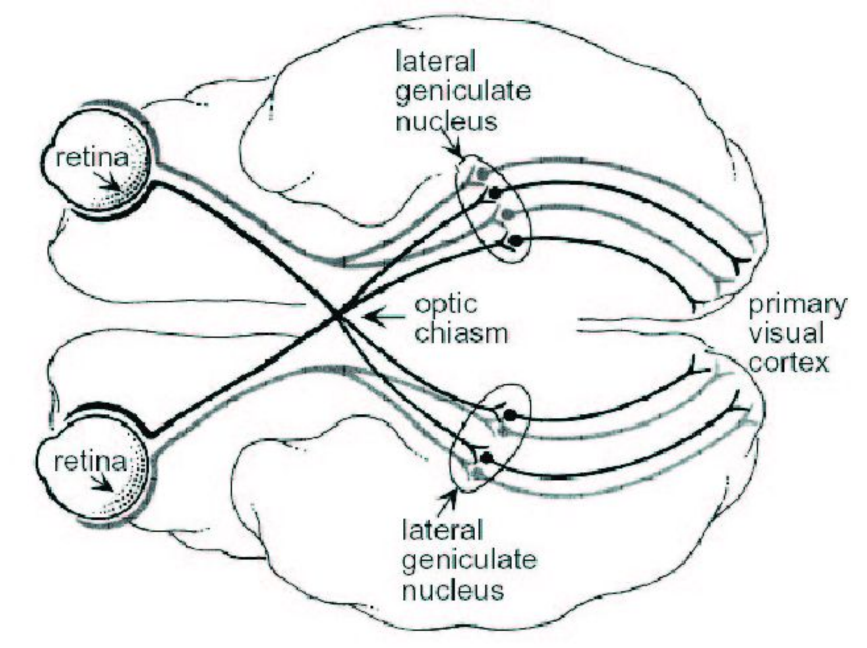
Where to use CNN?

- Visual Domain (Image classification, segmentation, transformation)
- Speech Recognition
- Text classification
- Time series classification

CNN and the Visual Cortex

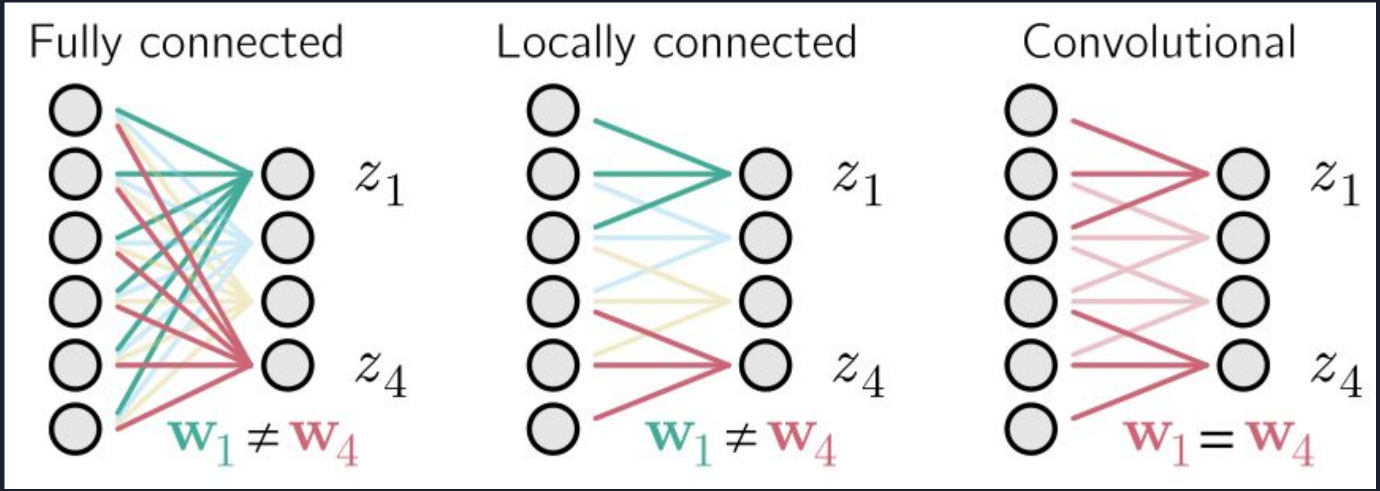
CNNs were inspired by the receptive fields and the neuronal connections in the Primary Visual Cortex (V1) and the Lateral Geniculate Nucleus (LGN)





