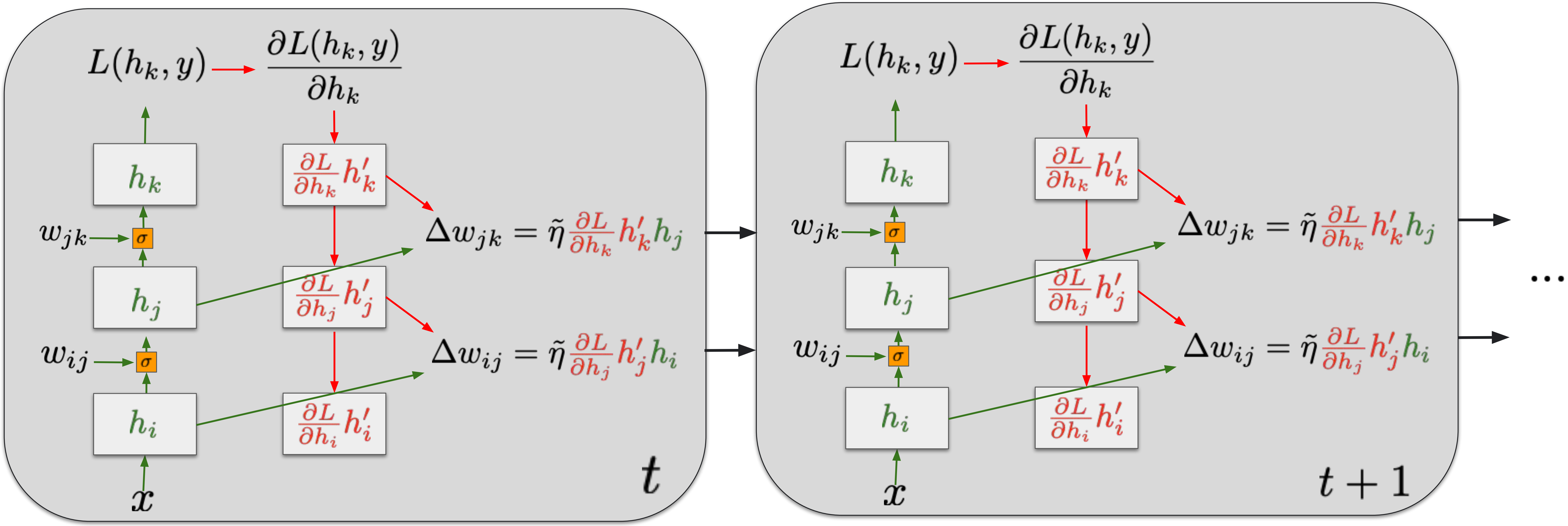
38th International Conference on Machine Learning, ICML 2021

Meta-Learning Bidirectional Update Rules

Mark Sandler
Max Vladymyrov
Andrey Zhmoginov
Nolan Miller
Andrew Jackson
Tom Madams
Blaise Agüera y Arcas

