Hebbian Learning Algorithms for Training Convolutional Neural Networks

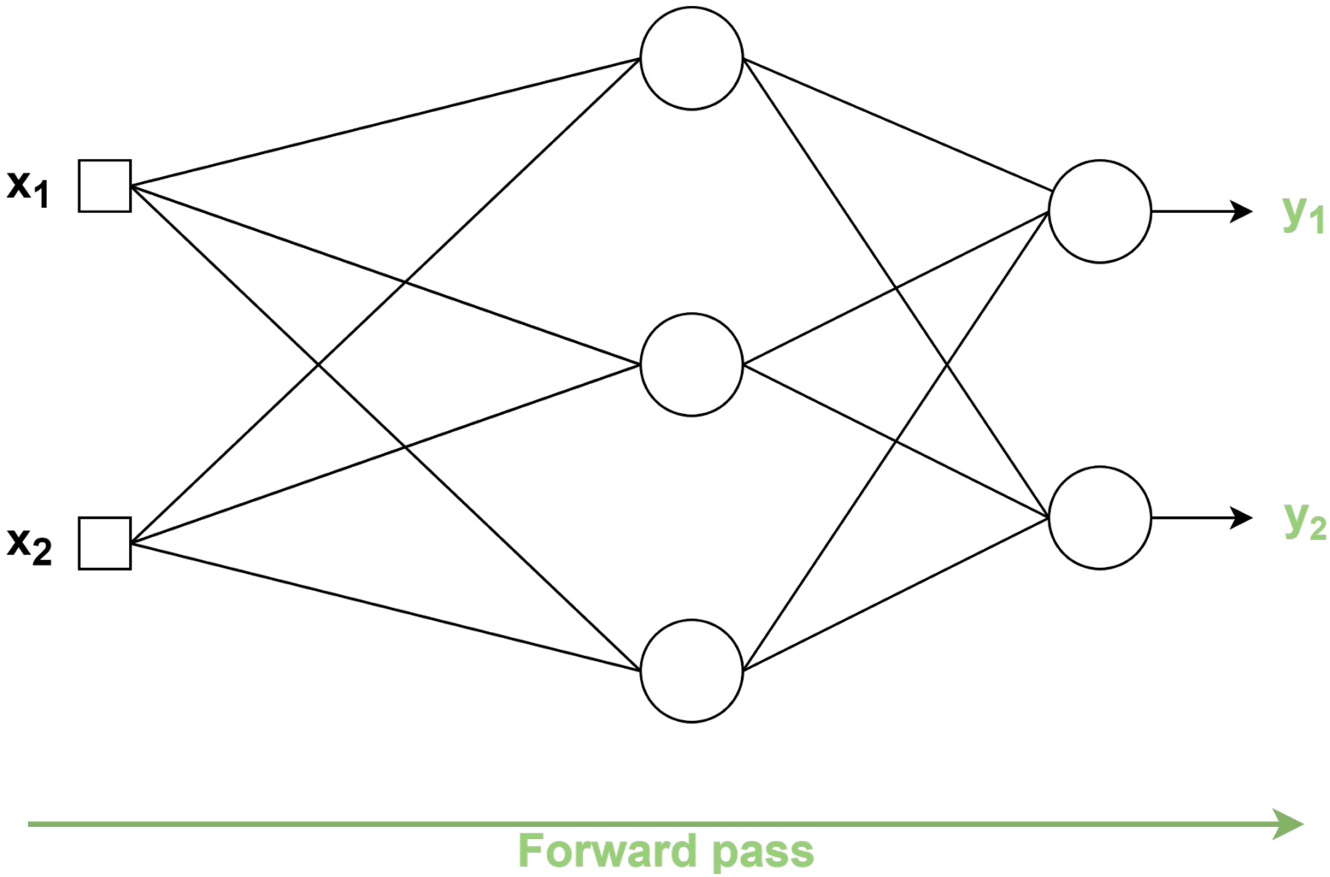
Gabriele Lagani
Computer Science PhD
University of Pisa

Outline

- SGD vs Hebbian learning
- Hebbian learning variants
- Training CNNs with Hebbian + WTA approach on image classification tasks (CIFAR-10 dataset)
- Comparison with CNNs trained with SGD
- Results and conclusions

SGD vs Hebbian Learning

- SGD training requires forward and backward pass



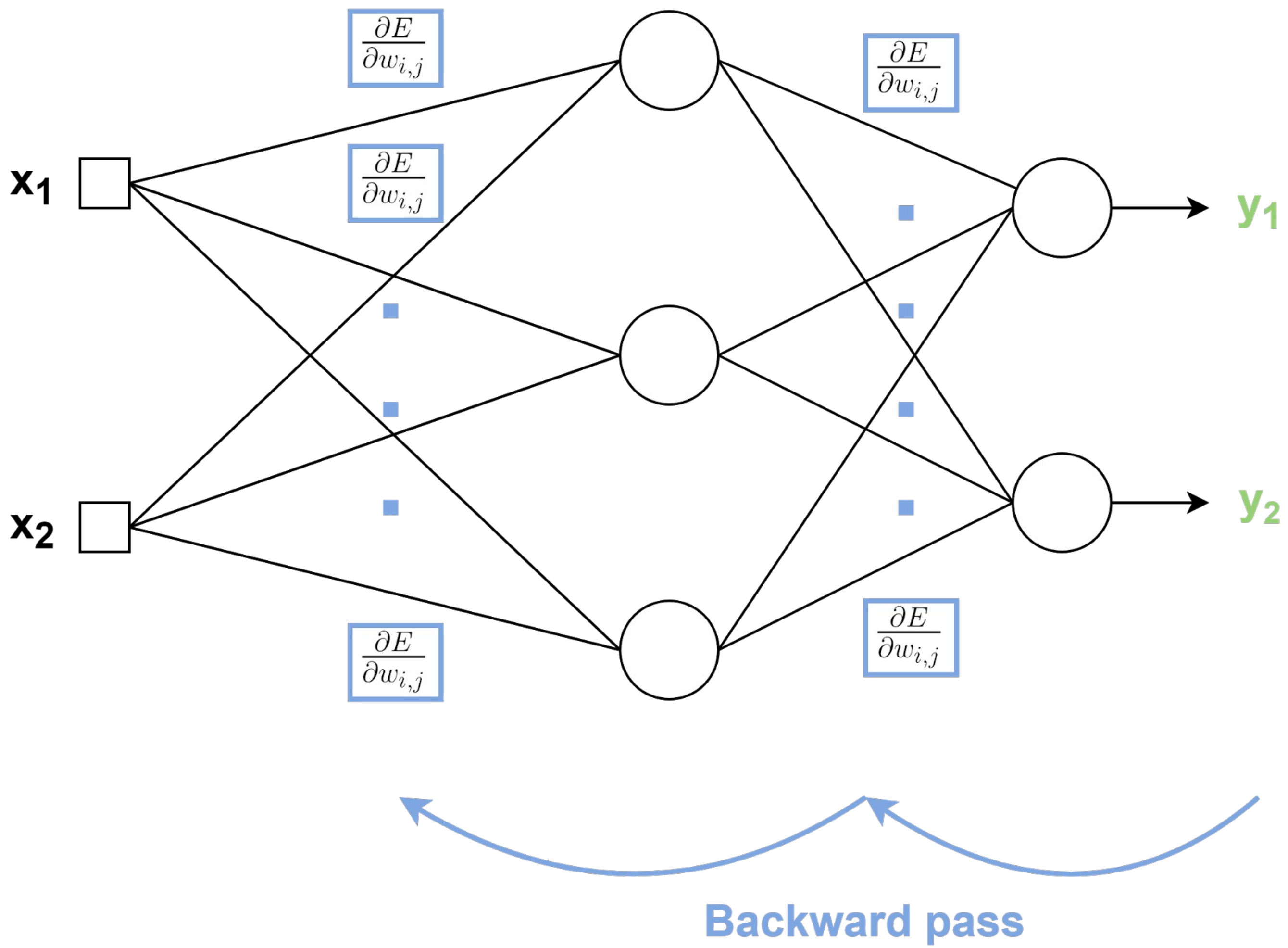
Error/loss computation: E(**y**, **t**)

