tinyML® On Device Learning Forum

Enabling Ultra-low Power Machine Learning at the Edge

"Forward Learning with Top-Down Feedback: Solving the Credit Assignment Problem without a Backward Pass"

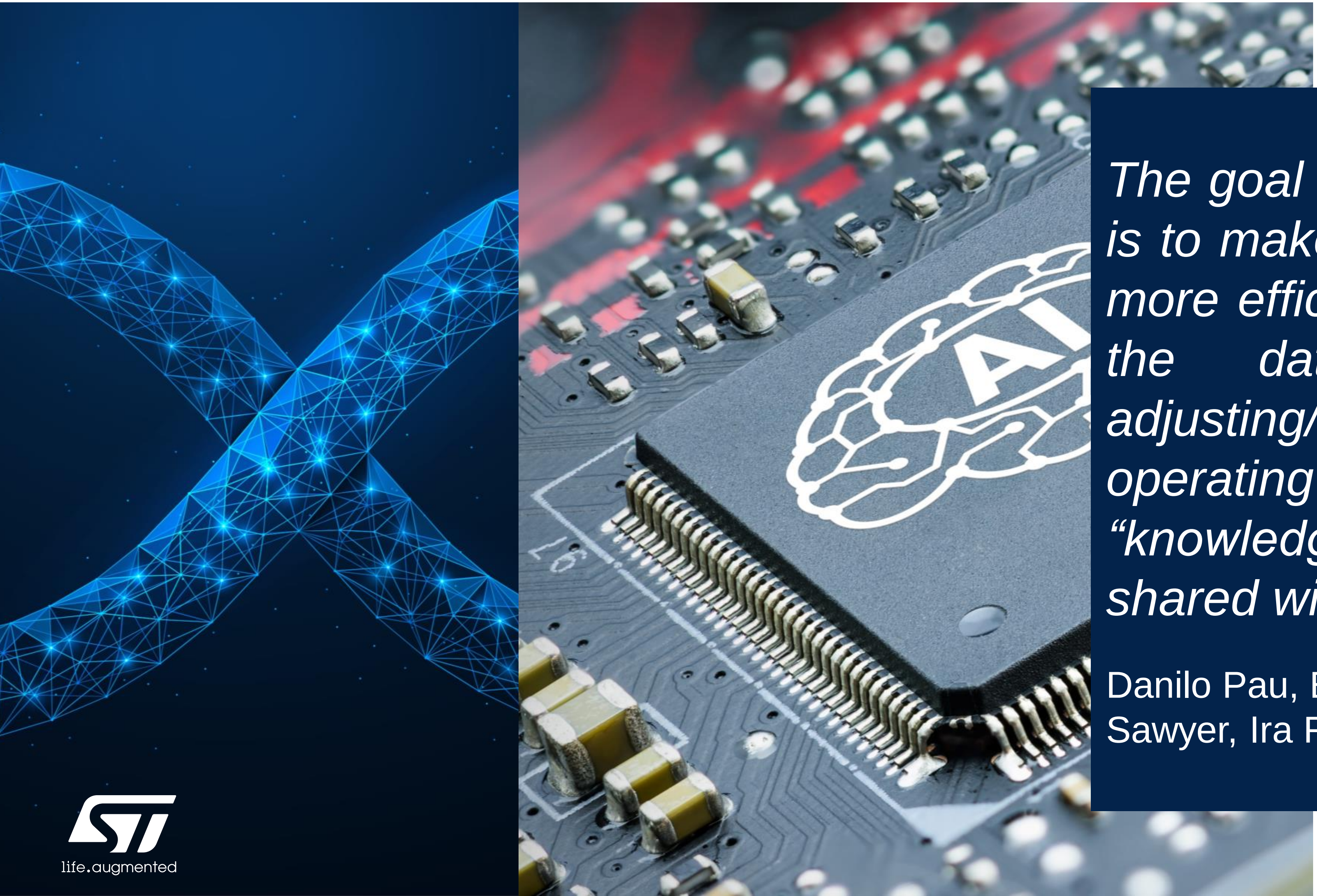
Giorgia Dellaferrera – Researcher, Institute of Neuroinformatics Zurich

May 16, 2023

TINY ML

www.tinyML.org

The Dawn of On Device Learning in TinyML



The goal of On Device Learning (ODL) is to make edge devices "smarter" and more efficient by observing changes in the data collected and self-adjusting/reconfiguring the device's operating model. Optionally the "knowledge" gained by the device is shared with other deployed devices.

Danilo Pau, Elias Fallon, Evgeni Gousev, Davis Sawyer, Ira Feldman, Christopher B. Rogers

tinyML On Device Learning Forum 8/31 – 9/1 , 2022 Online

T I N Y ODL

On device learning Forum

- Accademia on 8/31/2022
  - On-Device Learning Under 256KB Memory, Song HAN, Assistant Professor, MIT EECS
  - Neural Network ODL for Wireless Sensor Nodes ,Hiroki MATSUTANI, Professor, Keio University
  - Scalable, Heterogeneity-Aware and Trustworthy Federated Learning, Yiran CHEN, Professor, Duke University
  - On-Device Learning For Natural Language Processing with BERT, Warren J. GROSS, Professor, McGill University
  - Is on-device learning the next “big thing” in TinyML? Manuel ROVERI, Associate Professor, Politecnico di Milano
  - ODL Professors Panel
- Industry on 9/1/2022
  - TinyML ODL in industrial IoT, Haoyu REN, PhD Student, Technical University of Munich/Siemens
  - NeuroMem® wearable, hardwired sub milliwatt real time machine learning with wholly parallel access to “neuron memories” fully explainable, Guy PAILLET, Co-founder, General Vision
  - Using Coral Dev Board Micro for ODL innovations, Bill LUAN, Senior Program Manager, Google
  - Platform for Next Generation Analog AI Hardware Acceleration, Kaoutar EL MAGHRAOUI, Principal Research Scientist, IBM T.J Watson Research Center
  - Enabling on-device learning at scale, Joseph SORIAGA, Sr. Director of Technology, Qualcomm
  - Training models on tiny edge devices, Valeria TOMASELLI, Senior Engineer, STMicroelectronics

https://www.tinyml.org/event/on-device-learning/

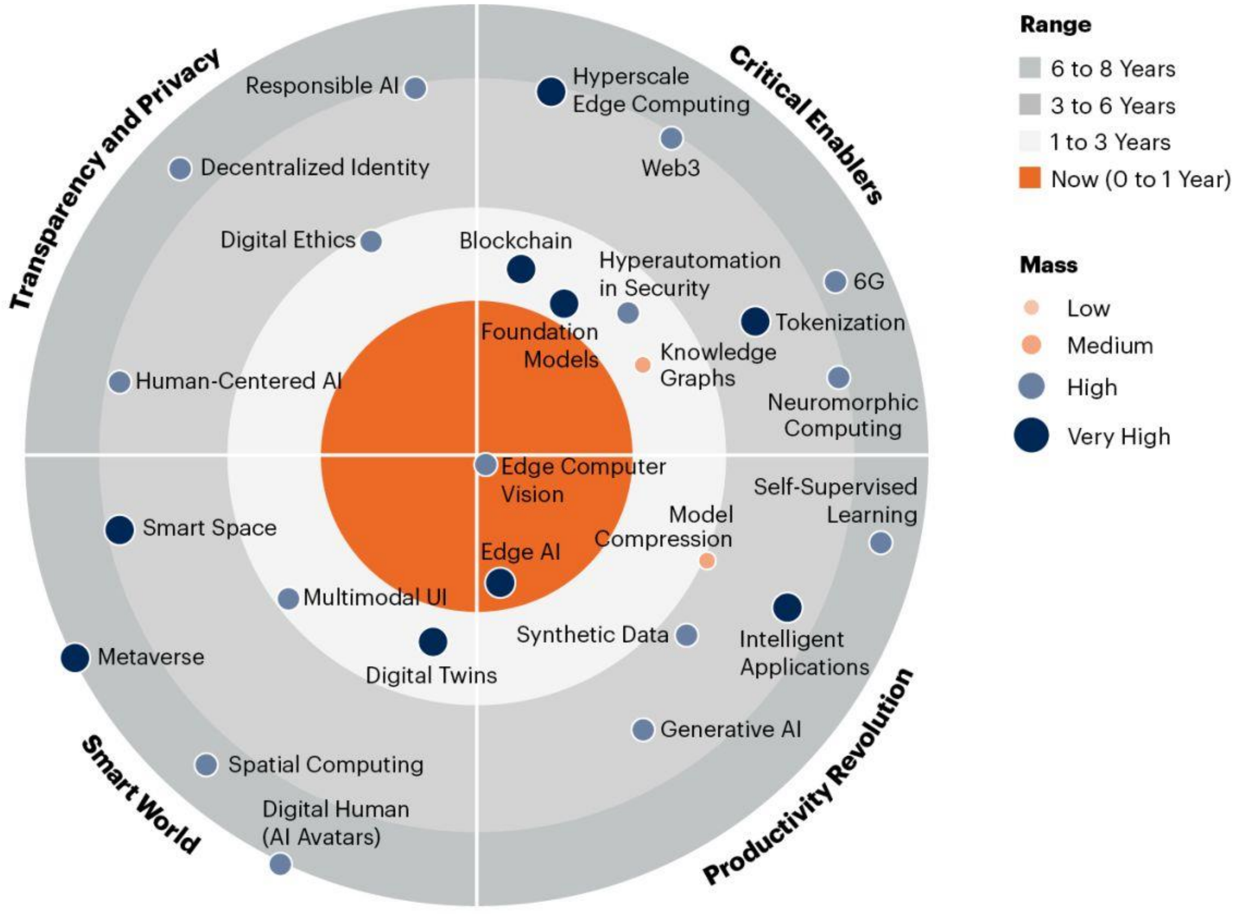
tinyML EMEA Forum - On Device Learning 9/12 , 2022 Cyprus, In person

TINY ML
On device learning Forum

- A framework of algorithms and associated tool for on-device tiny learning, Danilo PAU, Technical Director, IEEE and ST Fellow, STMicroelectronics
- In Sensor and On-device Tiny Learning for Next Generation of Smart Sensors Michele MAGNO, Head of the Project-based learning Center, ETH Zurich, D-ITET
- Continual On-device Learning on Multi- Core RISC-V MicroControllers Manuele RUSCI, Embedded Machine Learning Engineer, Greenwaves
- On-device continuous event-driven deep learning to avoid model drift, Bijan MOHAMMADI, CSO, Bondzai

On device learning Forum

2023 Gartner Emerging Technologies and Trends Impact Radar



gartner.com

Note: Range measures number of years it will take the technology/trend to cross over from early adopter to early majority adoption. Mass indicates how substantial the impact of the technology or trend will be on existing products and markets.

Source: Gartner

Gartner

On Device Learning Forum 2023, May 16 2023

- 8:00 - 8:10 Opening remarks by **Danilo Pau**
- 8:10 - 8:40 **Charlotte Frenkel** "Merging insights from artificial and biological neural networks for neuromorphic edge intelligence"
- 8:40 - 9:40 **Giorgia Dellaferrera** "Forward Learning with Top-Down Feedback: Solving the Credit Assignment Problem without a Backward Pass"
- 9:40 - 10:10 **Guy Paillet** "NeuroMem®, Ultra Low Power hardwired incremental learning and parallel pattern recognition"
- 10:10 - 10:40 **Aida Todri-Sanial** "On-Chip Learning and Implementation Challenges with Oscillatory Neural Networks"
- 10:40 - 11:10 **Eduardo S. Pereira** “Online Learning TinyML for Anomaly Detection Based on Extreme Values Theory”
- 11:10 - 11:15 Closing remarks by Danilo Pau

Pacific Time

life.augmented

TINY ML
On device learning Forum

Thank you, **tinyML Strategic Partners**,
for committing to take tinyML to the next Level, together

TINY ML

On device learning Forum

Executive Strategic Partners

TINY ML
On device learning Forum

EDGE IMPULSE

The Leading Development Platform for Edge ML

edgeimpulse.com

Qualcomm
AI research

Advancing AI research to make efficient AI ubiquitous

Power efficiency

Model design, compression, quantization, algorithms, efficient hardware, software tool

Personalization

Continuous learning, contextual, always-on, privacy-preserved, distributed learning

Efficient learning

Robust learning through minimal data, unsupervised learning, on-device learning

A platform to scale AI across the industry

Perception

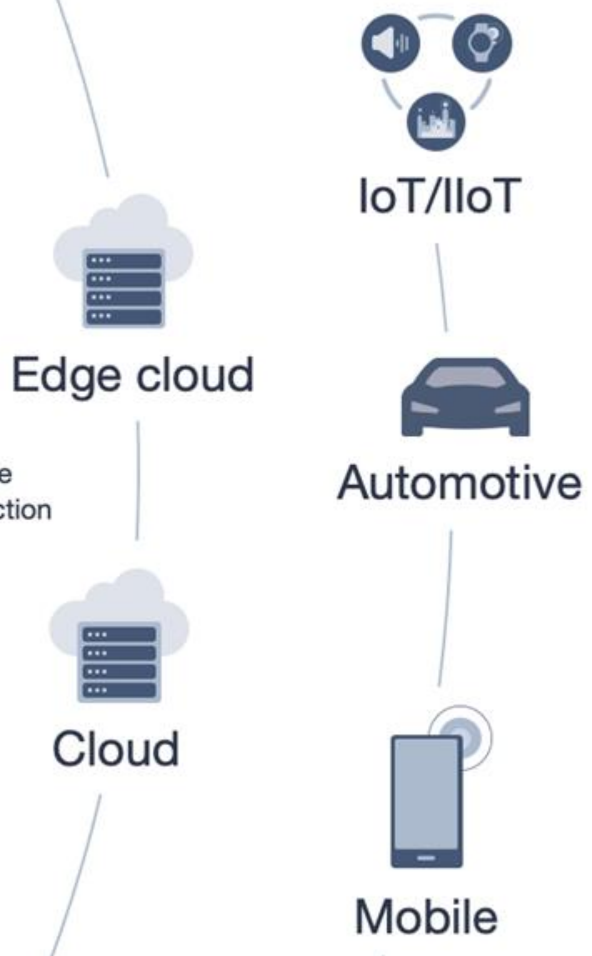
Object detection, speech recognition, contextual fusion

Reasoning

Scene understanding, language understanding, behavior prediction

Action

Reinforcement learning for decision making



Accelerate Your Edge Compute

SYNTIANT

Making Edge AI A Reality

www.syntiant.com

TINY ML
On device learning Forum

Platinum Strategic Partners

Renesas is enabling the next generation of AI-powered solutions that will revolutionize every industry sector.

renesas.com

BIG IDEAS FOR EVERY SPACE

AITRIOS

DEPLOY VISION AI
AT THE EDGE AT SCALE

SONY

TINY

On device learning Forum

Gold Strategic Partners

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Where what if becomes what is.

Witness potential made possible at analog.com.


ARDUINO
PRO
Easily deploy your tinyML solutions with Arduino Pro
arduino.cc/pro
Made In Italy

arm AI

Powering tinyML Innovation

Arm AI Virtual Tech Talks

The latest in AI trends, technologies & best practices from Arm and our Ecosystem Partners.

Demos, code examples, workshops, panel sessions and much more!

Fortnightly Tuesday @ 4pm GMT/8am PT

Find out more:
www.arm.com/techtalks

Driving decarbonization and digitalization. Together.

Infineon serving all target markets as
Leader in Power Systems and IoT

www.infineon.com

innatera

NEUROMORPHIC INTELLIGENCE FOR THE SENSOR-EDGE

www.innatera.com

Microsoft


The Right Edge AI Tools Can Make or Break Your Next Smart IoT Product
SensiML
Analytics Toolkit Suite
AutoML
Data Collection
Test & Validation
Data Labeling
Team Collaboration
Version Control and Model Management
sensiml.com/tinyML

STMicroelectronics provides extensive solutions to make tiny Machine Learning easy

life.augmented

www.st.com/ai

synaptics®

ENGINEERING EXCEPTIONAL EXPERIENCES

We engineer exceptional experiences for consumers in the home, at work, in the car, or on the go.

www.synaptics.com

TINY ML

On device learning Forum

Silver Strategic Partners

AIZIP

brainchip
The Neuromorphic Computing Company

GREENWAVES TECHNOLOGIES

Grovety Inc.

IBM

imagimob

Nota AI

NXP

POLYN TECHNOLOGY

Qeexo

Schneider Electric

SILICON LABS

Join Growing tinyML Communities:

meetup

14.7k members in
47 Groups in 39 Countries

tinyML - Enabling ultra-low Power ML at the Edge

https://www.meetup.com/tinyML-Enabling-ultra-low-Power-ML-at-the-Edge/

4k members
&
11.6k followers

The tinyML Community

https://www.linkedin.com/groups/13694488/

TINY ML

TINY ML

On device learning Forum

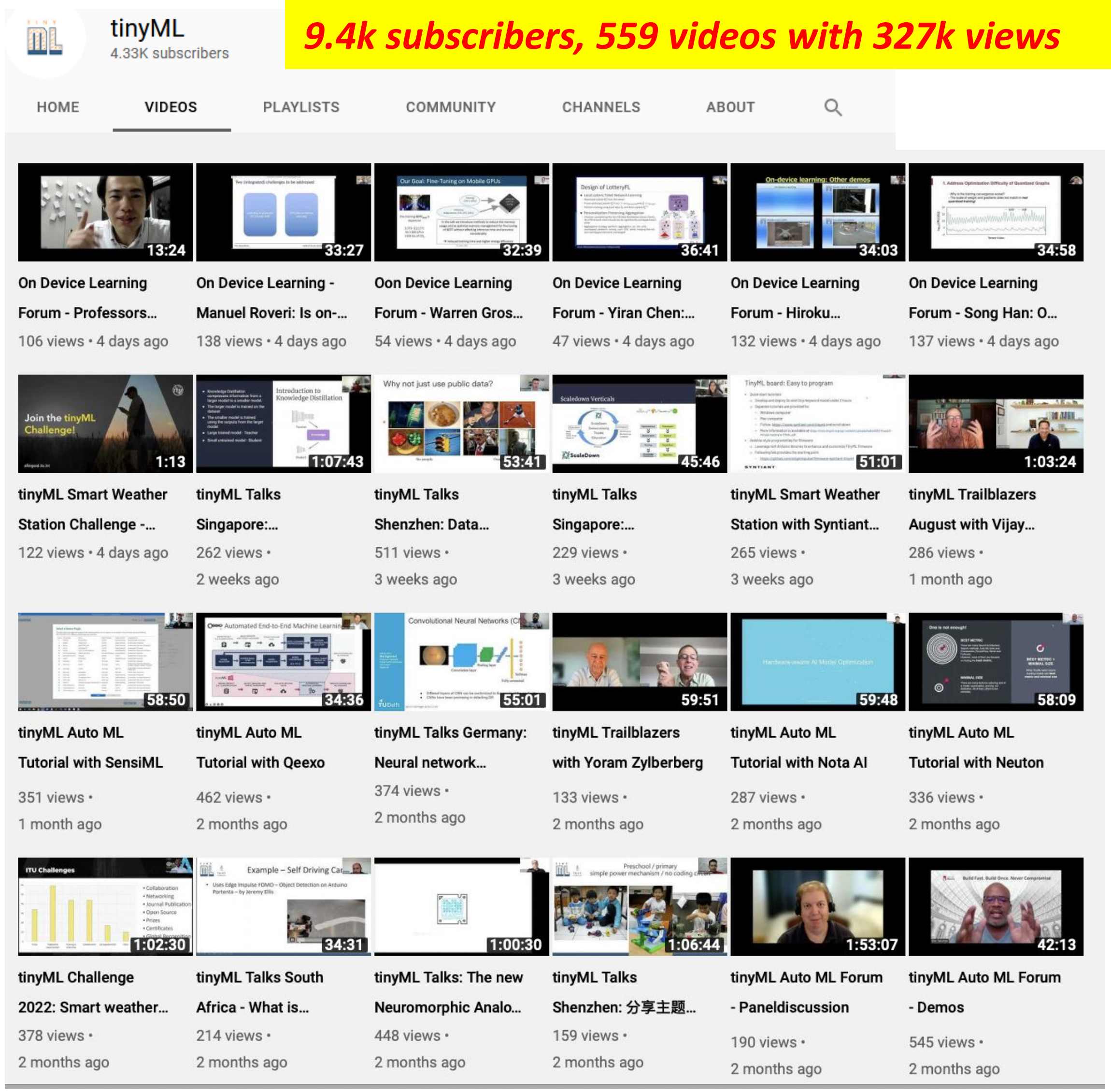
Subscribe to tinyML YouTube Channel for updates and notifications

(including this video)

www.youtube.com/tinyML

YouTube

9.4k subscribers, 559 videos with 327k views



On device learning Forum

FOUNDATION

EMEA 2023

https://www.tinyml.org/event/emea-2023

More sponsorships are available: sponsorships@tinyML.org

TINY ML

On device learning Forum

Reminders

Slides & Videos will be posted tomorrow

tinyml.org/forums youtube.com/tinyml

Please use the Q&A window for your questions

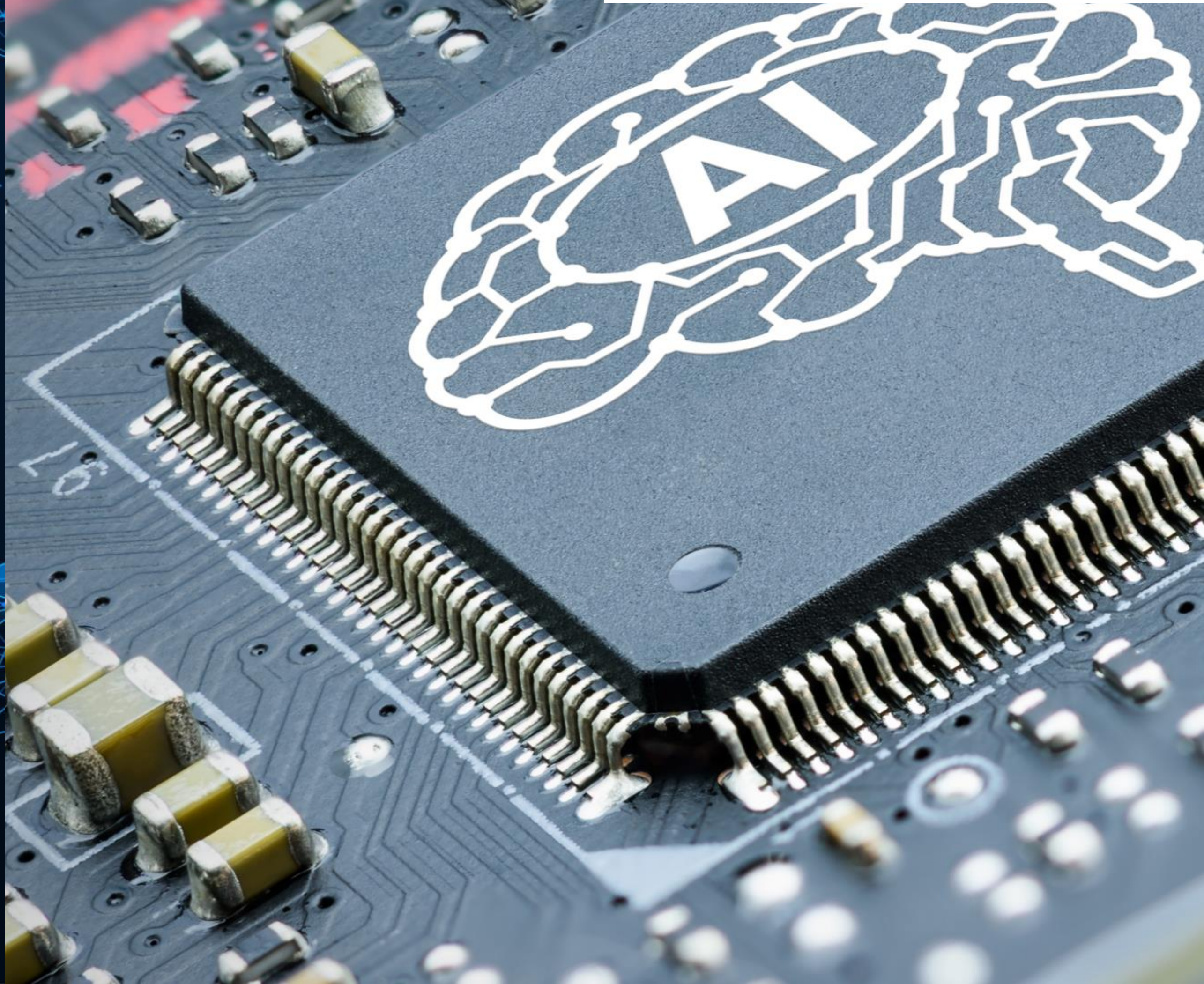
On device learning Forum

Giorgia Dellaferrera

Giorgia Dellaferrera has completed her PhD in computational neuroscience at the Institute of Neuroinformatics (ETH Zurich and the University of Zurich) and IBM Research Zurich with Prof. Indiveri, Prof. Eleftheriou and Dr. Pantazi. Her doctoral thesis focused on the interplay between neuroscience and artificial intelligence, with an emphasis on learning mechanisms in brains and machines. During her PhD, she visited the lab of Prof. Kreiman at the Harvard Medical School (US), where she developed a biologically inspired training strategy for artificial neural networks. Before her PhD, Giorgia obtained a master in Applied Physics at the Swiss Federal Institute of Technology Lausanne (EPFL) and worked as an intern at the Okinawa Institute of Science and Technology, Logitech, Imperial College London, and EPFL.

