Feedback alignment for training neural networks

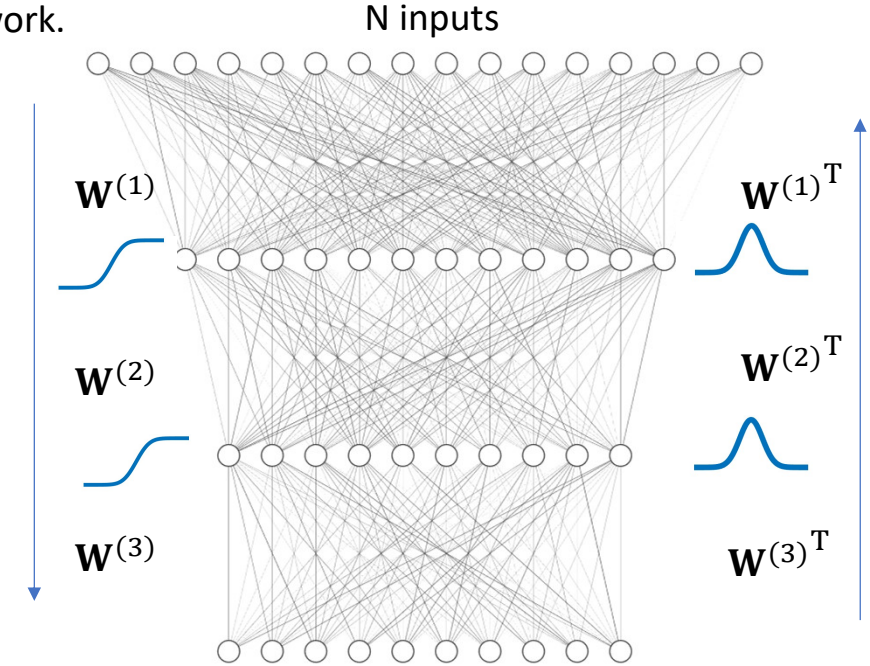
Backpropagation

$\mathbf{h}^{(0)} \in \mathbb{R}^{\mathrm{N}}$ is the input to the network.

Forward computation

For each layer i:

$\mathbf{a}^{(\mathrm{i})} = \mathbf{W}^{(\mathrm{i})}\mathbf{h}^{(\mathrm{i}-1)} + \mathbf{b}^{(\mathrm{i})}$,
$\mathbf{h}^{(\mathrm{i})} = f\left(\mathbf{a}^{(\mathrm{i})}\right)$



Last layer softmax $\hat{\mathbf{y}} = \mathrm{f}_y(\mathbf{a}^{(\mathrm{k})})$
Target label $\mathbf{y}$

Cross entropy loss

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{c=1}^{C} y_c \log \hat{\mathrm{y}}_c$$

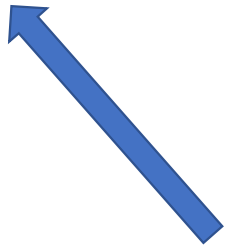
The weight matrices are used in the Computation of the gradients.

$$\boldsymbol{\delta}^{(\mathrm{k}-1)} = \frac{\partial \mathcal{L}(\mathbf{y},\hat{\mathbf{y}})}{\partial \mathbf{a}^{(\mathrm{k})}} = \mathbf{W}^{(\mathrm{k})^{T}} \boldsymbol{\delta}^{(\mathrm{k})} f'(\mathbf{a}^{(\mathrm{k})})$$

$$\boldsymbol{\delta}^{(3)} = \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \hat{\mathbf{y}}} \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{a}^{(3)}} = \hat{\mathbf{y}} - \mathbf{y}$$

Backpropagation biologically plausible?

$$\Delta \mathbf{W}^{(2)^T} \propto \mathbf{W}^{(3)^T} (\hat{\mathbf{y}} - \mathbf{y}) f'(\mathbf{a}^{(3)}) \mathbf{h}^{(1)^T}$$

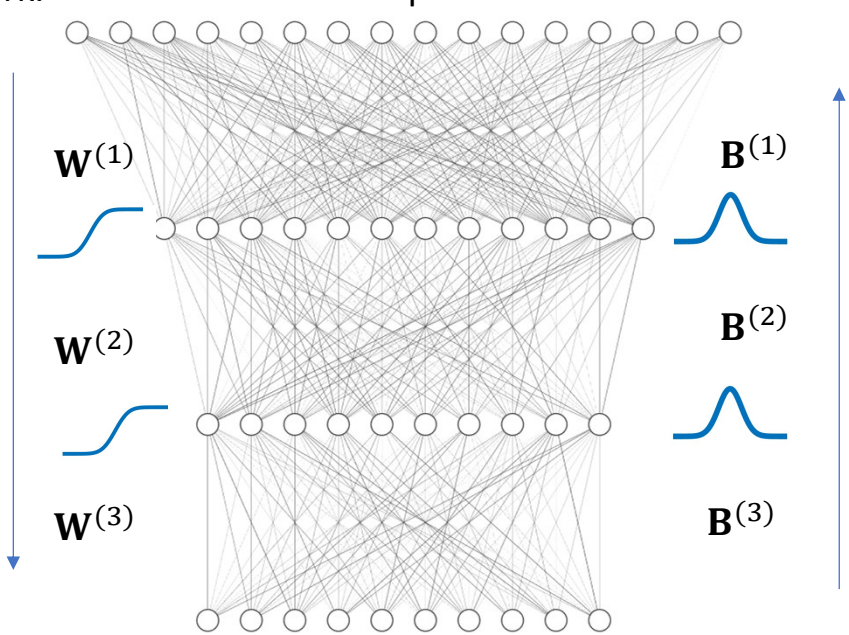


- Requires symmetric weights -> bi-directional synapses. Complete knowledge of downstream weights is necessary.

Feedback alignment

T. P. Lillicrap, D. Cownden, D.B. Tweed, C. J. Akerman, Random feedback weights support learning in deep neural networks

$\mathbf{h}^{(0)} \in \mathbb{R}^{\mathrm{N}}$ is the input to the network.