Gradient computation:

$$\frac{\partial E}{\partial w_{i,j}}$$

SGD vs Hebbian Learning

- SGD training requires forward and backward pass



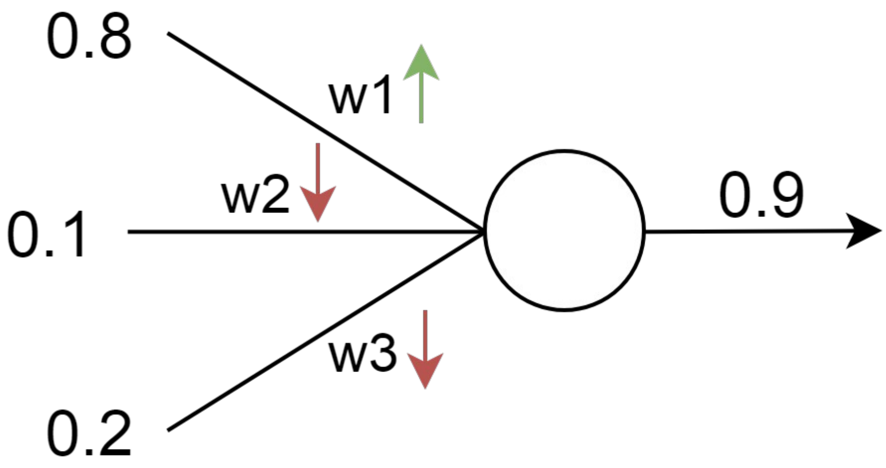
Error/loss computation: E(**y**, **t**)

Gradient computation:

$$\frac{\partial E}{\partial w_{i,j}}$$

SGD vs Hebbian Learning

- Hebbian learning rule: $\Delta w = \eta \, y(x, \, w) \, x$
- Unique local forward pass
- Advantage: layer-wise parallelizable



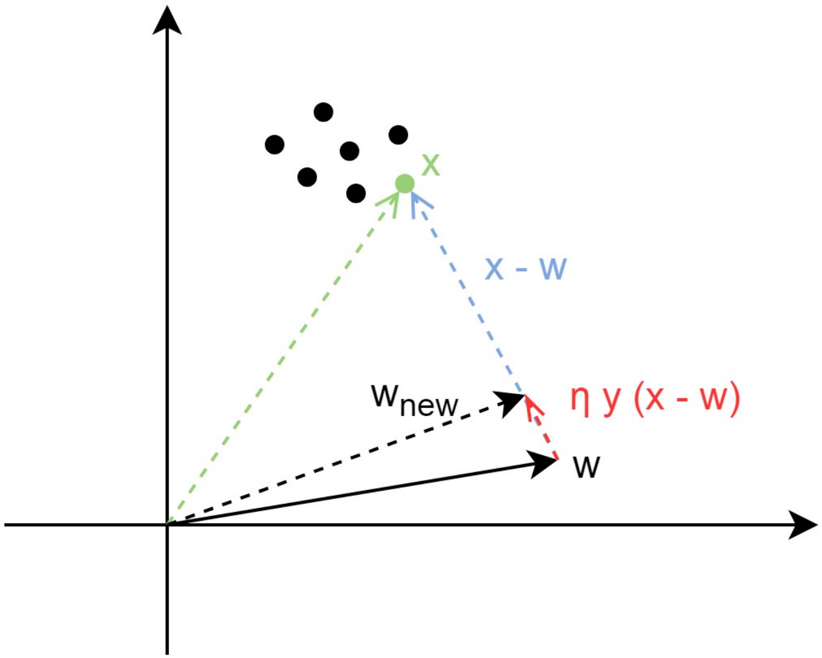
Hebbian Learning Variants

- Weight decay:

$$\Delta w = \eta \, y(x, w) \, x - \gamma(x, w)$$

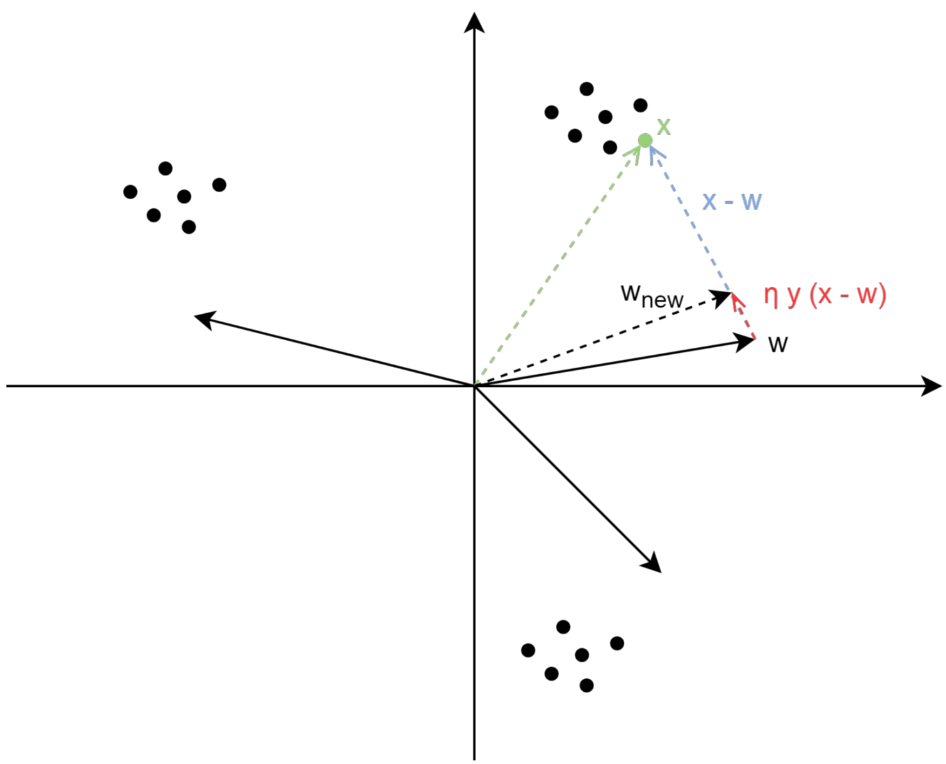
- Taking $\gamma(x, w) = \eta \, y(x, w) \, w$

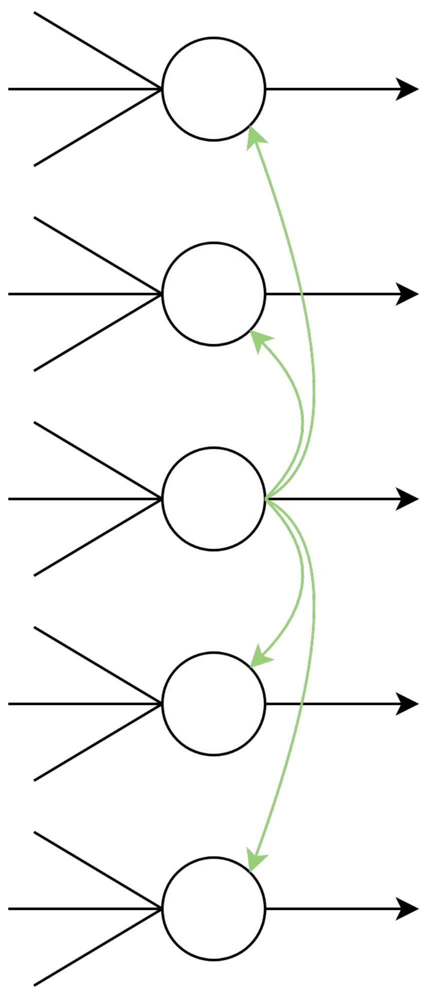
$$\Delta w = \eta \, y(x, w) \, (x - w)$$