Regularization in locally connected networks

Convolutional vs. locally connected networks



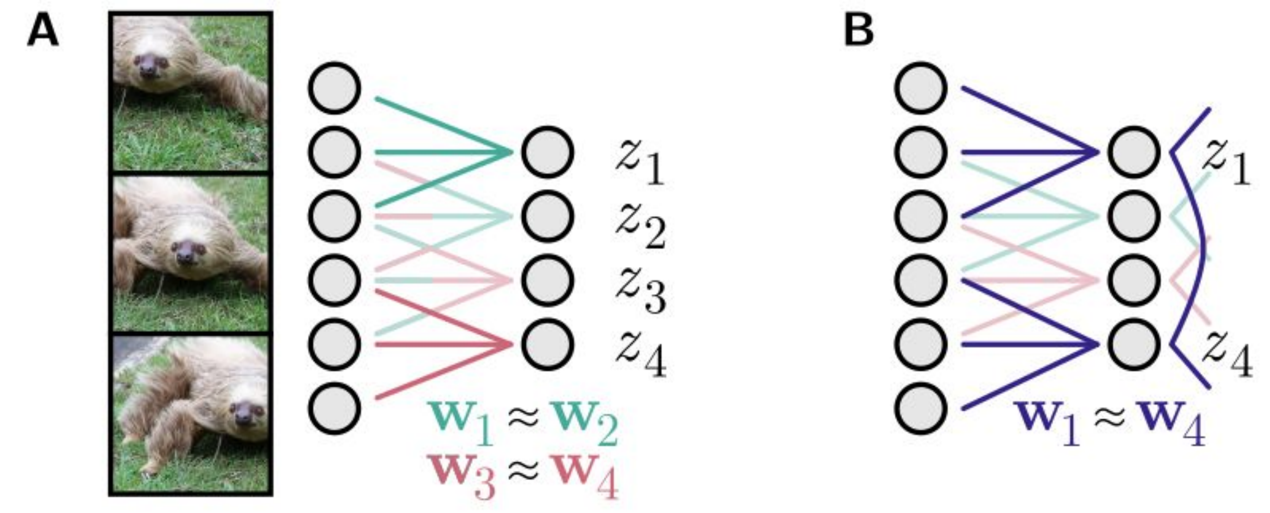
Convolution:

$$z_i = \sum_{j=1}^{N} w_{i-j} x_j$$

Non-Convolution:

$$z_i = \sum_{j=1}^{N} w_{ij} x_j$$

Data augmentation vs. dynamic weight sharing

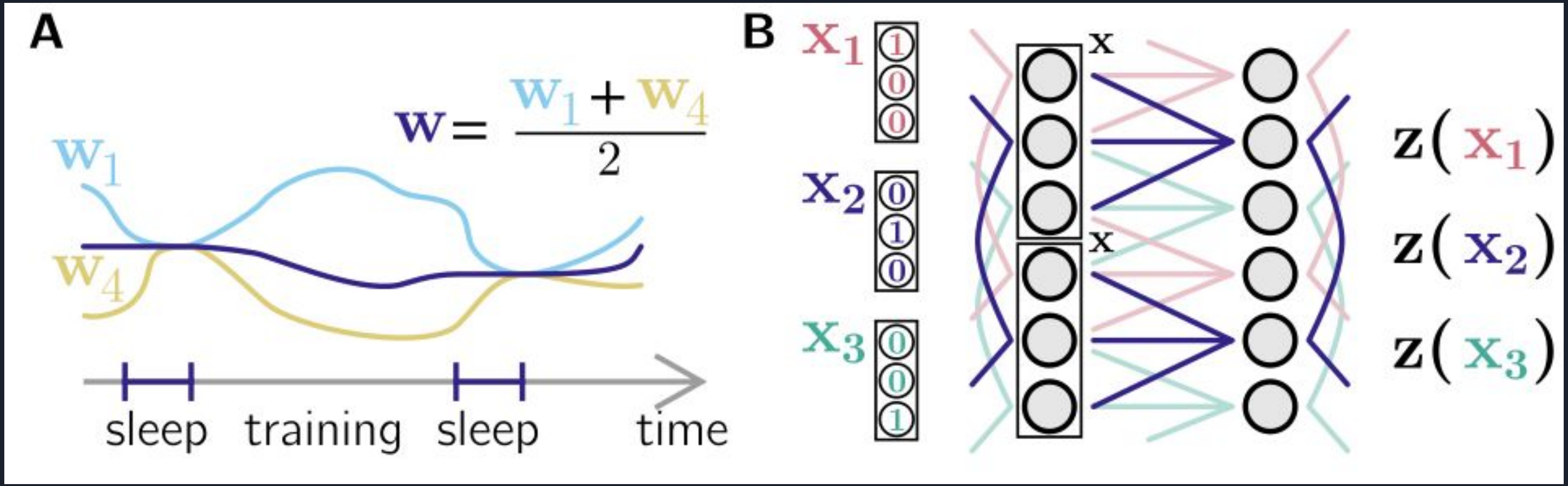


Is it possible for a locally connected network to develop approximately convolutional weights?

$$w_{ij} \approx w_{i-j}$$

A Hebbian Solution to Dynamic Weight Sharing

Normally, the weights of a locally connected network diverge. In order to make them convolutional (i.e. to make them converge) we introduce a an occasional “sleeping” period, during which the weights relax to their mean over output neurons.