Forward computation

For each layer i:

$$\mathbf{a}^{(\mathrm{i})} = \mathbf{W}^{(\mathrm{i})}\mathbf{h}^{(\mathrm{i}-1)} + \mathbf{b}^{(\mathrm{i})},$$
$$\mathbf{h}^{(\mathrm{i})} = f\left(\mathbf{a}^{(\mathrm{i})}\right)$$

Random coefficients are used in the feedback path.

$$\boldsymbol{\delta}^{(\mathrm{k}-1)} = \frac{\partial \mathcal{L}(\mathbf{y},\hat{\mathbf{y}})}{\partial \mathbf{a}^{(\mathrm{k})}} = \mathbf{B}^{(\mathrm{k})}\boldsymbol{\delta}^{(\mathrm{k})} f'(\mathbf{a}^{(\mathrm{k})})$$

$$\boldsymbol{\delta}^{(\mathrm{k})} = \frac{\partial \mathcal{L}(\mathbf{y},\hat{\mathbf{y}})}{\partial \hat{\mathbf{y}}}\frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{a}^{(\mathrm{k})}} = \hat{\mathbf{y}} - \mathbf{y}$$

Last layer softmax $\hat{\mathbf{y}} = \mathrm{f}_y(\mathbf{a}^{(\mathrm{k})})$

Target label $\mathbf{y}$

Cross entropy loss

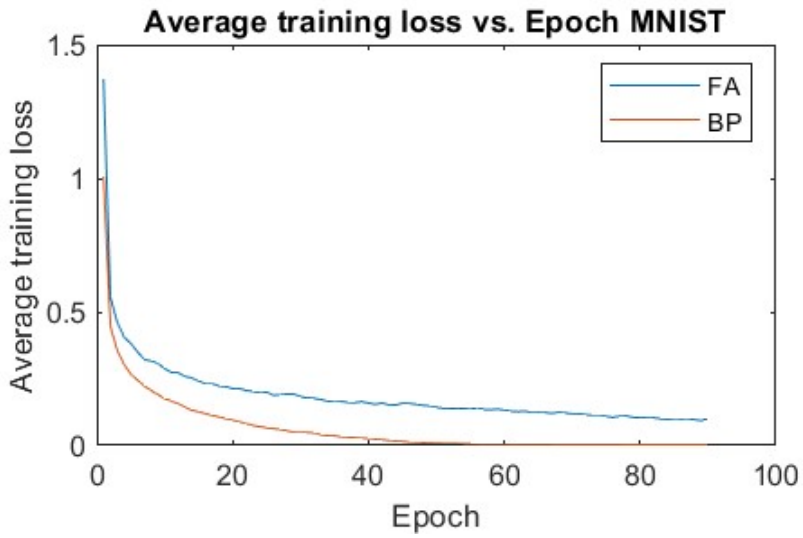
$$\mathcal{L}(\mathbf{y},\hat{\mathbf{y}}) = -\sum_{c=1}^{C} y_c \log \hat{\mathrm{y}}_c$$

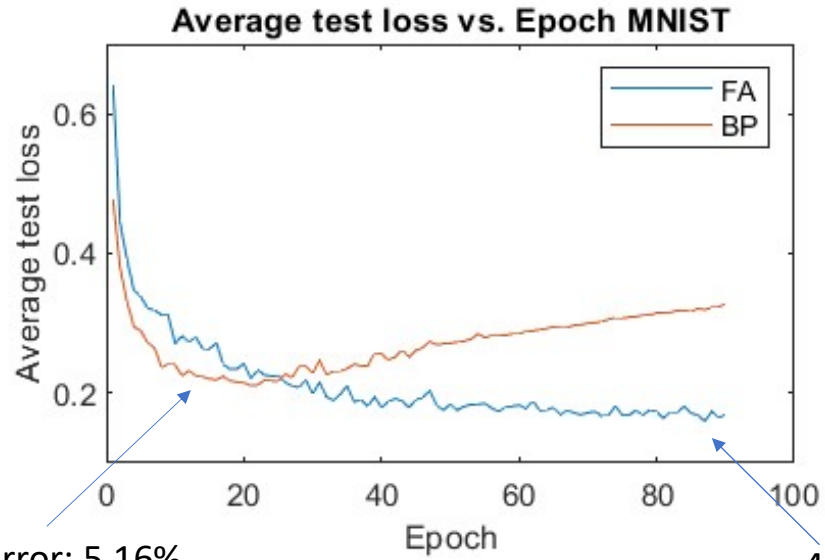
Feedback alignment questions

- To which solutions does it converge and why?
- How does their performance compare to back propagation?
- Can it somehow prevent over-fitting?
- …

Results on MNIST

-200-80-10 Architecture, tanh units hidden layers, softmax output
-Networks randomly initialized at the same weights in both experiments
-random normal feedback weights N(0, 1)
-Learning rate 0.01





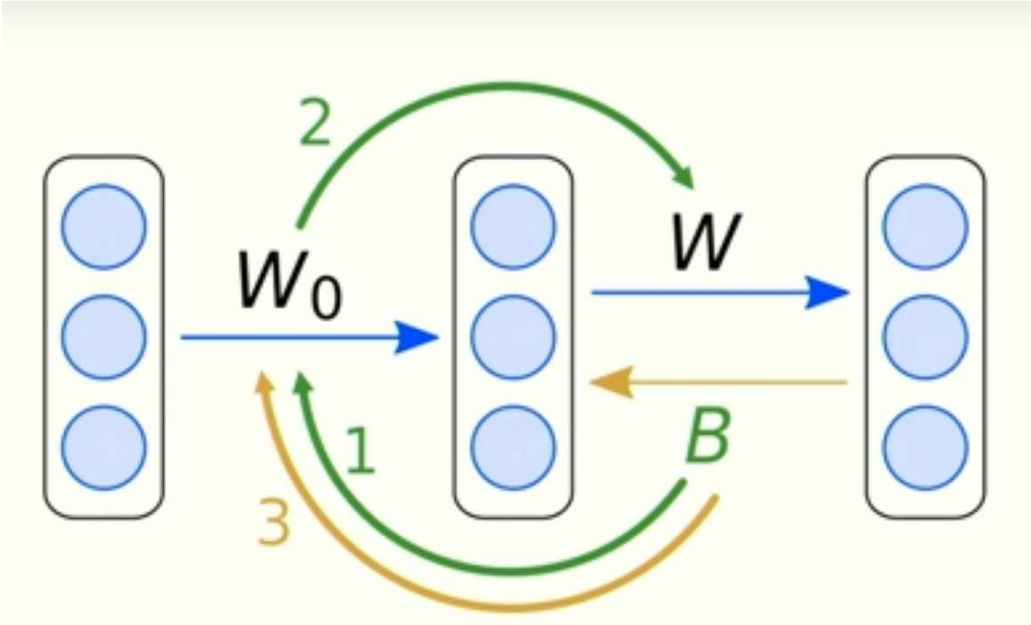
Test error: 5.16%

4.61%

Here feedback alignment mitigates over-fitting

Why would it work?