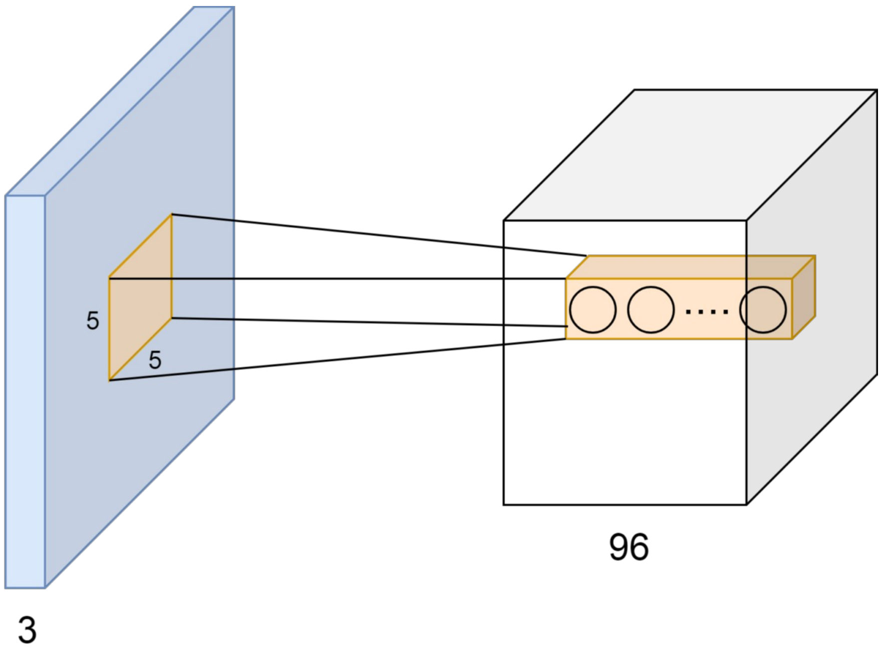
Lateral Interaction

- Competitive learning
  - Winner-Takes-All (WTA)
  - Self-Organizing Maps (SOM)





Convolutional Layers

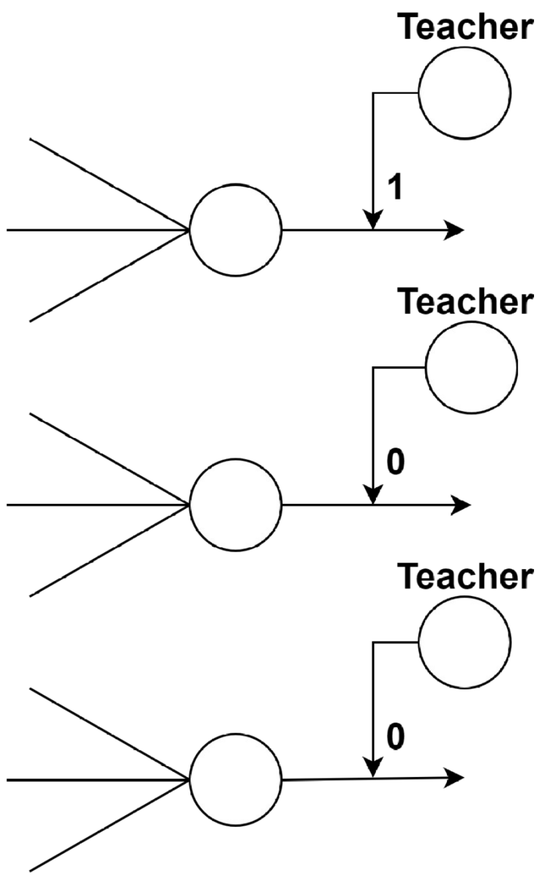


- Sparse connectivity
- Shared weights
- Translation invariance

Update aggregation by averaging in order to maintain shared weights

Final Classification Layer

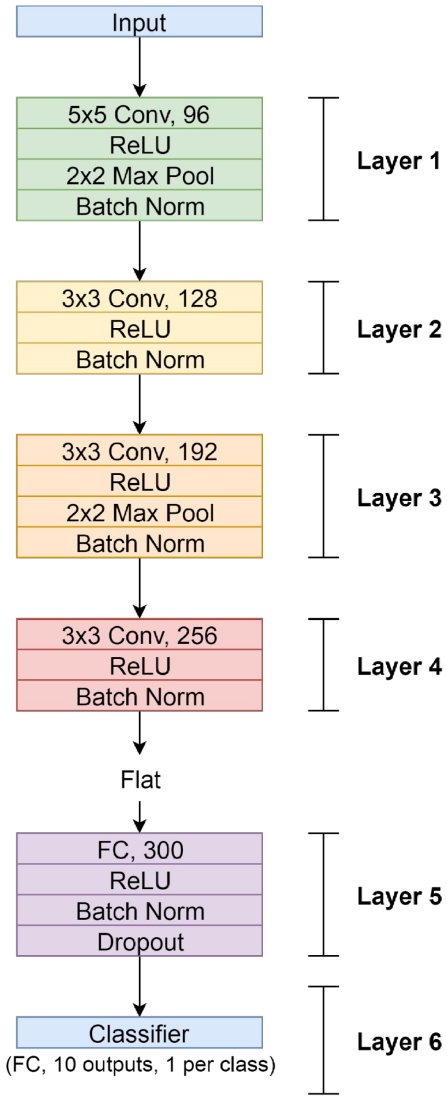
- Supervised Hebbian learning with **teacher neuron**