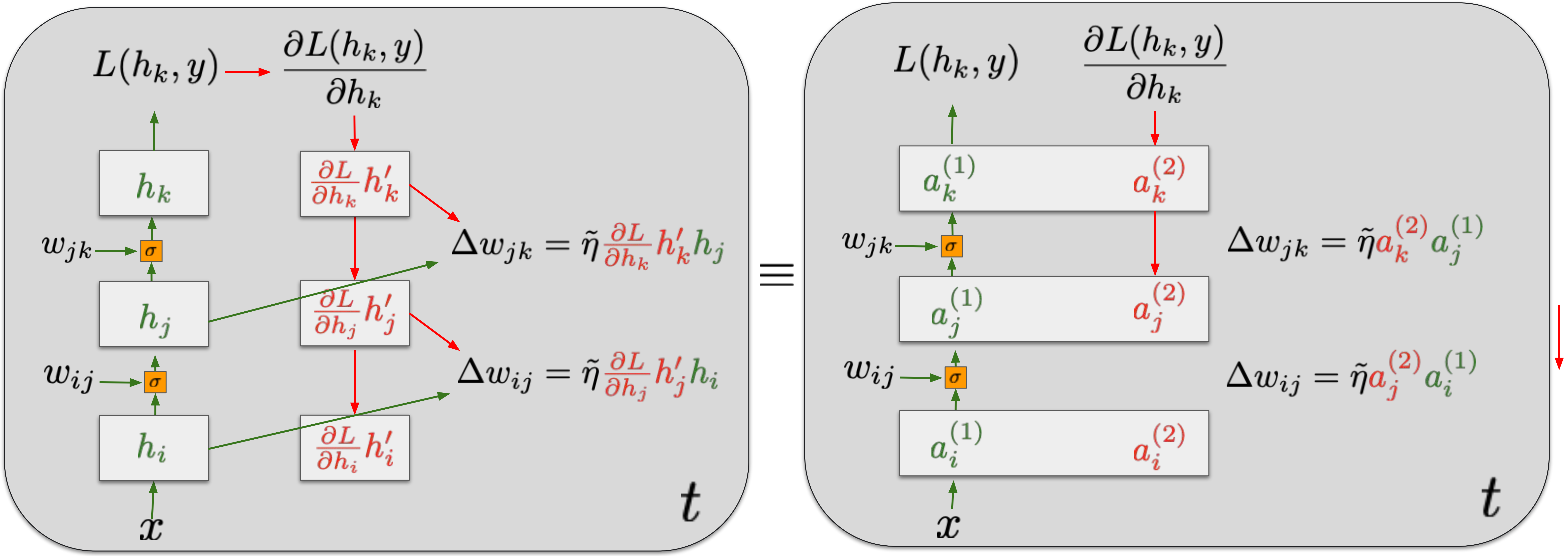
Stochastic Gradient Descent

- Requires a **predefined loss function** computed for every iteration.
- Synapse update is computed via **backpropagation** of the loss function.
- Optimization procedure is **independent** from the dataset.

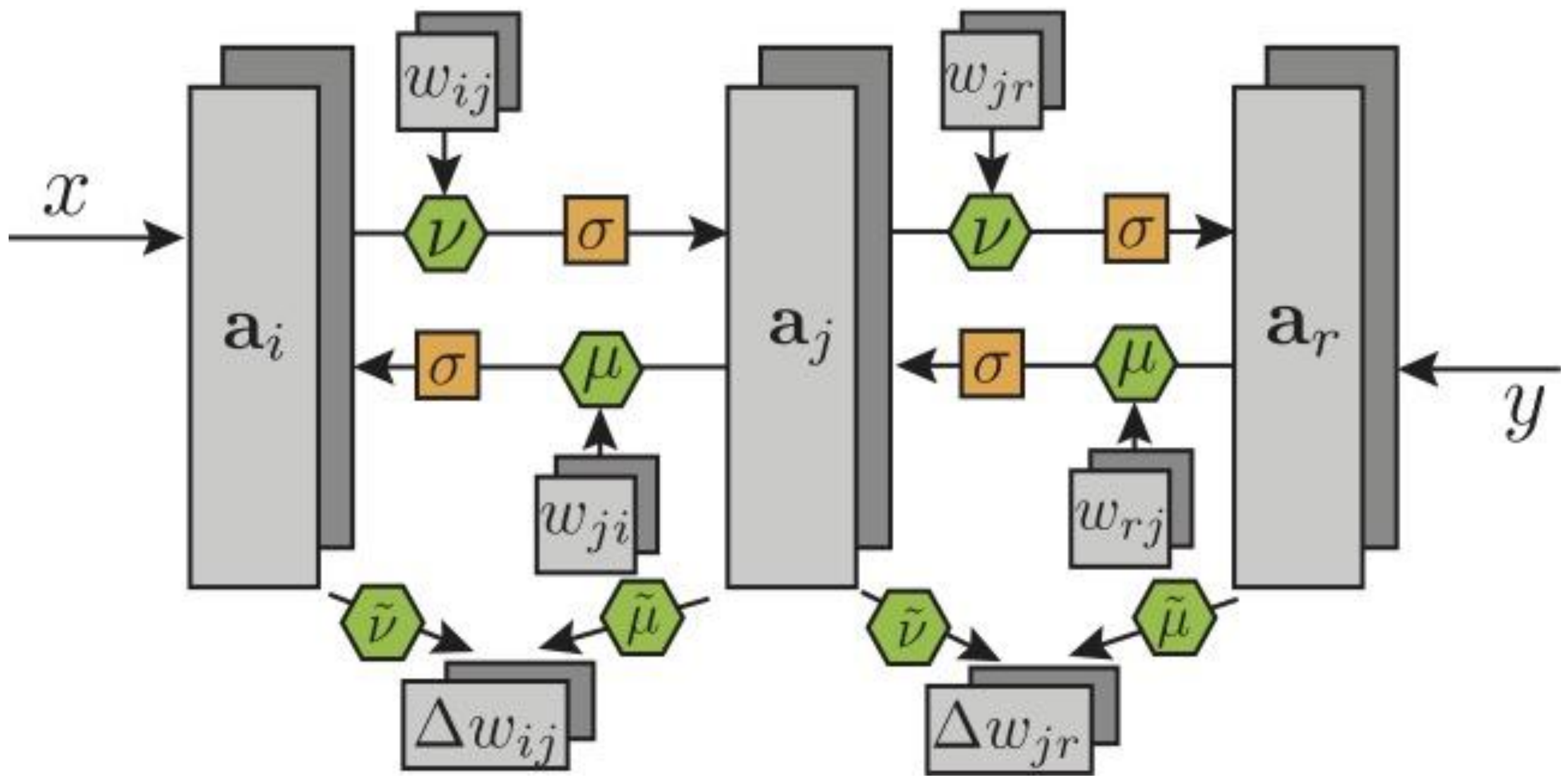


SGD using two-state neurons

Backpropagation can be equivantly reformulated with **generalized two-state neurons** a_j^c where j is a layer and $c \in \{0, 1\}$ is a state.