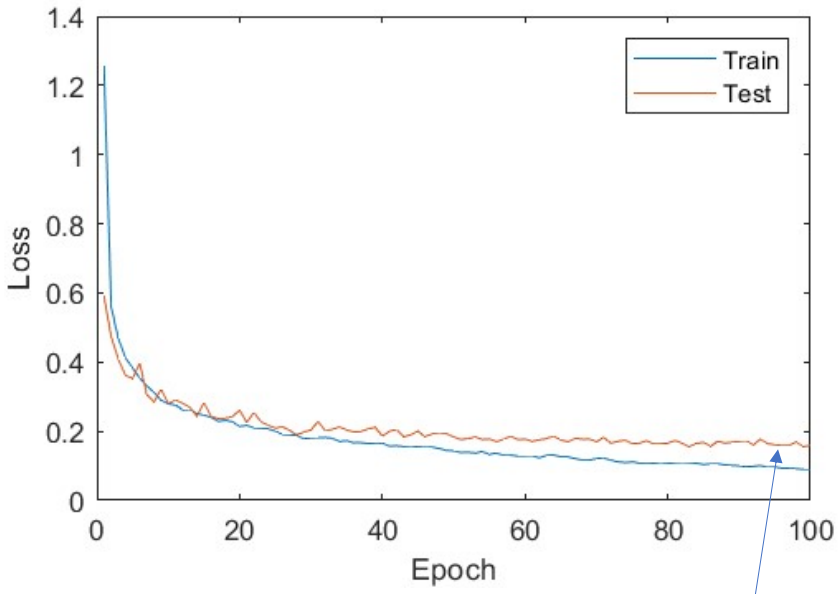
- *Alignment* of the forward weights with the *feedback* weights.



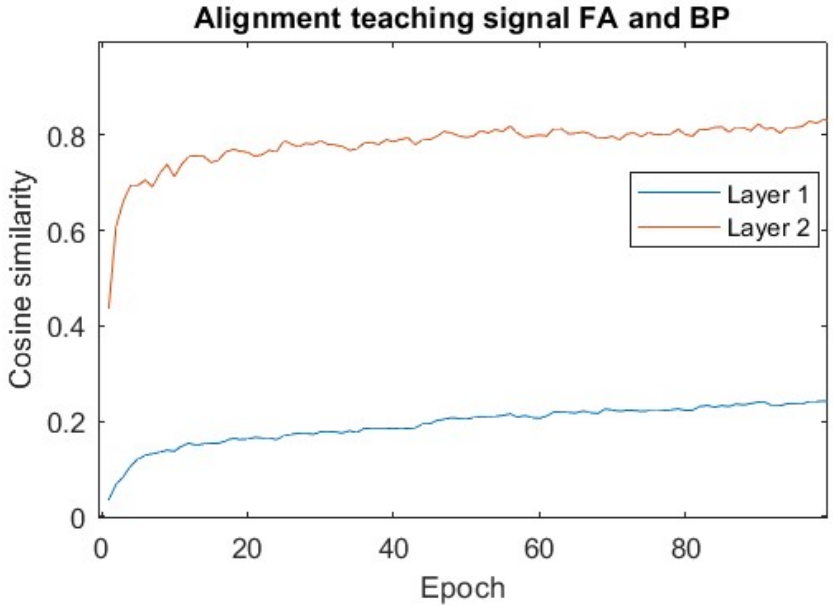
Lillicrap, Cownden, Tweed, Akerman, *Nat Comm. 2016*

Results MNIST Feedback Alignment

-200-80-10 Architecture, tanh units hidden layers, softmax output
-Random normal feedback weights N(0, 1)
-Learning rate 0.01

