FAIR Conference 20

The dynamics of Feedback Alignment

Stéphane d'Ascoli

Joint work with
Maria Refinetti (ENS Paris)
Ruben Ohana (ENS Paris)
Sebastian Goldt (SISSA Trieste)

FACEBOOK AI

Why feedback alignment ?

Backpropagation

🤒 **Limitations** of backprop :

- **Weight transport problem** : backward pass uses same weights as forward pass (biologically unlikely)
- **Backward locking** : need to perform backward pass sequentially

💡 **Alternatives** to backprop ?

Feedback alignment

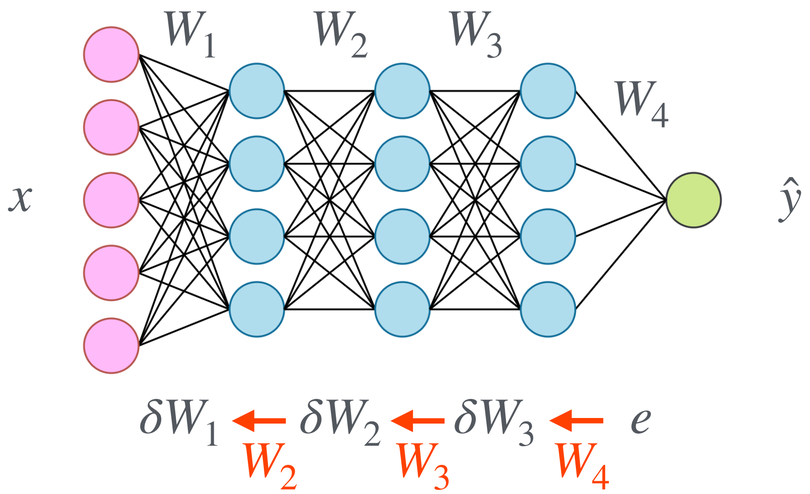
Consider a network with L layers.

$$a_l = W_l h_{l-1}, \quad h_l = \sigma(a_l), \quad \hat{y} = f_y(a_L)$$

SGD updates the weights as

$$\delta W_l = -\eta \delta a_l h_{l-1}^T \qquad \delta a_L = \hat{y} - y \equiv e$$

$$\delta a_l^{BP} = \left(W_{l+1}^T \delta a_{l+1}\right) \odot f'\left(a_l\right).$$



Feedback alignment

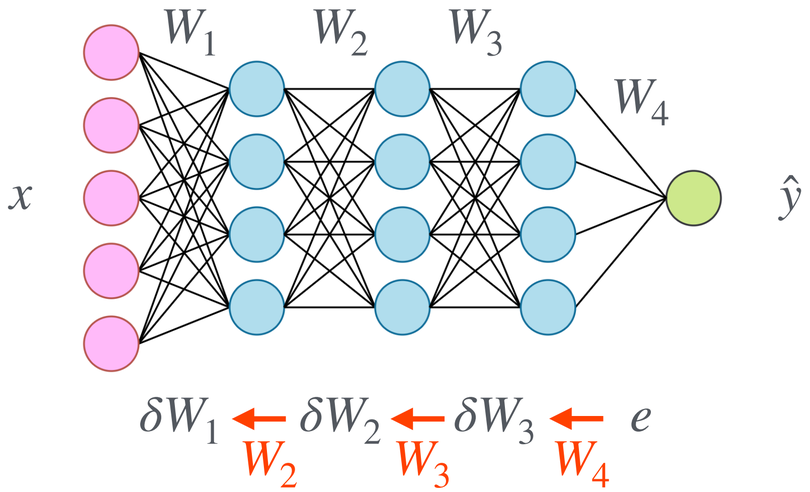
Consider a network with L layers.

$$a_l = W_l h_{l-1}, \quad h_l = \sigma(a_l), \quad \hat{y} = f_y(a_L)$$

SGD updates the weights as

$$\delta W_l = -\eta \delta a_l h_{l-1}^T \qquad \delta a_L = \hat{y} - y \equiv e$$

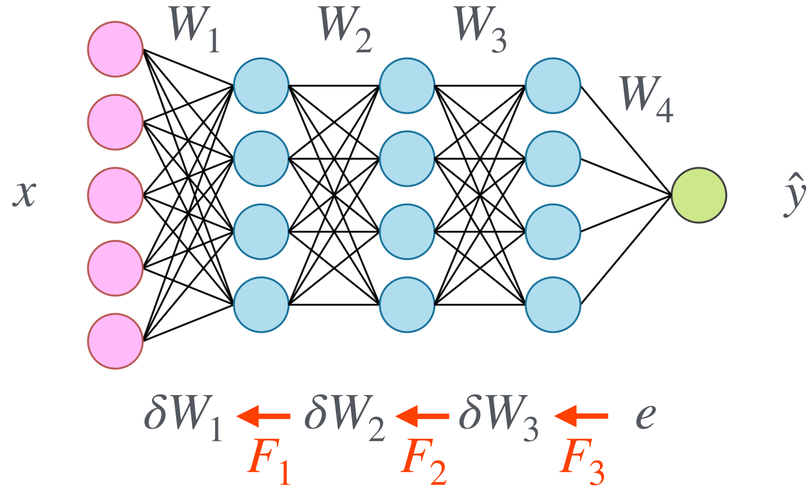
$$\delta a_l^{BP} = \left(W_{l+1}^T \delta a_{l+1}\right) \odot f'\left(a_l\right).$$



FA [LILLICRAP '14]

$$\delta a_l^{FA} = \left(F_l \delta a_{l+1}\right) \odot f'\left(a_l\right).$$

Random Feedback Matrix



Feedback alignment

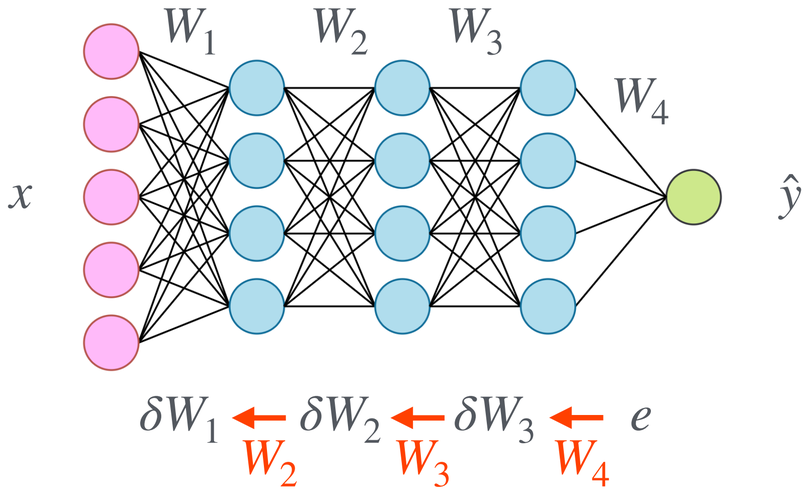
Consider a network with L layers.