Bidirectional Learning Update Rules



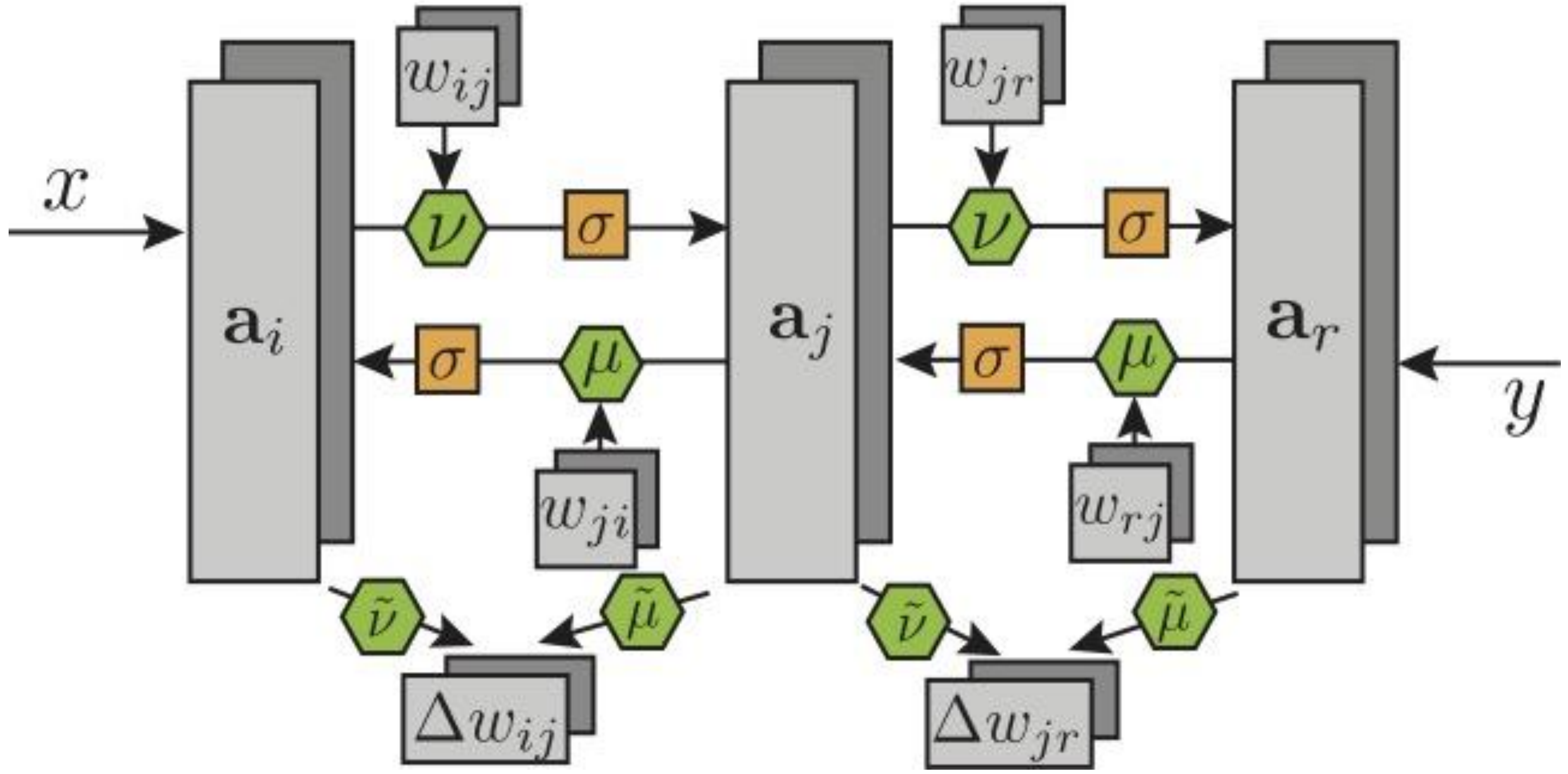
Forward pass: $$a_j^c \leftarrow \sigma\Big(f a_j^c + \eta \sum_{i,d} w_{ij}^c \nu^{cd} a_i^d\Big)$$

Backward pass: $$a_i^c \leftarrow \sigma\Big(f a_i^c + \eta \sum_{j,d} w_{ji}^c \mu^{cd} a_j^d\Big)$$

Weights update: $$w_{ij}^c \leftarrow \tilde{f} w_{ij}^c + \tilde{\eta} \sum_{e,d} a_i^e \tilde{\nu}^{ec} \cdot \tilde{\mu}^{cd} a_j^d$$

Bidirectional Learning Update Rules

Generalized formulation allows for *more than two* neuron states.