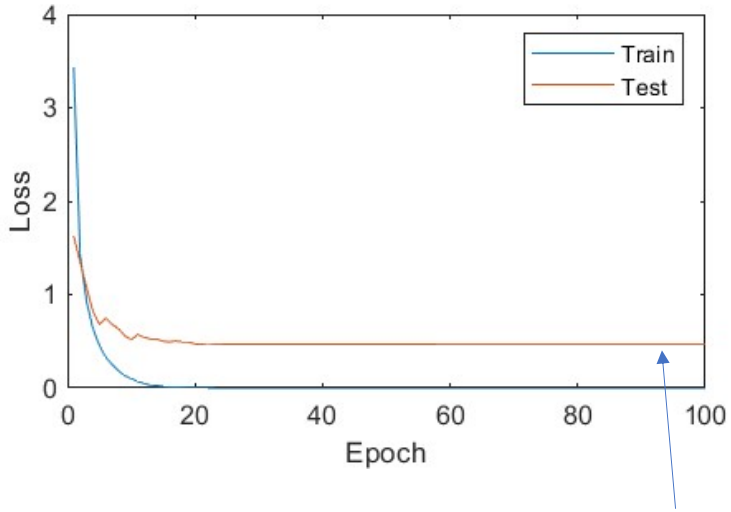


Test error: 4.41%



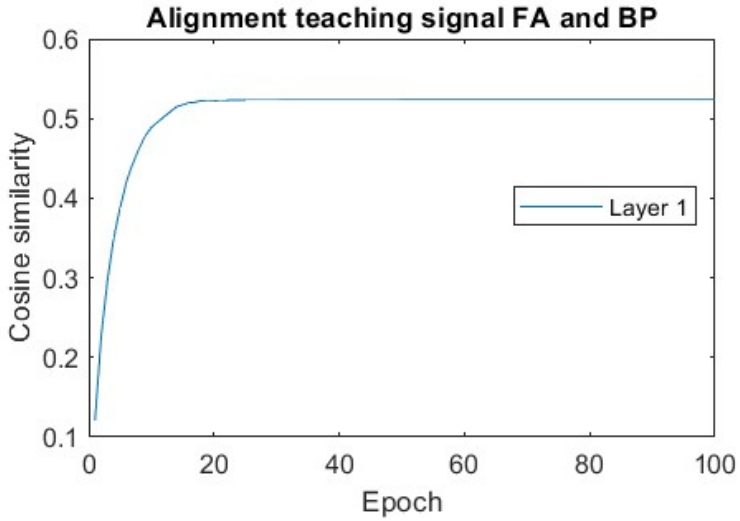
Epoch average alignment of neuron teaching signals per layer

-1000-10 Architecture, tanh units hidden layers, softmax output
-Random normal feedback weights N(0, 1)
-Learning rate 0.01

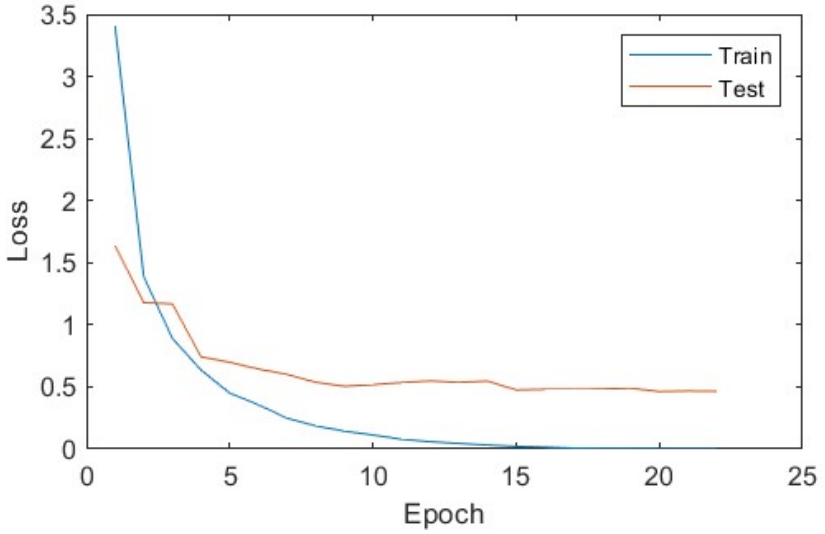


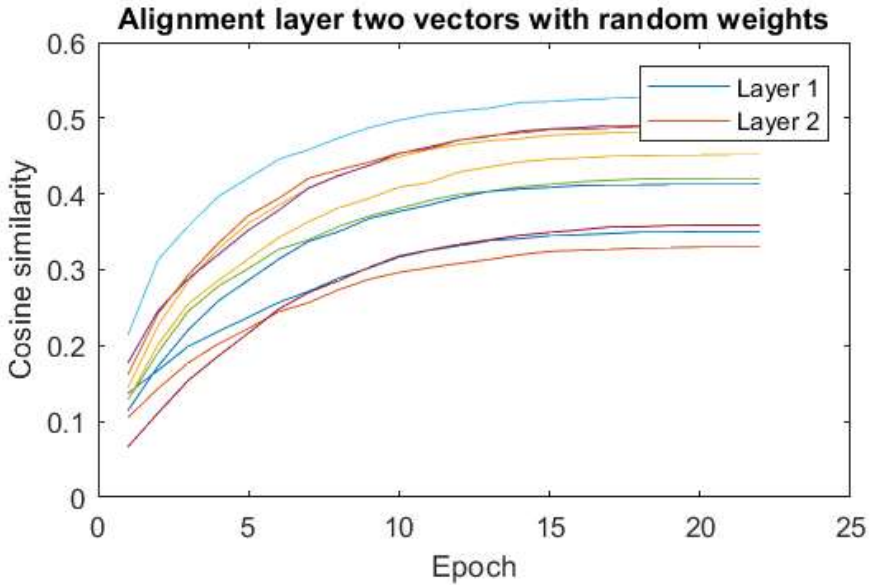
Test error 3.23%

Epoch average alignment



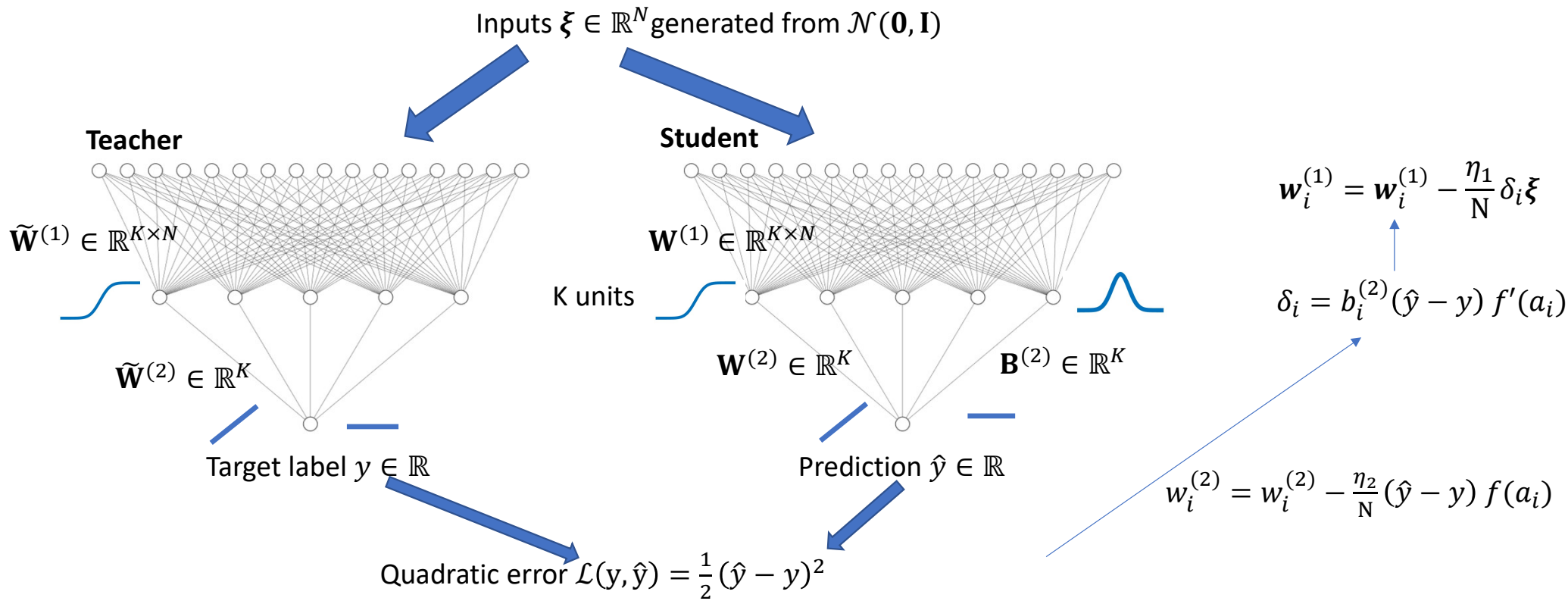
-1000-10 Architecture, tanh units hidden layers, softmax output
-Random normal feedback weights N(0, 1)
-Learning rate 0.01