How it's done?

$$z_i = \mathbf{w}_i^\intercal \mathbf{x} = \sum_{j=1}^{k} w_{ij} x_j$$

$$\Delta \mathbf{w}_i \propto \left(z_i - \frac{1}{N} \sum_{j=1}^{N} \right) \mathbf{x} - \gamma(\mathbf{w}_i - \mathbf{w}_i^{init})$$

This Hebbian update effectively implements SGD optimization over the sum of $(z_i - z_j)^2$

Learn more about learning rules here

If we have a set of M input vectors x_m, then the covariance matrix can be as follows;

$$\mathbf{C} = \frac{1}{M}\sum_{m} \mathbf{x}_m \mathbf{x}_m^{\mathsf{T}}$$

Then the weight dynamics converges to

$$\mathbf{w}_i^* = (\mathbf{C} + \gamma \mathbf{I})^{-1} \left(\mathbf{C} \frac{1}{N} \sum_{i=1}^{N} \mathbf{w}_i^{init} + \gamma \mathbf{w}_i^{init} \right)$$

For full rank covariance matrix and small regularization hyperparameter, we get

$$\mathbf{w}_i \approx \frac{1}{N} \sum_{i=1}^{N} \mathbf{w}_i^{init}$$

How quickly does it converge?

We plot the -log of signal-to-noise ratio of the weights defined as

$$\mathrm{SNR}_w = \frac{1}{k^2}\sum_{j=1}^{k} \frac{\left(\frac{1}{N}\sum_i (\mathbf{w}_i)_j\right)^2}{\frac{1}{N}\sum_i \left((\mathbf{w}_i)_j - \frac{1}{N}\sum_{i'}(\mathbf{w}_{i'})j\right)^2}$$

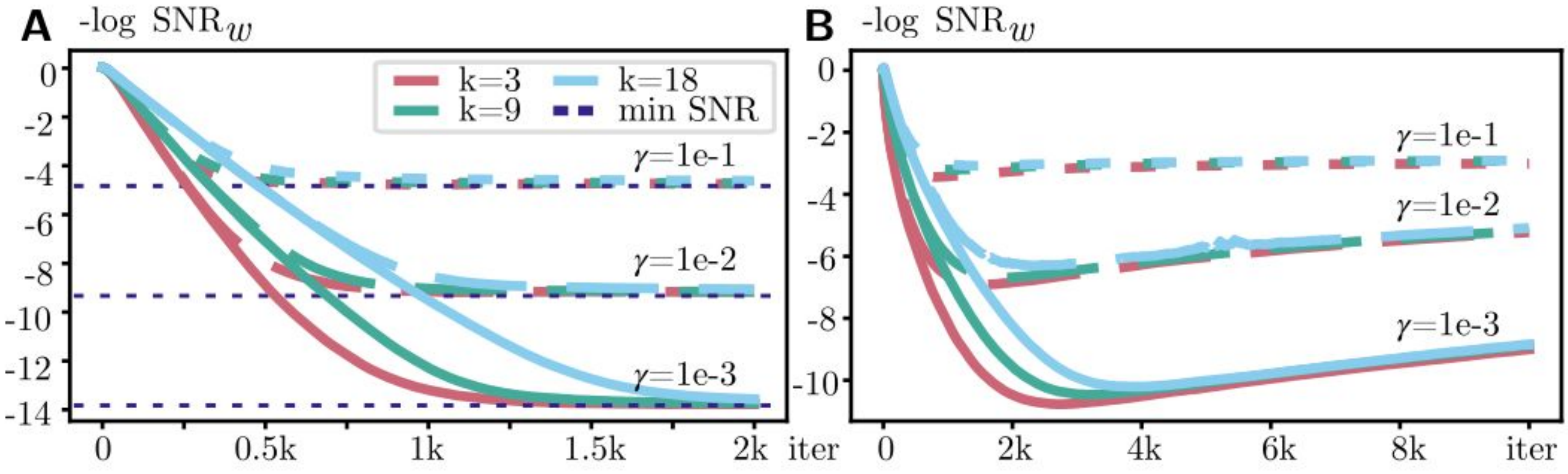


Figure 4: Logarithm of inverse signal-to-noise ratio (mean weight squared over weight variance, see Eq. (6)) for weight sharing objectives in a layer with 100 neurons. **A.** Dynamics of Eq. (4) for different kernel sizes k (meaning k^2 inputs) and γ. Dark dotted lines show the theoretical minimum. **B.** Dynamics of weight update that uses Eq. (9) for $\alpha = 10$, different kernel sizes k and γ. In each iteration, the input is presented for 150 ms.

A realistic model that implements the learning rule

$$\text{firing rate} = r_i = z_i - \frac{1}{N}\sum_{j=1}^{N} z_j \equiv \mathbf{w}_i^\intercal \mathbf{x} - \sum_{j=1}^{N} \mathbf{w}_j^\intercal \mathbf{x}$$

$$\tau \dot{r}_i = -r_i + \mathbf{w}_i^\intercal \mathbf{x} - \alpha r_{inh} + b,$$

$$\tau \dot{r}_{inh} = -r_{inh} + \frac{1}{N}\sum_j r_j + b$$

Thank you for your time :)