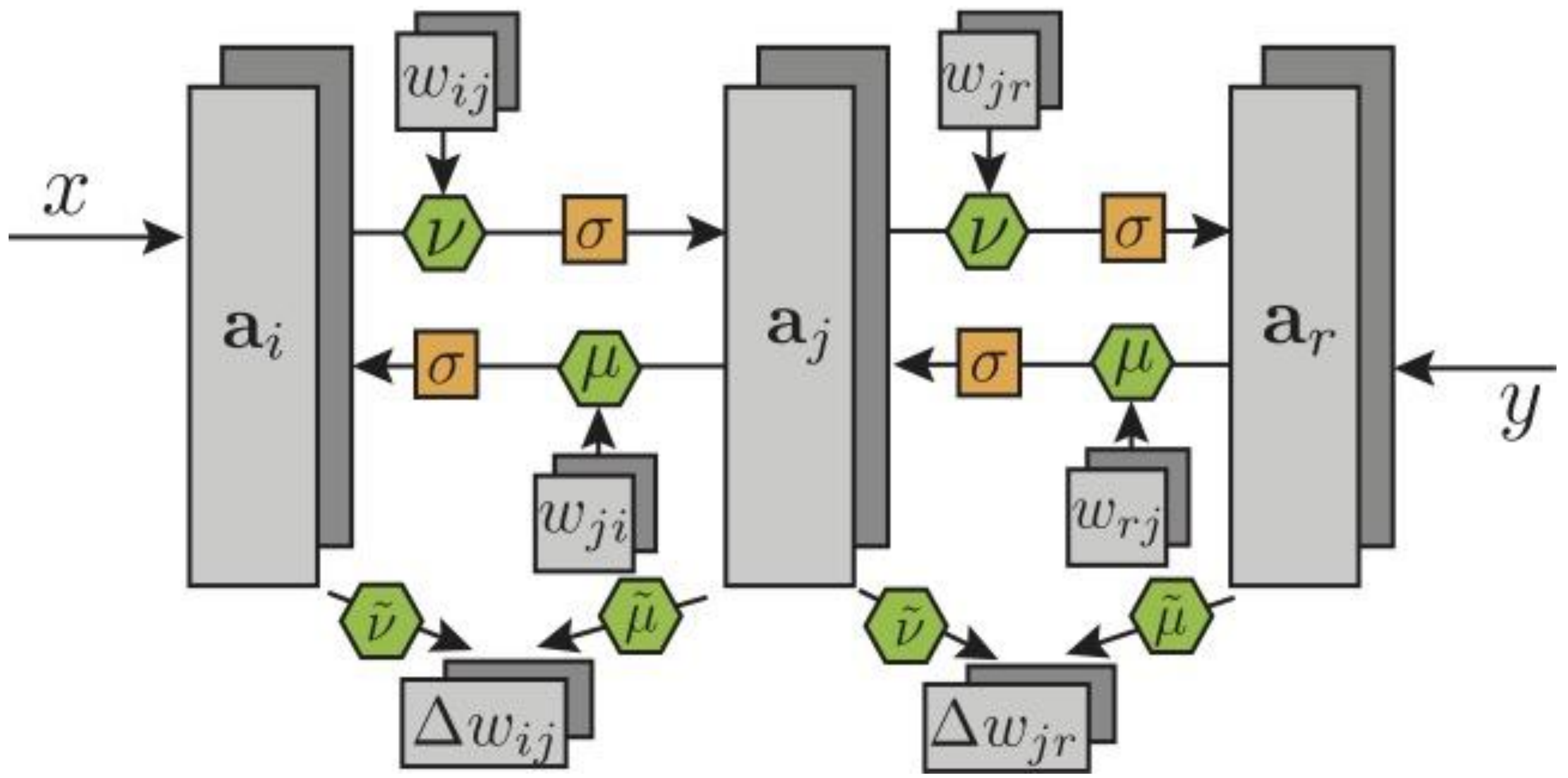
Forward pass: $a_j^c \leftarrow \sigma\left(f a_j^c + \eta \sum_{i,d} w_{ij}^c \nu^{cd} a_i^d\right)$

Backward pass: $a_i^c \leftarrow \sigma\left(f a_i^c + \eta \sum_{j,d} w_{ji}^c \mu^{cd} a_j^d\right)$

Weights update: $w_{ij}^c \leftarrow \tilde{f} w_{ij}^c + \tilde{\eta} \sum_{e,d} a_i^e \tilde{\nu}^{ec} \cdot \tilde{\mu}^{cd} a_j^d$

Bidirectional Learning Update Rules

Distinct *forward* and *backward* synapses (removing weight transport problem).