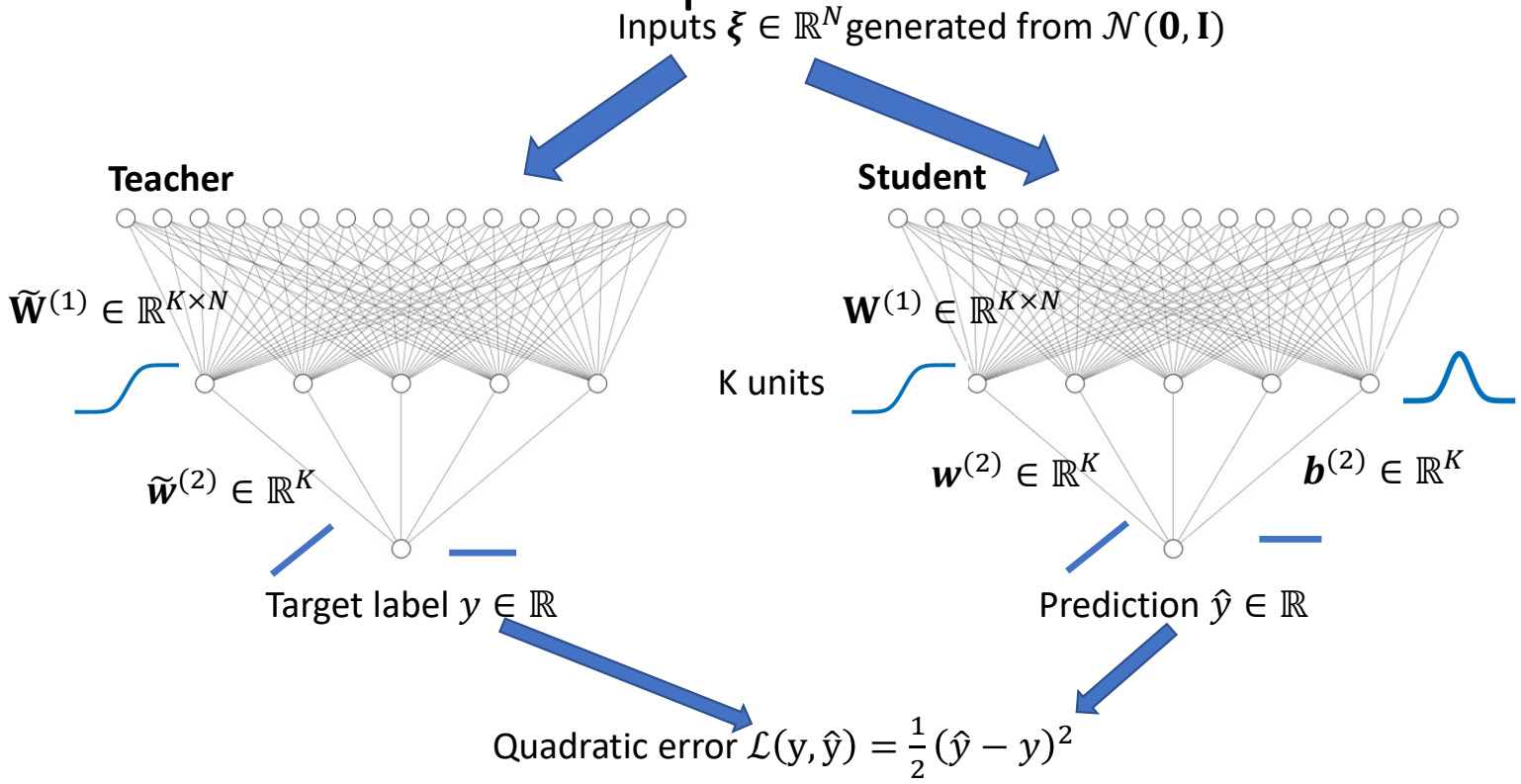


Alignment of second layer vectors with Feedback weights after each epoch.

Two-layer student teacher with sigmoidal units



Student teacher order parameters



During learning keep track of the weight vector overlaps:

$R_{in} = \boldsymbol{w}_i^{(1)} \cdot \widetilde{\boldsymbol{w}}_n^{(1)}$

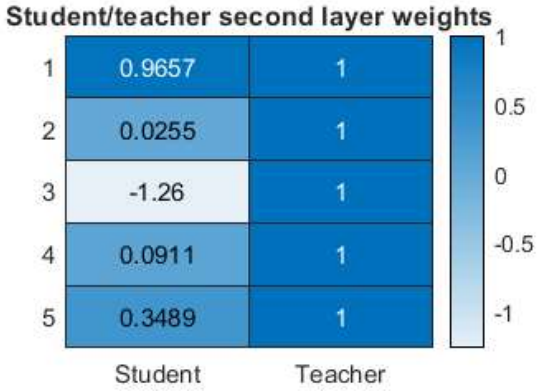
$Q_{ik} = \boldsymbol{w}_i^{(1)} \cdot \boldsymbol{w}_k^{(1)}$

In addition, we keep track of $F = \boldsymbol{w}^{(2)} \cdot \boldsymbol{b}^{(2)}$, the overlap between the second layer weights and the feedback vector.