$$a_l = W_l h_{l-1}, \quad h_l = \sigma(a_l), \quad \hat{y} = f_y(a_L)$$

SGD updates the weights as

$$\delta W_l = -\eta \delta a_l h_{l-1}^T \qquad \delta a_L = \hat{y} - y \equiv e$$

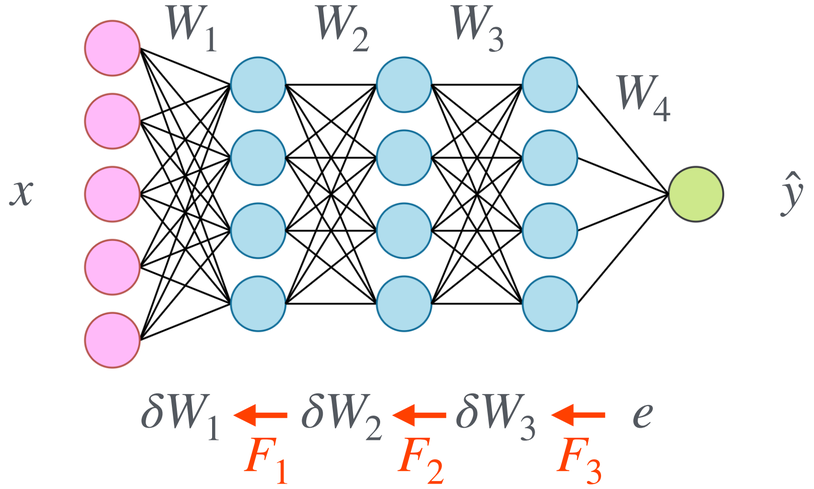
$$\delta a_l^{BP} = \left(W_{l+1}^T \delta a_{l+1}\right) \odot f'\left(a_l\right).$$



FA [LILLICRAP '14]

$$\delta a_l^{FA} = \left(F_l \delta a_{l+1}\right) \odot f'\left(a_l\right).$$

Random Feedback Matrix

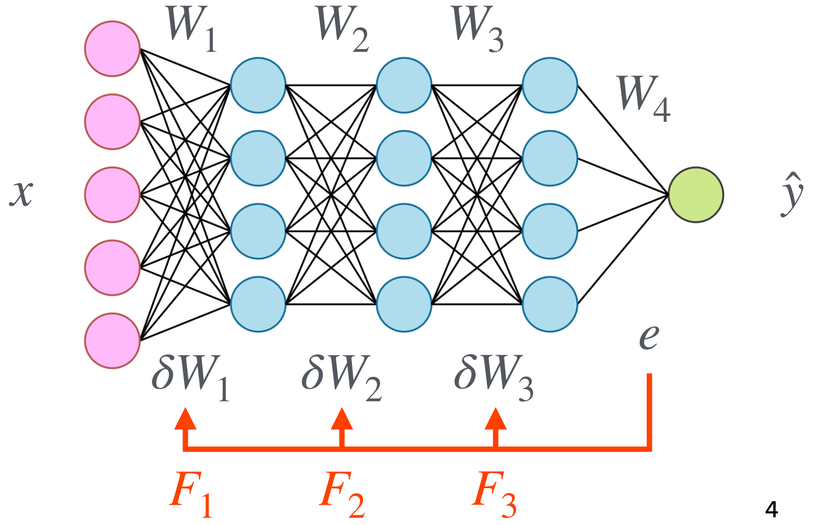


DFA [NOKLAND '16]

$$\delta a_l^{DFA} = \left(F_l e\right) \odot f'\left(a_l\right).$$

Random Feedback Matrix

Error Vector



Marvels and pitfalls of DFA

👍 **Gradient alignment** : gradient actually aligns with the true gradient during training !

➡️ DFA is a promising alternative to SGD, shown to scale to Transformers [Launay et al ’20]

👎 Pitfall : **DFA doesn’t work for CNNs** ! [Bartunov, ’18; Launay et al ’19]

➡️ Various alternatives have been proposed, by no thorough explanation of why it doesn’t work…

👨‍🔬 **What is the mechanism driving the success of DFA ?**

👨‍🔬 **Why doesn’t gradient alignment work for CNNs ?**

Align then Memorize

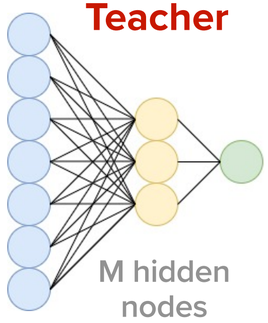
Teacher student setup

Setup

Inputs

$$x \in \mathbb{R}^N \quad \underset{\sim}{iid} \quad \mathcal{N}(0,1)$$

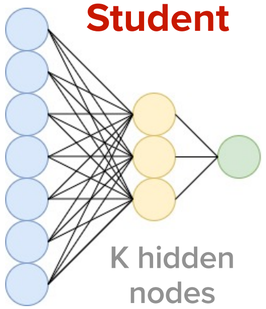
Teacher



Labels

$$y = \sum_{m=1}^{M} \tilde{W}_2^m g\left(\frac{\tilde{W}_1^{\,m} \cdot x}{\sqrt{N}}\right) \in \mathbb{R}$$

Student



Predictions

$$\hat{y} = \sum_{k=1}^{K} W_2^k g\left(\frac{W_1^k \cdot x}{\sqrt{N}}\right) \in \mathbb{R}$$

Generalisation Error

$$\epsilon_g = \frac{1}{2}\mathbb{E}\left[\left(y - \hat{y}\right)^2\right] = \frac{1}{2}\mathbb{E}\left[e^2\right]$$