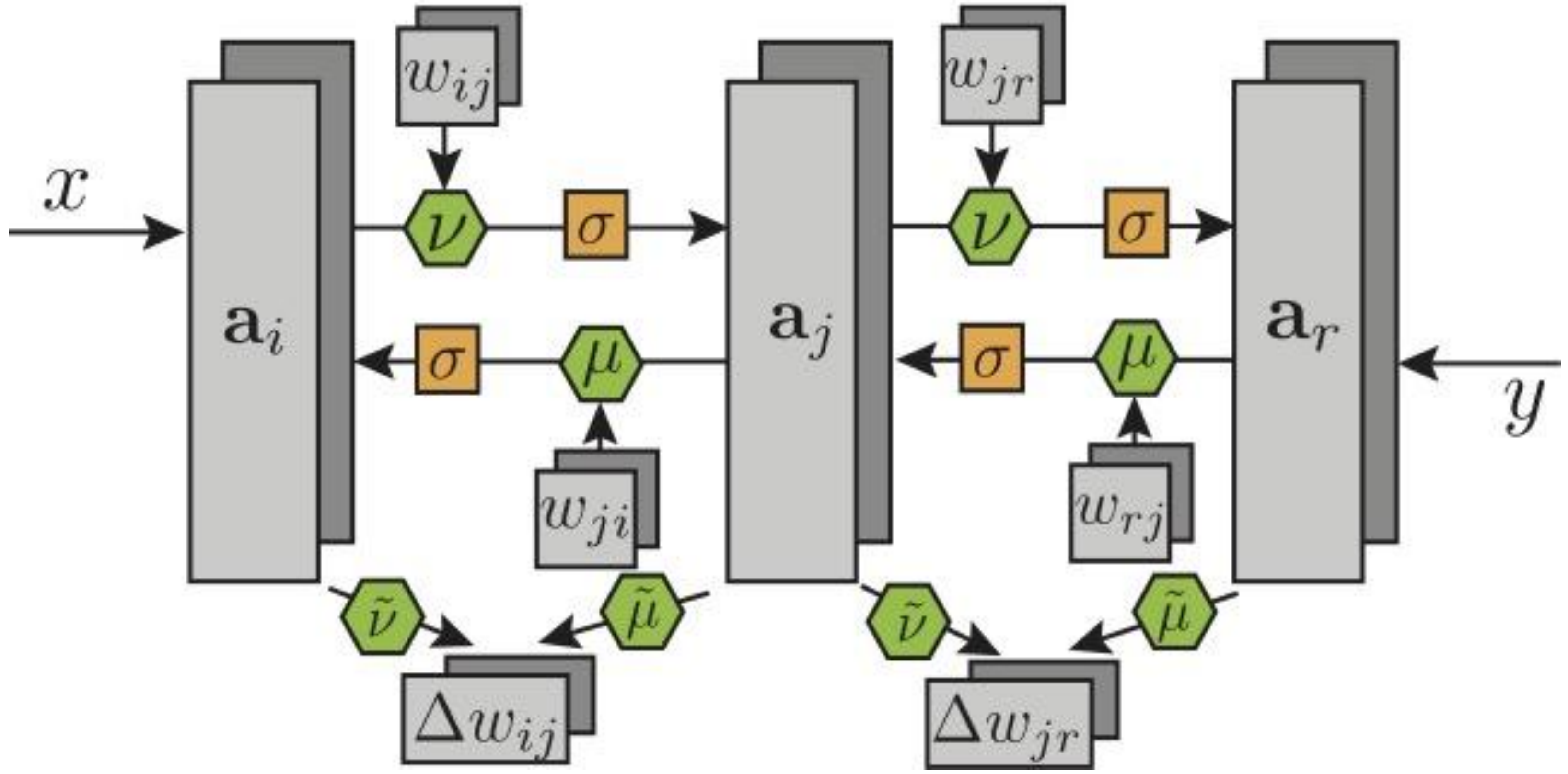
Forward pass: $a_j^c \leftarrow \sigma\left(f a_j^c + \eta \sum_{i,d} w_{ij}^c \nu^{cd} a_i^d\right)$

Backward pass: $a_i^c \leftarrow \sigma\left(f a_i^c + \eta \sum_{j,d} w_{ji}^c \mu^{cd} a_j^d\right)$

Weights update: $w_{ij}^c \leftarrow \tilde{f} w_{ij}^c + \tilde{\eta} \sum_{e,d} a_i^e \tilde{\nu}^{ec} \cdot \tilde{\mu}^{cd} a_j^d$

Bidirectional Learning Update Rules

Meta-learned *transform* matrices for interaction between the neuron states.