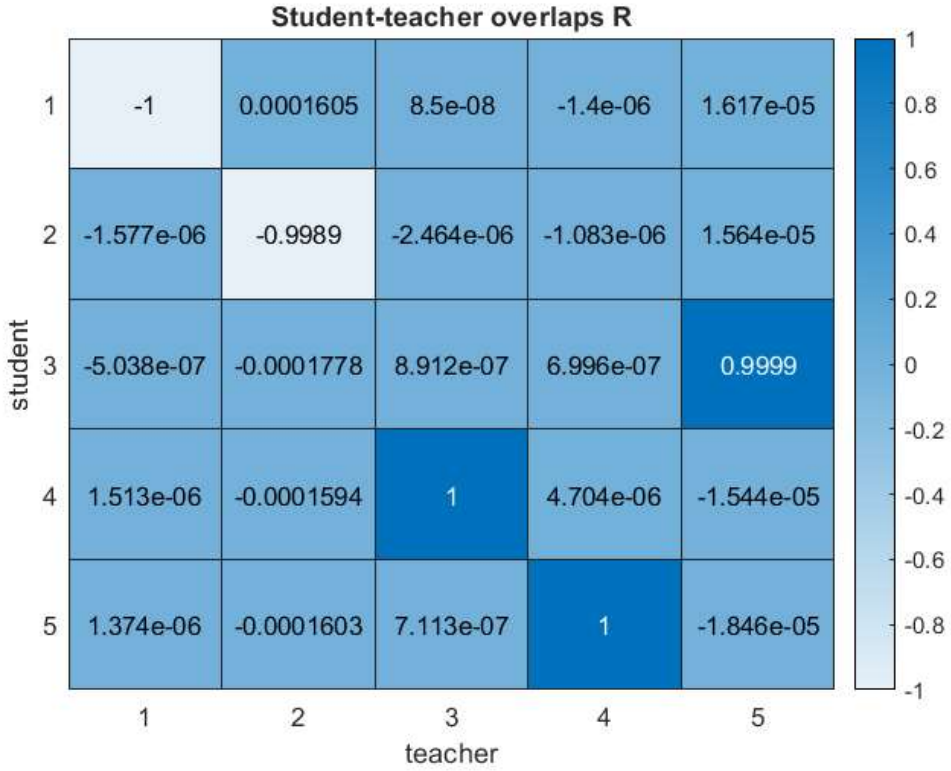
- Number of inputs N=500
- Start from same weight initialization in first layer with

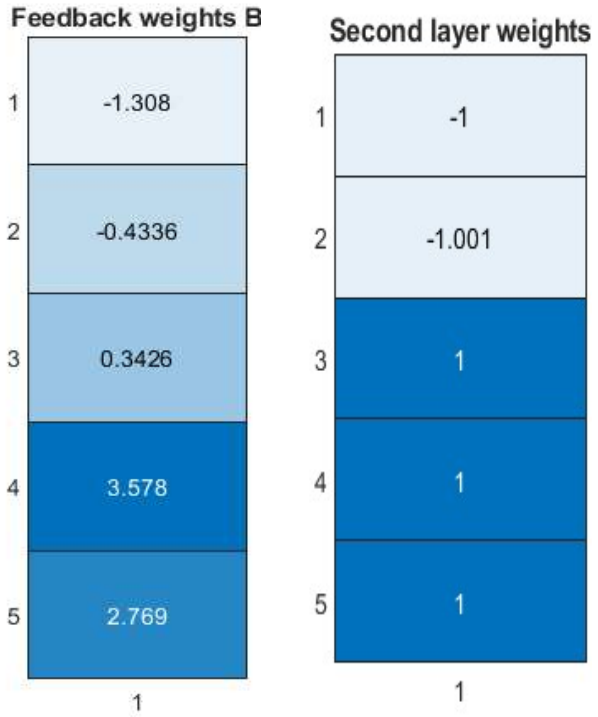
$$R_{in} = 0, Q_{ik} = 0.2\delta_{ik}, T_{nm} =, \text{for } 1 \leq \text{i, k, m, n,} \leq 5$$