Forward Learning with Top-Down Feedback: Solving the Credit Assignment Problem without a Backward Pass

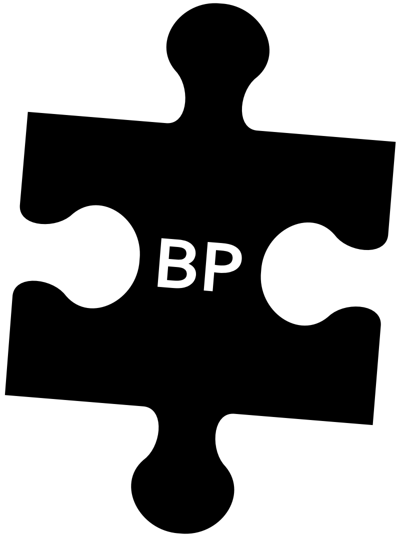
Dellaferrera & Kreiman, ICML 2022
Srinivasan, Mignacco, Sorbaro, Cooper, Refinetti, Kreiman, Dellaferrera, 2023 (arXiv:2302.05440)



TINY ML FOUNDATION
HARVARD MEDICAL SCHOOL
Boston Children's Hospital
Until every child is well
University of Zurich UZH
ETH zürich
institute of neuroinformatics
IBM Research | Zurich

Connecting the puzzle pieces of bio-inspired learning algorithms

1989



Weight symmetry!

Non local!

Activity is frozen!

Update locking!

Biologically unrealistic!

NATURE VOL. 337 12 JANUARY 1989 — COMMENTARY — 129

The recent excitement about neural networks

Francis Crick

Backpropagation and the brain

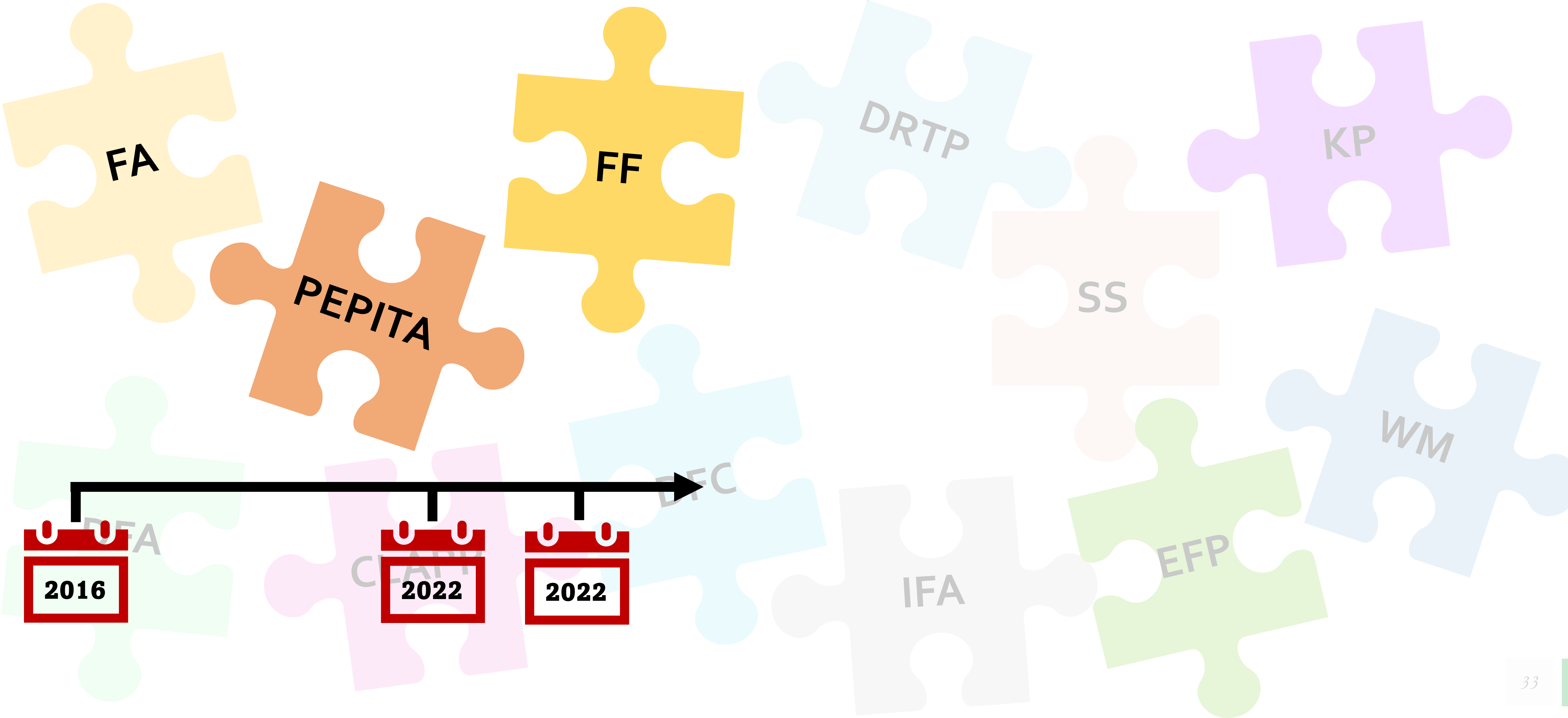
Timothy P. Lillicrap, Adam Santoro, Luke Marris, Colin J. Akerman and Geoffrey Hinton

Review

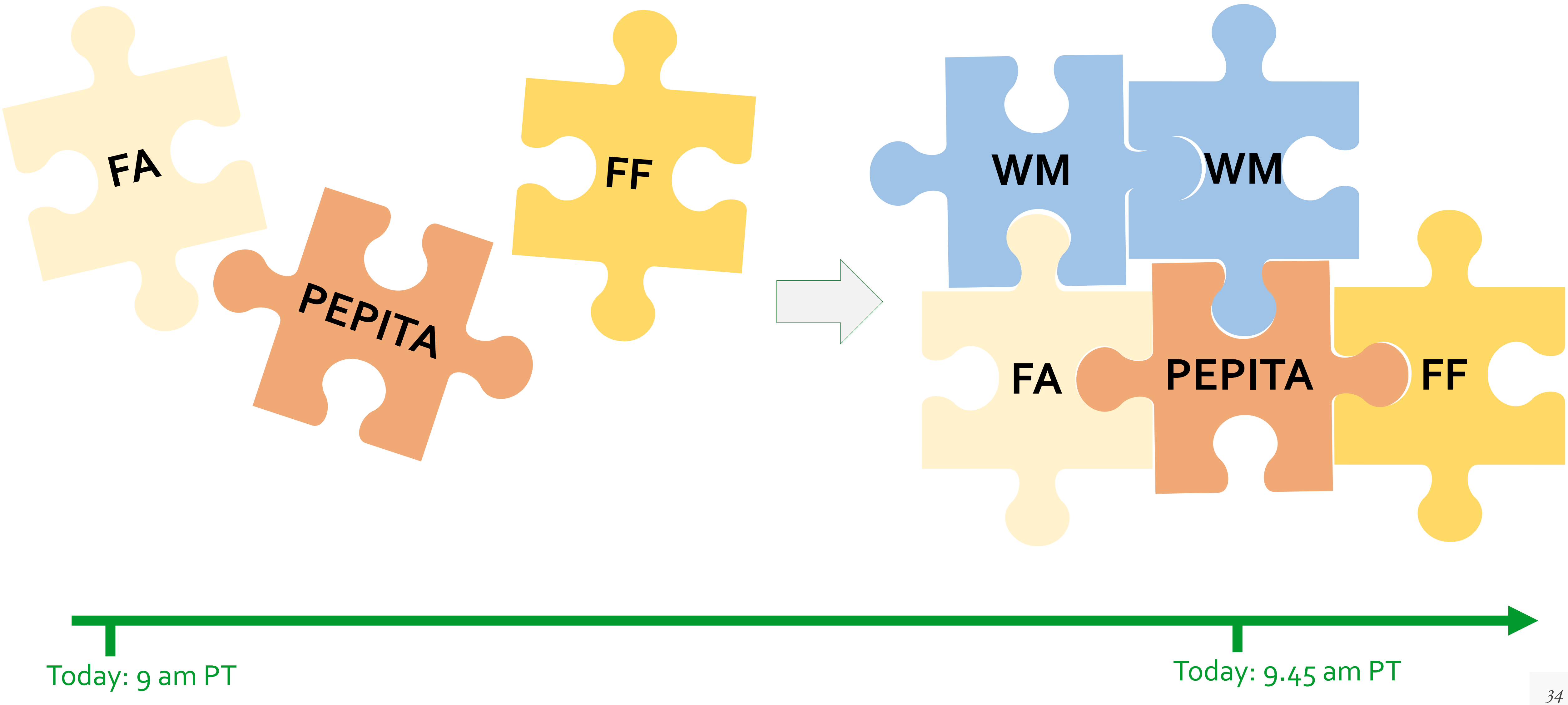
Theories of Error Back-Propagation in the Brain

James C.R. Whittington[1,2] and Rafal Bogacz[1,*]

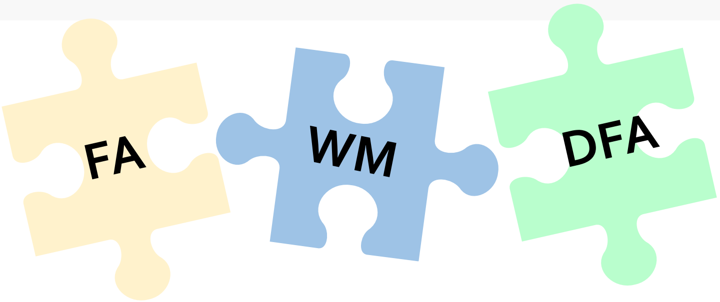
Connecting the puzzle pieces of bio-inspired learning algorithms



Connecting the puzzle pieces of bio-inspired learning algorithms



Outline

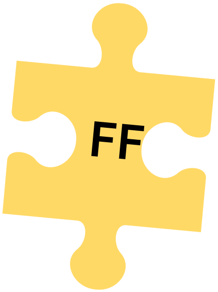


» Neuro-inspired AI

- Why Backpropagation is biologically implausible
- Overview of alternative solutions to credit assignment

» PEPITA: error-driven input modulation

- Replacing the backward pass with a second forward pass
- Results on image classification tasks
- Soft alignment dynamics
- Approximating PEPITA to *Adaptive Feedback Alignment*: analytical characterization
- Improving alignment with weight mirroring

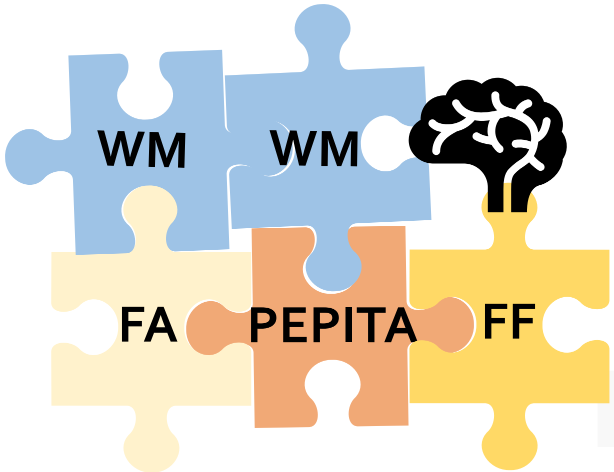


» Forward-Forward algorithm

- Idea and results
- Similarities with PEPITA's update rule

» Forward learning with top-down feedback

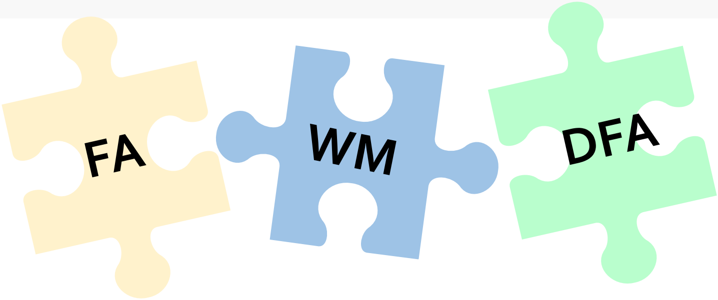
- Biological considerations



Outline

» **Neuro-inspired AI**

- Why Backpropagation is biologically implausible
- Overview of alternative solutions to credit assignment



» **PEPITA: error-driven input modulation**

- Replacing the backward pass with a second forward pass
- Results on image classification tasks
- Soft alignment dynamics
- Approximating PEPITA to *Adaptive Feedback Alignment*: analytical characterization
- Improving alignment with weight mirroring

» **Forward-Forward algorithm**

- Idea and results
- Similarities with PEPITA's update rule

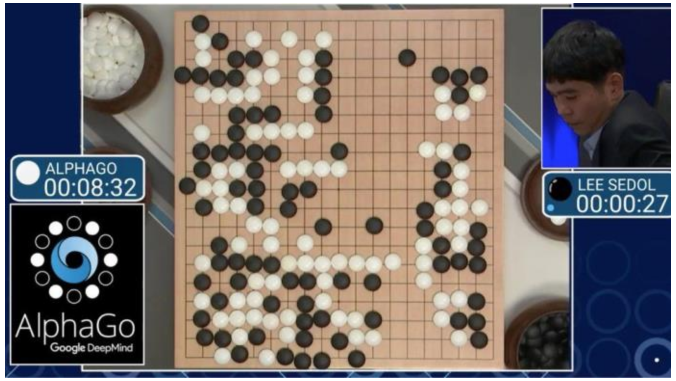
» **Forward learning with top-down feedback**

- Biological considerations

Backpropagation: successful but not biologically plausible

» **Success**
- The most effective training algorithm
- State-of-the-art performance in complex cognitive tasks

» **Algorithm**
- Chain rule of calculus
- Change in synaptic strength $\longleftrightarrow$ change of network's error



https://www.bbc.com/news/technology-35785875

Li et al., 2017

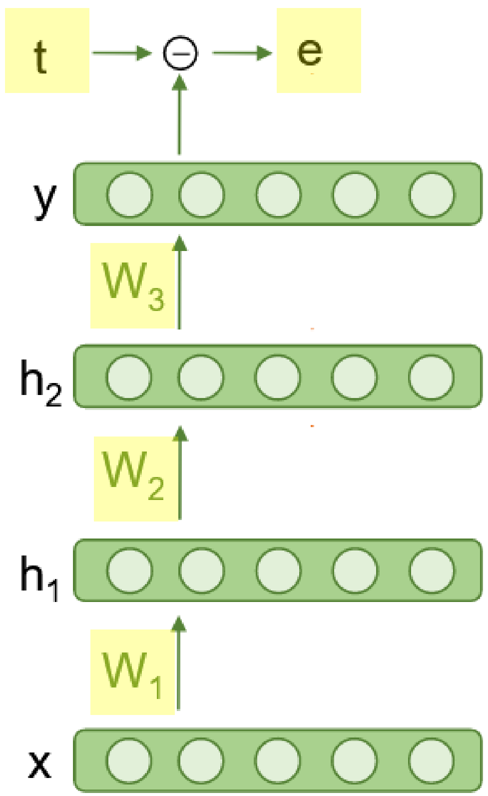
Rumelhart et al., 1989 Crick, 1989 Lillicrap et al., 2020

Glossary

» **Target (t)**
- Desired output of a network

» **Error (e)**
- Deviation of the network's output from the target

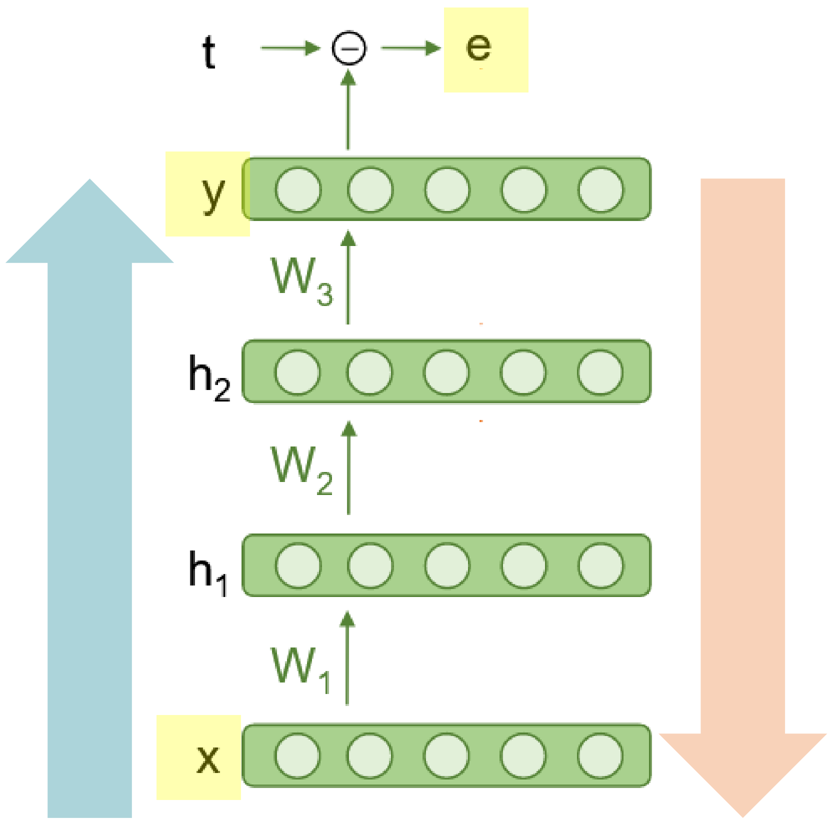
» **Weights (W)**
- Parameter corresponding to the strength of the connection between two nodes



The backpropagation algorithm

- **Forward pass**
  - Network's response to input
  - Error function $e = y - t$
  - Weight updates proportional to its negative gradient

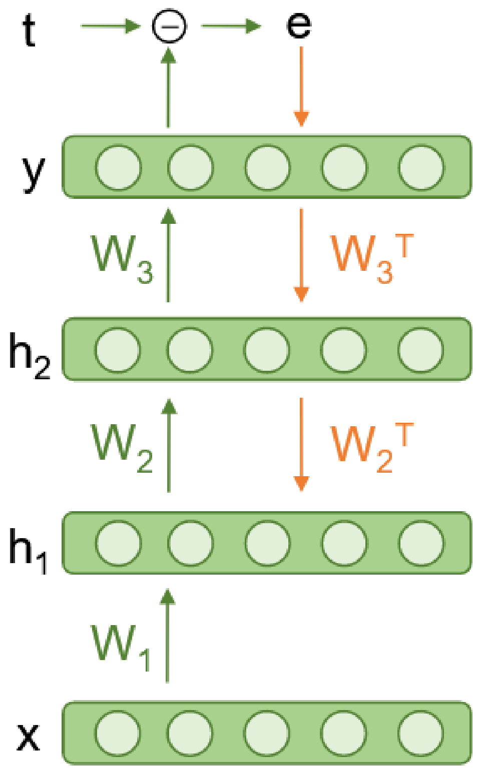
- **Backward pass**
  - Error signal flows backward through the network
  - Computed recursively via the chain rule
  - Update phase



Lillicrap et al., 2020

Backpropagation of the error is not biologically plausible

BP



Rumelhart et al., 1995

» **Weight transport problem**
- Symmetric weights for forward and backward computation

» **Non-local information used for the updates**
- Global error and downstream weights needed for learning

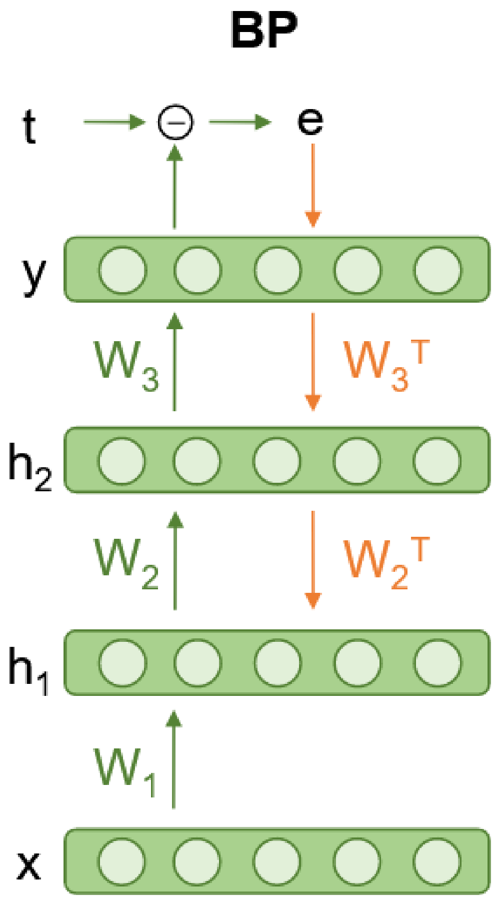
» **Frozen activity during error propagation and parameter updates**
- Separate forward and backward computation

» **Update locking problem**
- Backward computation needs to be complete before the next forward pass

Alternative Training Schemes

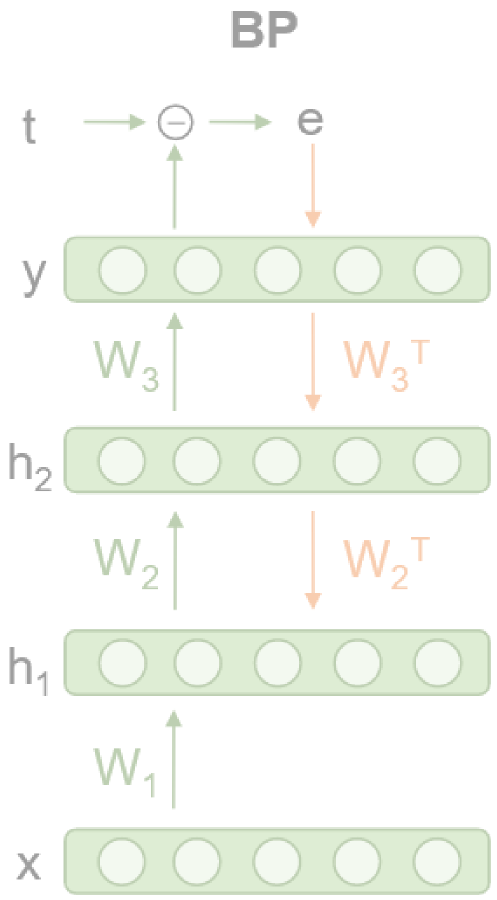
Lillicrap et al., 2020

Alternatives to BP: relaxing symmetry requirements

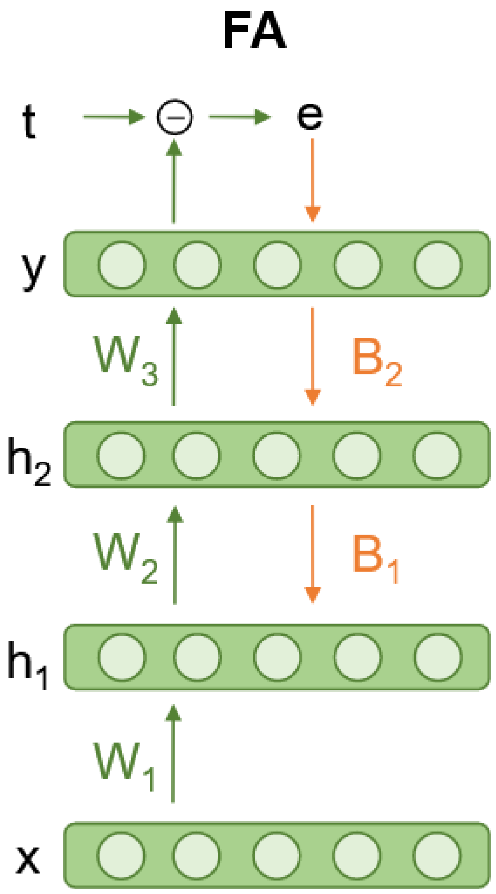


Rumelhart et al., 1995

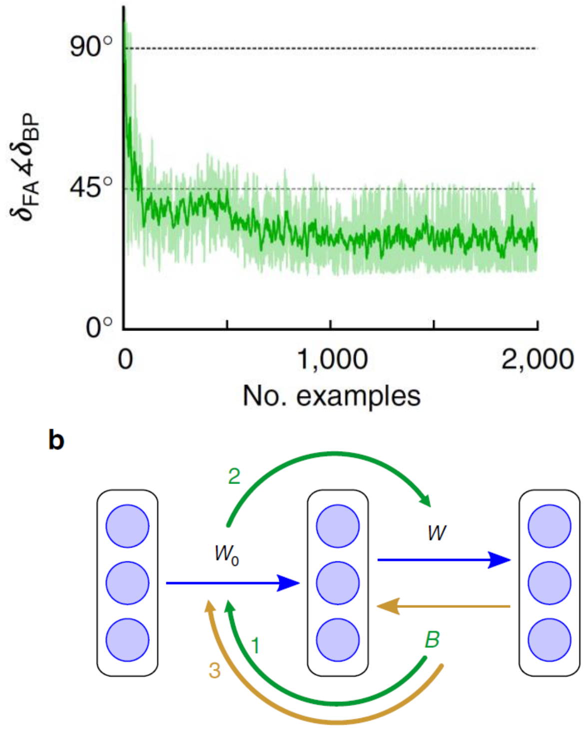
Alternatives to BP: relaxing symmetry requirements