Limit of infinite training data

$$K, M \sim \mathcal{O}(1) \quad N \to \infty$$

SGD updates

First layer

BP $\delta W_1 = -\eta \left[W_2^\top e \odot g'\left(a_1\right)\right] x^\top$

DFA $\delta W_1 = -\eta \left[F_1 e \odot g'\left(a_1\right)\right] x^\top$

Second layer

$$\delta W_2 = -\eta g\left(a_1\right) e^\top$$

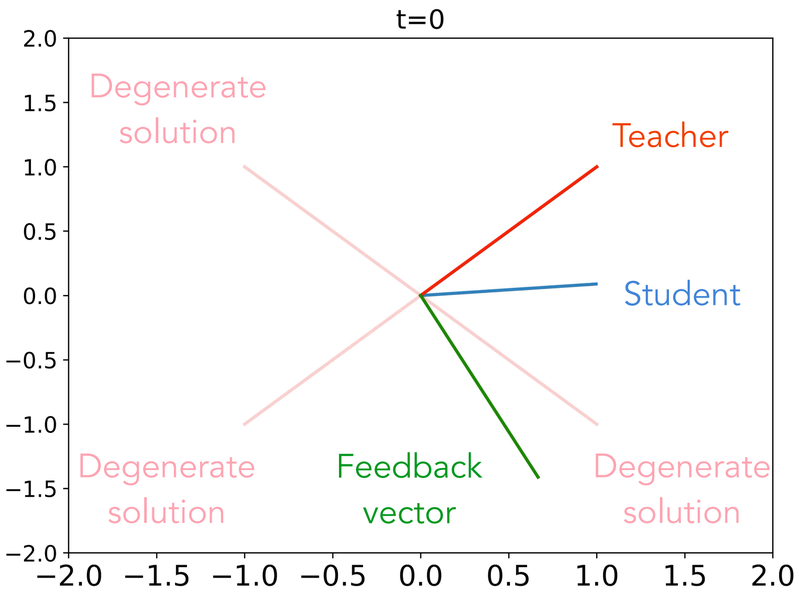
→ **Closed set of ODEs describing training dynamics**

Degeneracy breaking

- **In BP :** one constraint → Match teacher's output: $\hat{y} = y$

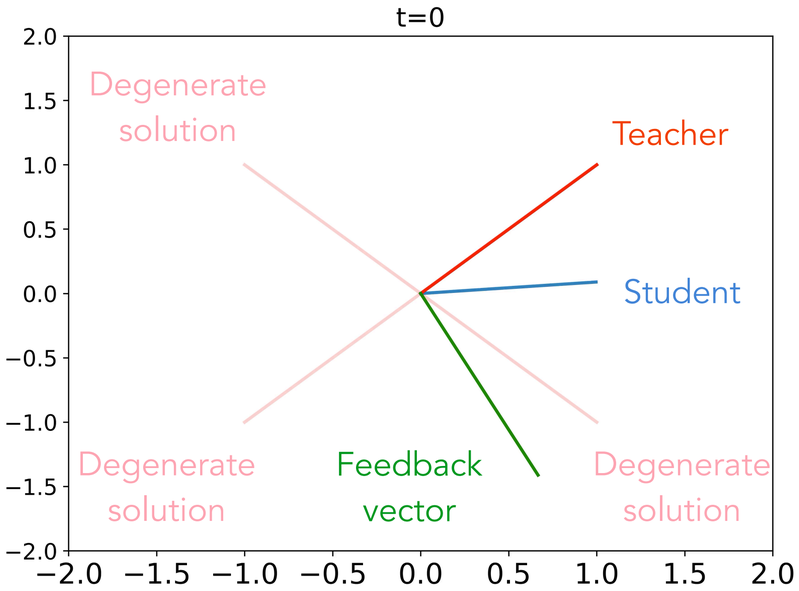
  solution → Match teacher's weights: $W_1 = \tilde{W}_1$ and $W_2 = \tilde{W}_2$

  other solutions → $\mathrm{erf}(x) = -\mathrm{erf}(-x)$ → $W_1 = -\tilde{W}_1$ and $W_2 = -\tilde{W}_2$ is also a solution!



Degeneracy breaking

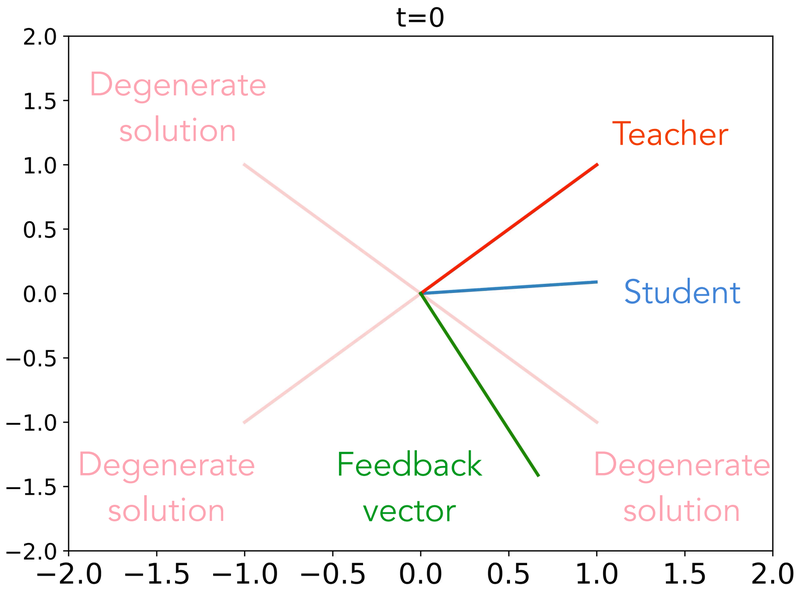
- **In BP :** one constraint → Match teacher's output: $\hat{y} = y$
  - solution → Match teacher's weights: $W_1 = \tilde{W}_1$ and $W_2 = \tilde{W}_2$
  - other solutions → $\mathrm{erf}(x) = -\mathrm{erf}(-x)$ → $W_1 = -\tilde{W}_1$ and $W_2 = -\tilde{W}_2$ is also a solution!
- **In DFA :** additional constraint → Recover BP gradient → $W_2 = F_1$



Degeneracy breaking

- **In BP :** one constraint → Match teacher's output: $\hat{y} = y$
  - solution → Match teacher's weights: $W_1 = \tilde{W}_1$ and $W_2 = \tilde{W}_2$
  - other solutions → $\mathrm{erf}(x) = -\mathrm{erf}(-x)$ → $W_1 = -\tilde{W}_1$ and $W_2 = -\tilde{W}_2$ is also a solution!
- **In DFA :** additional constraint → Recover BP gradient → $W_2 = F_1$