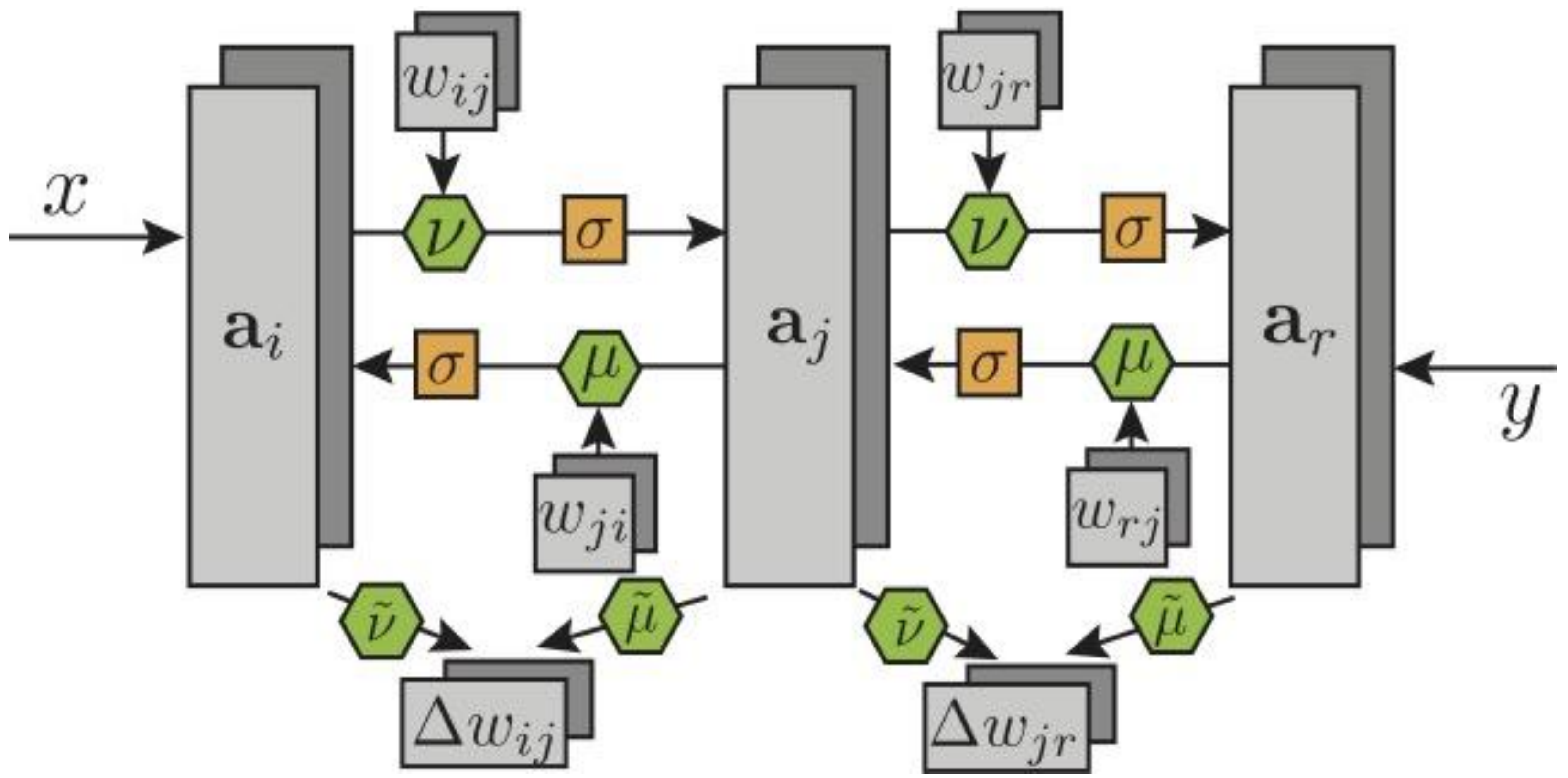
Forward pass: $a_j^c \leftarrow \sigma\left(f a_j^c + \eta \sum_{i,d} w_{ij}^c \nu^{cd} a_i^d\right)$

Backward pass: $a_i^c \leftarrow \sigma\left(f a_i^c + \eta \sum_{j,d} w_{ji}^c \mu^{cd} a_j^d\right)$

Weights update: $w_{ij}^c \leftarrow \tilde{f} w_{ij}^c + \tilde{\eta} \sum_{e,d} a_i^e \tilde{\nu}^{ec} \cdot \tilde{\mu}^{cd} a_j^d$

Bidirectional Learning Update Rules

Non-linear activations for forward *and* backward pass.