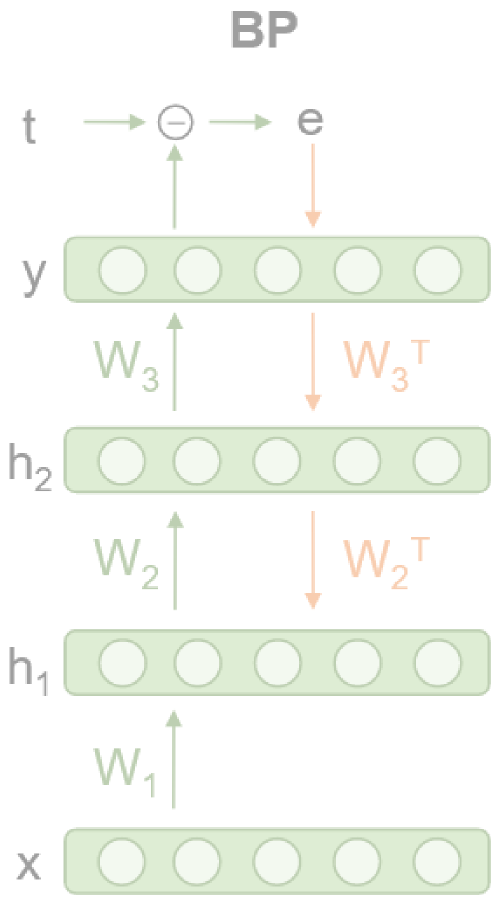
Rumelhart et al., 1995



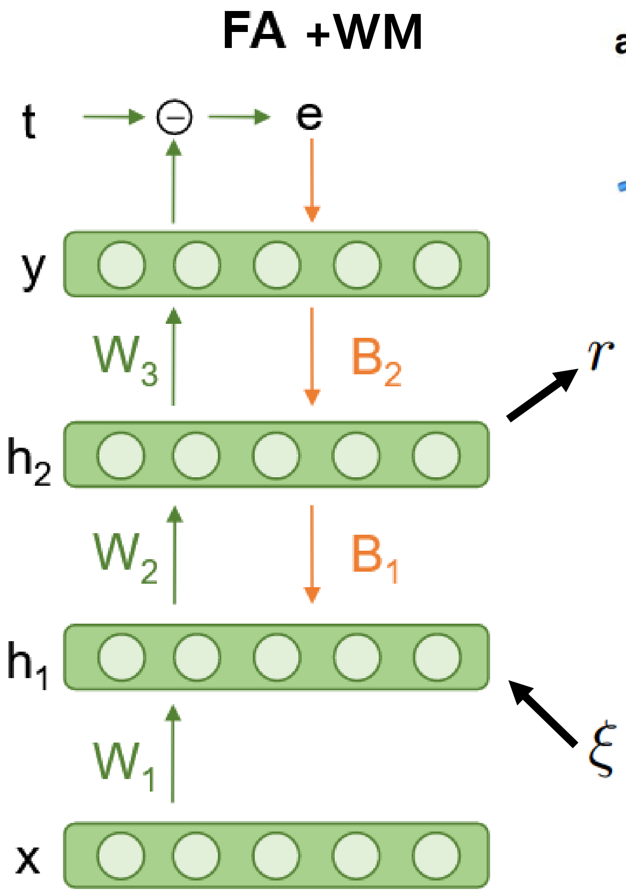
Lillicrap et al., 2016



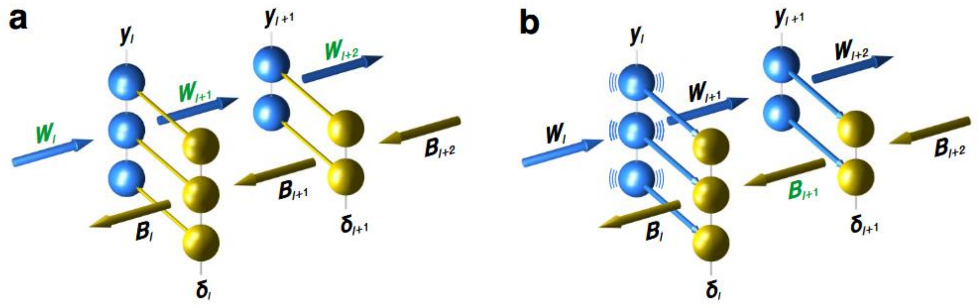
Alternatives to BP: relaxing symmetry requirements



Rumelhart et al., 1995

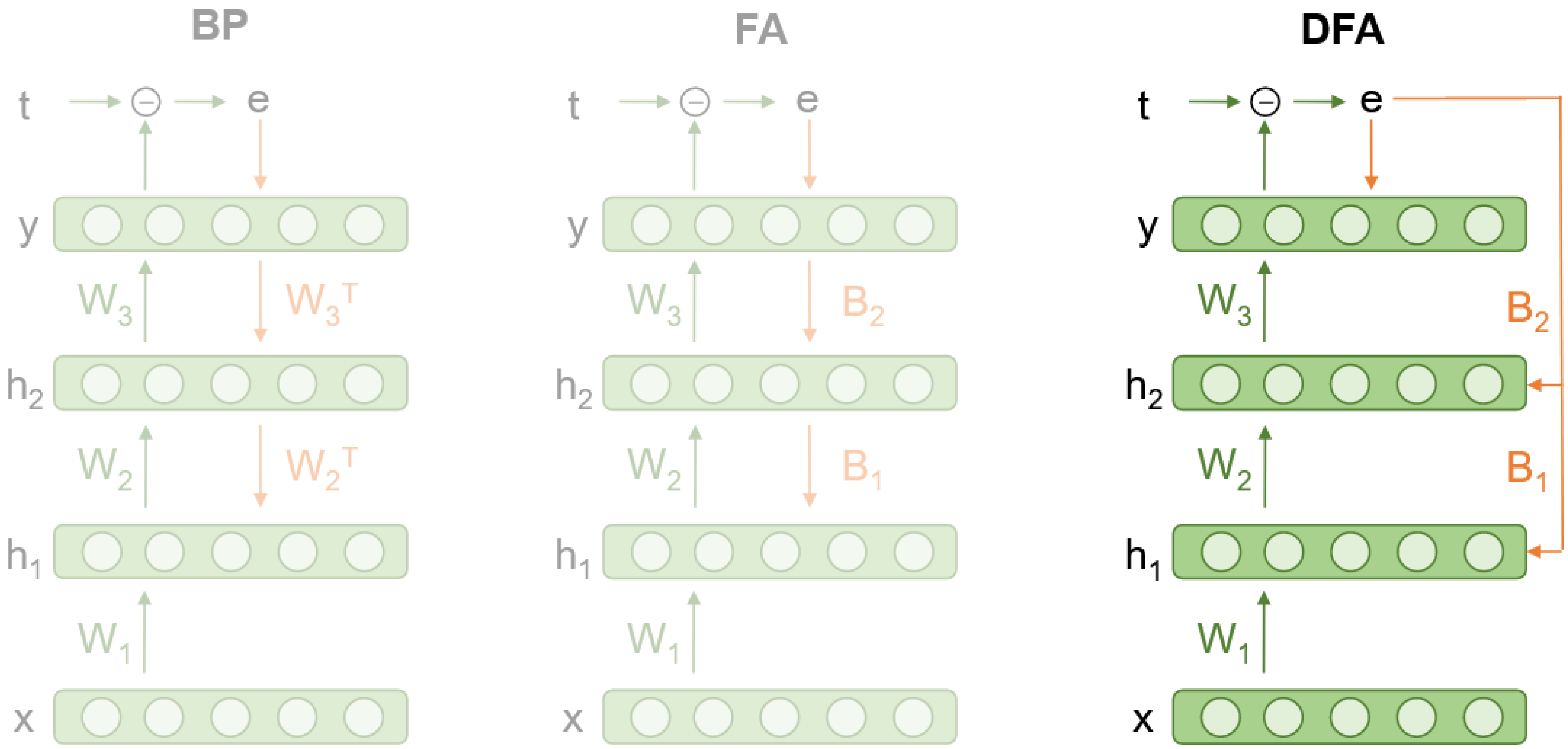


Lillicrap et al., 2016



Akrout et al., 2019

Alternatives to BP: relaxing symmetry requirements

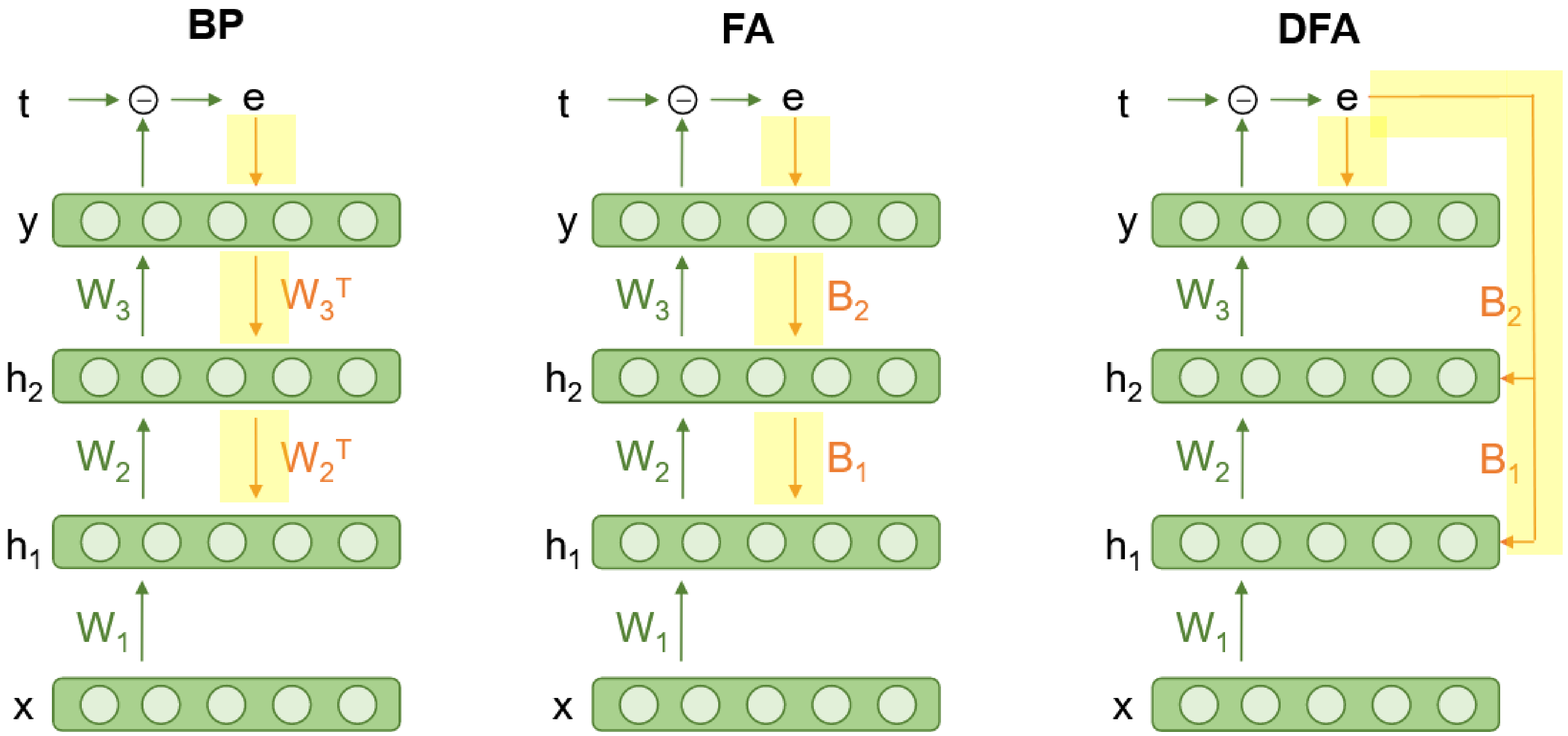


Rumelhart et al., 1995

Lillicrap et al., 2016

A. Nokland, 2016

Alternatives to BP: relaxing symmetry requirements



Rumelhart et al., 1995

Lillicrap et al., 2016

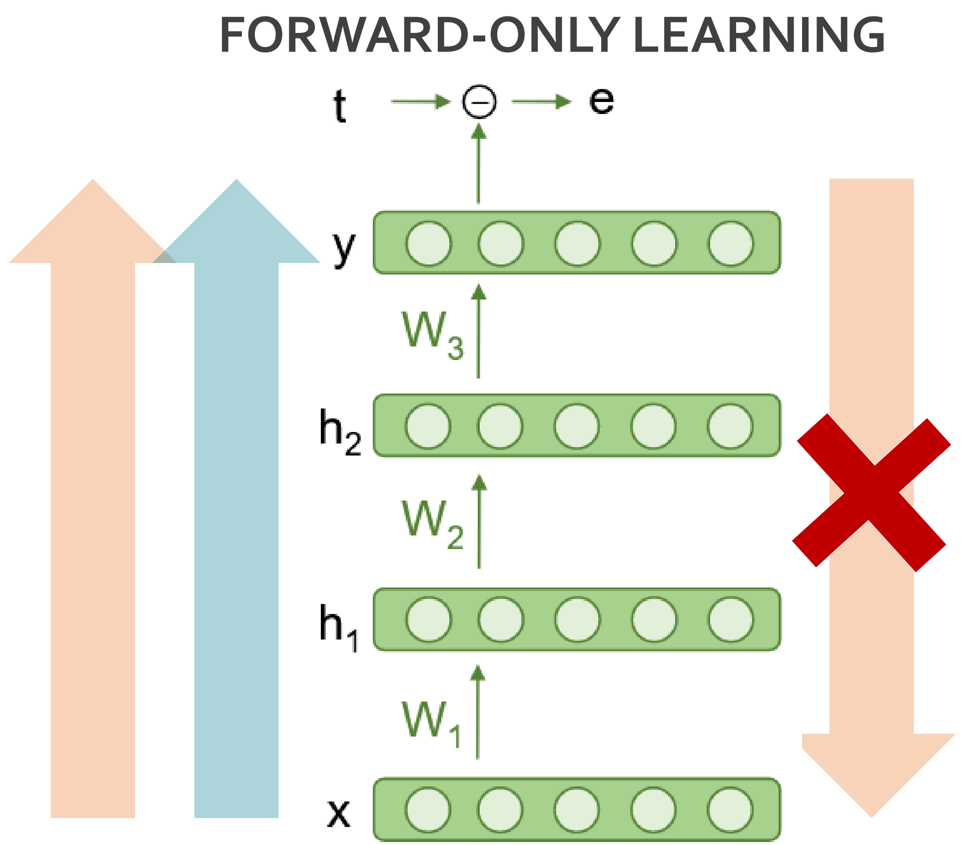
A. Nokland, 2016

The Backward Pass

The backward pass implies:

- Weight updates relying on **non-local** information
- **Freezing activity** for the update phase
- At least **partial update locking**

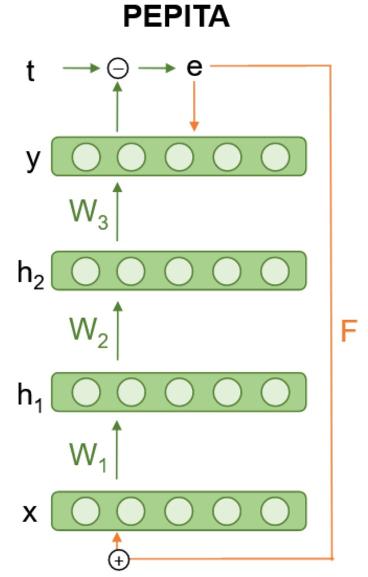
Remove the backward pass



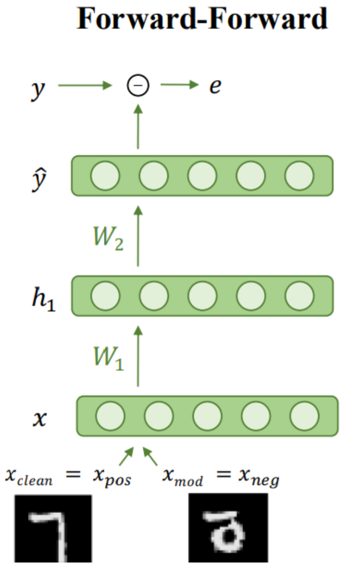
Dellaferrera & Kreiman, ICML (2022)

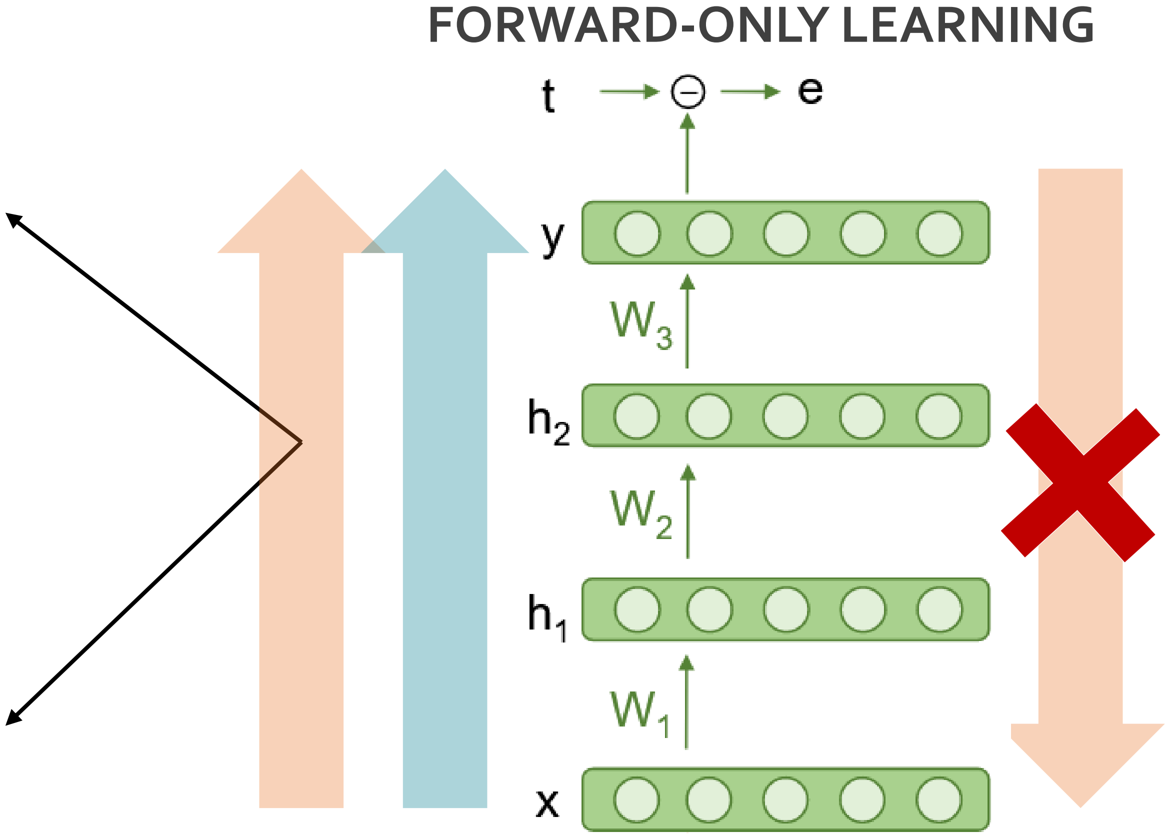
The Backward Pass

- PEPITA: Top down feedback connections



- FF: Input sample corruption





Dellaferrera & Kreiman, ICML (2022)
Hinton, 2022

Outline

» Neuro-inspired AI
- Why Backpropagation is biologically implausible
- Overview of alternative solutions to credit assignment

» **PEPITA: error-driven input modulation**
- Replacing the backward pass with a second forward pass
- Results on image classification tasks
- Soft alignment dynamics
- Approximating PEPITA to *Adaptive Feedback Alignment*: analytical characterization
- Improving alignment with weight mirroring

» Forward-Forward algorithm
- Idea and results
- Similarities with PEPITA's update rule

» Forward learning with top-down feedback
- Biological considerations

The PEPITA learning rule

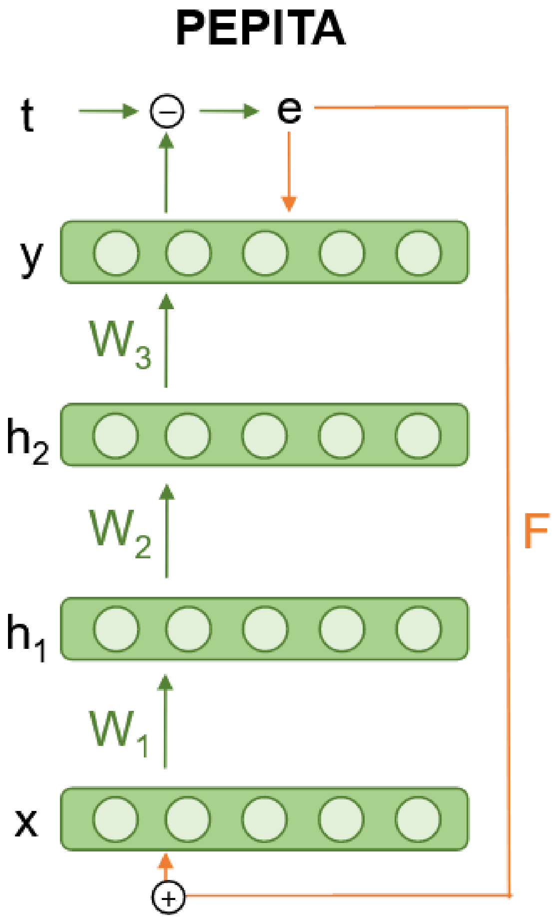
» **PEPITA** = Present the Error to Perturb the Input To modulate Activity

» Substitutes the standard Forward+Backward scheme with **two Forward Passes**

- *Standard Forward pass* → same as for standard algorithms
- *Modulated Forward pass* → input is modulated by the error

» F = **projection matrix** to add the error onto the input

» Update relies **on difference of activations** between *Standard* and *Modulated pass*



The PEPITA learning rule for Fully Connected Neural Networks

» **PEPITA** = Present the Error to Perturb the Input To modulate Activity

Algorithm 1 Implementation of PEPITA

Given: Input (x) and label ($target$)

#standard forward pass

$h_0 = x$

for $\ell = 1, ..., L$

$\quad h_\ell = \sigma_\ell(W_\ell h_{\ell-1})$

$e = h_L - target$ — » Present the Error ...

#modulated forward pass

$h_0^{err} = x + Fe$ — » ... to Perturb the Input...

for $\ell = 1, ..., L$

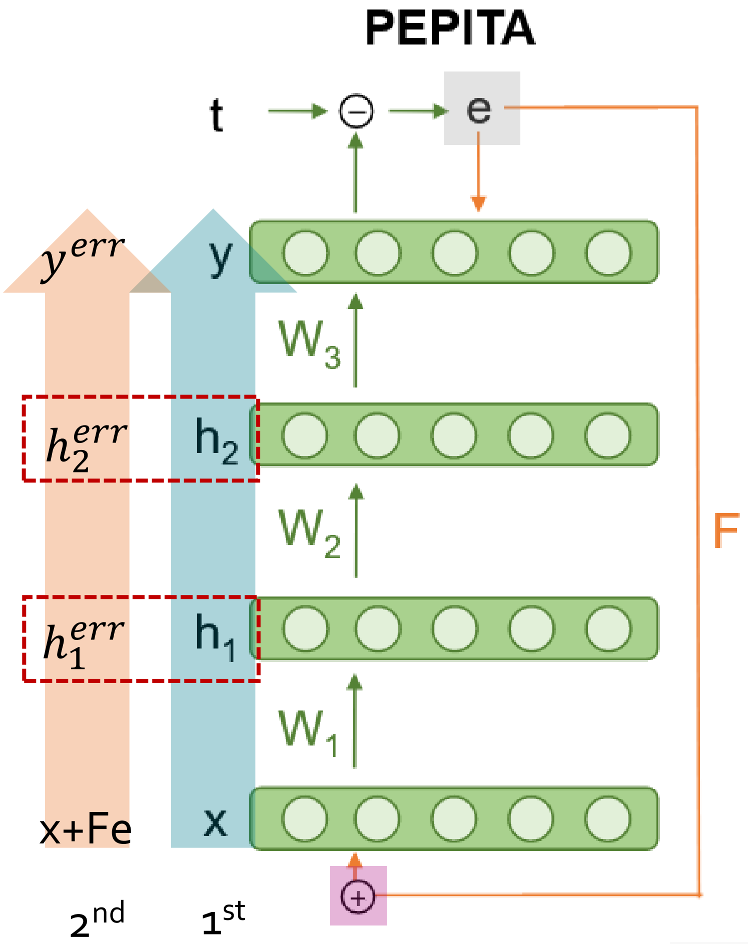
$\quad h_\ell^{err} = \sigma_\ell(W_\ell h_{\ell-1}^{err})$ — » ... To modulate Activity

$\quad$ **if** $\ell < L$:

$\quad\quad \Delta W_\ell = \boxed{(h_\ell - h_\ell^{err})} \cdot (h_{\ell-1}^{err})^T$

$\quad$ **else:**

$\quad\quad \Delta W_\ell = e \cdot (h_{\ell-1}^{err})^T$



Testing PEPITA on image classification tasks - experimental results

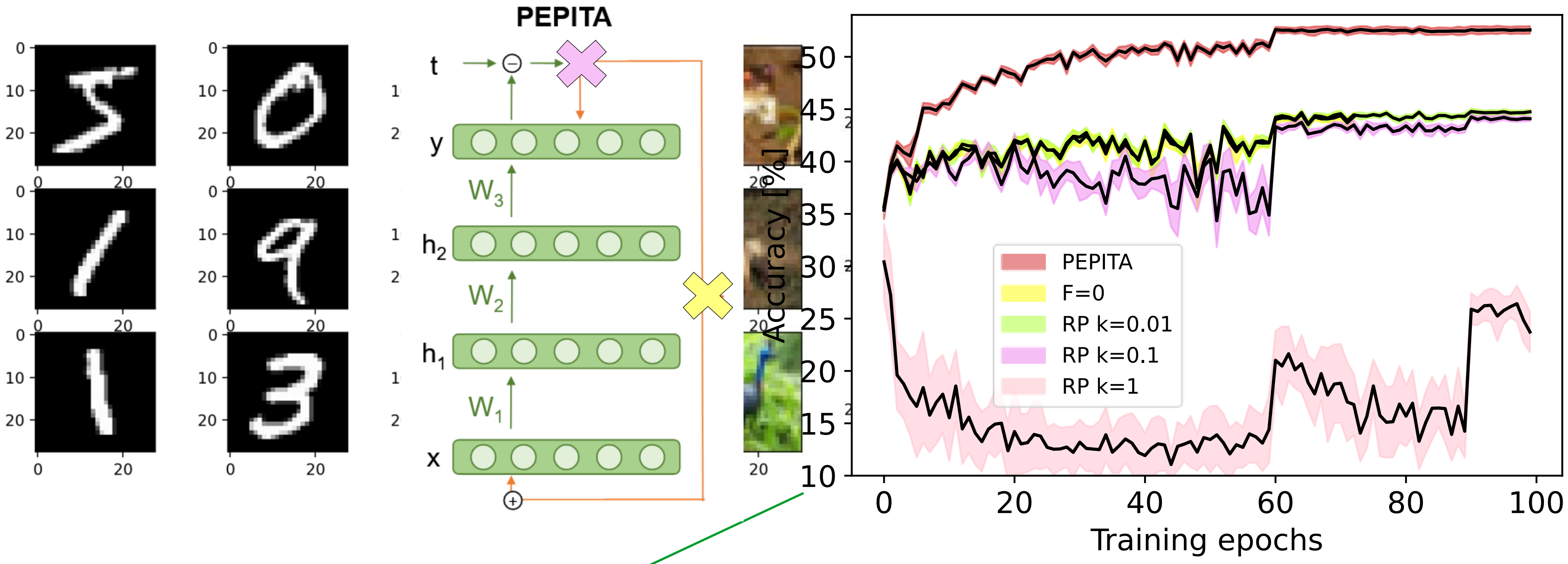
» Results for PEPIT

» In some tasks, PE

» PEPITA always o

» The PEPITA conv

- Useful 2D featu



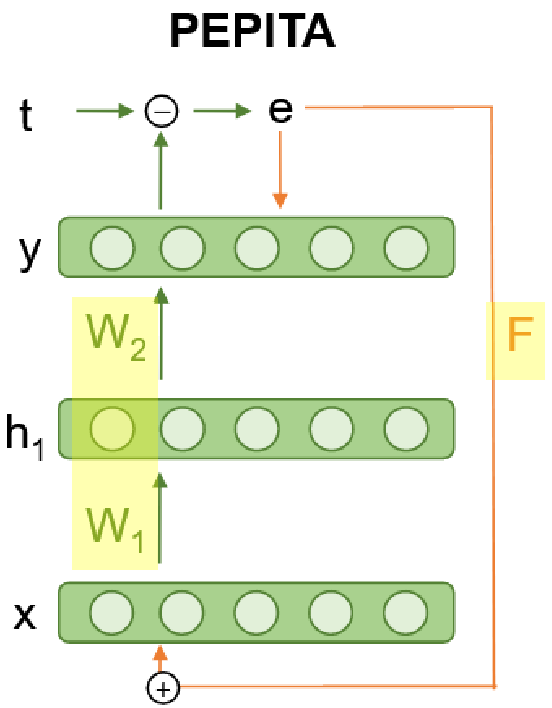
Architecture:
1 hidden layer +
1 output layer