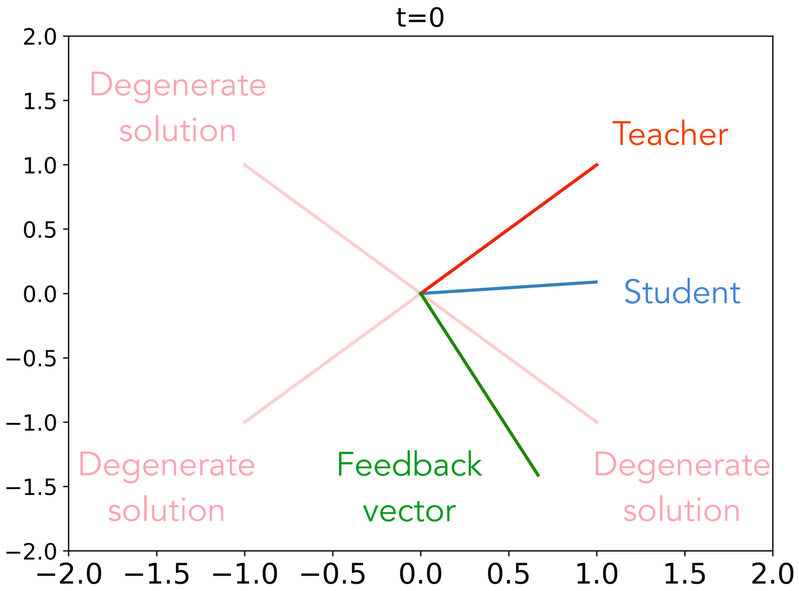
🤔 **Problem** : how can the second layer both align and match ?



Degeneracy breaking

- **In BP :** one constraint → Match teacher's output: $\hat{y} = y$
  - solution → Match teacher's weights: $W_1 = \tilde{W}_1$ and $W_2 = \tilde{W}_2$
  - other solutions → $\mathrm{erf}(x) = -\mathrm{erf}(-x)$ → $W_1 = -\tilde{W}_1$ and $W_2 = -\tilde{W}_2$ is also a solution!
- **In DFA :** additional constraint → Recover BP gradient → $W_2 = F_1$

🤔 **Problem** : how can the second layer both align and match ?



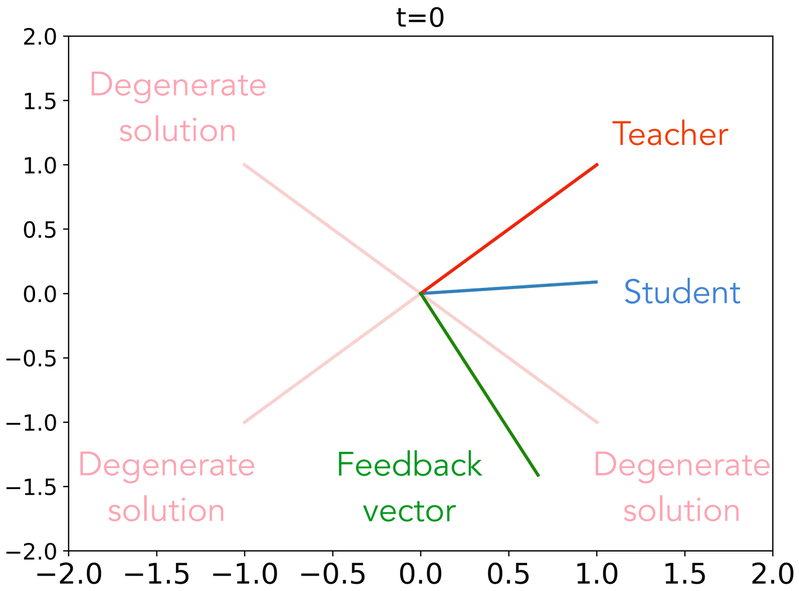
Degeneracy breaking

- **In BP :** one constraint → Match teacher's output: $\hat{y} = y$
  - solution → Match teacher's weights: $W_1 = \tilde{W}_1$ and $W_2 = \tilde{W}_2$
  - other solutions → $\mathrm{erf}(x) = -\mathrm{erf}(-x)$ → $W_1 = -\tilde{W}_1$ and $W_2 = -\tilde{W}_2$ is also a solution!
- **In DFA :** additional constraint → Recover BP gradient → $W_2 = F_1$

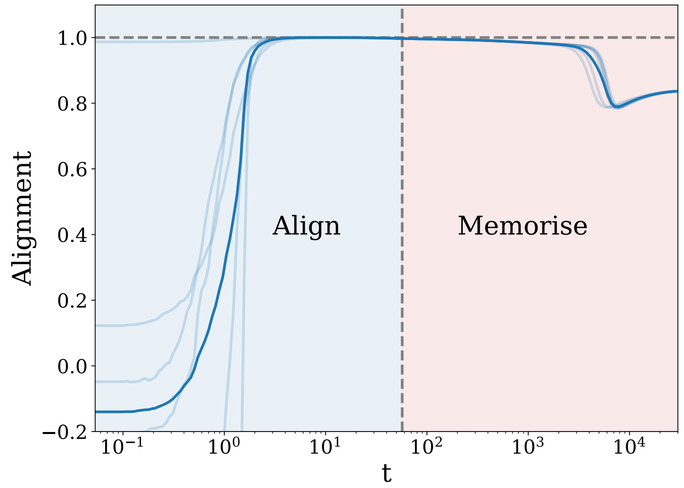
🤔 **Problem** : how can the second layer both align and match ?