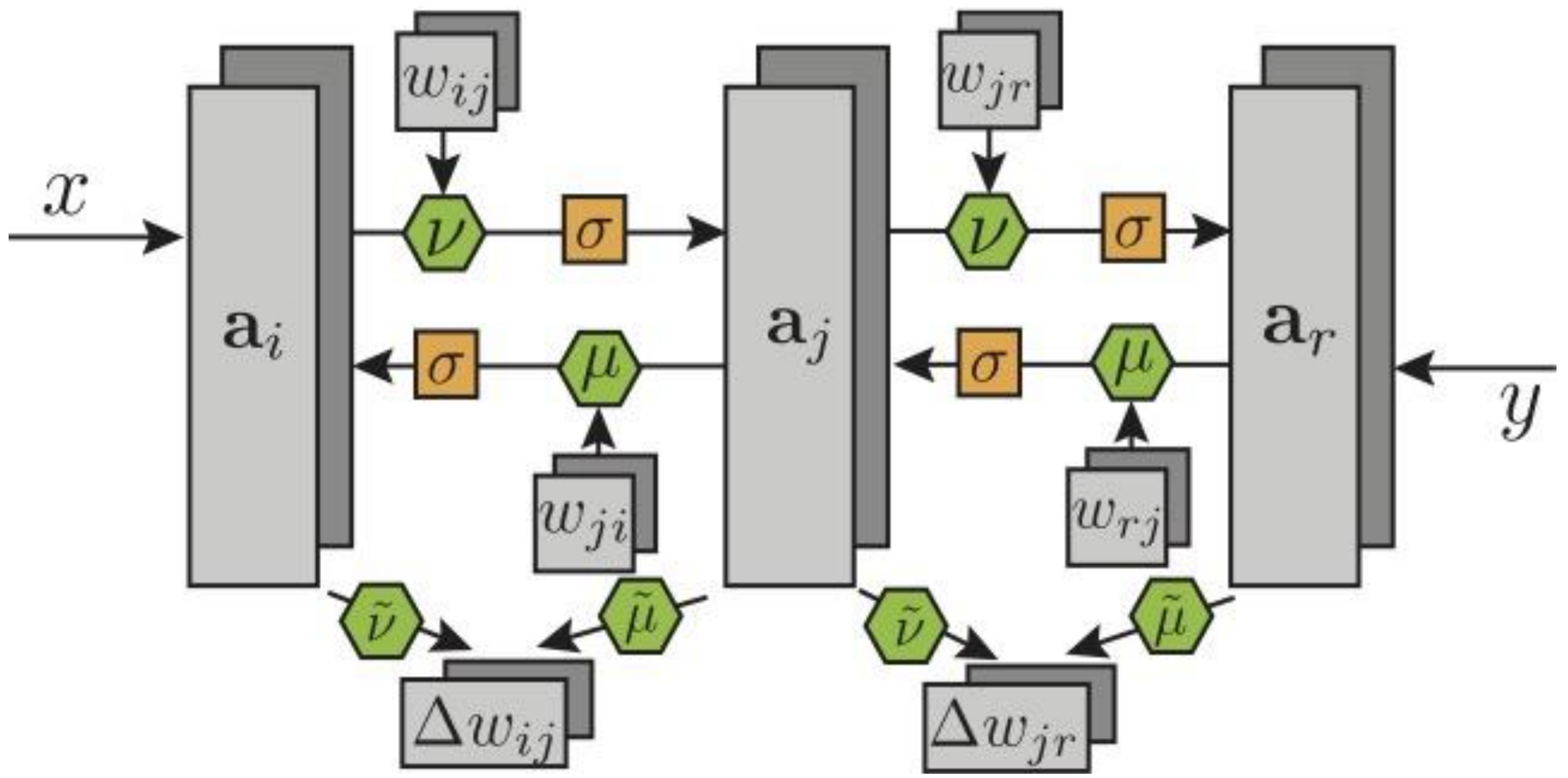
Forward pass: $a_j^c \leftarrow \sigma\left(f a_j^c + \eta \sum_{i,d} w_{ij}^c \nu^{cd} a_i^d\right)$

Backward pass: $a_i^c \leftarrow \sigma\left(f a_i^c + \eta \sum_{j,d} w_{ji}^c \mu^{cd} a_j^d\right)$

Weights update: $w_{ij}^c \leftarrow \tilde{f} w_{ij}^c + \tilde{\eta} \sum_{e,d} a_i^e \tilde{\nu}^{ec} \cdot \tilde{\mu}^{cd} a_j^d$

Bidirectional Learning Update Rules

Meta-learned `keep` and `update` parameters for neuron update.