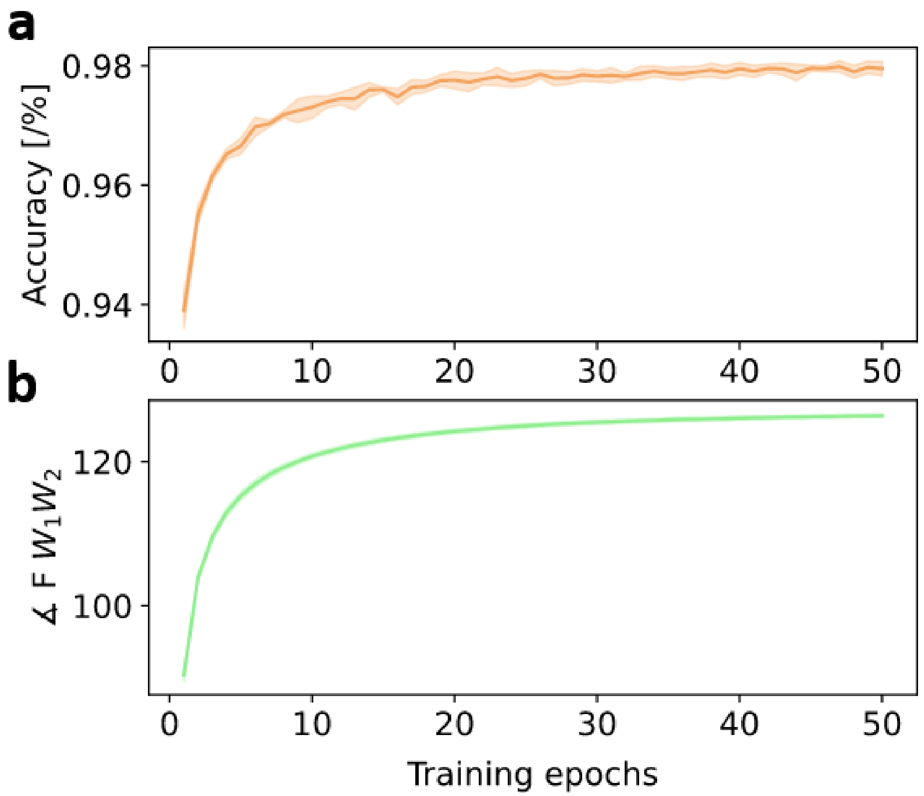
	Fully connected models			Convolutional models		
	MNIST	CIFAR10	CIFAR100	MNIST	CIFAR10	CIFAR100
BP						
FA						
DRTP						
PEPITA	98.01±0.09	52.57±0.36	24.91±0.22	98.29±0.13	56.33±1.35	27.56±0.60

Why it works: soft-antialignment

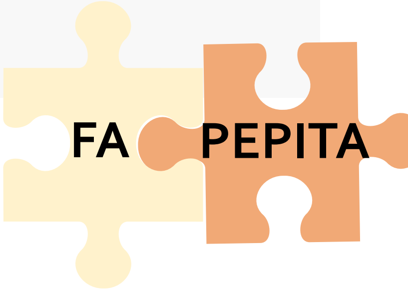
» **Soft-antialignment**

- Angle between
  - projection matrix F and
  - product between the forward weight matrices
- Evolution during learning → soft antialignment
- <u>Analytically proven</u> for one-hidden layer linear network





Approximating PEPITA to an Adaptive Feedback Alignment algorithm



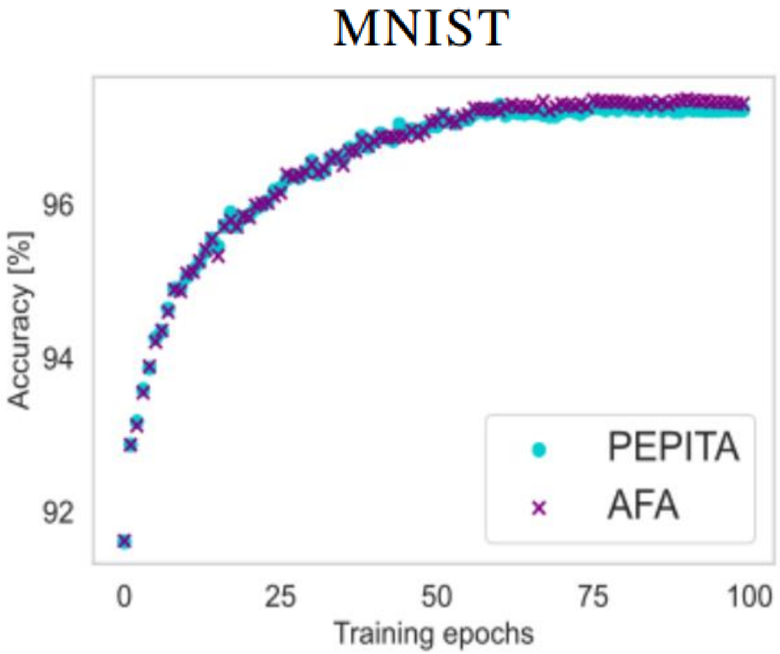
» First order Taylor expansion

$$\Delta W_\ell = (h_\ell - h_\ell^{err}) \cdot (h_{\ell-1}^{err})^T$$

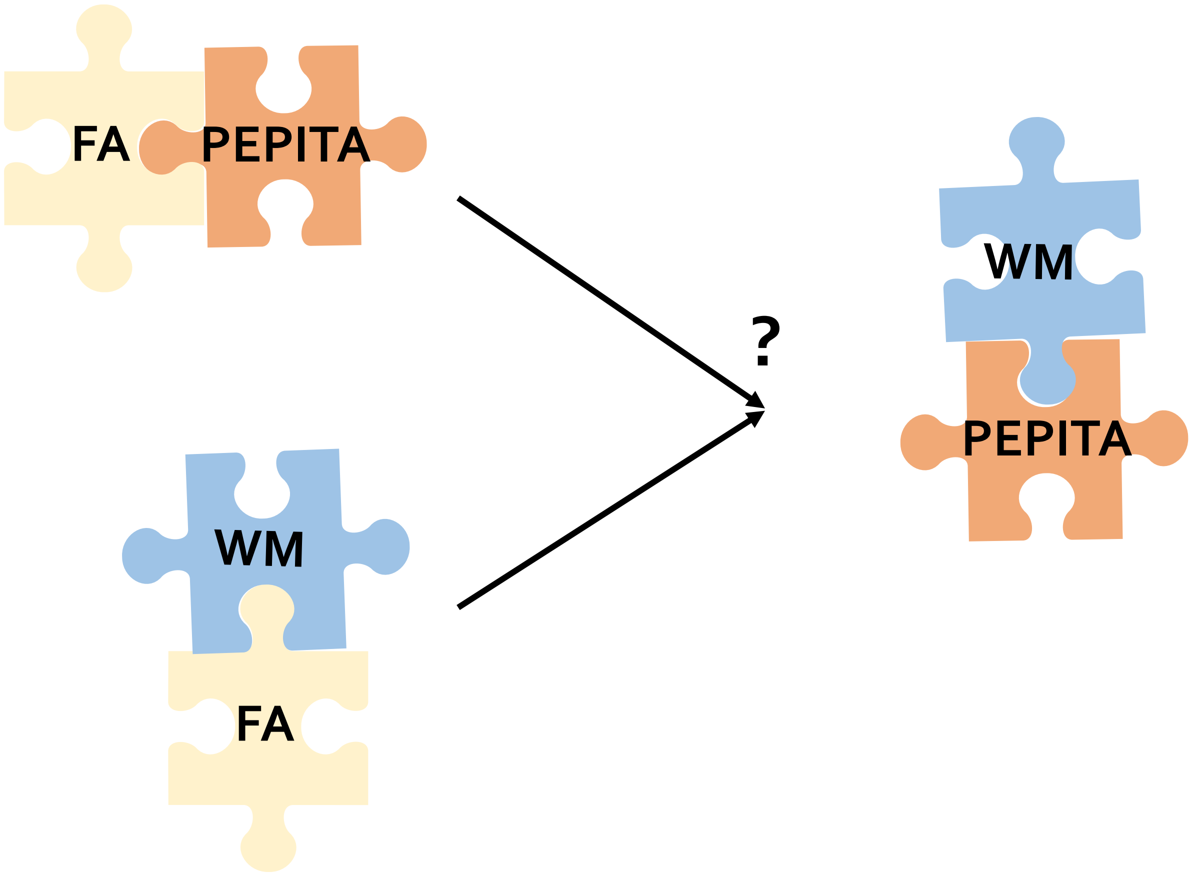
$$f(a+h) \simeq f(a) + h f'(a)$$

$$\begin{aligned}
h_1 - h_1^{err} &= \sigma_1(W_1 x) - \sigma_1(W_1(x - Fe)) = \\
&= \sigma_1(W_1 x) - \sigma_1(W_1 x - W_1 F e)) = \\
&\simeq \cancel{\sigma_1(W_1 x)} - [\cancel{\sigma_1(W_1(x))} - W_1 F e \sigma_1'(W_1 x)] = \\
&= W_1 F e \sigma_1'(W_1 x) = \\
&= \underbrace{W_1 \underbrace{F e}_{\text{FA}}} h_1'.
\end{aligned}$$

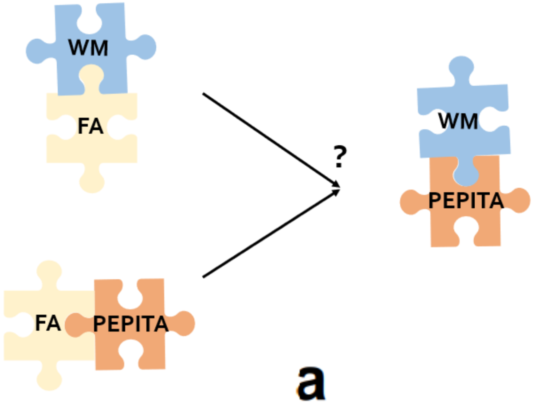
AFA = Adaptive (W) Feedback Alignment



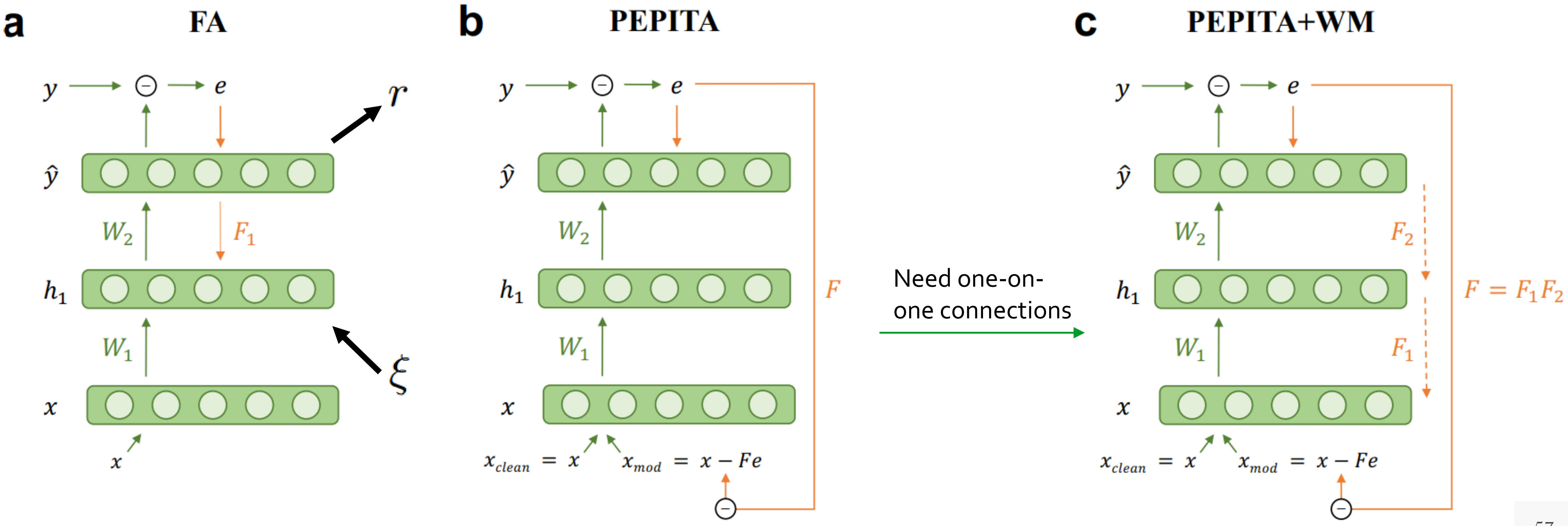
Improving PEPITA's alignment with weight mirroring



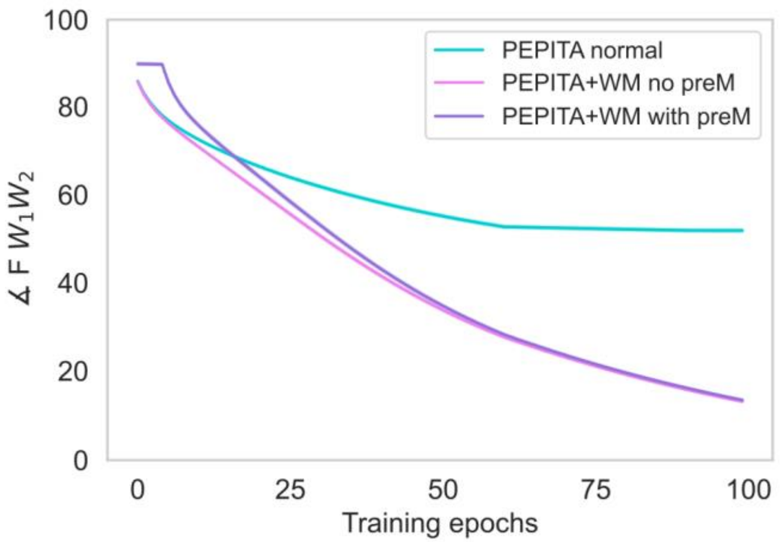
Improving PEPITA's alignment with weight mirroring



$$\Delta F_\ell = \eta_F \xi r^\top$$



Improving PEPITA's alignment with weight mirroring



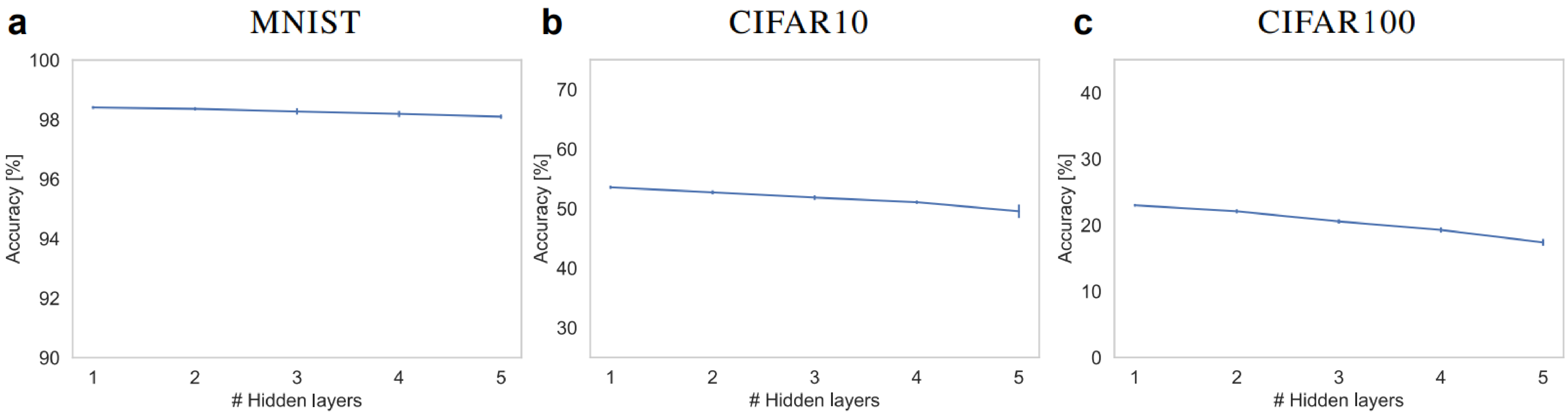
← Note: the sign is flipped x^{err}=x-Fe

Architecture:
1 hidden layer +
1 output layer

	W. Decay	Norm.	Mirror	MNIST	CIFAR10	CIFAR100
PEPITA	✗	✗	✗	98.02±0.08	52.45±0.25	24.69±0.17
	✓	✗	✗	98.12±0.08	53.05±0.23	24.86±0.18
	✗	✓	✗	**98.41**±0.08	**53.51**±0.23	22.87±0.25
	✗	✗	✓	98.05±0.08	52.63±0.30	**27.07**±0.11
	✓	✗	✓	98.10±0.12	53.46±0.26	**27.04**±0.19
	✗	✓	✓	**98.42**±0.05	**53.80**±0.25	24.20±0.36

Training deeper fully connected models

» Adding activation normalization allows to train up to 6 layer networks

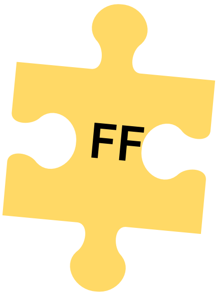


Outline

» **Neuro-inspired AI**
- Why Backpropagation is biologically implausible
- Overview of alternative solutions to credit assignment

» **PEPITA: error-driven input modulation**
- Replacing the backward pass with a second forward pass
- Results on image classification tasks
- Soft alignment dynamics
- Approximating PEPITA to *Adaptive Feedback Alignment*: analytical characterization
- Improving alignment with weight mirroring

» **Forward-Forward algorithm**
- Idea and results
- Similarities with PEPITA's update rule



» **Forward learning with top-down feedback**
- Biological considerations

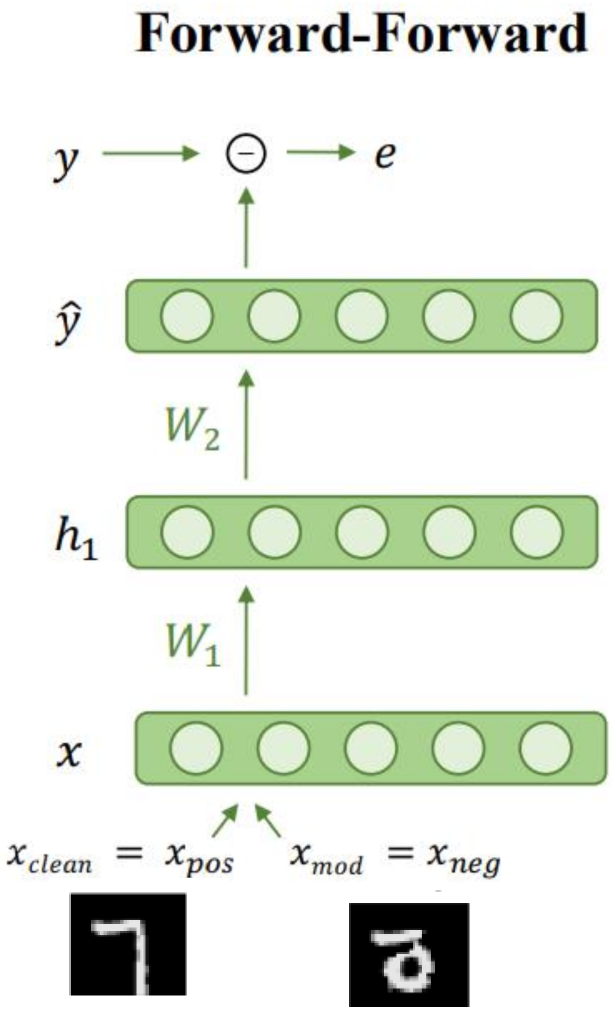
The Forward-Forward algorithm

» Two forward passes per sample:
- the positive pass operate on real data
- the negative pass operates on "negative data"

» In the positive pass:
- weights updated to increase the goodness in hidden layers

» In the negative pass:
- weights updated to decrease the goodness in hidden layers

» One measure of goodness
- sum of the squared neural activities



PEPITA: weight update equivalent to the Forward-Forward framework

Forward-Forward framework

- Goodness as the sum of squared neural activities
  - $\|h_\ell\|^2$ for the positive pass and
  - $\|h_\ell^{err}\|^2$ for the negative pass.
- Local loss function J_l for layer l = the sum of
  - loss function of the positive pass J^+_l and
  - loss function of the negative pass J^-_l

$$J_\ell = \|h_\ell\|^2 - \|h_\ell^{err}\|^2$$

Equivalence of weight update

$$\begin{aligned}
\frac{1}{2}\frac{\partial J_\ell}{\partial W_\ell} &= \frac{1}{2}\left(\frac{\partial\|h_\ell\|^2}{\partial W_\ell} - \frac{\partial\|h_\ell^{err}\|^2}{\partial W_\ell}\right) \\
&= \frac{1}{2}\left(\frac{\partial\|\sigma(W_\ell h_{\ell-1})\|^2}{\partial W_\ell} - \frac{\partial\|\sigma(W_\ell h_{\ell-1}^{err})\|^2}{\partial W_\ell}\right) \\
&= \sigma(W_\ell h_{\ell-1}) \odot \sigma'(W_\ell h_{\ell-1}) h_{\ell-1}^\top \\
&\quad - \sigma(W_\ell h_{\ell-1}^{err}) \odot \sigma'(W_\ell h_{\ell-1}^{err}) h_{\ell-1}^{err\top} \\
&= (\sigma'(W_\ell h_{\ell-1}) \odot h_\ell h_{\ell-1}^\top \\
&\quad - \sigma'(W_\ell h_{\ell-1}^{err}) \odot h_\ell^{err} h_{\ell-1}^{err}) \\
&= (h_\ell' \odot h_\ell) h_{\ell-1}^\top - (h_\ell^{err\prime} \odot h_\ell^{err}) h_{\ell-1}^{err\top}.
\end{aligned} \tag{9}$$

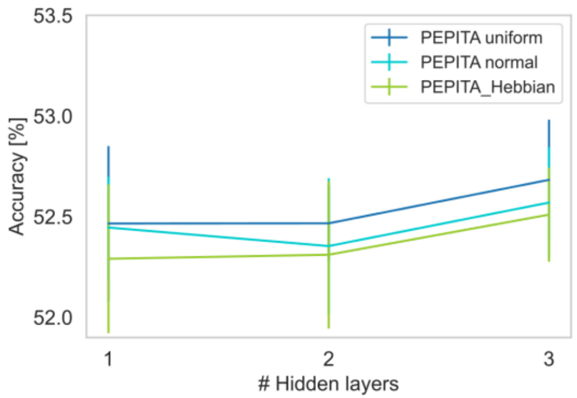
If ReLU non-linearity:

$$\frac{1}{2}\frac{\partial J_\ell}{\partial W_\ell} = h_\ell h_{\ell-1}^\top - h_\ell^{err} h_{\ell-1}^{err\top}$$

The Hebbian modification of PEPITA

Algorithm 1 Implementation of PEPITA

Given: Input (x) and label ($target$)
#standard forward pass
$h_0 = x$
for $\ell = 1, ..., L$
  $h_\ell = \sigma_\ell(W_\ell h_{\ell-1})$
$e = h_L - target$
#modulated forward pass
$h_0^{err} = x + Fe$
for $\ell = 1, ..., L$
  $h_\ell^{err} = \sigma_\ell(W_\ell h_{\ell-1}^{err})$
  if $\ell < L$:
    $\Delta W_\ell = (h_\ell - h_\ell^{err}) \cdot (h_{\ell-1}^{err})^T$
  else:
    $\Delta W_\ell = e \cdot (h_{\ell-1}^{err})^T$
  #apply update
  $W_\ell(t+1) = W_\ell(t) - \eta \Delta W_\ell$



$$\Delta W_\ell = h_\ell \cdot h_{\ell-1}^{err T} - h_\ell^{err} \cdot h_{\ell-1}^{err T}$$
$$\simeq h_\ell \cdot h_{\ell-1}^{T} - h_\ell^{err} \cdot h_{\ell-1}^{err T}$$