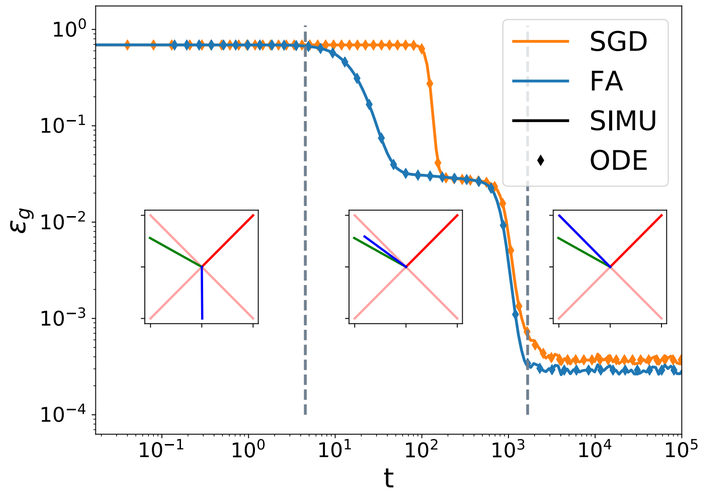
💡 **Solution** :

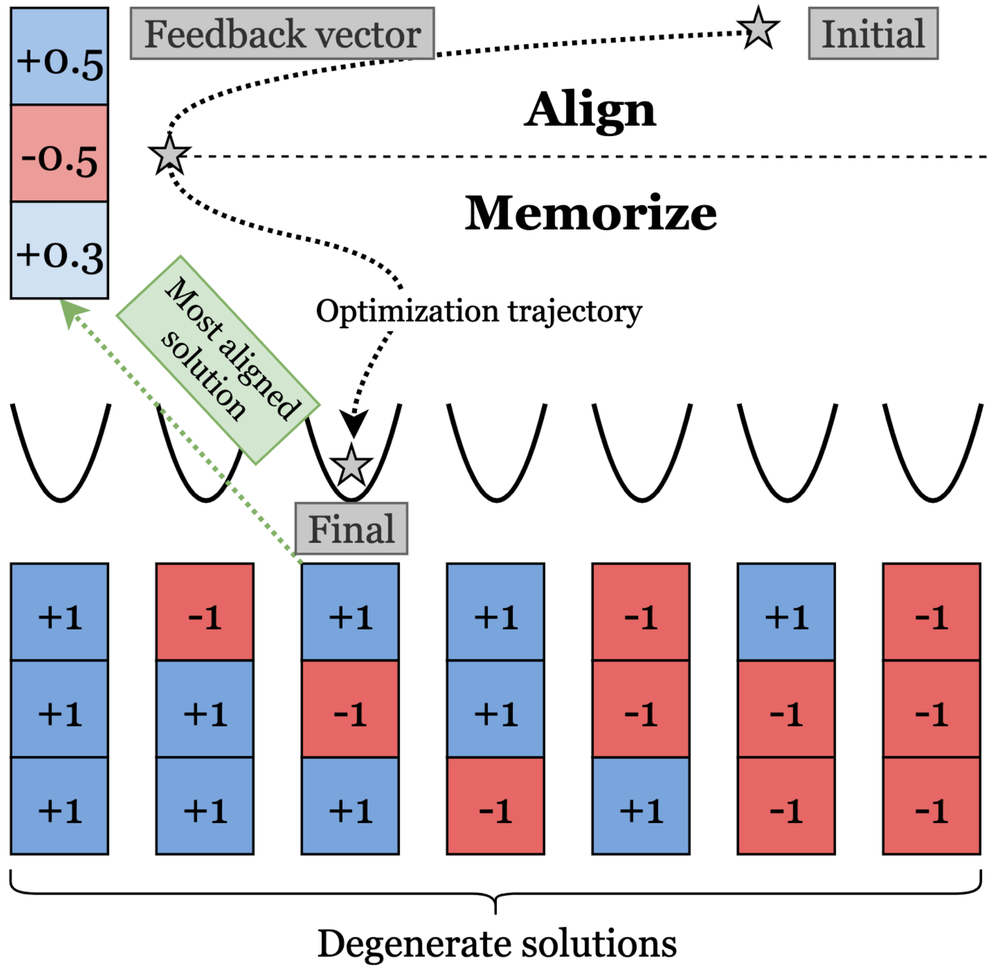
1. **Align** : the student first grows towards F
2. **Memorize** : then it falls to the nearest solution



Align-then-memorize







The alignment phase

Linear setup

In shallow networks, aligning second layer suffices to align gradients.

🧐 **What happens for deep networks ?**

Warm-up : consider the deep linear setup

$$\delta W_1^t = -\eta (F_1 e_t) x_t^T \qquad \delta W_2^t = -\eta (F_2 e_t)(W_1 x_t)^\top$$

Start from zero weights. The first layer performs **Hebbian learning** 🧠

$$W_1^t = -\eta \sum_{t'=0}^{t-1} F_1 e_{t'} x_{t'}^\top = F_1 A_1^t, \qquad A_1^t = -\eta \sum_{t'=0}^{t-1} e_{t'} x_{t'}^\top$$

From this we obtain the dynamics of second layer :

$$W_2^t = -\eta \sum_{t'=0}^{t-1} F_2 e_t (A_1^{t'} x_{t'})^\top F_1^\top = F_2 A_2^t F_1^\top, \qquad A_2^t = \eta^2 \sum_{t'=0}^{t-1} \sum_{t''=0}^{t'-1} (x_{t'} \cdot x_{t''}) e_{t'} e_{t''}^\top .$$

Weak and strong weight alignment

By generalizing the above calculation, we obtain the following :

WEAK WEIGHT ALIGNMENT

$$W_l^t = F_l A_l^t F_{l-1}^\top \quad l = 2,...L-1 \qquad\qquad W_L^t = A_L^t F_{L-1}^\top,$$

Where we defined the **alignment matrices**

$$A_{l\geq 2}^t = \eta^2 \sum_{t'=0}^{t-1} \sum_{t''=0}^{t'-1} (B_l^{t'} x_{t'}) \cdot (B_l^{t''} x_{t''}) e_{t'} e_{t''}^\top \in \mathbb{R}^{n_L \times n_L}$$

If the alignment matrices become proportional to the identity, we have :

STRONG WEIGHT ALIGNMENT

$$W_l^t \propto F_l F_{l-1}^\top, \quad l = 2,...L-1 \qquad\qquad W_L^t \propto F_{L-1}^\top$$

Weight alignment underlies gradient alignment

🧐 What is special about strong weight alignment ?