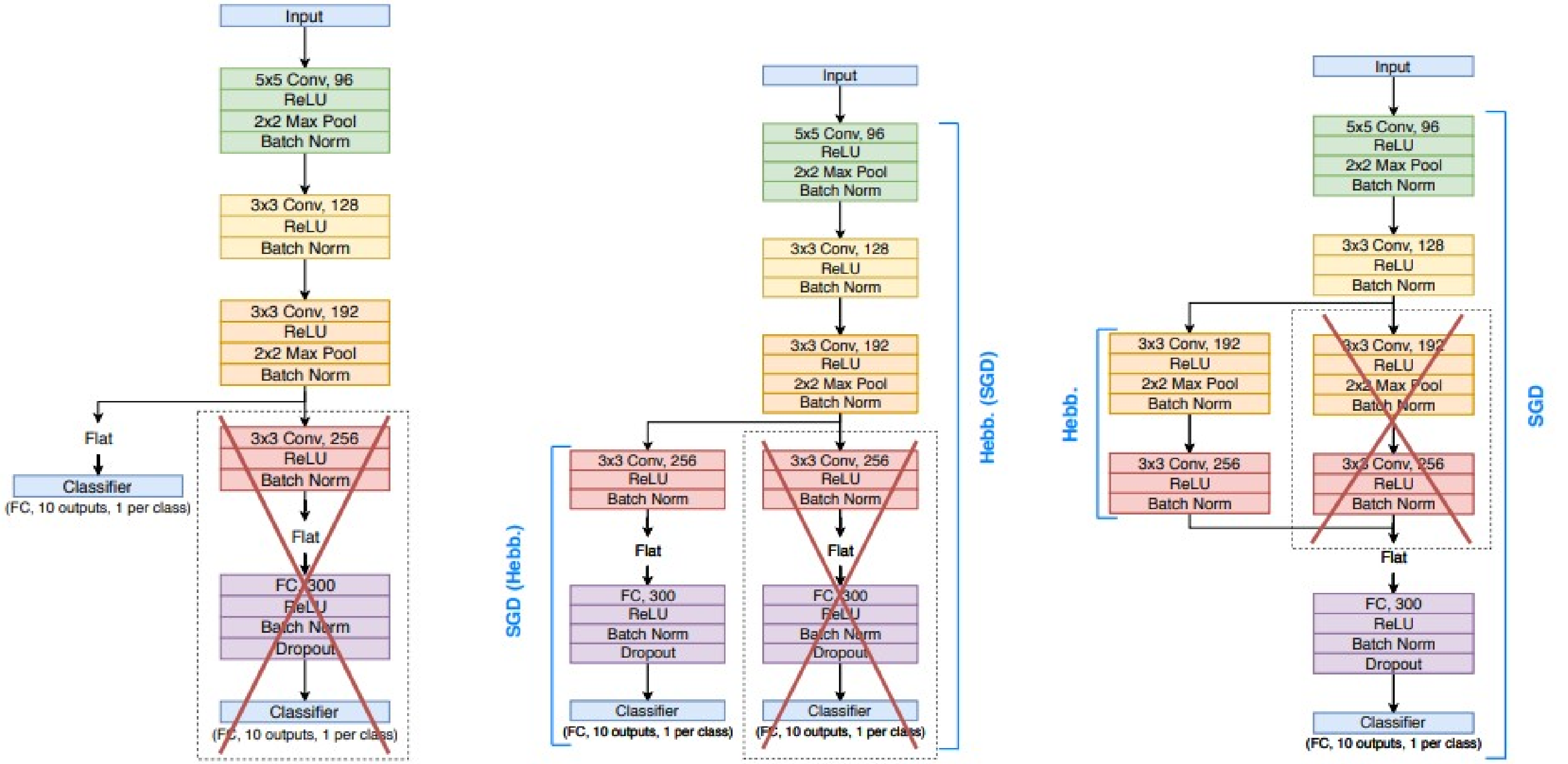
Experimental Setup

- Hebbian + WTA approach applied to **deep CNNs**
- Extension of Hebbian rules to convolutional layers with shared kernels: **update aggregation**
- **Teacher neuron** for supervised Hebbian learning
- **Hybrid** network architectures (Hebbian + SGD layers)

Network Architecture



Different Configurations

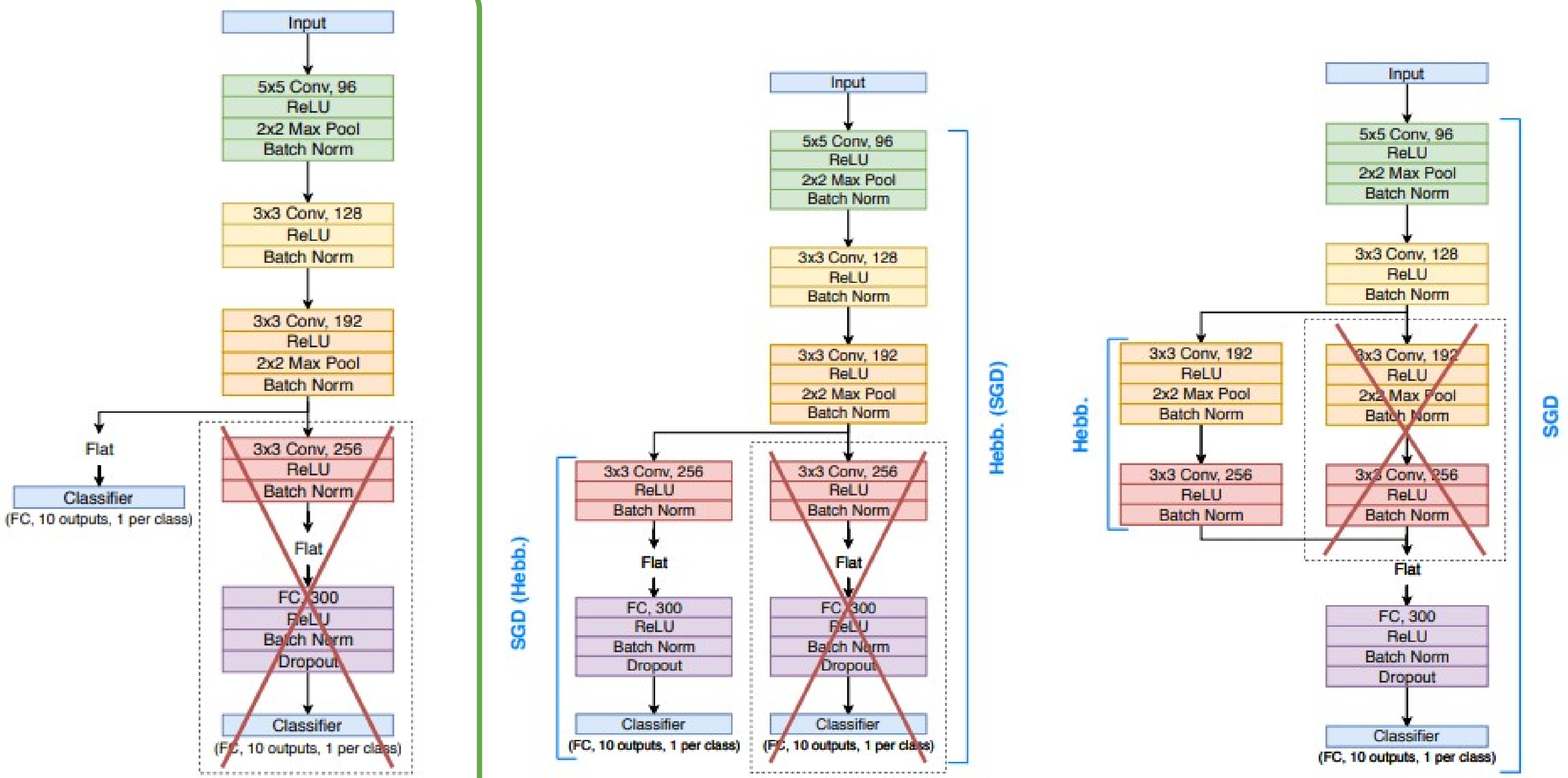


(a) A classifier on top on the i-th layer of the network

(b) SGD layers on top of Hebbian-trained layers (and vice-versa)