Second layer teacher weights δ_{nm}



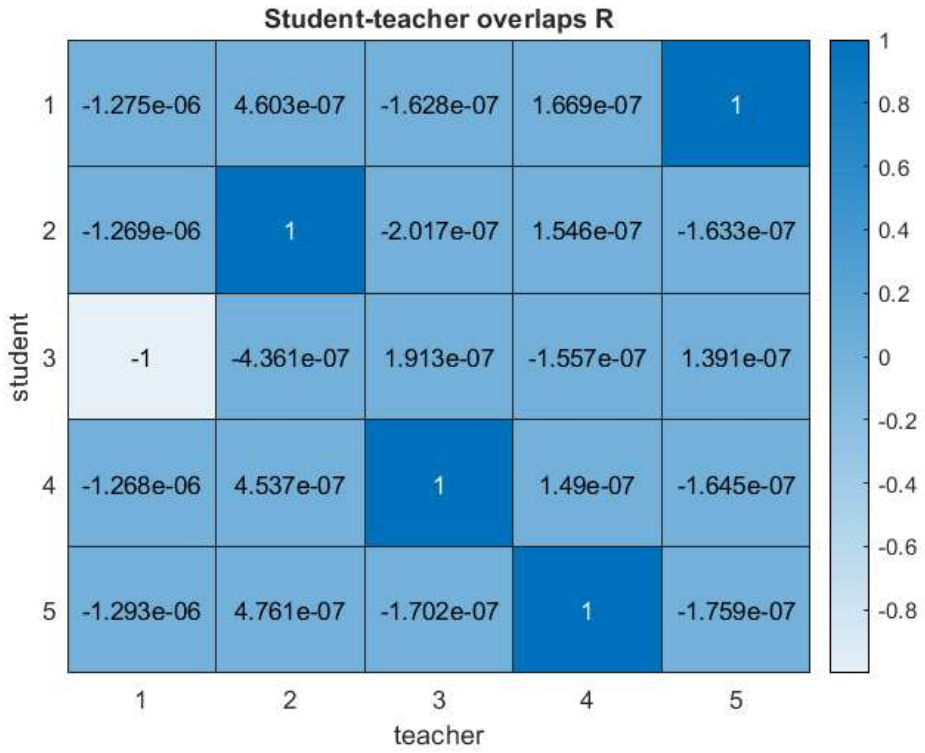
Student-Teacher FA

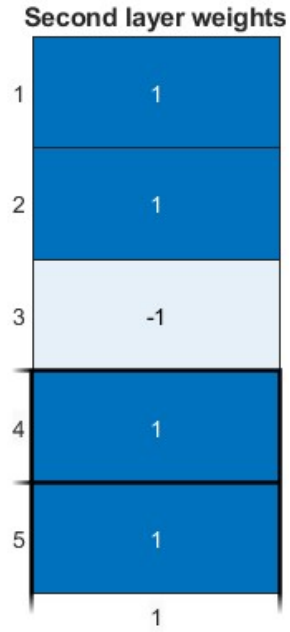




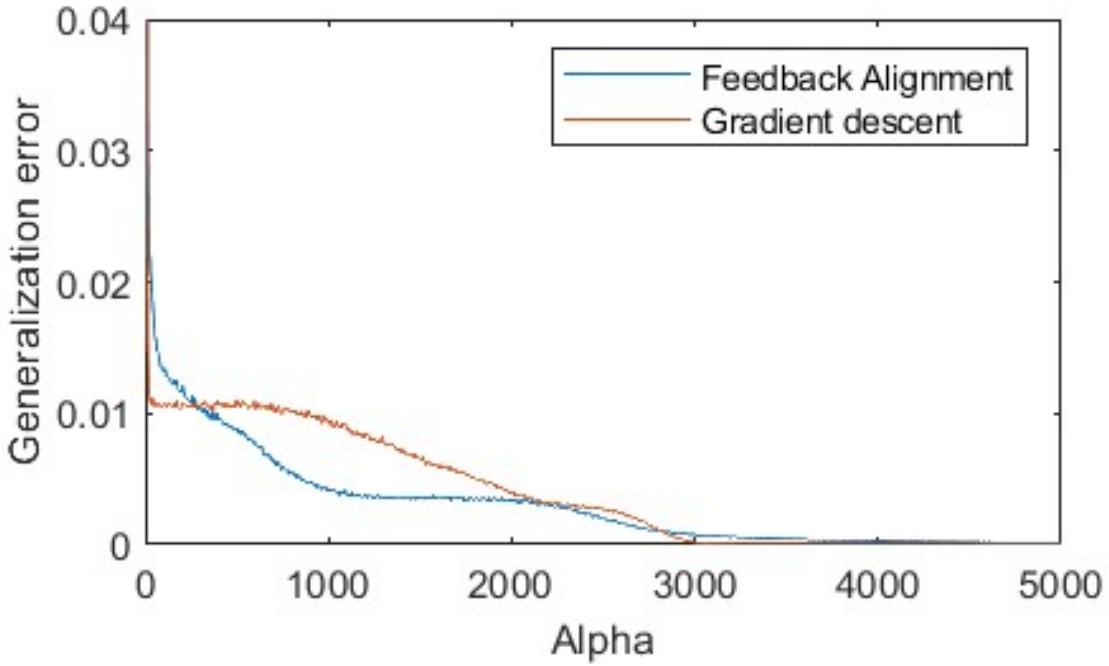
Second layer weights align with the feedback vector.