💡 **Strong weight alignment** ➡️ **Gradient alignment** 👏👏👏

Sketch of proof : $\delta W_l^{BP} = \left(W_{l+1}^\top \dots W_L^T e\right)\left(W_{l-1} \dots W_1 x\right)^\top \qquad \delta W_l^{DFA} = \left(F_l e\right)\left(W_{l-1} \dots W_1 x\right)^\top$

Strong WA: $W_l^t \propto F_l F_{l-1}^\top, \quad l = 2,\dots L-1$

$W_L^t \propto F_{L-1}^\top .$

➡️ $W_{l+1}^T W_{l+2}^T \dots W_L^\top \propto (F_l \boxed{F_{l+1}^\top})(F_{l+1}\boxed{F_{l+2}^\top}) \dots (F_{L-2}^\top \boxed{F_{L-1}^\top})F_{L-1} = F_l$

Identity **Identity**

🧐 **Is this also the case in deep nonlinear networks ?**

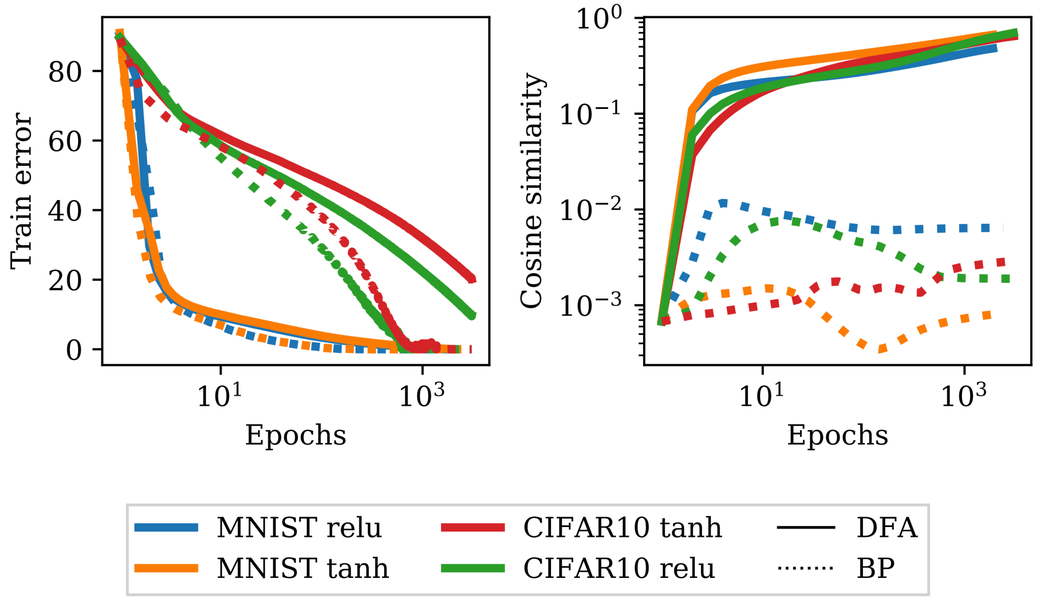
The deep nonlinear case

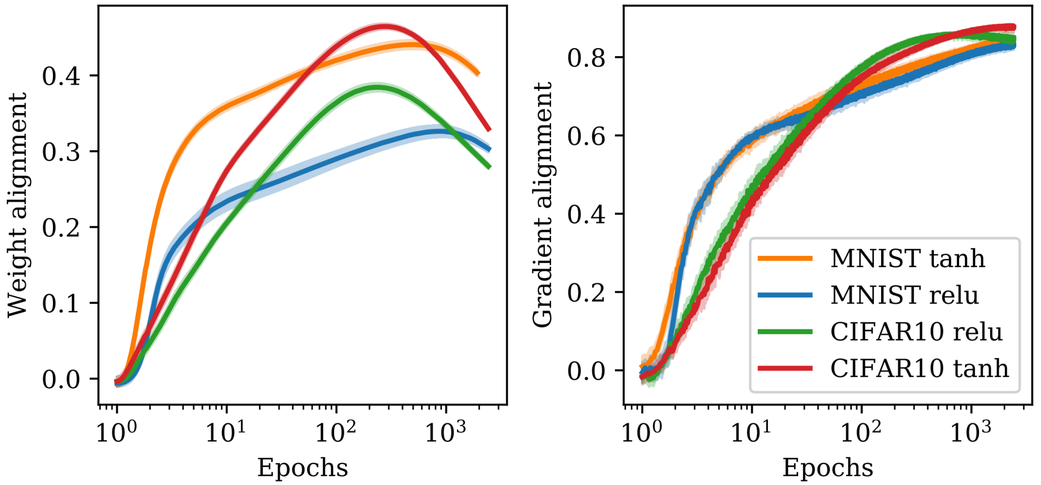
Gradient alignment holds

Run a 4-layer MLP with 100 nodes per layer on MNIST and CIFAR10. Measure :

➡️ Cosine similarity between final weights of 10 seeds

➡️ Weight alignment : $$\frac{W_l \cdot \left(F_l F_{l-1}^\top\right)}{\|W_l\| \, \|F_l F_{l-1}^\top\|}$$