1st pass 2nd pass

PEPITA: weight update equivalent to the Forward-Forward framework

Forward-Forward framework

- Goodness as the sum of squared neural activities
  - h^2_l for the positive pass and
  - $(h^{err}_l)^2$ for the negative pass.
- Local loss function J_l for layer l = the sum of
  - loss function of the positive pass J^+_l and
  - loss function of the negative pass J^-_l

$$J_\ell = \|h_\ell\|^2 - \|h_\ell^{err}\|^2$$

PEPITA- Hebbian

$$\begin{aligned}\Delta W_\ell &= h_\ell \cdot h_{\ell-1}^{errT} - h_\ell^{err} \cdot h_{\ell-1}^{errT} \\ &\simeq h_\ell \cdot h_{\ell-1}^{T} - h_\ell^{err} \cdot h_{\ell-1}^{errT}\end{aligned}$$

Equivalence of weight update

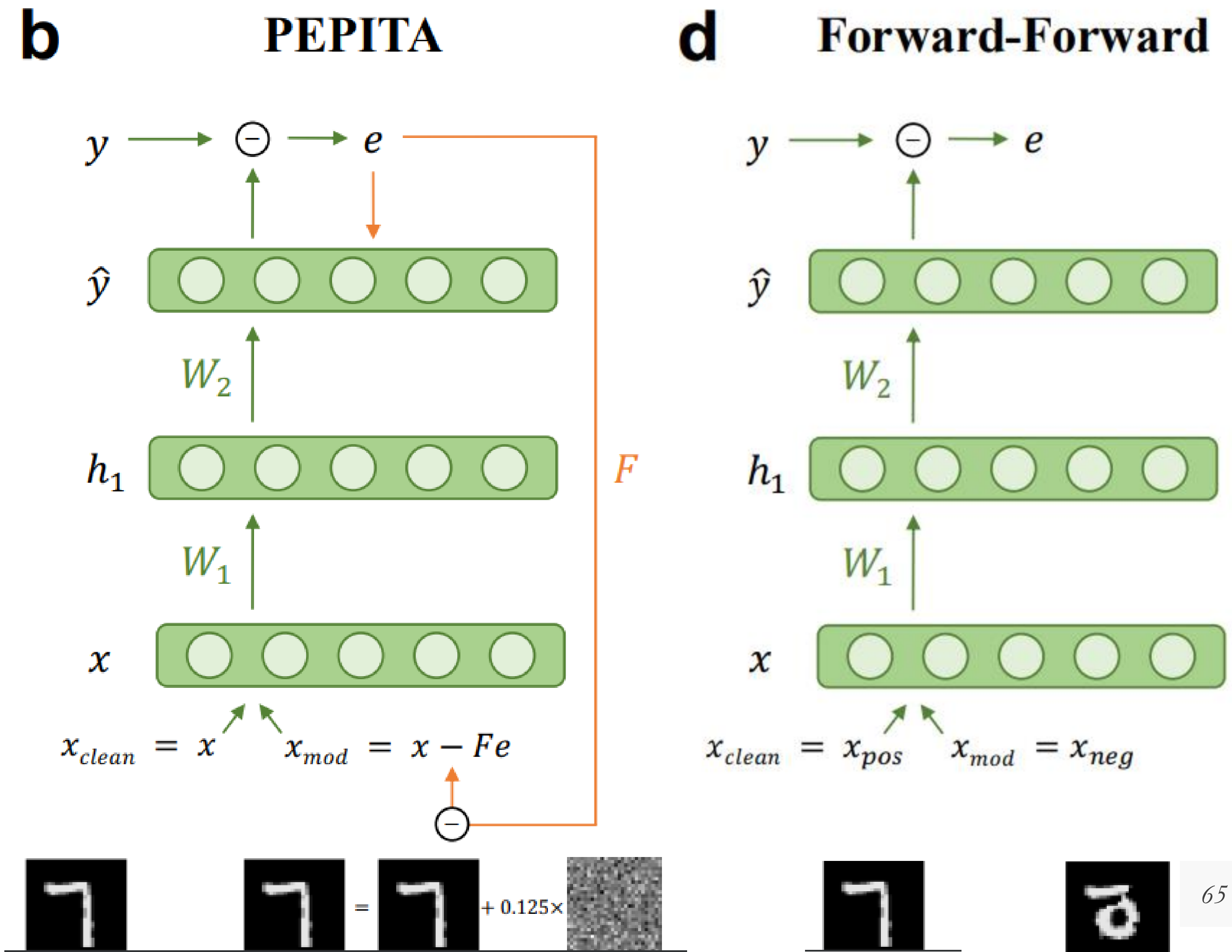
$$\begin{aligned}\frac{1}{2}\frac{\partial J_\ell}{\partial W_\ell} &= \frac{1}{2}\left(\frac{\partial\|h_\ell\|^2}{\partial W_\ell} - \frac{\partial\|h_\ell^{err}\|^2}{\partial W_\ell}\right) \\ &= \frac{1}{2}\left(\frac{\partial\|\sigma(W_\ell h_{\ell-1})\|^2}{\partial W_\ell} - \frac{\partial\|\sigma(W_\ell h_{\ell-1}^{err})\|^2}{\partial W_\ell}\right) \\ &= \sigma(W_\ell h_{\ell-1}) \odot \sigma'(W_\ell h_{\ell-1}) h_{\ell-1}^\top \\ &\quad - \sigma(W_\ell h_{\ell-1}^{err}) \odot \sigma'(W_\ell h_{\ell-1}^{err}) h_{\ell-1}^{err\top} \\ &= (\sigma'(W_\ell h_{\ell-1}) \odot h_\ell h_{\ell-1}^\top \\ &\quad - \sigma'(W_\ell h_{\ell-1}^{err}) \odot h_\ell^{err} h_{\ell-1}^{err}) \\ &= (h_\ell' \odot h_\ell) h_{\ell-1}^\top - (h_\ell^{err\prime} \odot h_\ell^{err}) h_{\ell-1}^{err\top}.\end{aligned} \tag{9}$$

If ReLU non-linearity:

$$\boxed{\frac{1}{2}\frac{\partial J_\ell}{\partial W_\ell} = h_\ell h_{\ell-1}^\top - h_\ell^{err} h_{\ell-1}^{err\top}}$$

Differences between PEPITA and FF

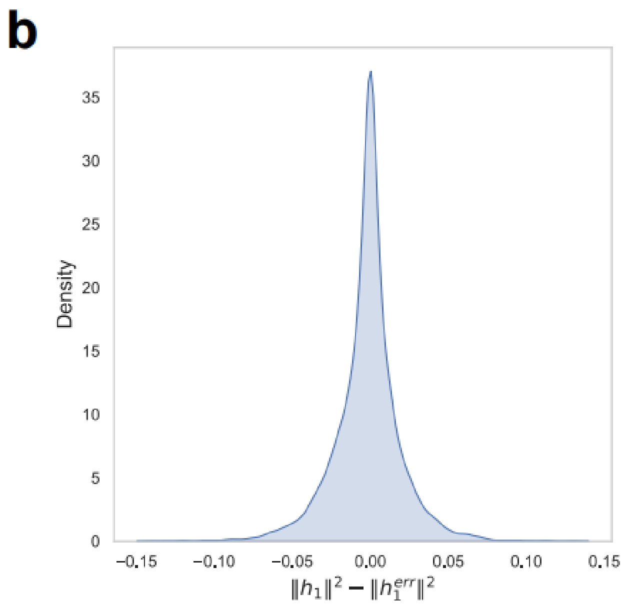
» Method to generate the modulated sample
- PEPITA → add error → top-down feedback
- FF → hybrid mask

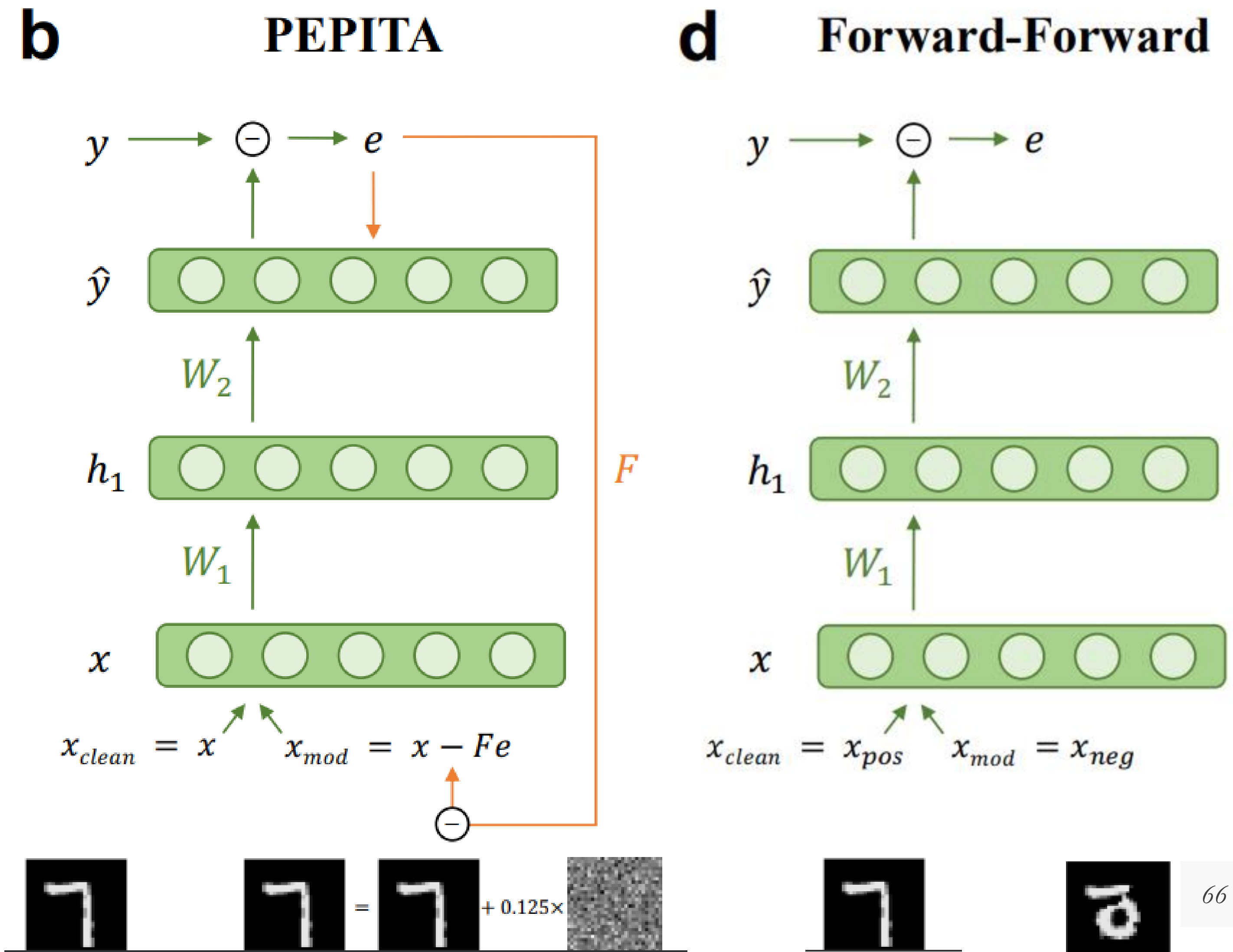


Differences between PEPITA and FF

» Method to generate the modulated sample
  - PEPITA → add error → top-down feedback
  - FF → hybrid mask

» PEPITA does not maximize (resp. minimize) the activations squared in the clean (resp. modulated) pass





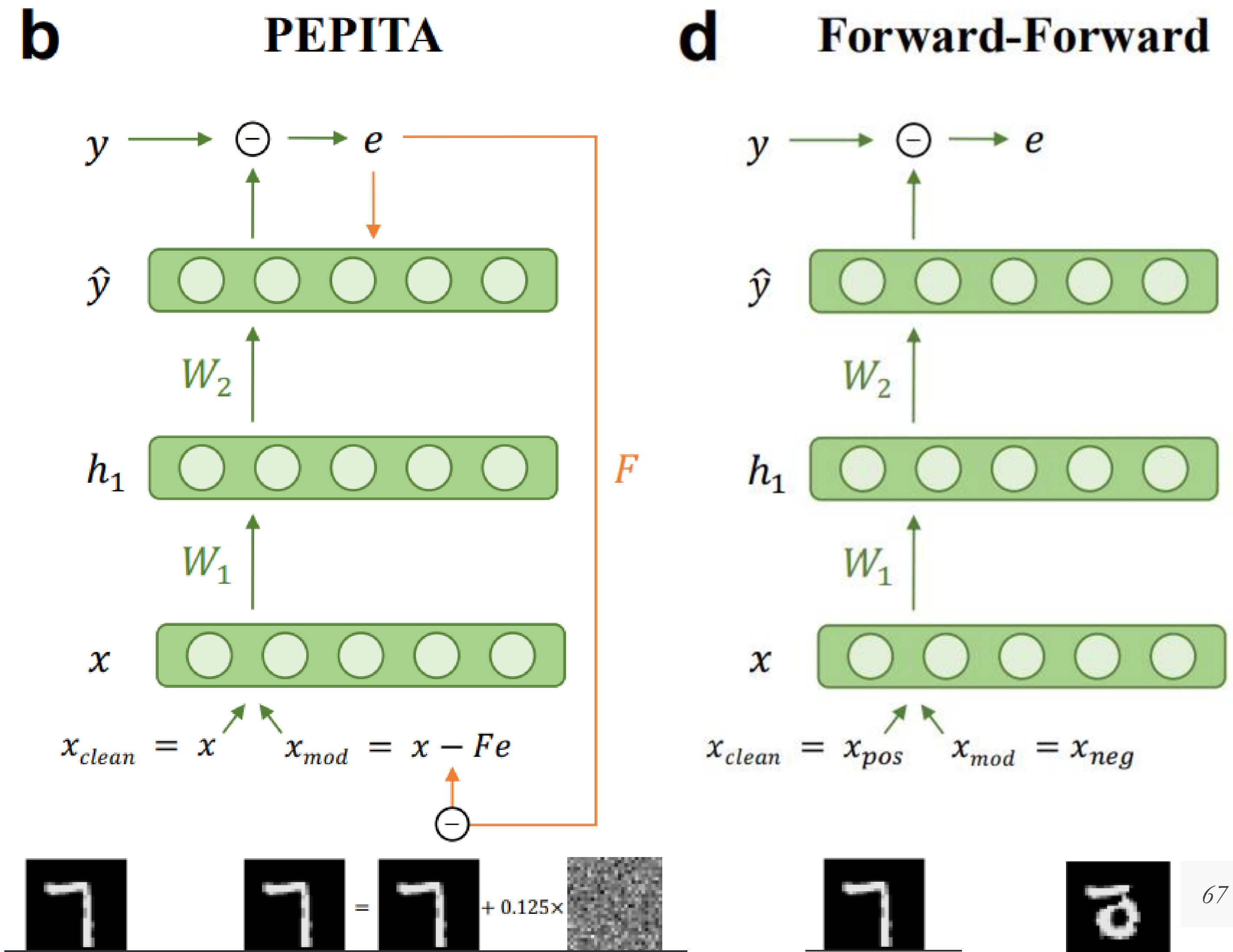
Differences between PEPITA and FF

- Method to generate the modulated sample
  - PEPITA → add error → top-down feedback
  - FF → hybrid mask

- PEPITA does not maximize (resp. minimize) the activations squared in the clean (resp. modulated) pass

- Note: FF chooses a loss based on the logistic function applied to the goodness, minus a threshold → analytical differences

$$p = \sigma\left(\|h_l\|^2 - \theta\right)$$



Outline

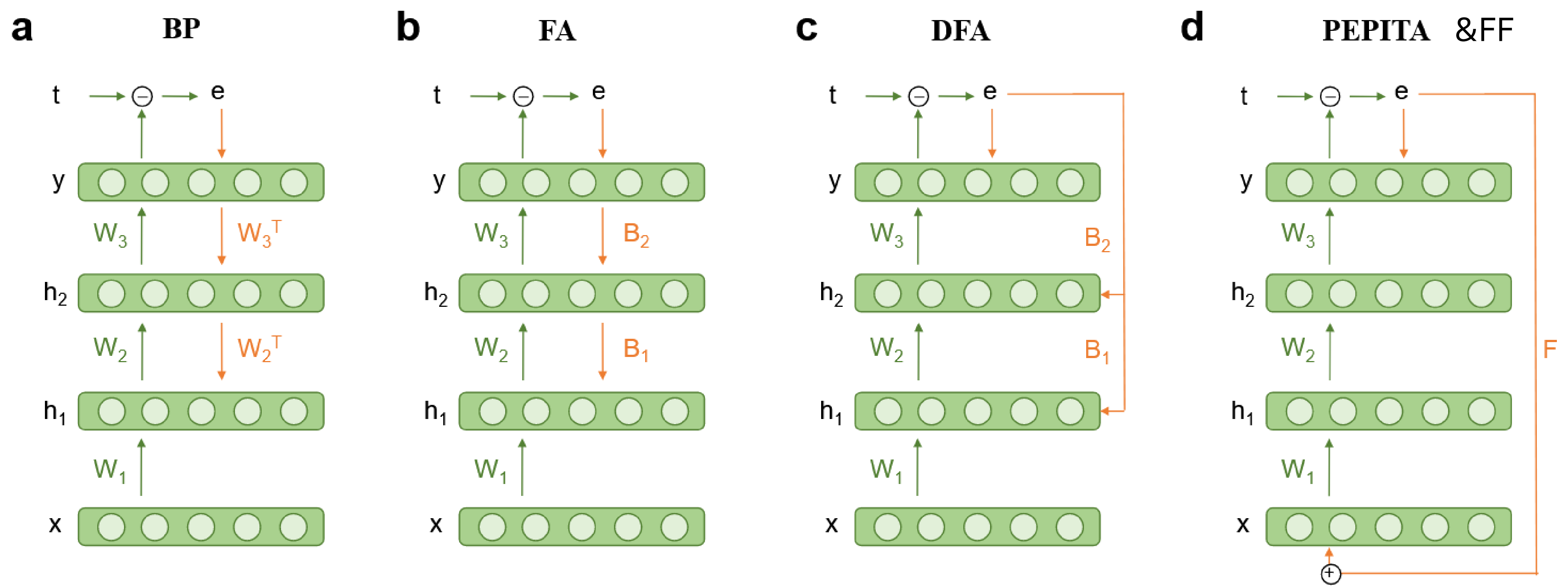
» **Neuro-inspired AI**
- Why Backpropagation is biologically implausible
- Overview of alternative solutions to credit assignment

» **PEPITA: error-driven input modulation**
- Replacing the backward pass with a second forward pass
- Results on image classification tasks
- Soft alignment dynamics
- Approximating PEPITA to *Adaptive Feedback Alignment*: analytical characterization
- Improving alignment with weight mirroring

» **Forward-Forward algorithm**
- Idea and results
- Similarities with PEPITA's update rule

» **Forward learning with top-down feedback**
- Biological considerations

PEPITA solves the biologically implausible aspects of BP

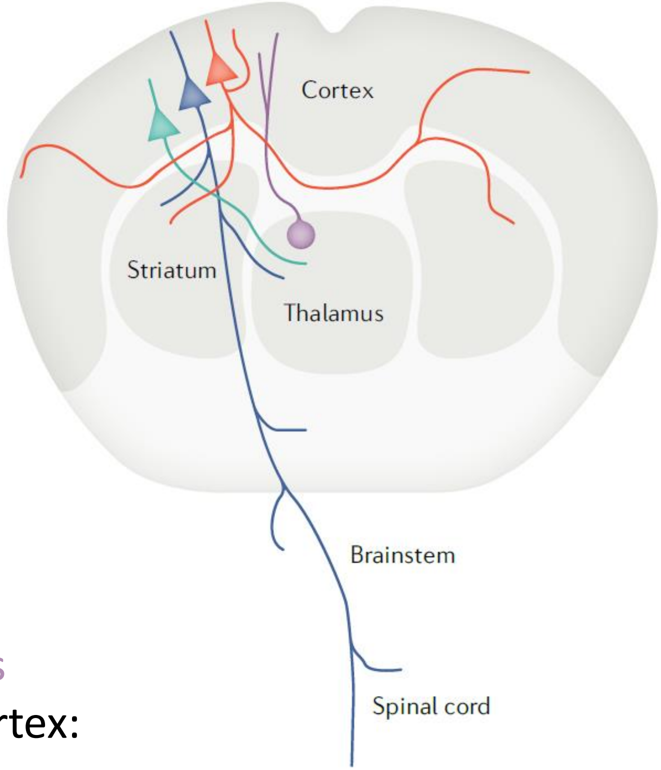


	BP	FA	DFA	PEPITA
$\Delta W_\ell =$	$-(W_{\ell+1}^T \delta a_{\ell+1}) \odot f'(a_\ell) h_{\ell-1}^T$	$-(B_\ell^T \delta a_{\ell+1}) \odot f'(a_\ell) h_{\ell-1}^T$	$-(B_\ell^T e) \odot f'(a_\ell) h_{\ell-1}^T$	$(h_\ell - h_\ell^{err}) \cdot (h_{\ell-1}^{err\,T})$
WEIGHT-TRANSPORT-FREE	✘	✓	✓	✓
LOCAL RULE	✘	✘	✘	✓
FREEZING OF ACTIVITY	✘	✘	✘	✓
UPDATE-UNLOCKED	✘	✘	PARTIALLY	PARTIALLY

Analysis of PEPITA from a biological standpoint

PEPITA solves the BP's issues of biological plausibility, but it introduces additional elements:

» Projection of the error onto the input through a fixed random matrix
  - Reminiscent of cortico-thalamo-cortical loops



In the thalamus:

- Thalamocortical (TC) neurons

Excitatory neurons in the neocortex:

- Intratelencephalic (IT)
- Pyramidal tract (PT)
- Corticothalamic (CT) neurons

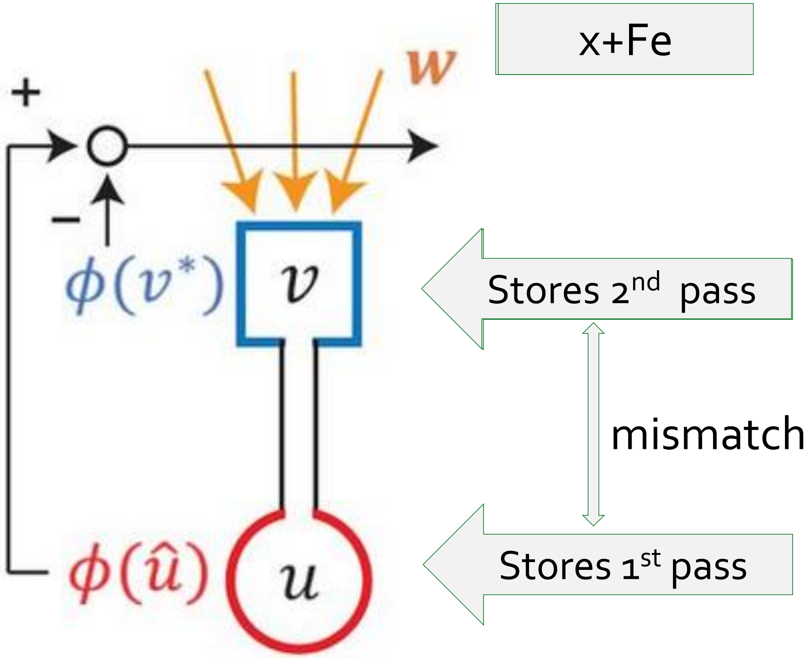
Shepherd & Yamawaki, 2021

Analysis of PEPITA from a biological standpoint

PEPITA solves the BP's issues of biological plausibility, but it introduces additional elements:

» Projection of the error onto the input through a fixed random matrix
  - Reminiscent of cortico-thalamo-cortical loops

» Storing of the activation of the *Standard pass* until the *Modulated pass*
  - Can be implemented in biological neurons through mismatch between dendritic and somatic activity



Asabuki & Fukai, 2020

Summary and Outlook

» **PEPITA and FF**

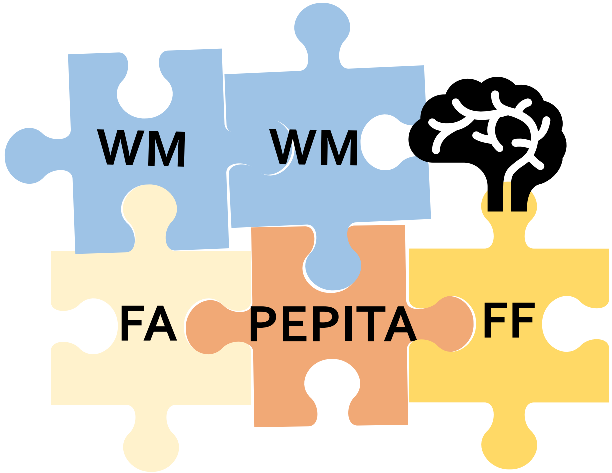
- Are novel training schemes relying only on **forward computations**
- Solve weight transport, freezing of neural activity, non-local weight updates and backward locking
- Achieve performance on-par with FA on simple image classification tasks
- PEPITA and FF share the same principles for the weight updates
- PEPITA can be approximated to an Adaptive Feedback Alignment

» **Challenges**

- Performance **does not improve with depth**
- Residual connection, intermediate error-driven modulation, training the F matrix

» **Promising avenues for exploration**

- PEPITA is not gradient-based: could it be more robust to gradient-based adversarial attacks?
- Application to object recognition on videos:
  - Consecutive frames need only one forward pass
- Implementation in unconventional physical analog hardware



Acknowledgements

Gabriel Kreiman
Harvard, Boston Children's Hospital

Ravi Srinivasan
Harvard, ETH

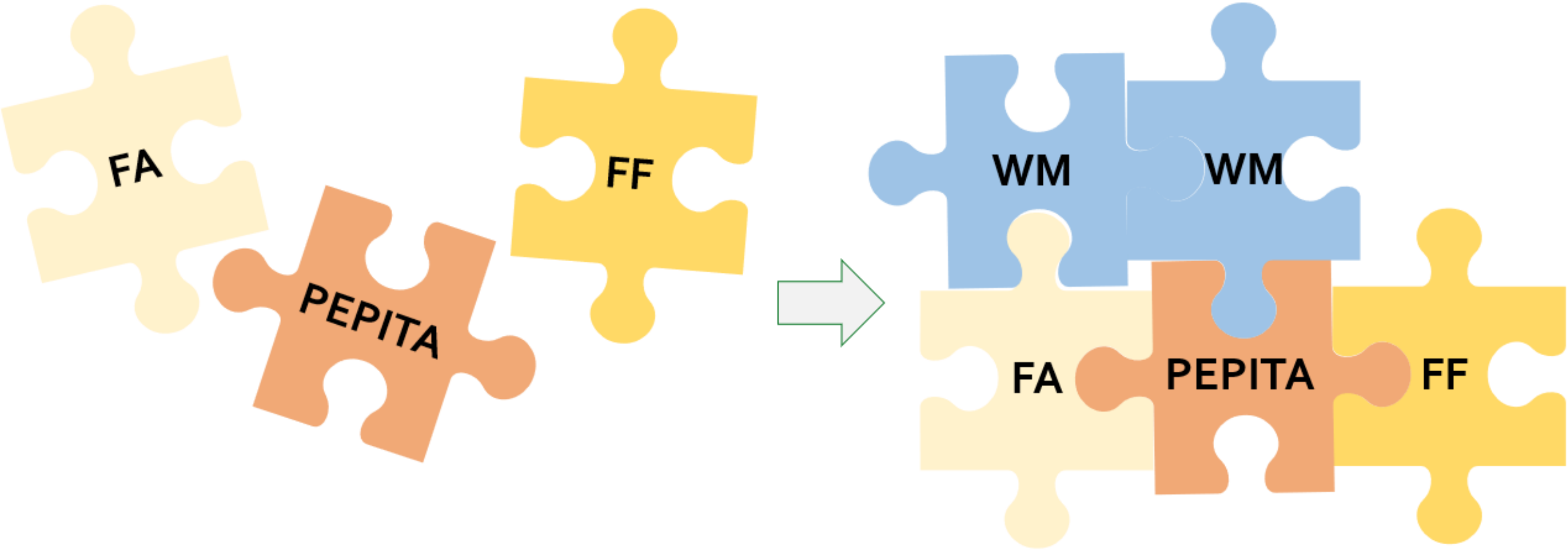
Martino Sorbaro
ETH AI Center

Francesca Mignacco
Princeton

Will Xiao
Harvard Medical School

Thank you for your attention!