(c) Hebbian trained layers in between SGD layers

Classifiers on top of Deep Layers

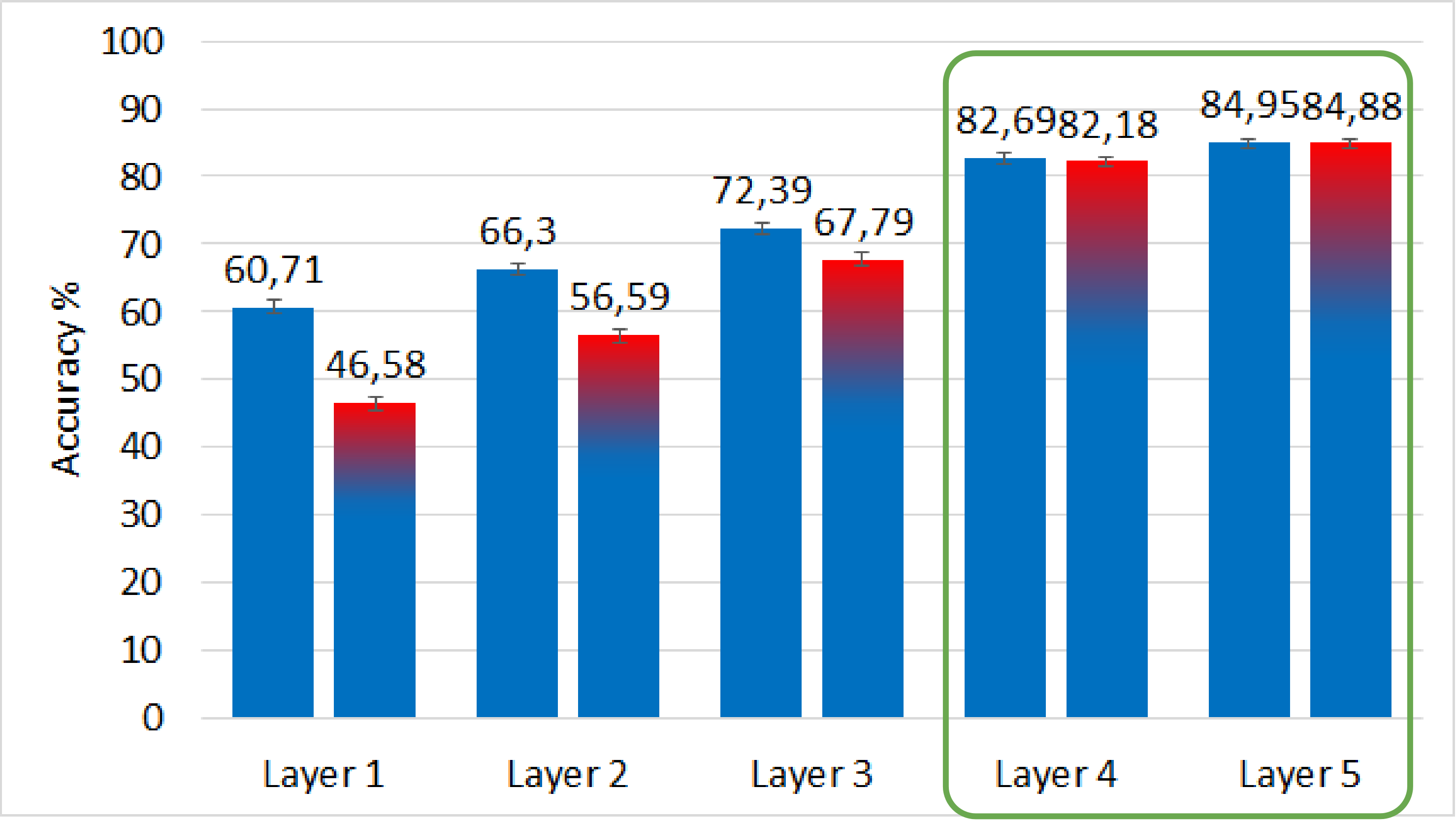


(a) A classifier on top on the i-th layer of the network

(b) SGD layers on top of Hebbian-trained layers (and vice-versa)

(c) Hebbian trained layers in between SGD layers

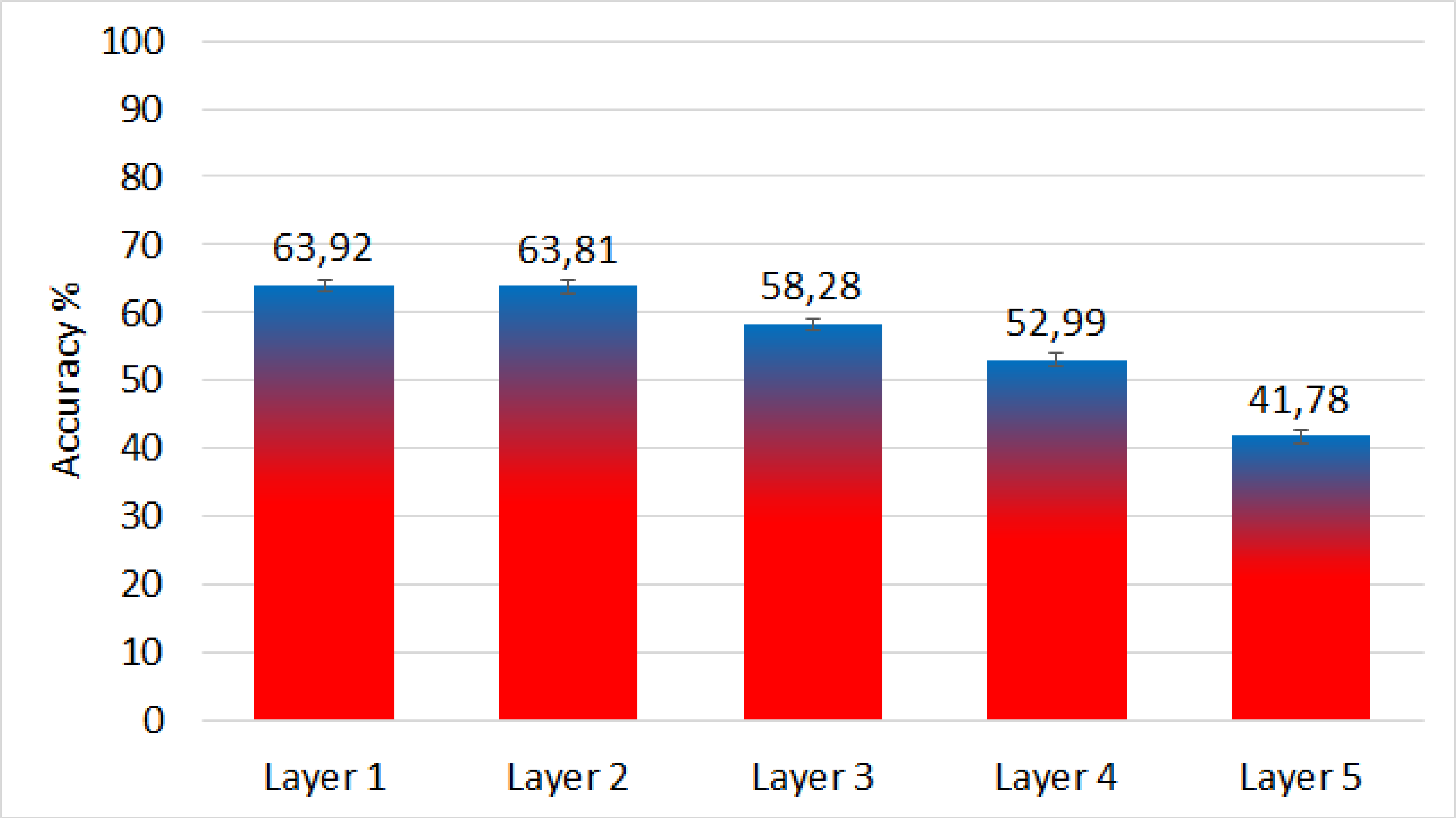
Classifiers on Deep Layers Trained with SGD



Considerations on Hebbian classifier:

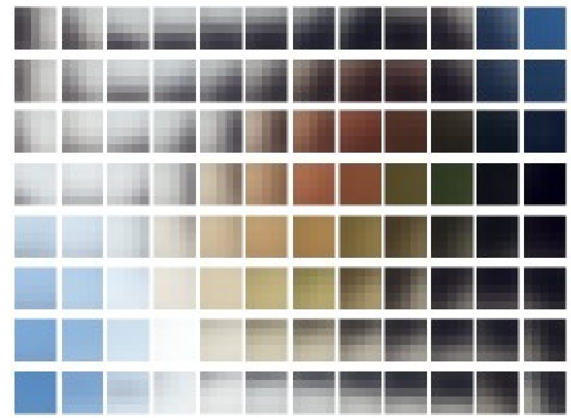
- Pros: good on **high-level features**, fast training (1-2 epochs)
- Cons: bad on low-level features

Classifiers on Hebbian Deep Layers

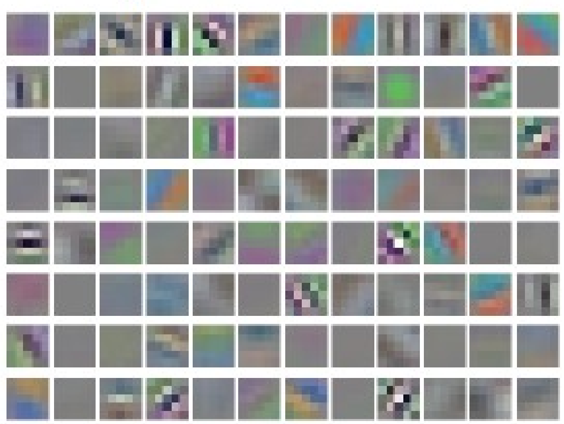


Layer 1 Kernels

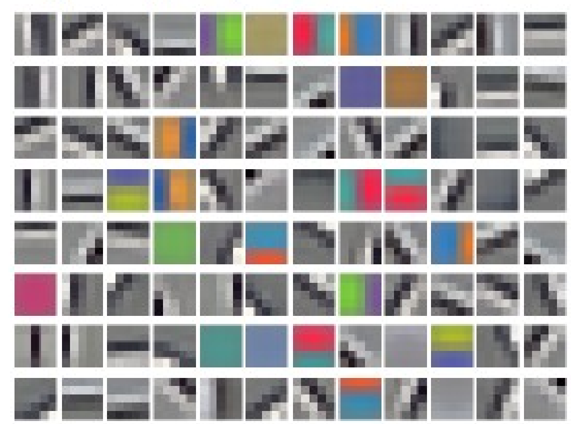
(a) Kernels obtained using Hebbian-WTA algorithm.