Student-Teacher BP

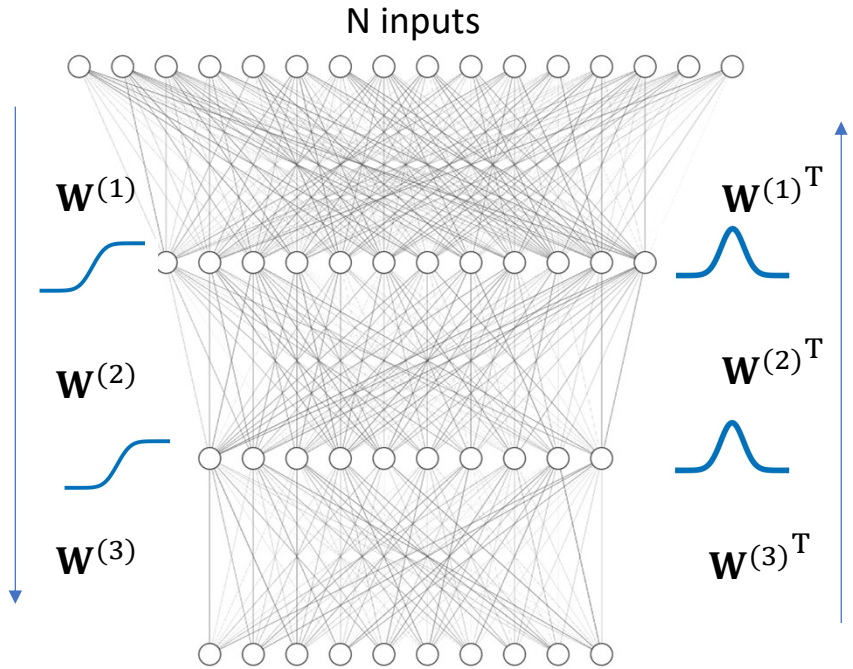




Student-teacher (K=5) FA vs. BP generalization error



MNIST experiment



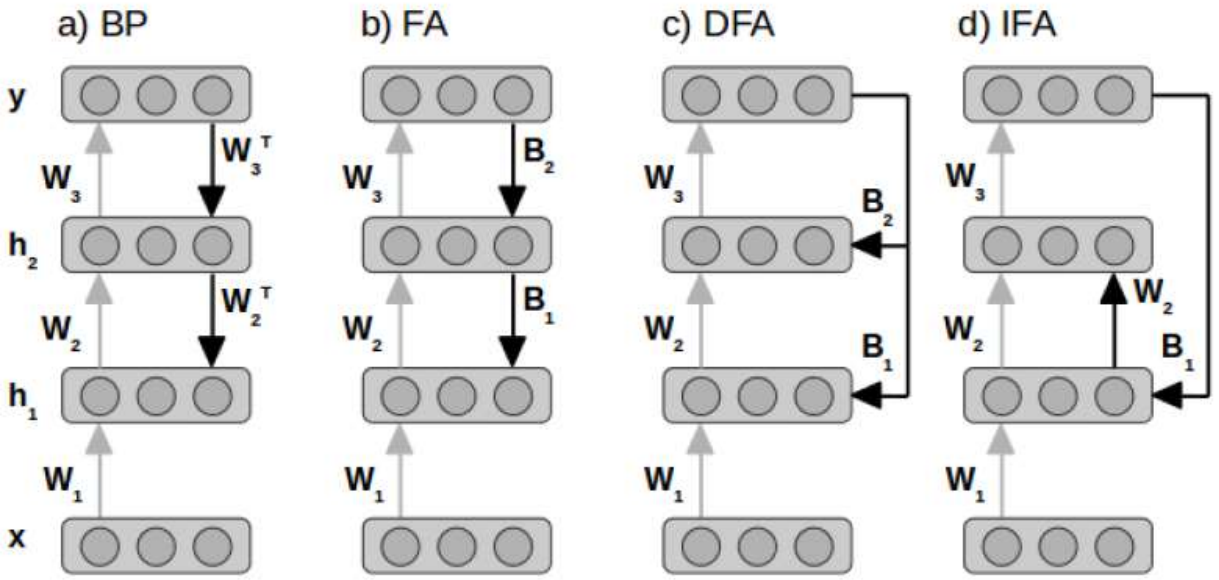
Last layer softmax $\hat{\mathbf{y}} = \mathrm{f}_y(\mathbf{a}^{(\mathrm{k})})$

Target label $\mathbf{y}$

Cross entropy loss

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{c=1}^{C} y_c \log \hat{y}_c$$

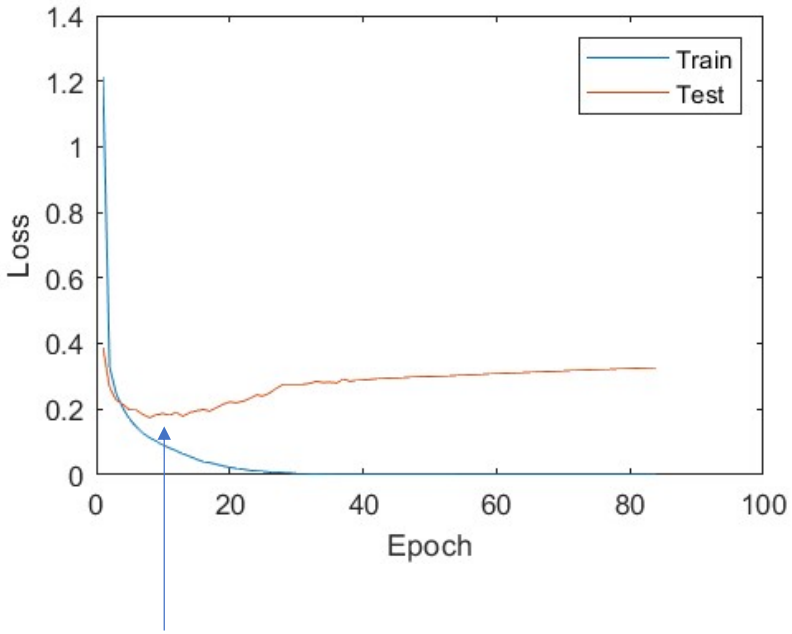
Direct Feedback Alignment



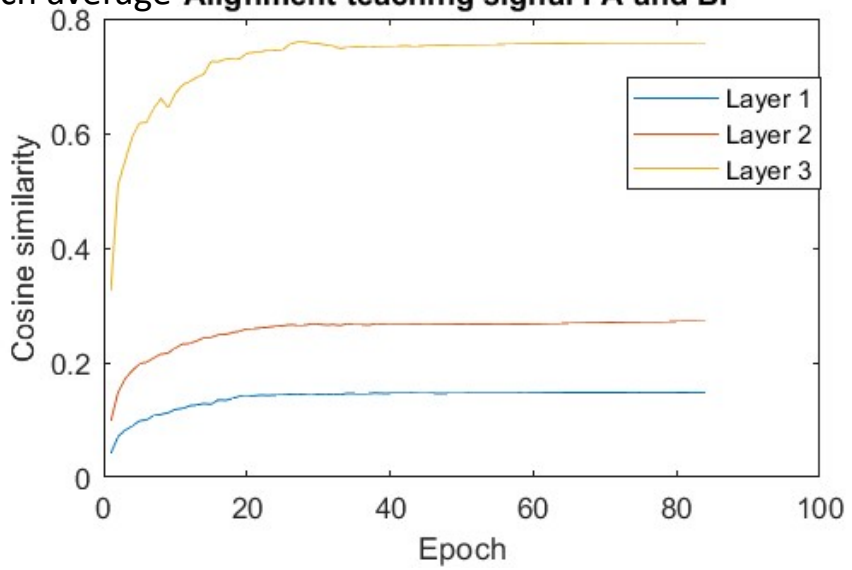
A. Nøkland, Direct Feedback Alignment Provides Learning in Deep Neural Networks

Results MNIST Direct Feedback Alignment

-200-200-100-10 Architecture, tanh units hidden layers, softmax output
-Random normal feedback weights N(0, 1)
-Learning rate 0.01



Epoch average **Alignment teaching signal FA and BP**



Test error: 4.31%

References

[1] T. P. Lillicrap, D. Cownden, D.B. Tweed, C. J. Akerman, Random feedback weights support learning in deep neural networks

[2] A. Nøkland, Direct Feedback Alignment Provides Learning in Deep Neural Networks