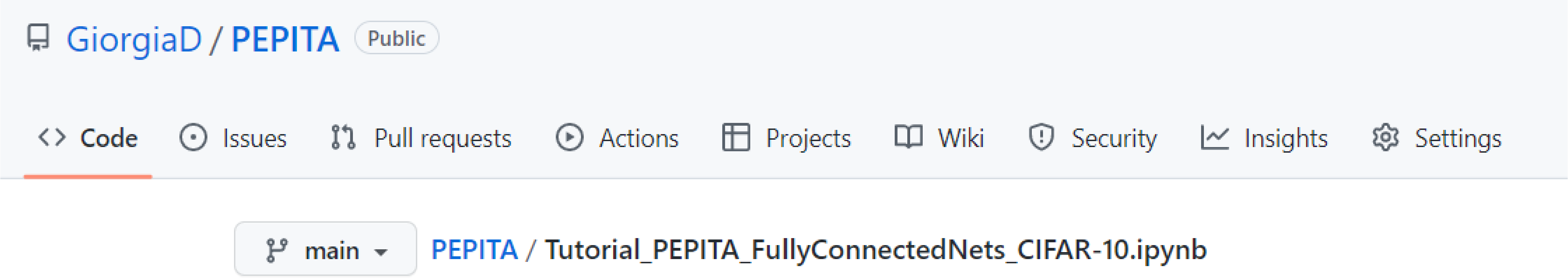
- Questions?
- Ideas?
- Suggestions?



giorgia.dellaferrera@gmail.com

Coding tutorial: Implementing PEPITA with Pytorch 1/11

- Today → Code (ICML 2022): https://github.com/GiorgiaD/PEPITA
- Code with Pytorch lightning (arXiv:2302.05440): https://drive.google.com/drive/u/1/folders/1wqHqtZx2NVuxpdjQuYUVVf1A8v-88oFS



Coding tutorial: Implementing PEPITA with Pytorch 1/11

Import libraries

In [1]:
```
# import torch libraries
import torch
import torchvision
import torchvision.transforms as transforms
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch.autograd import Variable

# import other libraries
import numpy as np
import matplotlib.pyplot as plt
import copy
```

Coding tutorial: Implementing PEPITA with Pytorch 2/11

Define Network architecture

In [2]:

```
# models with Dropout
class NetFC1x1024DOcust(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(32*32*3,1024,bias=False)
        self.fc2 = nn.Linear(1024, 10,bias=False)

        # initialize the layers using the He uniform initialization scheme
        fc1_nin = 32*32*3 # Note: if dataset is MNIST --> fc1_nin = 28*28*1
        fc1_limit = np.sqrt(6.0 / fc1_nin)
        torch.nn.init.uniform_(self.fc1.weight, a=-fc1_limit, b=fc1_limit)
        fc2_nin = 1024
        fc2_limit = np.sqrt(6.0 / fc2_nin)
        torch.nn.init.uniform_(self.fc2.weight, a=-fc2_limit, b=fc2_limit)

    def forward(self, x, do_masks):
        x = F.relu(self.fc1(x))
        # apply dropout --> we use a custom dropout implementation because we ne
        if do_masks is not None:
            x = x * do_masks[0]
        x = F.softmax(self.fc2(x))
        return x
```

Coding tutorial: Implementing PEPITA with Pytorch 3/11

Set hyperparameters and train+test the model

In [3]:
```
# set hyperparameters
## learning rate
eta = 0.01
eta_decay = 0.1
eta_decay_epochs = [60,90]
## number of epochs
num_epochs = 3
## dropout keep rate
keep_rate = 0.9
## loss --> used to monitor performance, but not for parameter updates (PEPITA do
criterion = nn.CrossEntropyLoss()
## optimizer (choose 'SGD' o 'mom')
optim = 'mom' # --> default in the paper
if optim == 'SGD':
    gamma = 0
elif optim == 'mom':
    gamma = 0.9
## batch size
batch_size = 64 # --> default in the paper
```

Coding tutorial: Implementing PEPITA with Pytorch 4/11

In [4]:

```
# initialize the network
net = NetFC1x1024DOcust()
```

In [5]:

```
# define B --> this is the F projection matrix in the paper (here named B because
nin = 32*32*3
sd = np.sqrt(6/nin)
B = (torch.rand(nin,10)*2*sd-sd)*0.05  # B is initialized with the He uniform in
```

In [6]:

```
# load the dataset - CIFAR-10
transform = transforms.Compose(
    [transforms.ToTensor()]) # this normalizes to [0,1]
trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                        download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=batch_size,
                                          shuffle=True, num_workers=2)
testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                       download=True, transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=batch_size,
                                         shuffle=False, num_workers=2)
```

```
Files already downloaded and verified
Files already downloaded and verified
```

Coding tutorial: Implementing PEPITA with Pytorch 5/11

In [7]:

```
# define function to register the activations --> we need this to compare the ac
activation = {}
def get_activation(name):
    def hook(model, input, output):
        activation[name] = output.detach()
    return hook
for name, layer in net.named_modules():
    layer.register_forward_hook(get_activation(name))
```

In [8]:

```
# do one forward pass to get the activation size needed for setting up the dropou
dataiter = iter(trainloader)
images, labels = next(dataiter)
images = torch.flatten(images, 1) # flatten all dimensions except batch
outputs = net(images,do_masks=None)
layers_act = []
for key in activation:
    if 'fc' in key or 'conv' in key:
        layers_act.append(F.relu(activation[key]))
```

Coding tutorial: Implementing PEPITA with Pytorch 6/11

In [9]:
```
# set up for momentum
if optim == 'mom':
    gamma = 0.9
    v_w_all = []
    for l_idx,w in enumerate(net.parameters()):
        if len(w.shape)>1:
            with torch.no_grad():
                v_w_all.append(torch.zeros(w.shape))
```

In [10]:
```
# Train and test the model
test_accs = []
for epoch in range(num_epochs):  # loop over the dataset multiple times

    # learning rate decay
    if epoch in eta_decay_epochs:
        eta = eta*eta_decay
        print('eta decreased to ',eta)

    # loop over batches
    running_loss = 0.0
    for i, data in enumerate(trainloader, 0):
        # get the inputs; data is a list of [inputs, labels]
        inputs, target = data
        inputs = torch.flatten(inputs, 1) # flatten all dimensions except batch
        target_onehot = F.one_hot(target,num_classes=10)
```

Coding tutorial: Implementing PEPITA with Pytorch 7/11

```
# create dropout mask for the two forward passes --> we need to use the s
do_masks = []
if keep_rate < 1:
    for l in layers_act[:-1]:
        input1 = l
        do_mask = Variable(torch.ones(inputs.shape[0],input1.data.new(in
        do_masks.append(do_mask)
    do_masks.append(1) # for the last layer we don't use dropout --> jus

# forward pass 1 with original input --> keep track of activations
outputs = net(inputs,do_masks)
layers_act = []
cnt_act = 0
for key in activation:
    if 'fc' in key or 'conv' in key:
        layers_act.append(F.relu(activation[key])* do_masks[cnt_act]) # 
        cnt_act += 1

# compute the error
error = outputs - target_onehot

# modify the input with the error
error_input = error @ B.T
mod_inputs = inputs + error_input
```

Coding tutorial: Implementing PEPITA with Pytorch 8/11

```
# forward pass 2 with modified input --> keep track of modulated activat
mod_outputs = net(mod_inputs,do_masks)
mod_layers_act = []
cnt_act = 0
for key in activation:
    if 'fc' in key or 'conv' in key:
        mod_layers_act.append(F.relu(activation[key])* do_masks[cnt_act])
        cnt_act += 1
mod_error = mod_outputs - target_onehot
```

Coding tutorial: Implementing PEPITA with Pytorch 9/11

```
    # compute the delta_w for the batch
    delta_w_all = []
    v_w = []
    for l_idx,w in enumerate(net.parameters()):
        v_w.append(torch.zeros(w.shape))

    for l in range(len(layers_act)):

        # update for the last layer
        if l == len(layers_act)-1:

            if len(layers_act)>1: # if network has more than one layer
                delta_w = -mod_error.T @ mod_layers_act[-2]
            else: # if only one layer network
                delta_w = -mod_error.T @ mod_inputs

        # update for the first layer
        elif l == 0:
            delta_w = -(layers_act[l] - mod_layers_act[l]).T @ mod_inputs

        # update for the hidden layers (not first, not last)
        elif l>0 and l<len(layers_act)-1:
            delta_w = -(layers_act[l] - mod_layers_act[l]).T @ mod_layers_ac

        delta_w_all.append(delta_w)
```

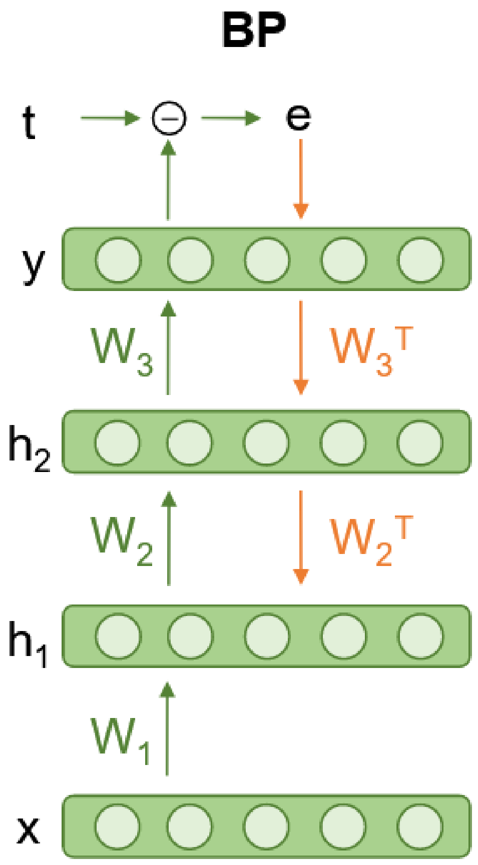
Coding tutorial: Implementing PEPITA with Pytorch 10/11

```python
        # apply the weight change
        if optim == 'SGD': # if SGD without momentum
            for l_idx,w in enumerate(net.parameters()):
                with torch.no_grad():
                    w += eta * delta_w_all[l_idx]/batch_size # specify for which

        elif optim == 'mom': # if SGD with momentum
            for l_idx,w in enumerate(net.parameters()):
                with torch.no_grad():
                    v_w_all[l_idx] = gamma * v_w_all[l_idx] + eta * delta_w_all[l
                    w += v_w_all[l_idx]


        # keep track of the loss
        loss = criterion(outputs, target)
        # print statistics
        running_loss += loss.item()
        if i%500 == 499:
            print('[%d, %5d] loss: %.3f' %
                  (epoch + 1, i + 1, running_loss / 500))
            running_loss = 0.0
```

Coding tutorial: Implementing PEPITA with Pytorch 11/11

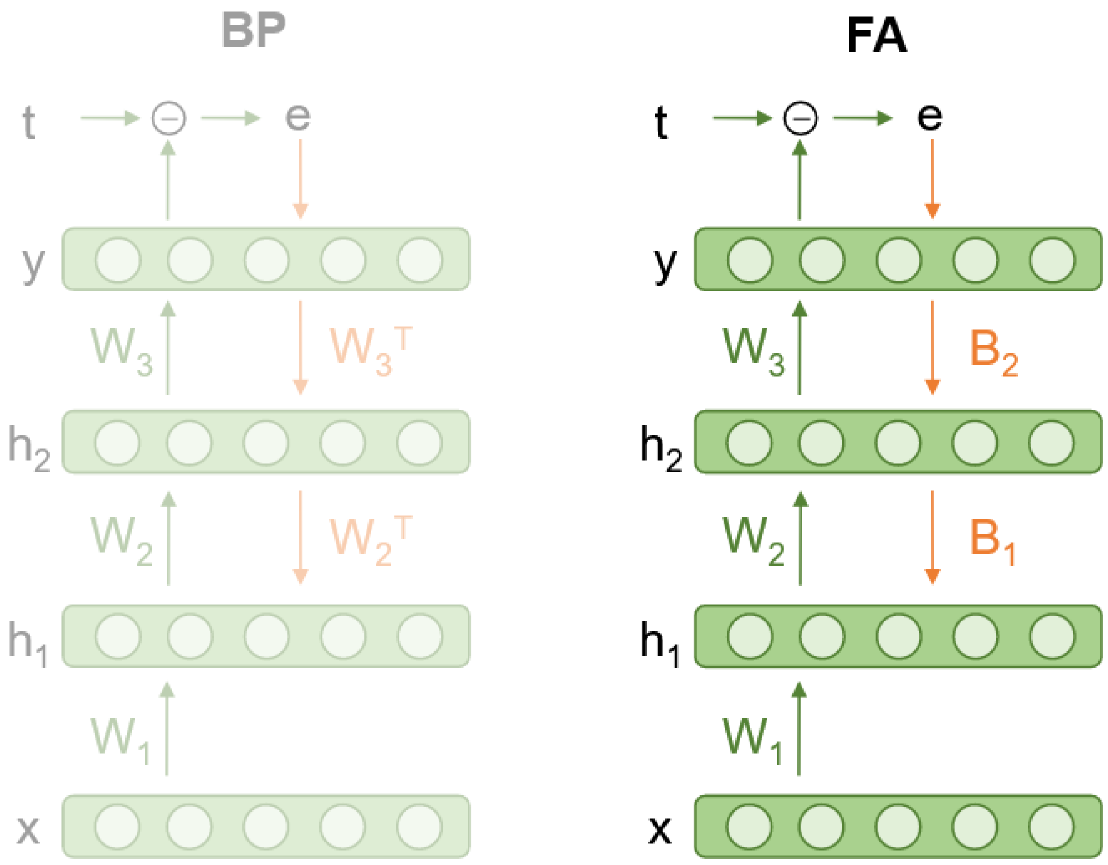
```
    print('Testing...')
    correct = 0
    total = 0
    # since we're not training, we don't need to calculate the gradients for our
    with torch.no_grad():
        for test_data in testloader:
            test_images, test_labels = test_data
            test_images = torch.flatten(test_images, 1) # flatten all dimensions
            # calculate outputs by running images through the network
            test_outputs = net(test_images,do_masks=None)
            # the class with the highest energy is what we choose as prediction
            _, predicted = torch.max(test_outputs.data, 1)
            total += test_labels.size(0)
            correct += (predicted == test_labels).sum().item()

    print('Test accuracy epoch {}: {} %'.format(epoch, 100 * correct / total))
    test_accs.append(100 * correct / total)

print('Finished Training')
```

Alternatives to BP: relaxing symmetry requirements



Rumelhart et al., 1995

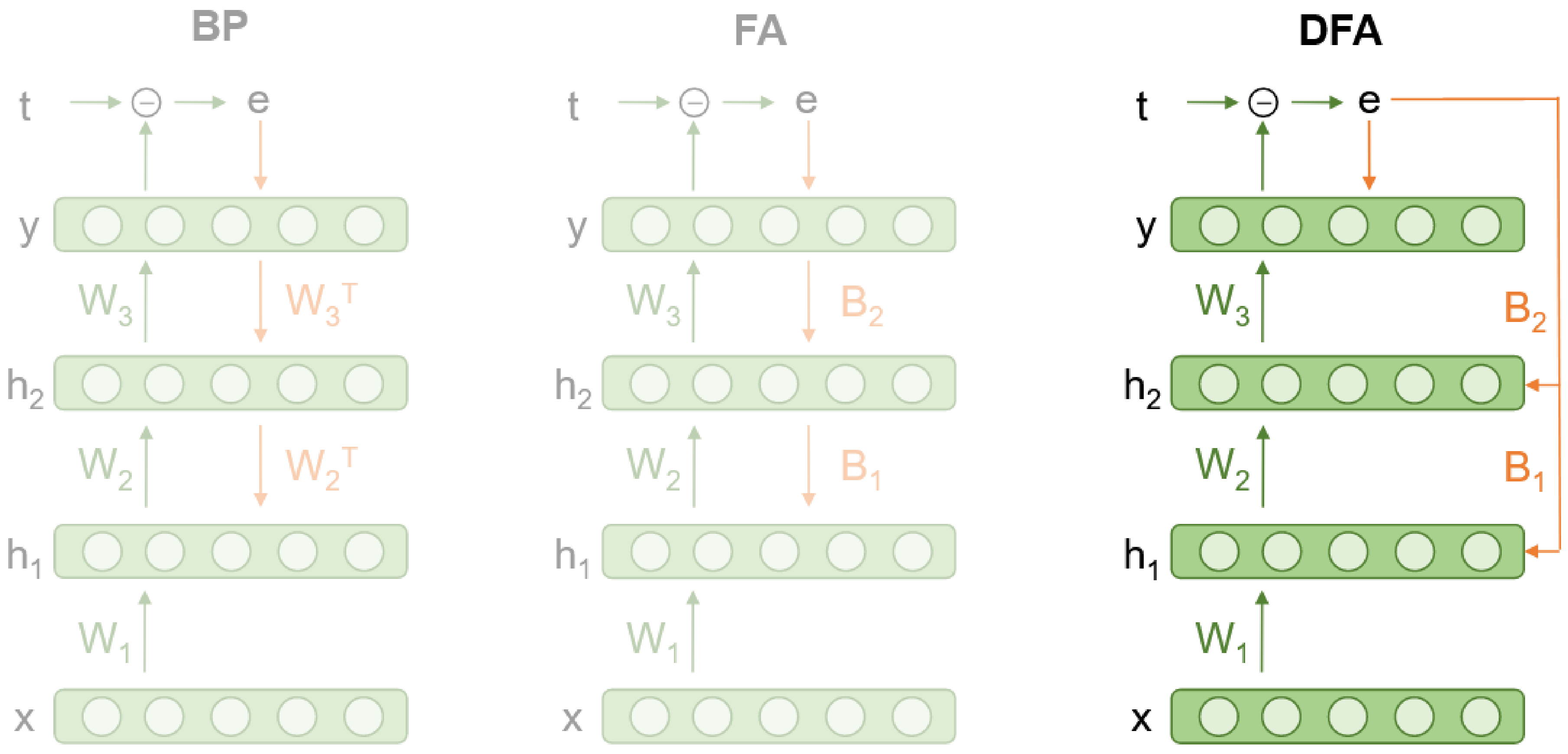
Alternatives to BP: relaxing symmetry requirements



Rumelhart et al., 1995

Lillicrap et al., 2016

Alternatives to BP: relaxing symmetry requirements

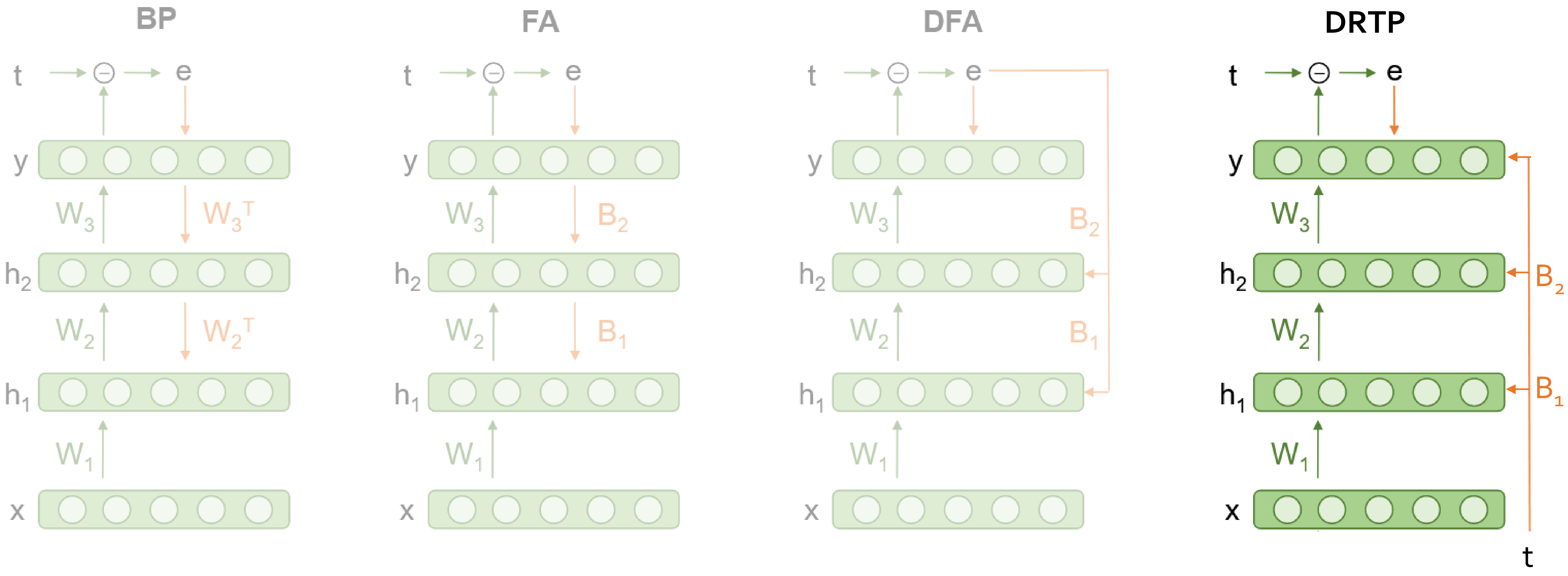


Rumelhart et al., 1995

Lillicrap et al., 2016

A. Nokland, 2016

Alternatives to BP: relaxing symmetry requirements



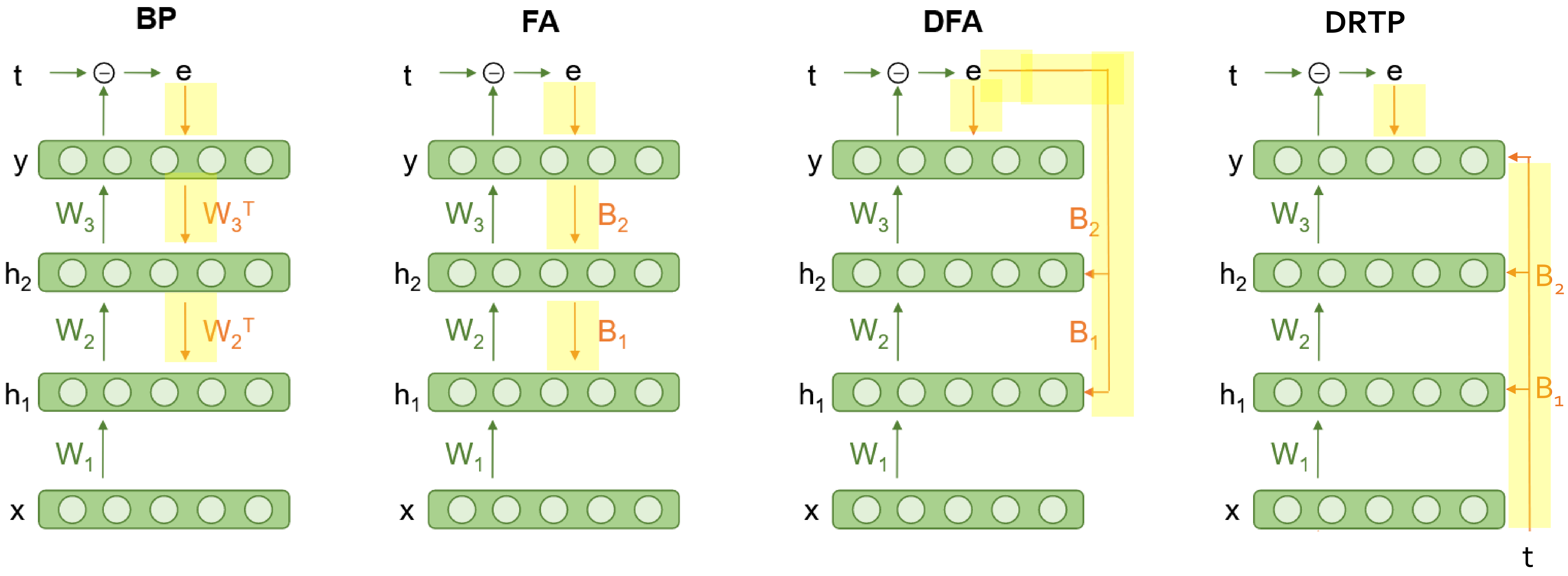
Rumelhart et al., 1995

Lillicrap et al., 2016

A. Nokland, 2016

Frenkel et al., 2019

Alternatives to BP: relaxing symmetry requirements



Rumelhart et al., 1995

Lillicrap et al., 2016

A. Nokland, 2016

Frenkel et al., 2019

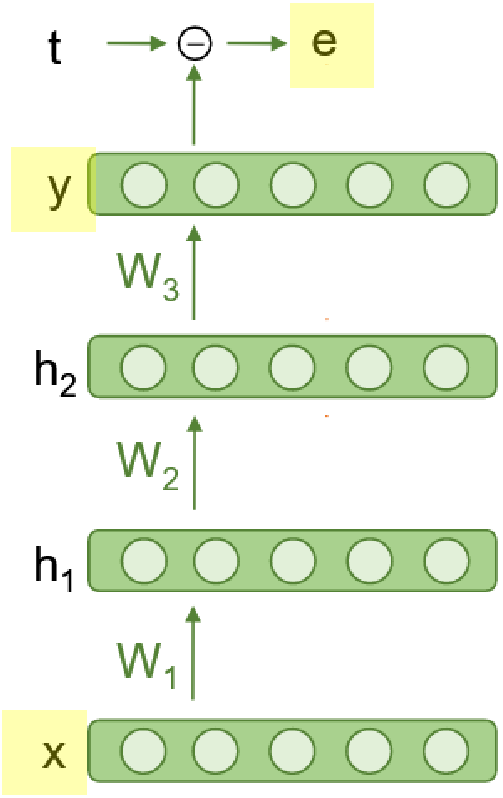
The backpropagation algorithm

» **Forward pass**

- Network's response to input
- Error function $e = y - t$
- Weight updates proportional to its negative gradient