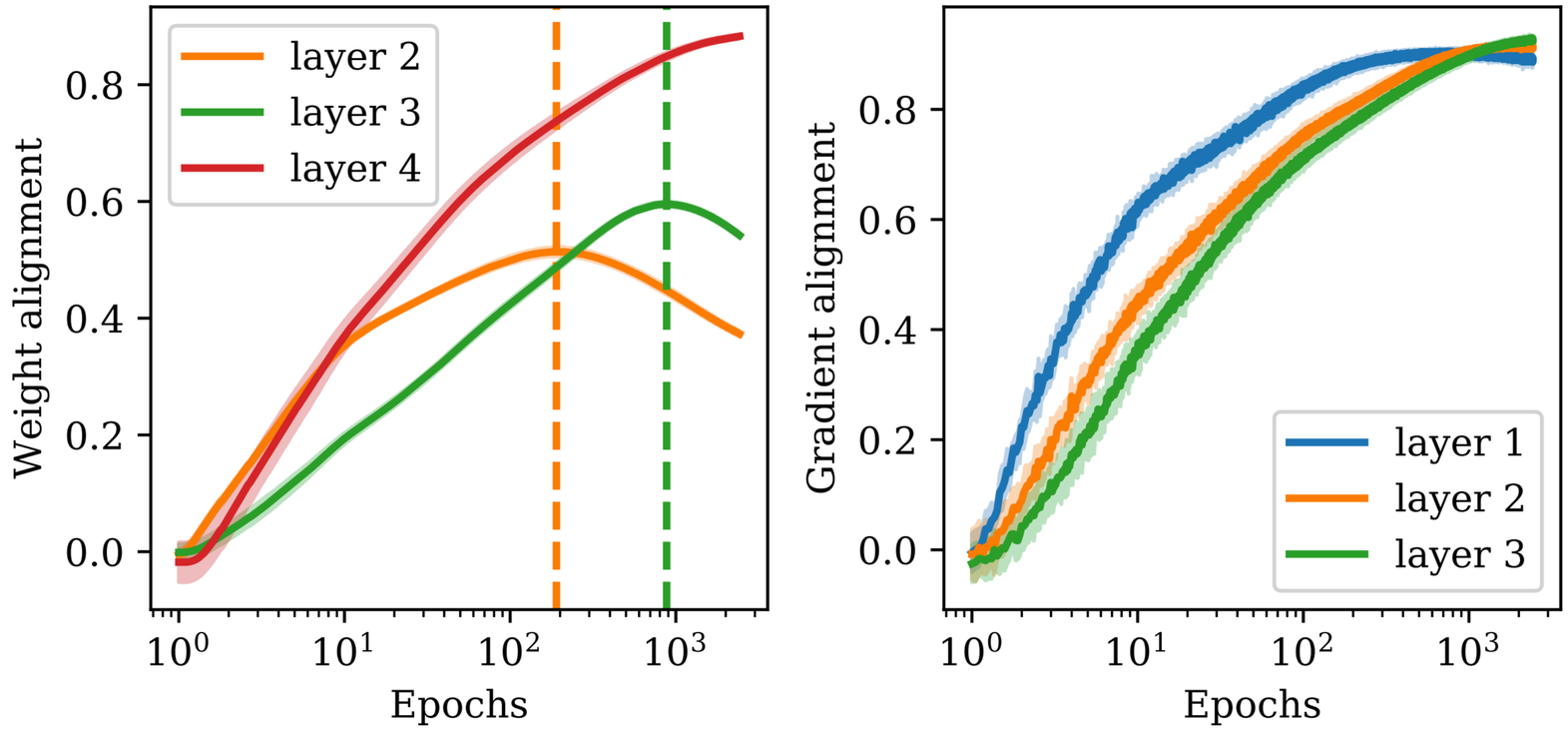


- **The weights correlate during training, signalling degeneracy breaking ! 😲**
- **Weight alignment is of order one, and drives gradient alignment ! 😲**

Layerwise match-then-memorize

The align-then-memorize process occurs layerwise, from bottom to top!

This allows GA to stay high (upper layers are the most crucial for GA)



What can cause failure ?

Structure of data

👀 To study GA, study the conditioning of the alignment matrices

$$A^t_{l\geq 2} = \eta^2 \sum_{t'=0}^{t-1} \sum_{t''=0}^{t'-1} (B_l^{t'} x_{t'}) \cdot (B_l^{t''} x_{t''}) e_{t'} e_{t''}^\top \in \mathbb{R}^{n_L \times n_L}$$

🧐 What if we only have one output class (e.g. scalar regression / binary classification) ?

💡 Then the alignment matrices are just scalars : we have **strong WA, therefore perfect gradient alignment** !

🧐 What about the general case ?

💡 Alignment requires good conditioning of the error statistics

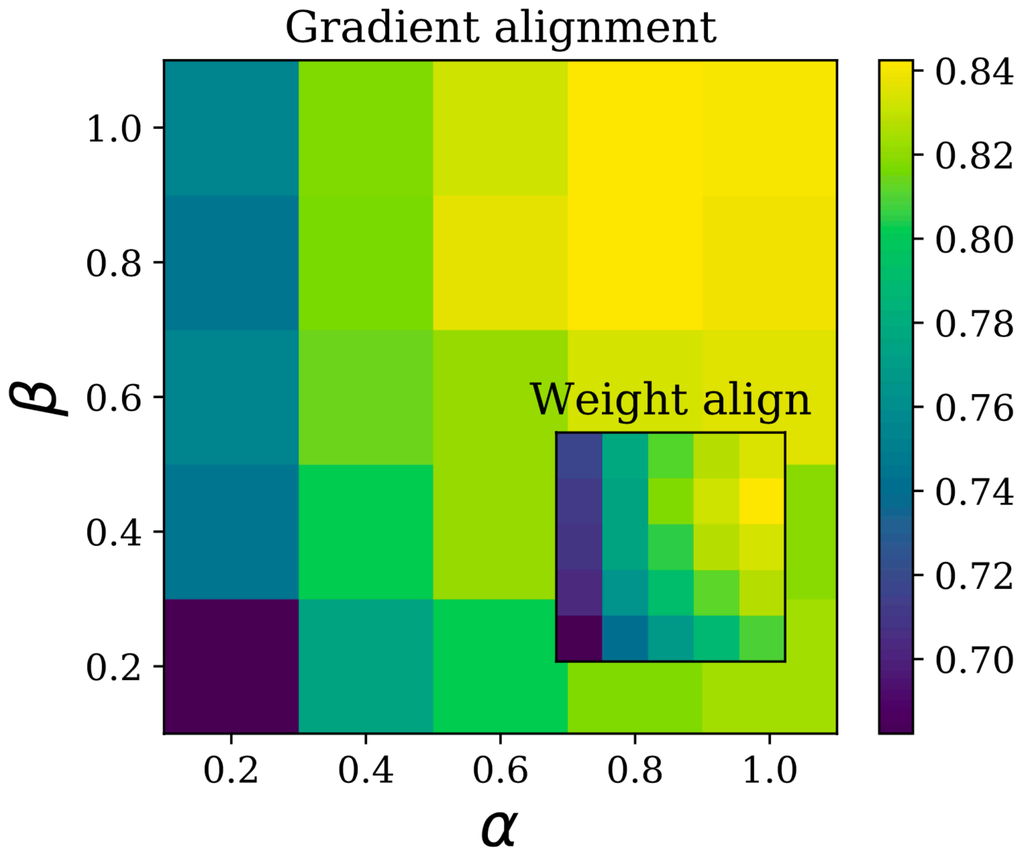
$$e = \hat{y} - y$$

Structure of the targets

Train a ReLU net on the following synthetic dataset :

$$x \in \mathbb{R}^{10} \sim \mathcal{N}(0,1)$$
$$y \in \mathbb{R}^{2} \sim \mathcal{N}(0,\Sigma),$$
$$\Sigma = \begin{bmatrix} 1 & \alpha(1-\beta) \\ \alpha(1-\beta) & \alpha^2 \end{bmatrix}$$