(b) Kernels obtained using a Self-Organizing Map.

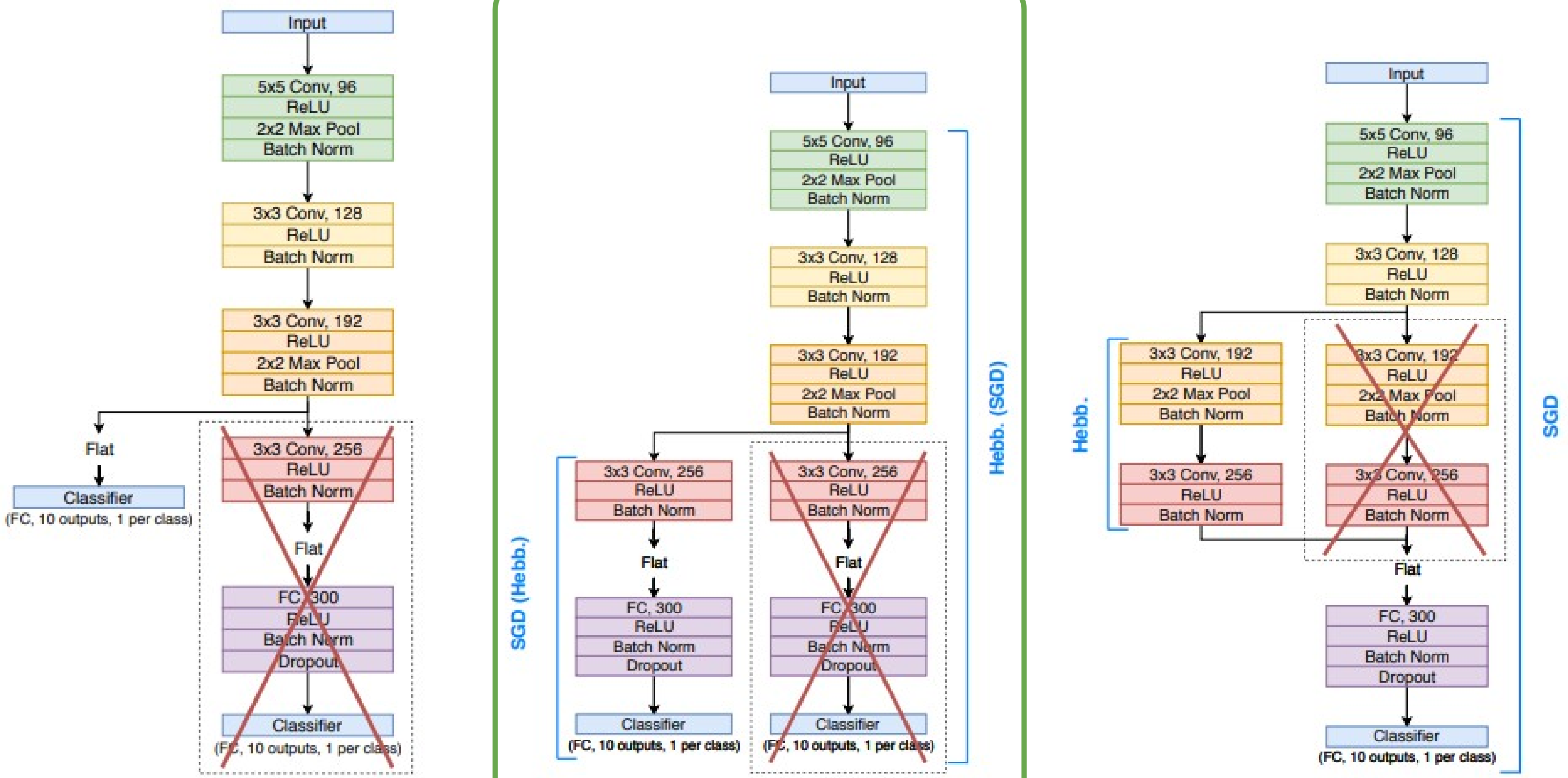


(c) Kernels obtained using Gradient Descent.



(d) Kernels obtained using Hebbian-WTA algorithm with image whitening.