» **Backward pass**

- Error signal flows backward through the network
- Computed recursively via the chain rule
- Update phase



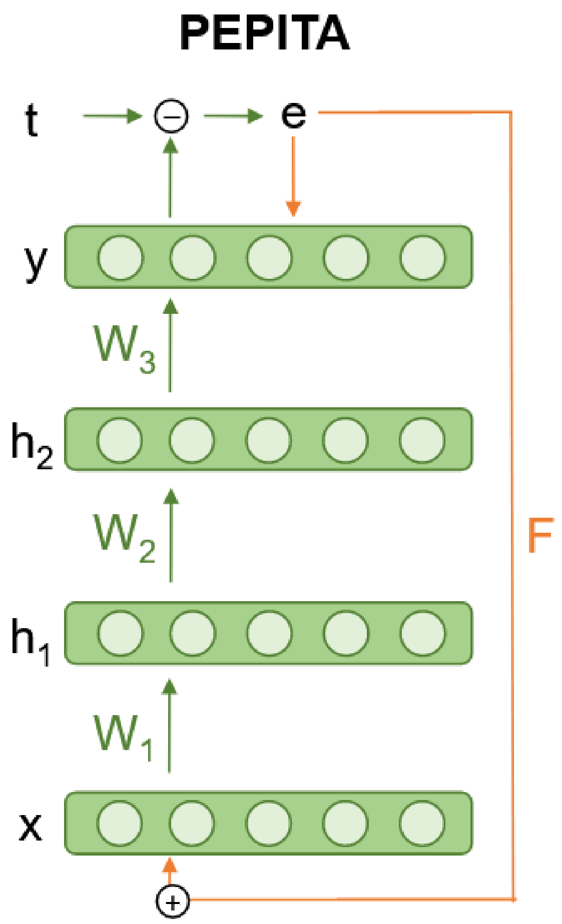
Lillicrap et al., 2020

The Backward Pass

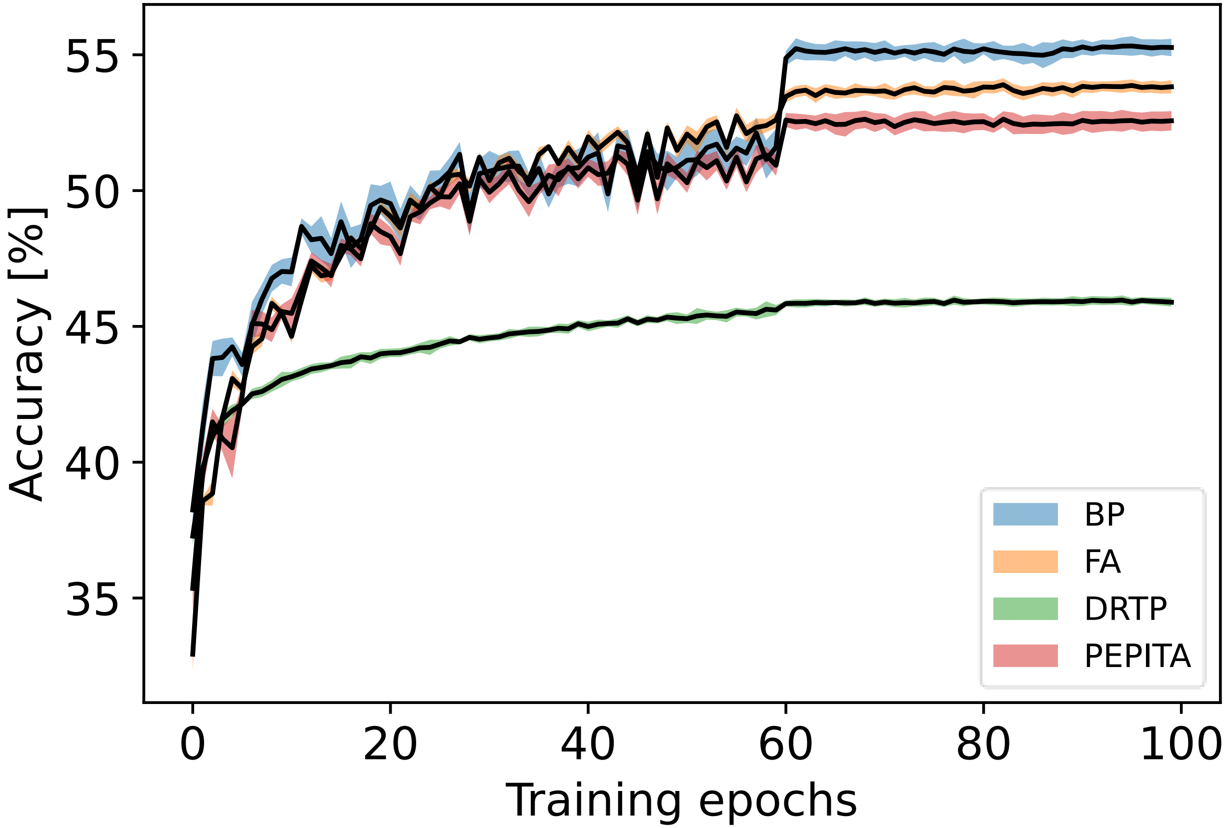
The backward pass implies:

- **Non-locality**
- **Frozen activity**
- At least **partial update locking**

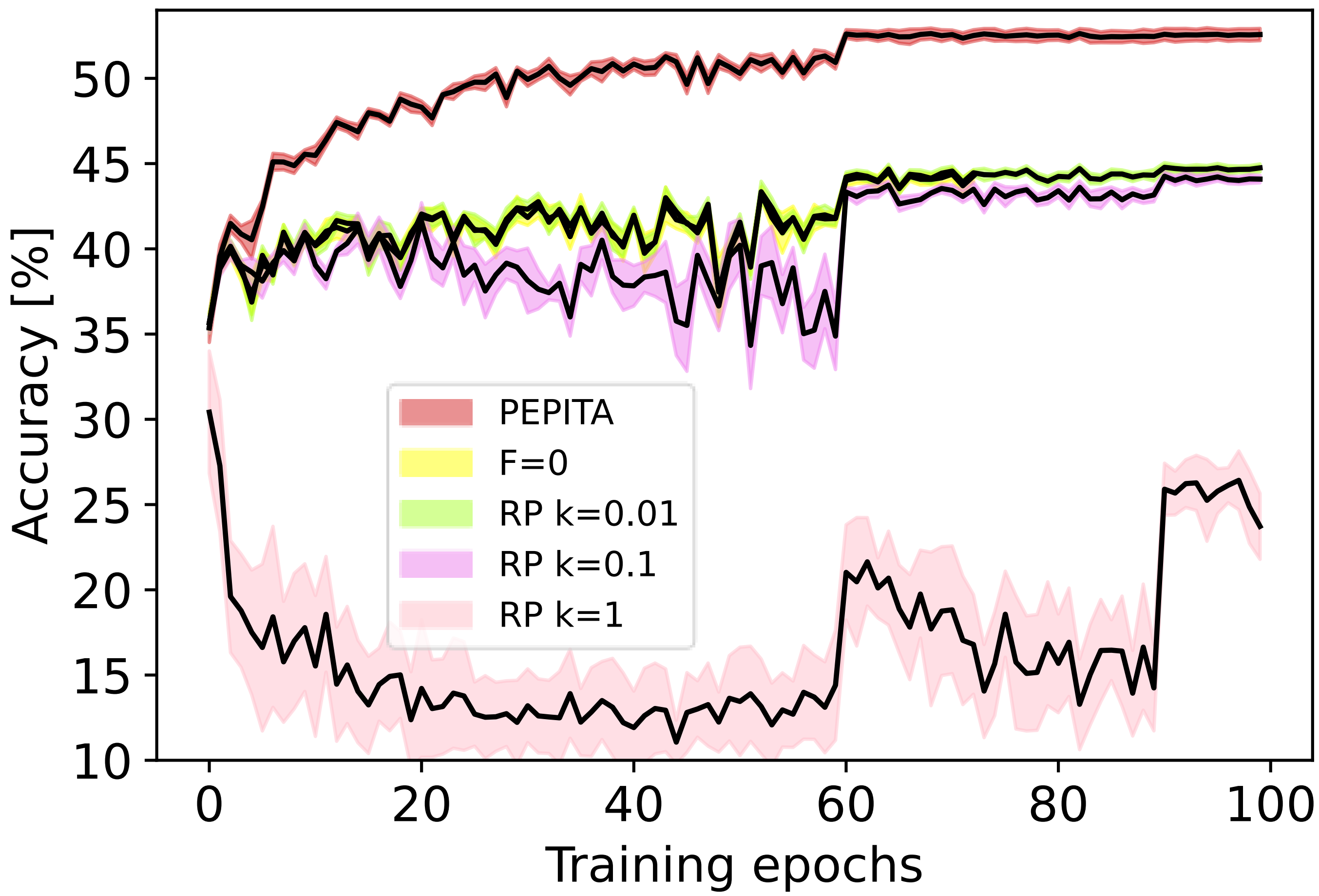
Remove the backward pass



Test curves on CIFAR-10



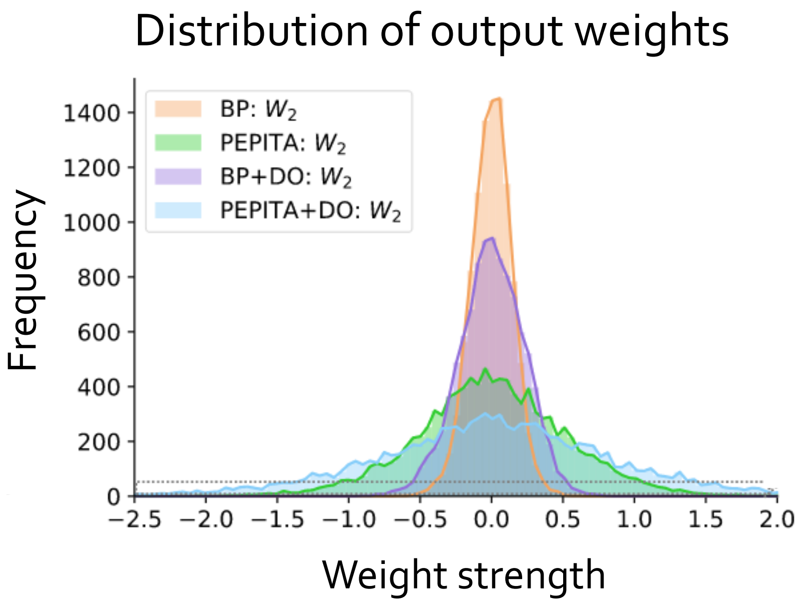
Error-based modulation is key for good performance

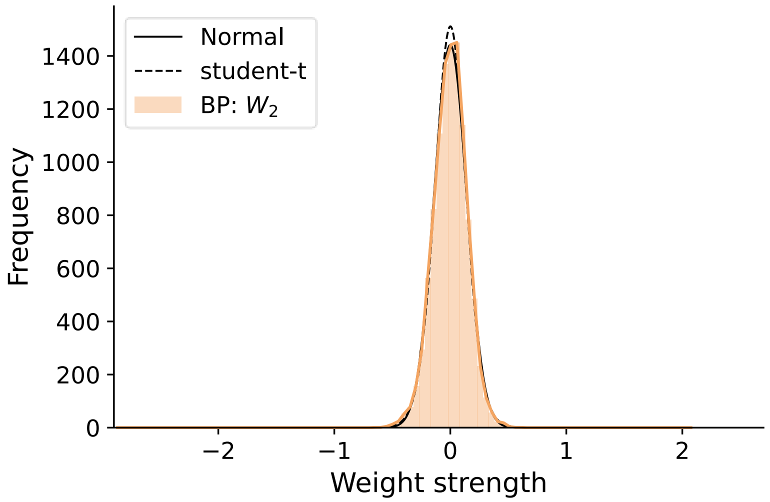


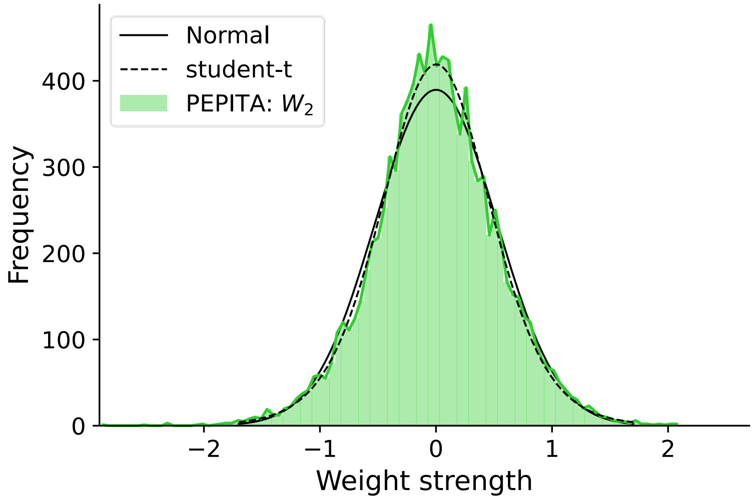
Weight distribution after training

» **Wider weight distribution**

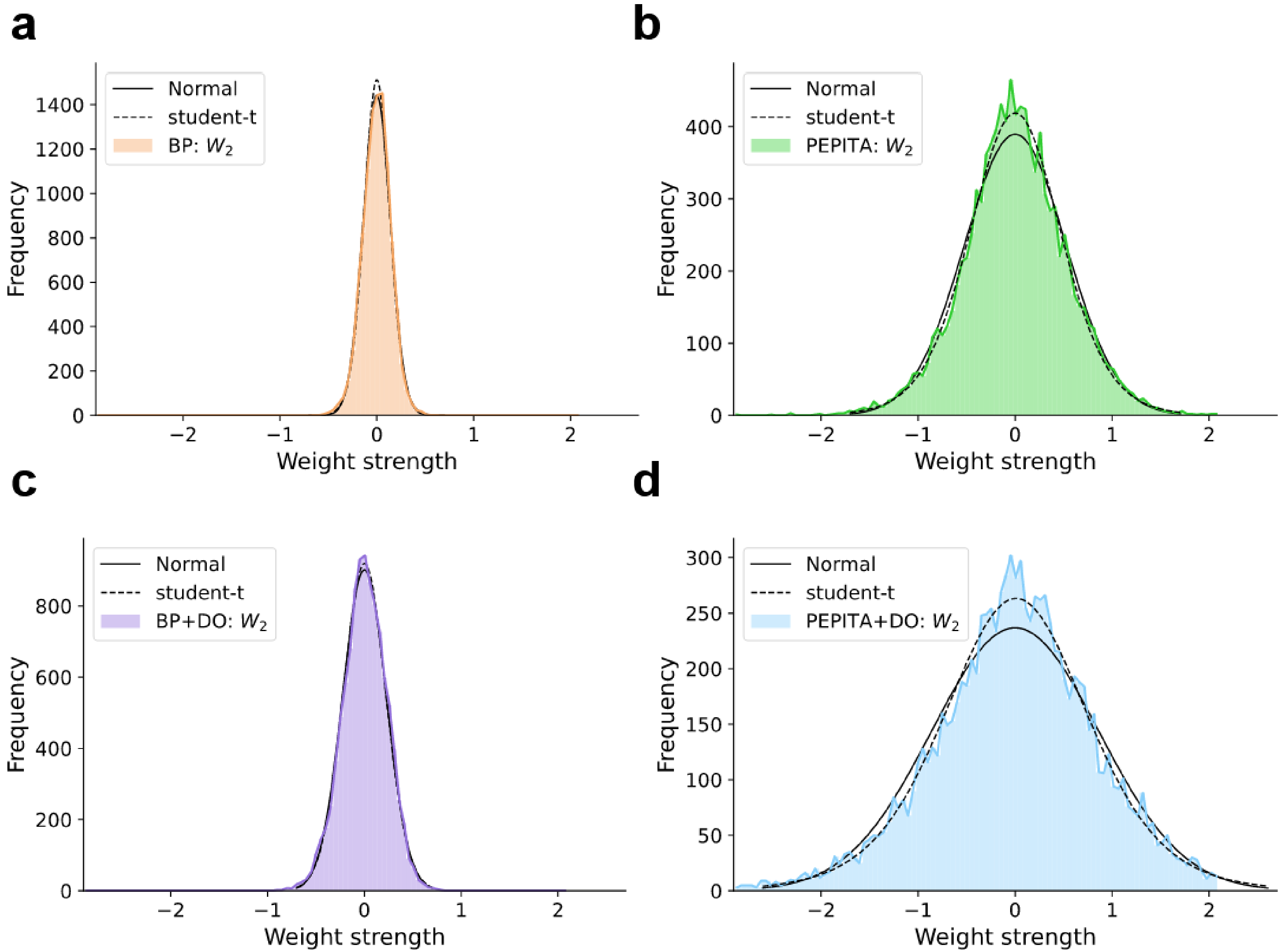
- PEPITA learns different solutions compared to BP
- Sub exponential distribution







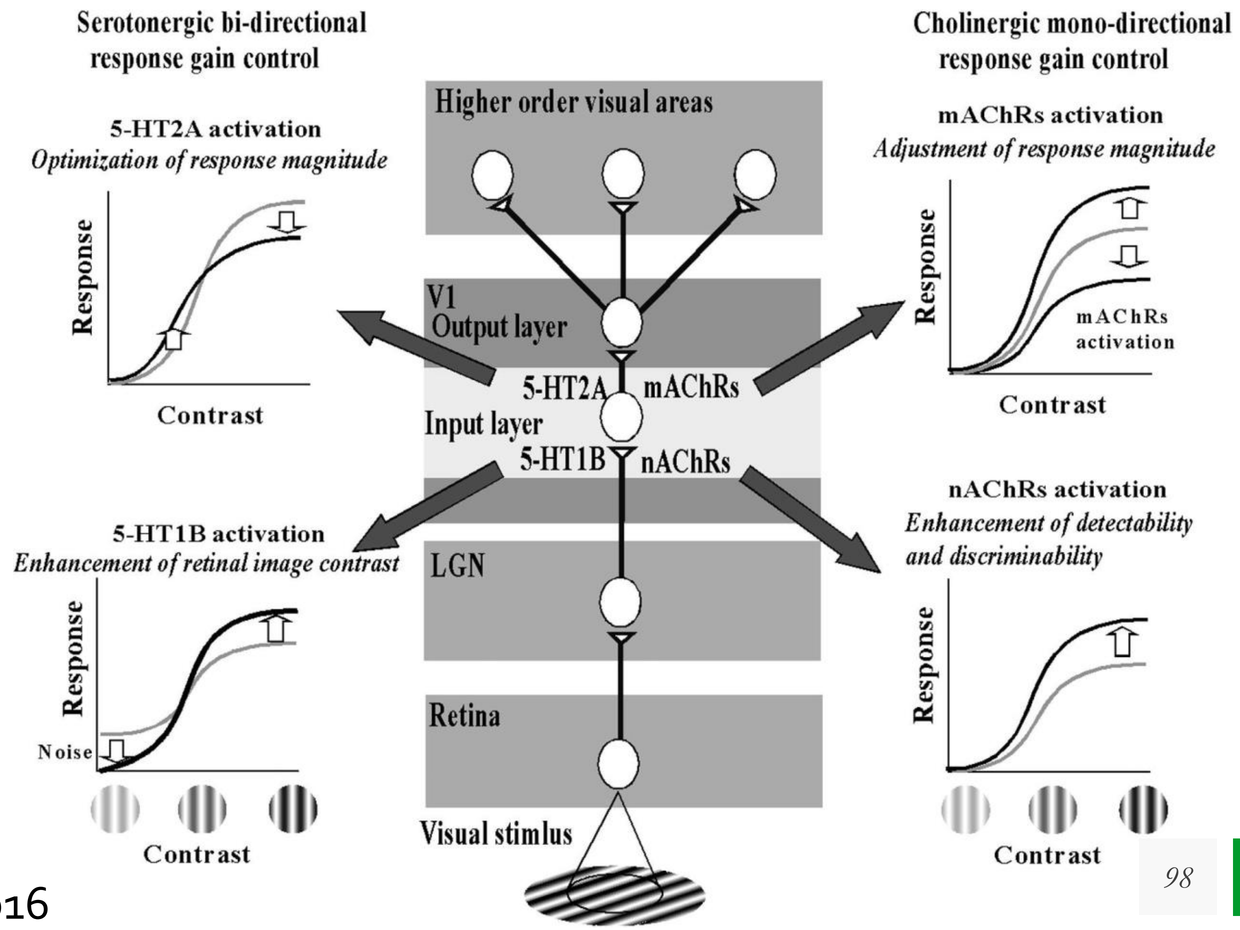
Weight distribution – heavy tailed



Analysis of PEPITA from a biological standpoint

PEPITA solves the BP's issues of biological plausibility, but it introduces additional elements:

» Projection of the error onto the input through a fixed random matrix
- Global neuromodulatory signals modulate activity in V1

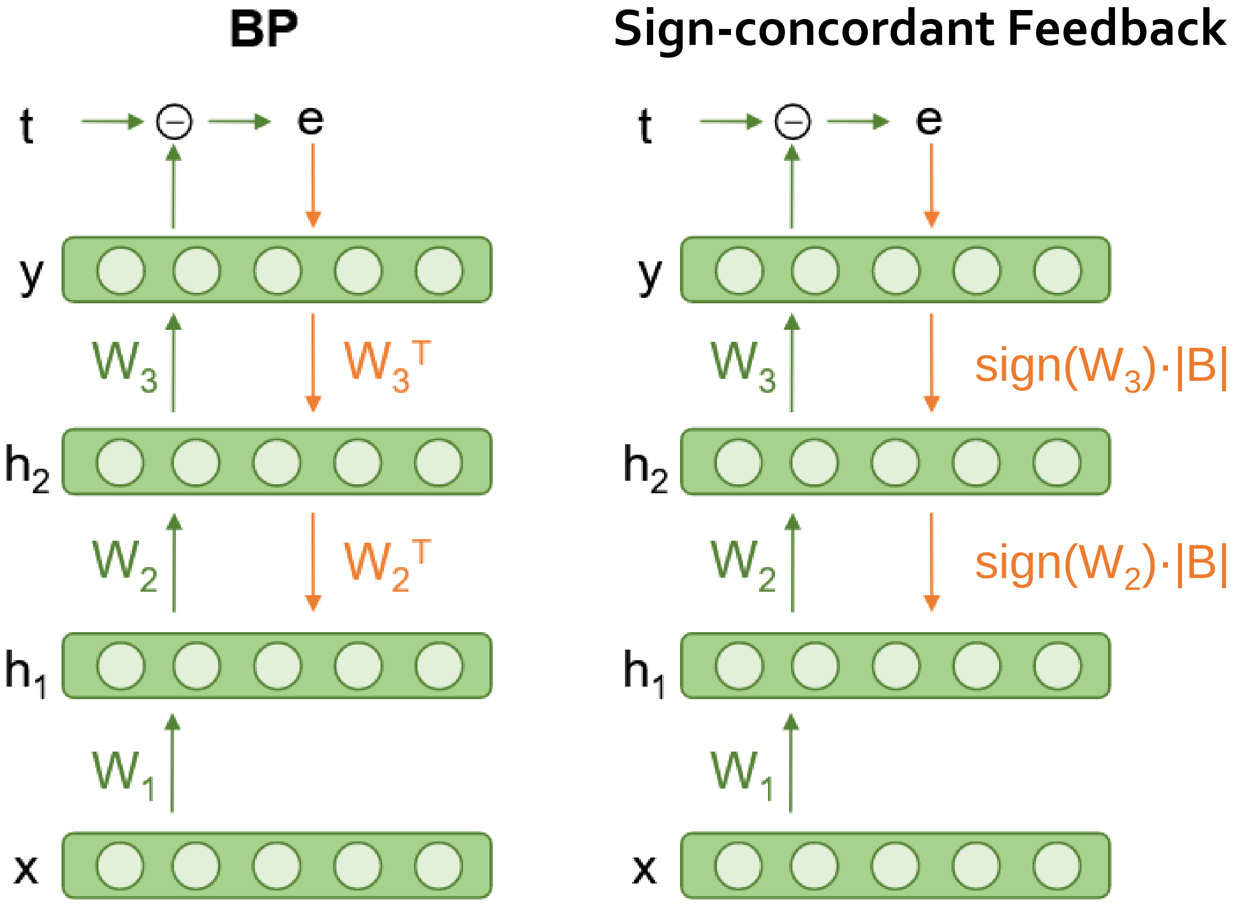


Shimegi et al., 2016

Alternatives to BP: Sign symmetry

» **Asymmetric backpropagation**
- Sign-concordant Feedback

» **Relax weight symmetry requirement**
- the magnitudes of feedback weights do not matter to performance
- the signs of feedback weights do matter — the more concordant signs between feedforward and their corresponding feedback connection