Increasing variance imbalance (α) and correlation (β) hampers the WA and GA



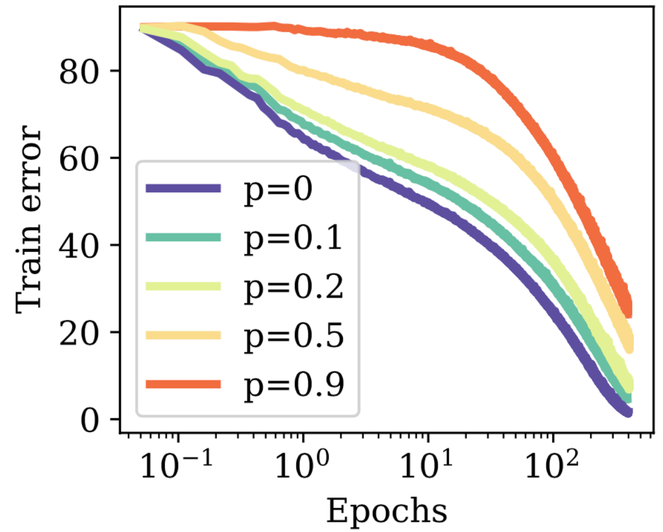
Structure of predictions

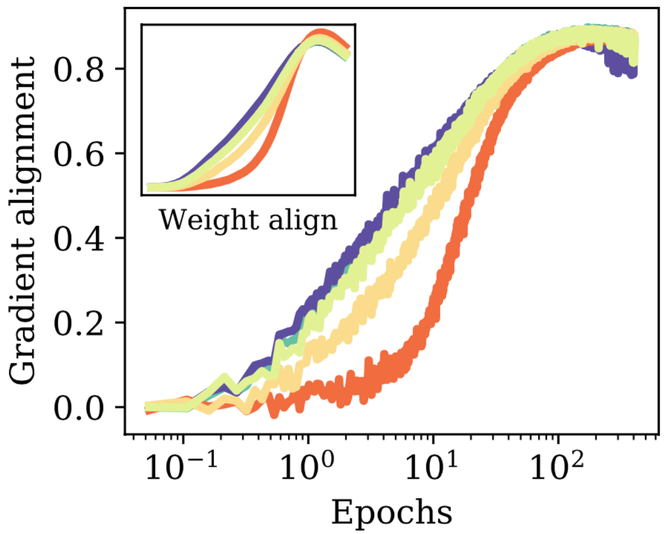
Train a ReLU network on CIFAR10, with a fraction p of the labels shuffled

➡️ What happens at high p ?

First epoch of training : network assigns probability 0.1 to each of the classes ➡️ **The correlation of output classes hampers WA**

After an epoch : the network starts to overfit (vertical line), the probabilities of the different classes decouple ➡️ **WA starts to grow**





What is wrong with conv layers ?

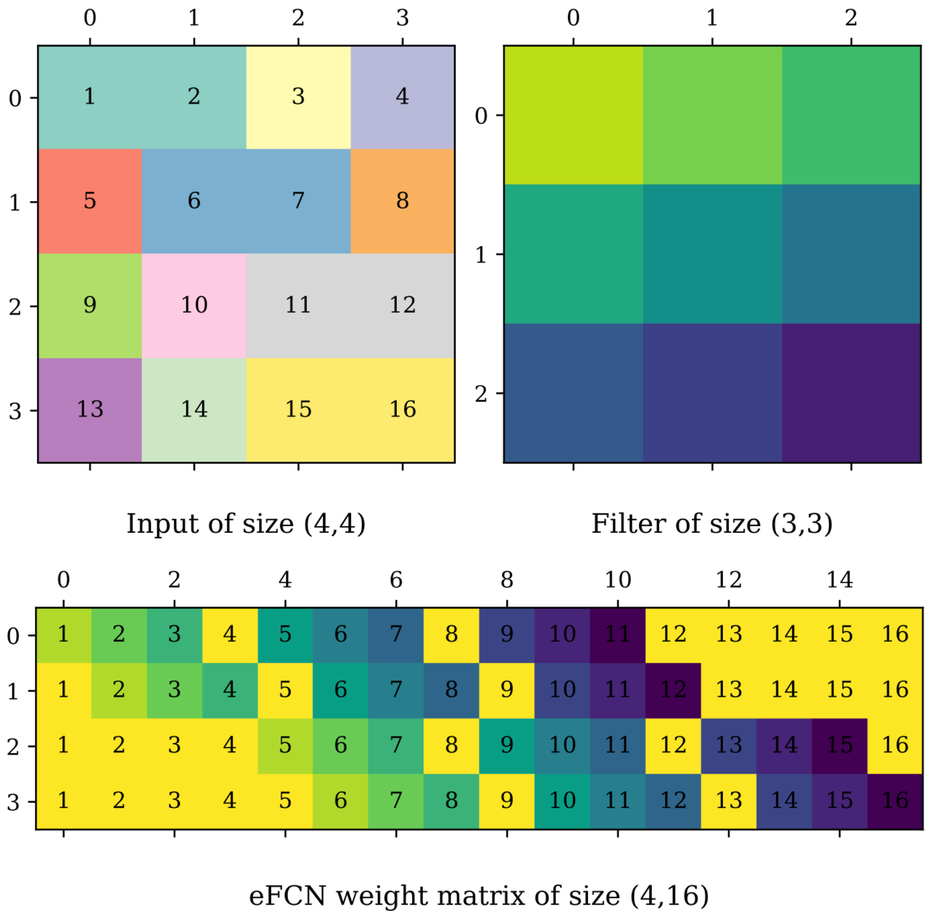
A conv layer W can be represented by a fully connected layer ϕ(W)

ϕ(W) has a **constrained structure** (locality + weight sharing). The condition for WA is the following :

$$\phi(W_l) \propto F_l F_{l-1}^{\top}$$

🚨 **WA cannot be ensured** due to the constraints on ϕ(W) !

💡 **Future work** : can we impose the structure in the feedback matrices ?



W

$\phi(W)$

[d'Ascoli et al, 2019]

Takeaways

Takeaways

DFA learning dynamics occurs in two stages :

Alignment phase : forward weights adapt to the backward (feedback) weights

- ➡️ Requires **balanced and uncorrelated** statistics of the errors
- ➡️ Requires flexibility of forward weights : **impossible for CNNs**

Memorization phase : the learner breaks the degeneracy by converging to the most aligned solution (possibly giving up some alignment)

- ➡️ Drives to a special region of the loss landscape
- ➡️ Depends on how hard the dataset is to overfit

Thank you !