Hybrid Networks

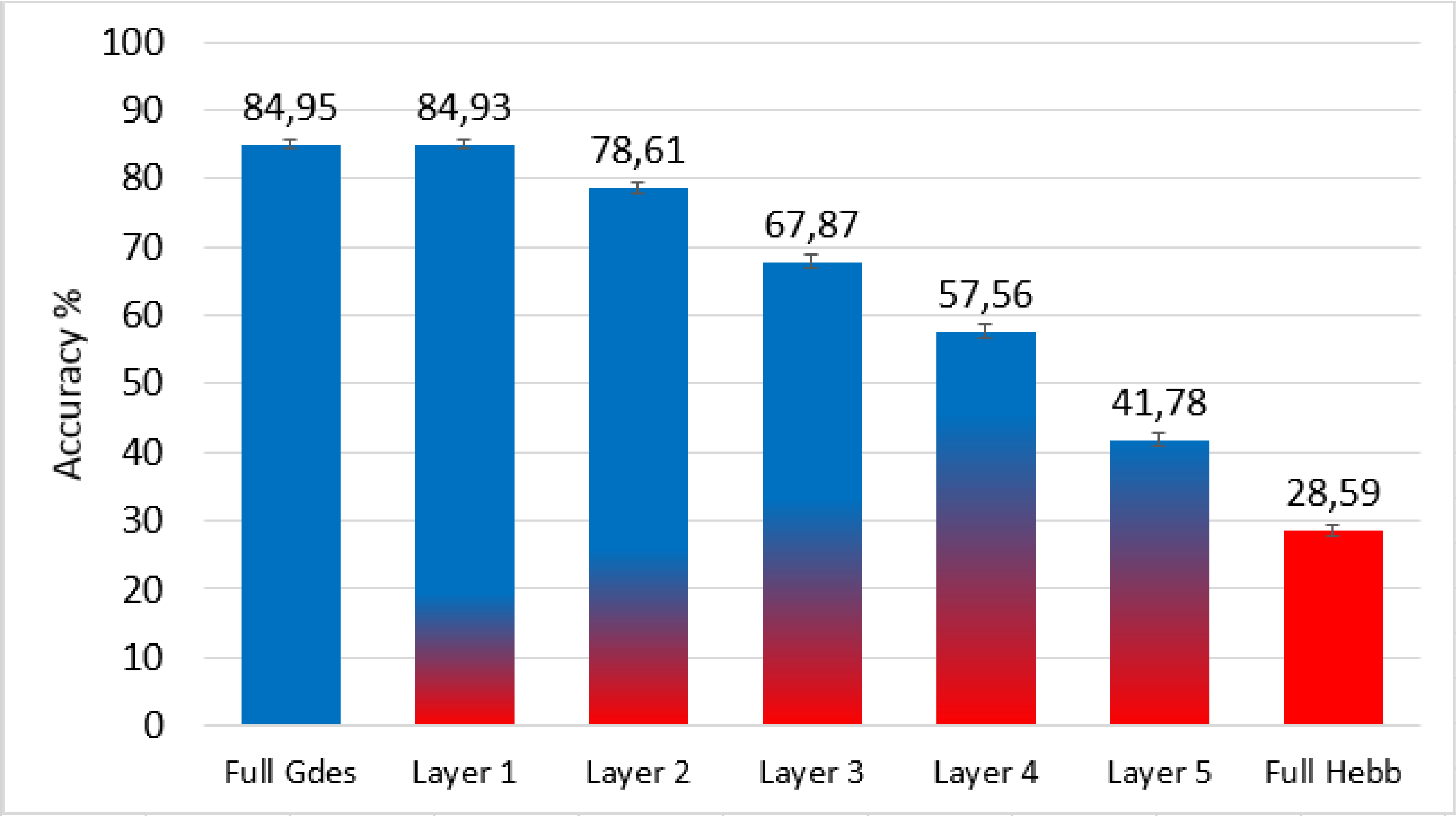


(a) A classifier on top on the i-th layer of the network

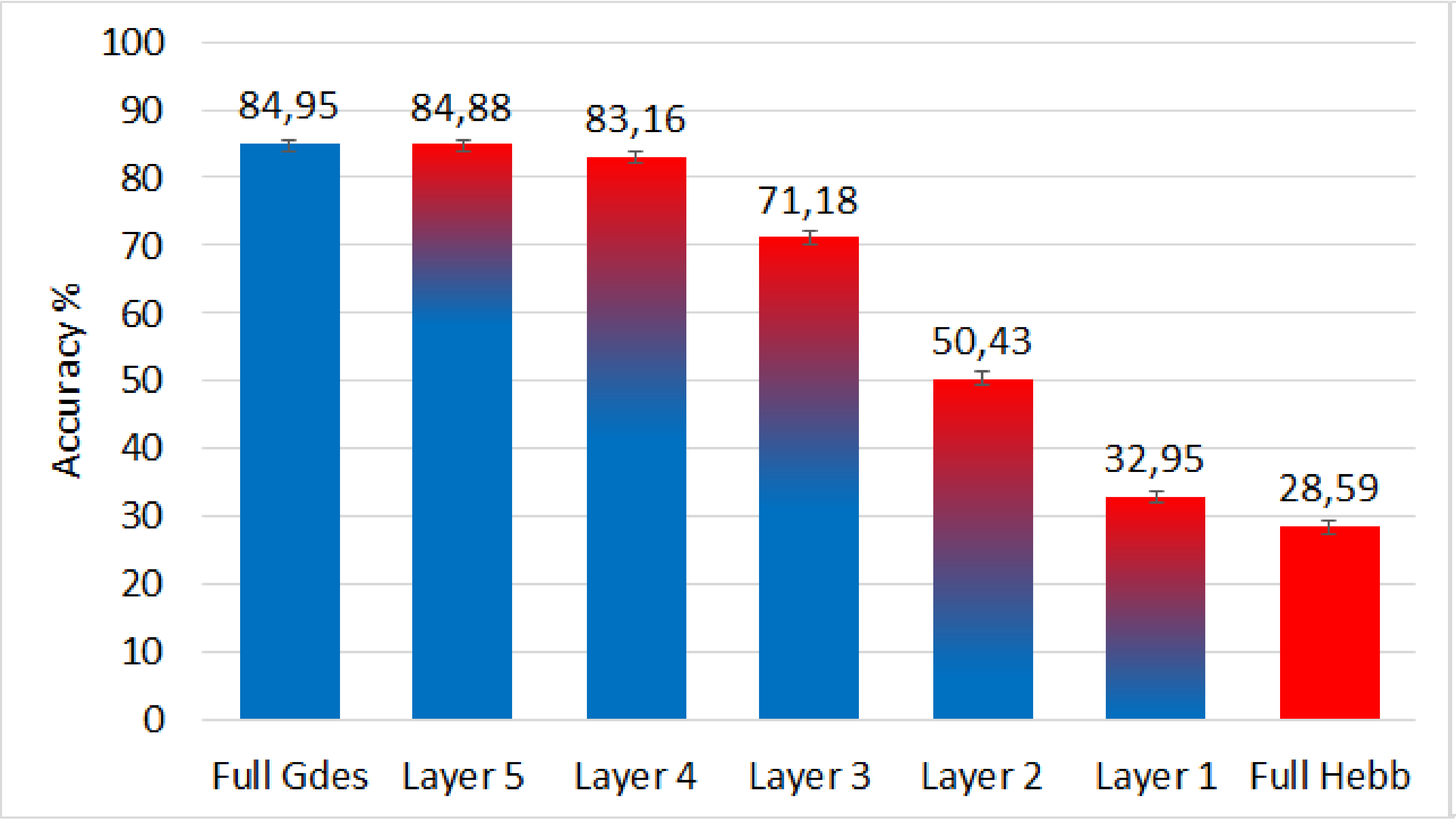
(b) SGD layers on top of Hebbian-trained layers (and vice-versa)

(c) Hebbian trained layers in between SGD layers

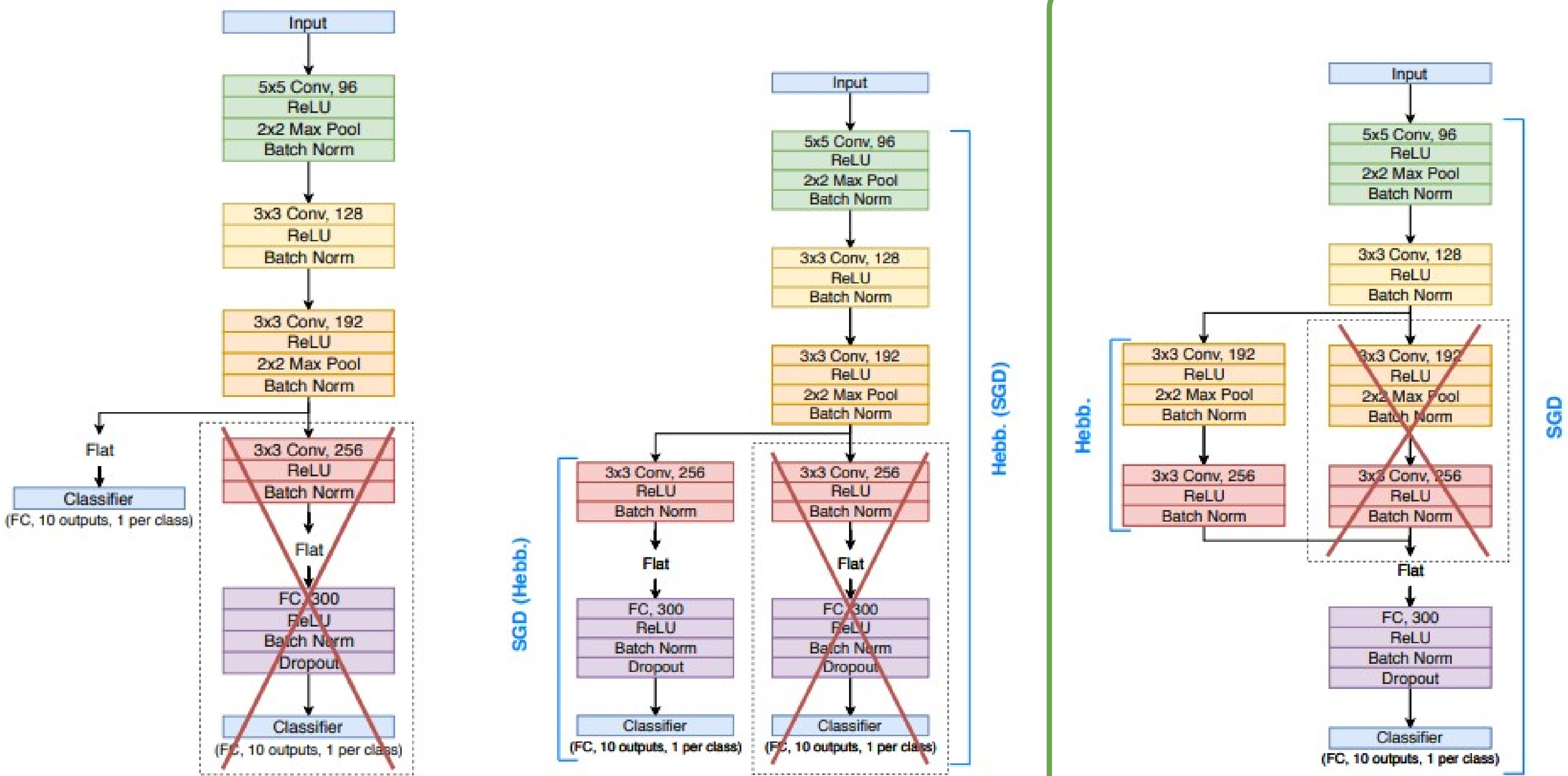
Hybrid Networks: Bottom Hebb. - Top SGD



Hybrid Networks: Bottom SGD - Top Hebb.