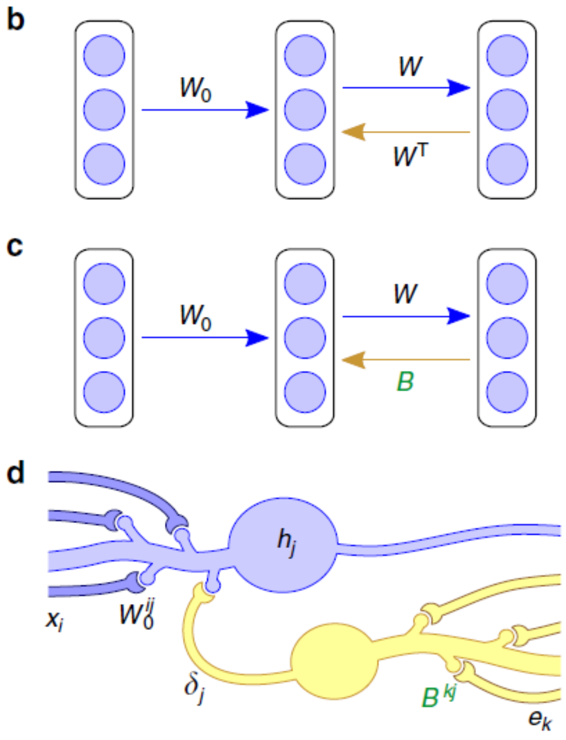
Liao et al., 2016
Xiao et al., 2019

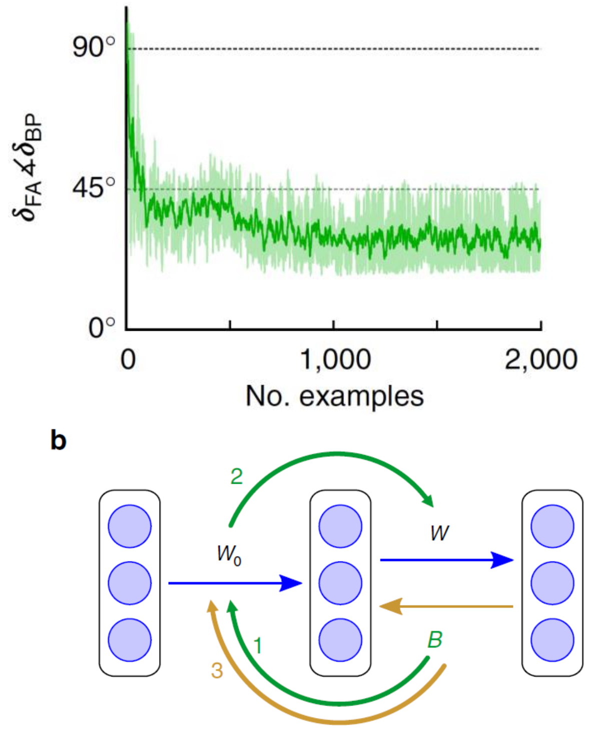
Alternatives to BP: Feedback Alignment

» Precise symmetric connectivity between connected layers is not required to obtain quick learning

» **Replaces W^T with a matrix of fixed random weights B**
- Each neuron in the hidden layer receives a random projection of the error vector
- Avoids all transport of synaptic weight information

» **The circuit learns by encouraging a soft alignment of W with B^T**
- The angle between modulator vectors prescribed by feedback alignment and backprop decreases
- As W aligns with B^T, B begins to act like W^T, sending useful teaching signals to the hidden units





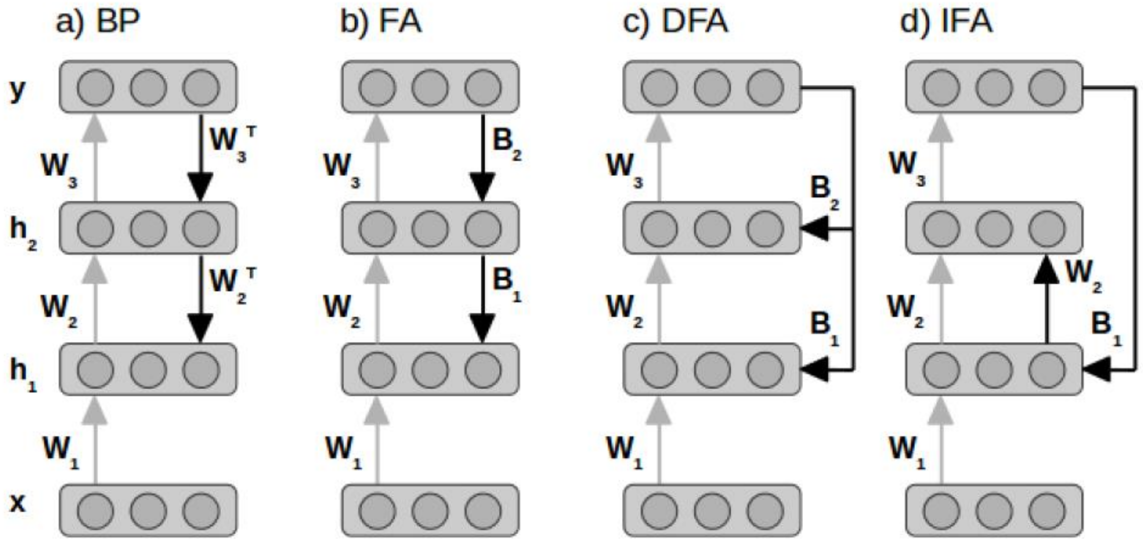
Lillicrap et al., 2016

Alternatives to BP: Direct and Indirect Feedback Alignment

» The FA principle is used for training hidden layers more independently from the rest of the network

» **Feedback path disconnected from the forward path**

- Possibility that the error in the feedback layer is represented by neurons not participating in the forward pass
- layer is no longer reciprocally connected to the layer above



» **DFA**

- direct feedback path to each hidden layer

» **IFA**

- direct feedback path connecting to the first hidden layer
- then visiting every layer on its way forward

MODEL	BP	FA	DFA
7x240 Tanh	$2.16 \pm 0.13\%$	$2.20 \pm 0.13\%$ (0.02%)	$2.32 \pm 0.15\%$ (0.03%)
100x240 Tanh			$3.92 \pm 0.09\%$ (0.12%)
1x800 Tanh	$1.59 \pm 0.04\%$	$1.68 \pm 0.05\%$	$1.68 \pm 0.05\%$
2x800 Tanh	$1.60 \pm 0.06\%$	$1.64 \pm 0.03\%$	$1.74 \pm 0.08\%$
3x800 Tanh	$1.75 \pm 0.05\%$	$1.66 \pm 0.09\%$	$1.70 \pm 0.04\%$
4x800 Tanh	$1.92 \pm 0.11\%$	$1.70 \pm 0.04\%$	$1.83 \pm 0.07\%$ (0.02%)
2x800 Logistic	$1.67 \pm 0.03\%$	$1.82 \pm 0.10\%$	$1.75 \pm 0.04\%$
2x800 ReLU	$1.48 \pm 0.06\%$	$1.74 \pm 0.10\%$	$1.70 \pm 0.06\%$
2x800 Tanh + DO	$1.26 \pm 0.03\%$ (0.18%)	$1.53 \pm 0.03\%$ (0.18%)	$1.45 \pm 0.07\%$ (0.24%)
2x800 Tanh + ADV	$1.01 \pm 0.08\%$	$1.14 \pm 0.03\%$	$1.02 \pm 0.05\%$ (0.12%)

Test error on MNIST

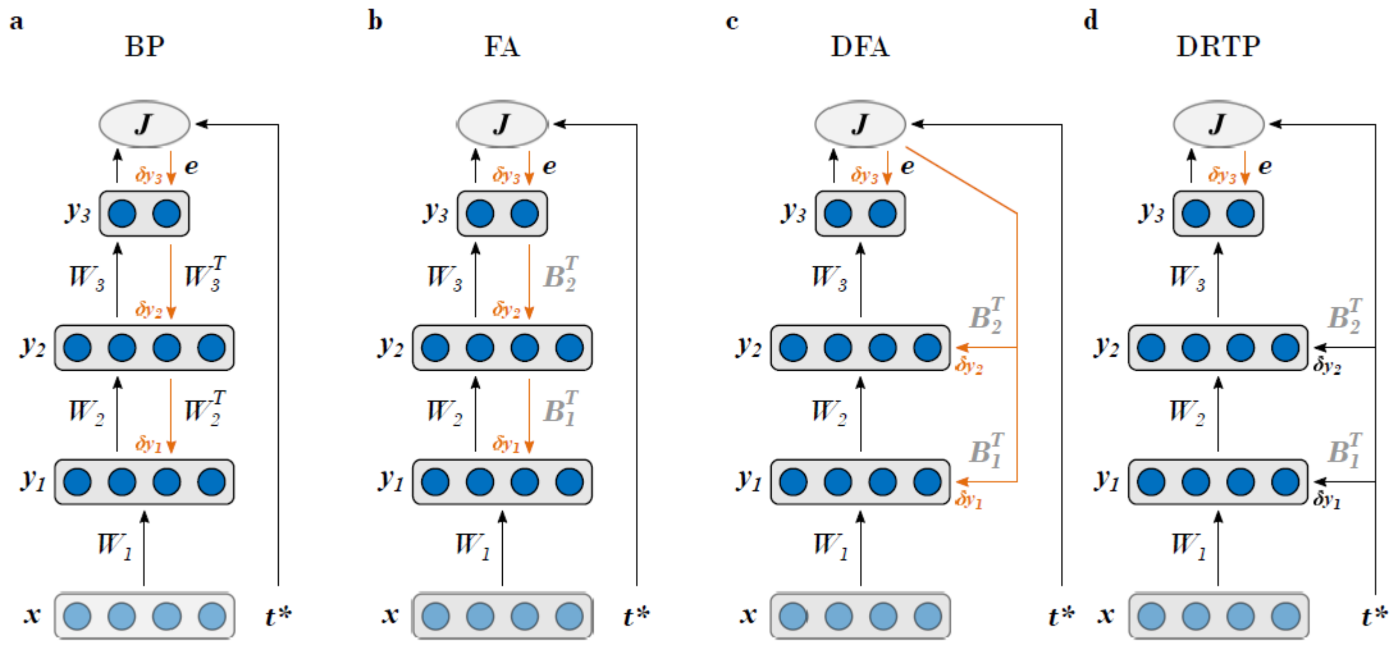
A. Nokland, 2016

Alternatives to BP: Direct Random Target Propagation

» **The error sign provides useful modulatory signals to multi-layer networks**
  - Targets (i.e. one-hot-encoded labels) used in place of the output error
  - Targets are projected onto the hidden layers

» Fully solves both the weight transport and the update locking problems

» BUT: lower performance than BP, FA, DFA

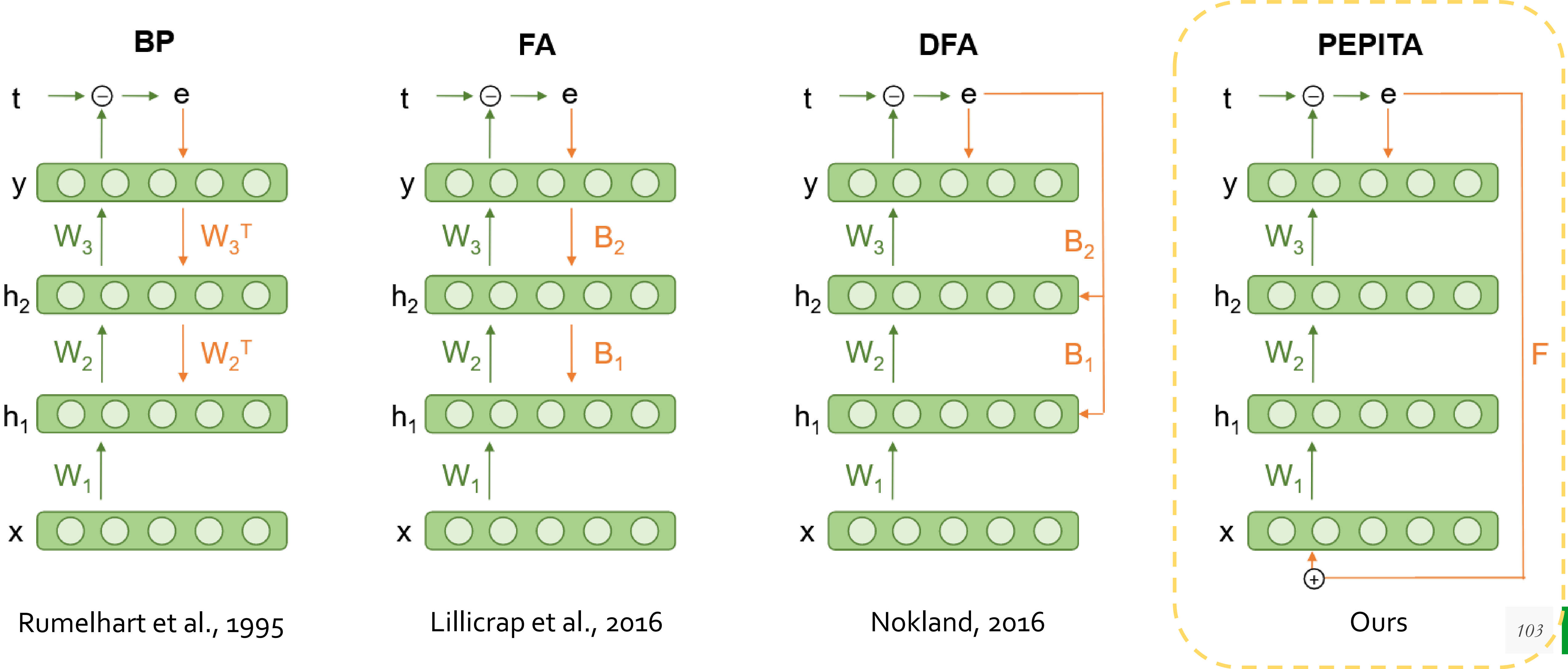


Network		BP	FA	DFA	DRTP
FC1-500	DO 0.0	1.72±0.08%	1.92±0.08%	2.59±0.11%	4.58±0.12%
	DO 0.1	1.55±0.03%	1.66±0.06%	2.17±0.10%	4.65±0.13%
	DO 0.25	1.64±0.06%	1.73±0.05%	2.32±0.08%	5.36±0.11%

Test error on MNIST

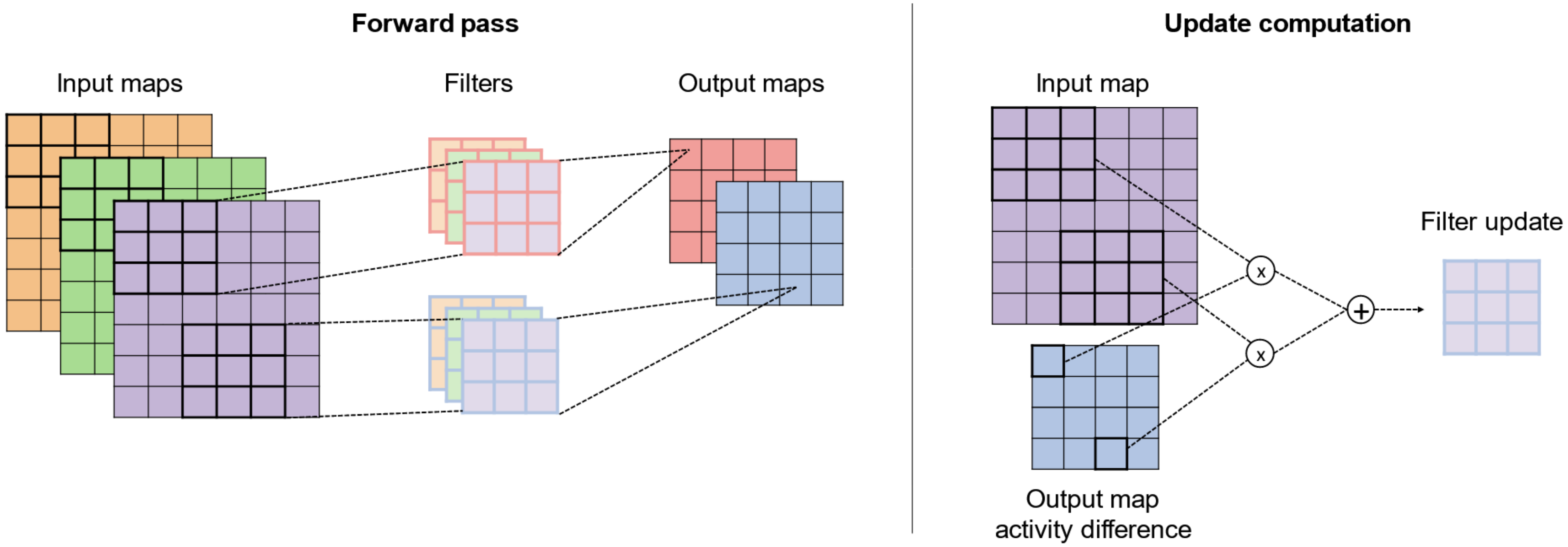
Frenkel et al., 2019

Training without a backward path: modulating the input through the error



The PEPITA learning rule for Convolutional Neural Networks

- Same approach with *Standard* and *Modulated pass*
- Takes into account *weight sharing* of convolutional layers
- Each filter is updated based on the contributions of each *input-map-region – output-map-element* pair



Testing PEPITA on image classification tasks - experimental results

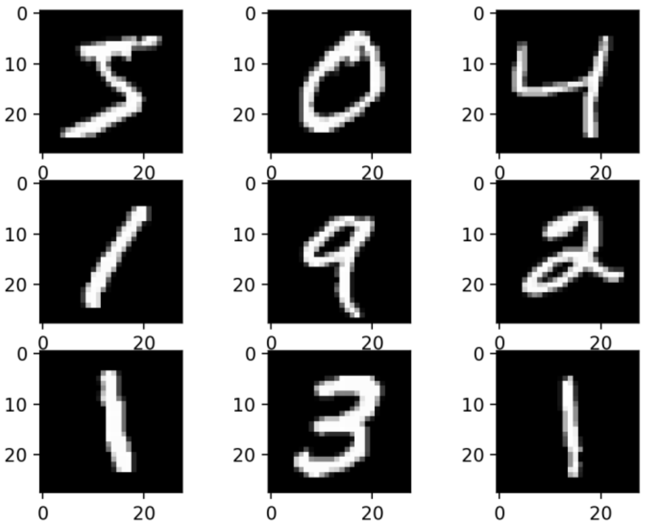
» Res ... nce

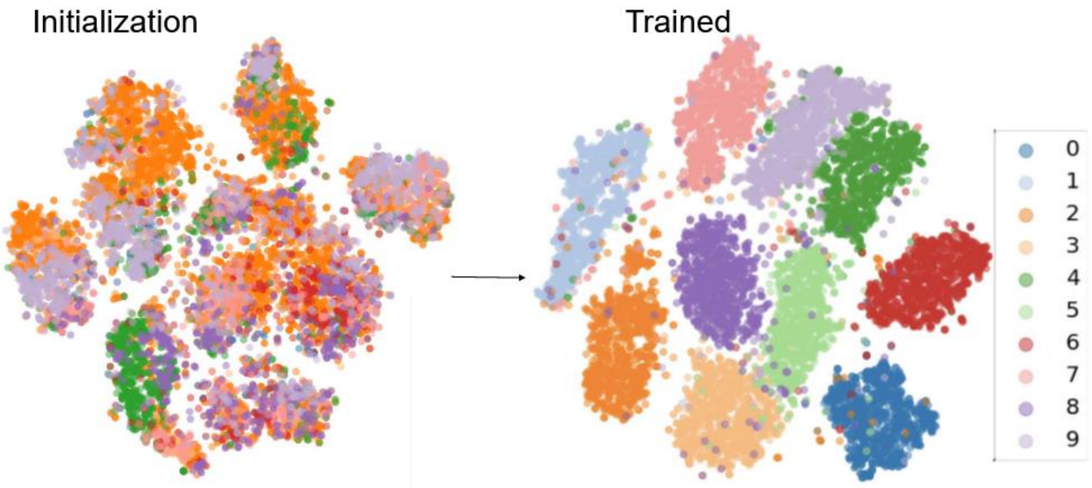
» In s

» PEF

» The

- (
- I





	Fully connected models			Convolutional models		
	MNIST	CIFAR10	CIFAR100	MNIST	CIFAR10	CIFAR100
BP	98.63±0.03	55.27±0.32	27.58±0.09	98.86±0.04	64.99±0.32	34.20±0.20
FA	98.42±0.07	53.82±0.24	24.61±0.28	98.50±0.06	57.51±0.57	27.15±0.53
DRTP	95.10±0.10	45.89±0.16	18.32±0.18	97.32±0.25	50.53±0.81	20.14±0.68
PEPITA	98.01±0.09	52.57±0.36	24.91±0.22	98.29±0.13	56.33±1.35	27.56±0.60

Beyond PEPITA: towards time locality

PEPITA

Algorithm 1 Implementation of PEPITA

Given: Input (x) and label ($target$)

#standard forward pass

$h_0 = x$

for $\ell = 1, ..., L$

$\quad h_\ell = \sigma_\ell(W_\ell h_{\ell-1})$

$e = h_L - target$

#modulated forward pass

$h_0^{err} = x + Fe$

for $\ell = 1, ..., L$

$\quad h_\ell^{err} = \sigma_\ell(W_\ell h_{\ell-1}^{err})$

$\quad$ **if** $\ell < L$:

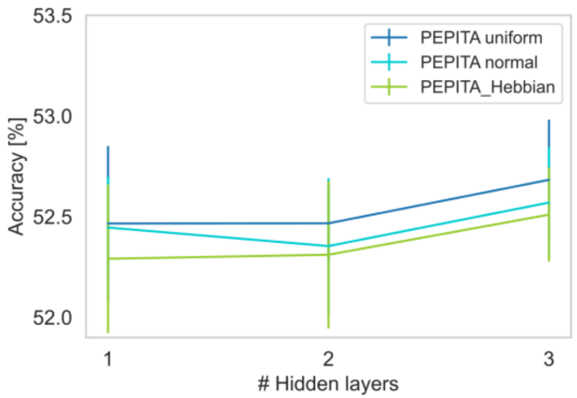
$\quad\quad \Delta W_\ell = (h_\ell - h_\ell^{err}) \cdot (h_{\ell-1}^{err})^T$

$\quad$ **else:**

$\quad\quad \Delta W_\ell = e \cdot (h_{\ell-1}^{err})^T$

$\quad$ #apply update

$\quad W_\ell(t+1) = W_\ell(t) - \eta \Delta W_\ell$



$$\Delta W_\ell = h_\ell \cdot h_{\ell-1}^{errT} - h_\ell^{err} \cdot h_{\ell-1}^{errT}$$
$$\simeq h_\ell \cdot h_{\ell-1}^{T} - h_\ell^{err} \cdot h_{\ell-1}^{errT}$$

PEPITA 2.0

Algorithm 2 Implementation of PEPITA local in space

Given: Input (x) and label ($target$)

#standard forward pass

$h_0 = x$

for $\ell = 1, ..., L$

$\quad h_\ell = \sigma_\ell(W_\ell h_{\ell-1})$

$\quad \Delta W_\ell^+ = h_\ell \cdot h_{\ell-1}^T$

$\quad$ #apply update for the positive phase

$\quad W_\ell^+(t+1) = W_\ell(t) - \eta \Delta W_\ell^+$

$e = h_L - target$

#modulated forward pass

$h_0^{err} = x + Fe$

for $\ell = 1, ..., L$

$\quad h_\ell^{err} = \sigma_\ell(W_\ell^+ h_{\ell-1}^{err})$

$\quad$ **if** $\ell < L$:

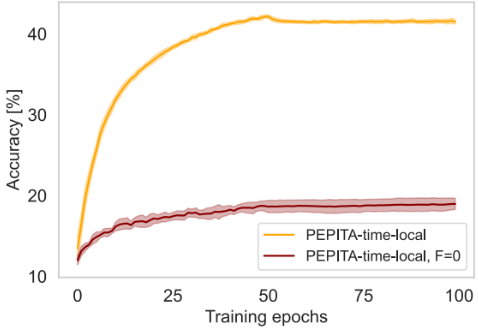
$\quad\quad \Delta W_\ell^- = -h_\ell^{err} \cdot h_{\ell-1}^{errT}$

$\quad$ **else:**

$\quad\quad \Delta W_\ell^- = -target \cdot h_{\ell-1}^{errT}$

$\quad$ #apply update for the negative phase

$\quad W_\ell(t+1) = W_\ell^+(t+1) - \eta \Delta W_\ell^-$



Acknowledgements

Gabriel Kreiman
Harvard, Boston Children's Hospital

Will Xiao
Harvard Medical School

TINY ML
On device learning Forum

Copyright Notice

www.tinyml.org

T I N Y

On device learning Forum

Copyright Notice

www.tinyml.org