Hybrid Networks: SGD - Hebb. - SGD

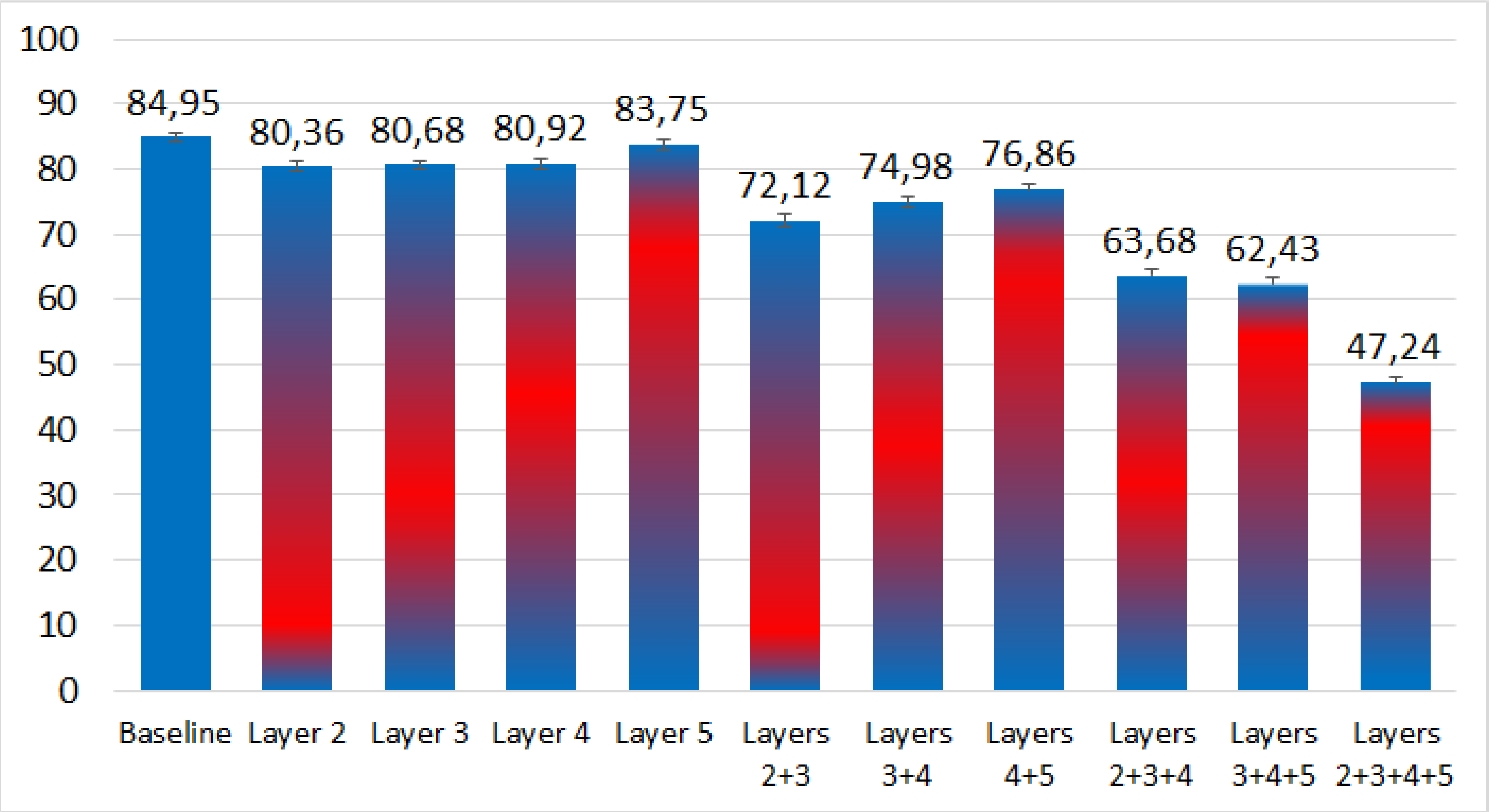


(a) A classifier on top on the i-th layer of the network

(b) SGD layers on top of Hebbian-trained layers (and vice-versa)

(c) Hebbian trained layers in between SGD layers

Hybrid Networks: SGD - Hebb. - SGD

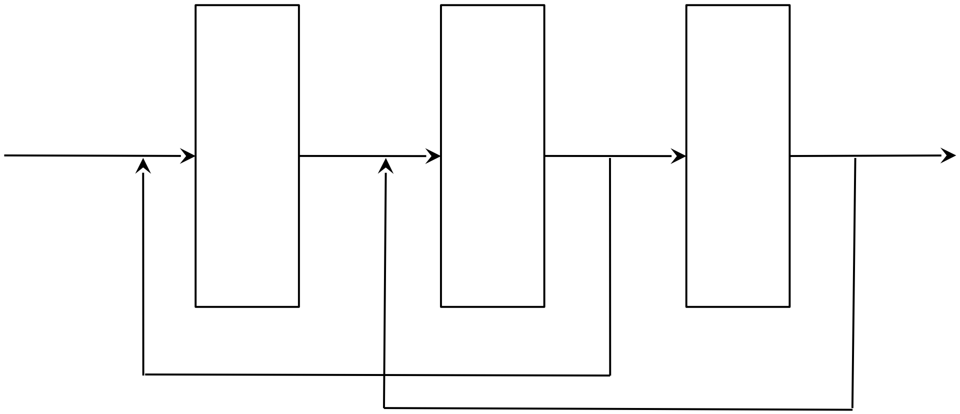


Conclusions

- **Pros** of Hebbian + WTA:
  - Effective for low level feature extraction
  - Effective for training higher network layers, including a classifier on top of high-level features
  - Takes fewer epochs than SGD (2 vs 10)
    → useful for *transfer learning*
- **Cons** of Hebbian + WTA:
  - Not effective for training intermediate network layers
  - Not effective for training a classier on top of low-level features.

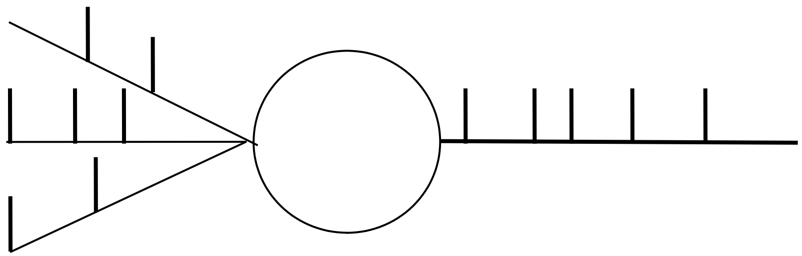
Future Works

- Explore other Hebbian learning variants
  - **Hebbian PCA**
    - Can achieve distributed coding at intermediate layers
  - **Contrastive Hebbian Learning** (CHL)
    - Free phase + clamped phase
    - Update step: $\Delta w = \eta(y^+ x^+ - y^- x^-)$
    - Equivalent to Gradient Descent



Future Works

- Switch to **Spiking Neural Networks** (SNN)



  - Spike Time Dependent Plasticity (**STDP**)

$$\Delta w = \begin{cases} A_{+}e^{-(t_{out}-t_{in})/\tau_{+}} & \text{if } t_{out} > t_{in} \\ A_{-}e^{(t_{out}-t_{in})/\tau_{-}} & \text{if } t_{out} \leq t_{in} \end{cases}$$

  - Higher biological plausibility
  - Low power consumption
    - Good for neuromorphic hardware implementation
    - Ideal for applications on constrained devices

References

- G. Amato, F. Carrara, F. Falchi, C. Gennaro,G. Lagani; Hebbian Learning Meets Deep Convolutional Neural Networks (2019) http://www.nmis.isti.cnr.it/falchi/Draft/2019-ICIAP-HLMSD.pdf
- S. Haykin; Neural Networks and Learning Machines (2009)
- W.Gerstner, W. Kistler; Spiking Neuron Models (2002)
- X. Xie, S. H. Seung; Equivalence of Backpropagation and Contrastive Hebbian Learning in a Layered Network (2003)