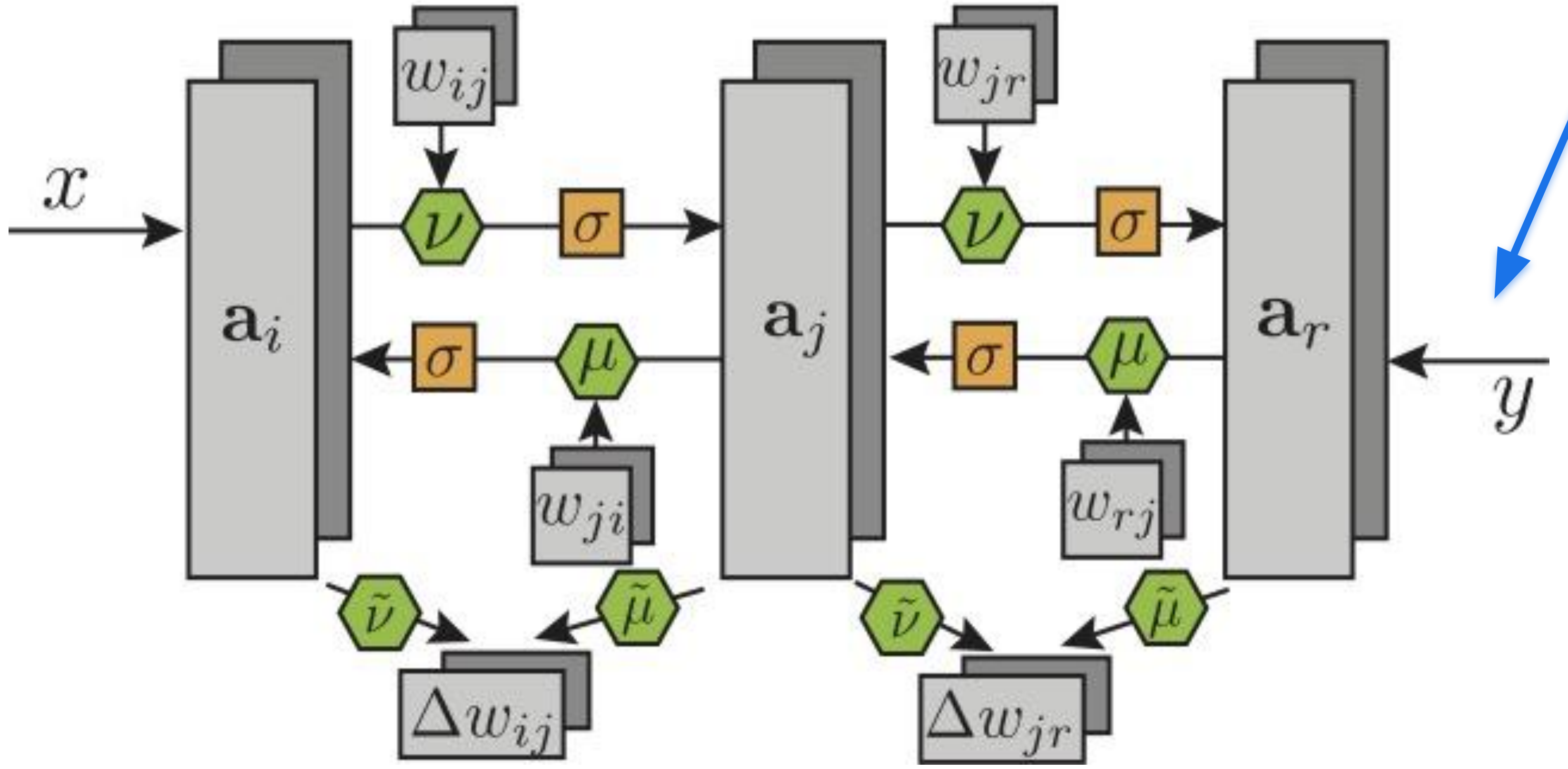
Forward pass: $$a_j^c \leftarrow \sigma\left(f a_j^c + \eta \sum_{i,d} w_{ij}^c \nu^{cd} a_i^d\right)$$

Backward pass: $$a_i^c \leftarrow \sigma\left(f a_i^c + \eta \sum_{j,d} w_{ji}^c \mu^{cd} a_j^d\right)$$

Weights update: $$w_{ij}^c \leftarrow \tilde{f} w_{ij}^c + \tilde{\eta} \sum_{e,d} a_i^e \tilde{\nu}^{ec} \cdot \tilde{\mu}^{cd} a_j^d$$

Bidirectional Learning Update Rules

No more loss function.
Feedback is passed directly
to one of the states.

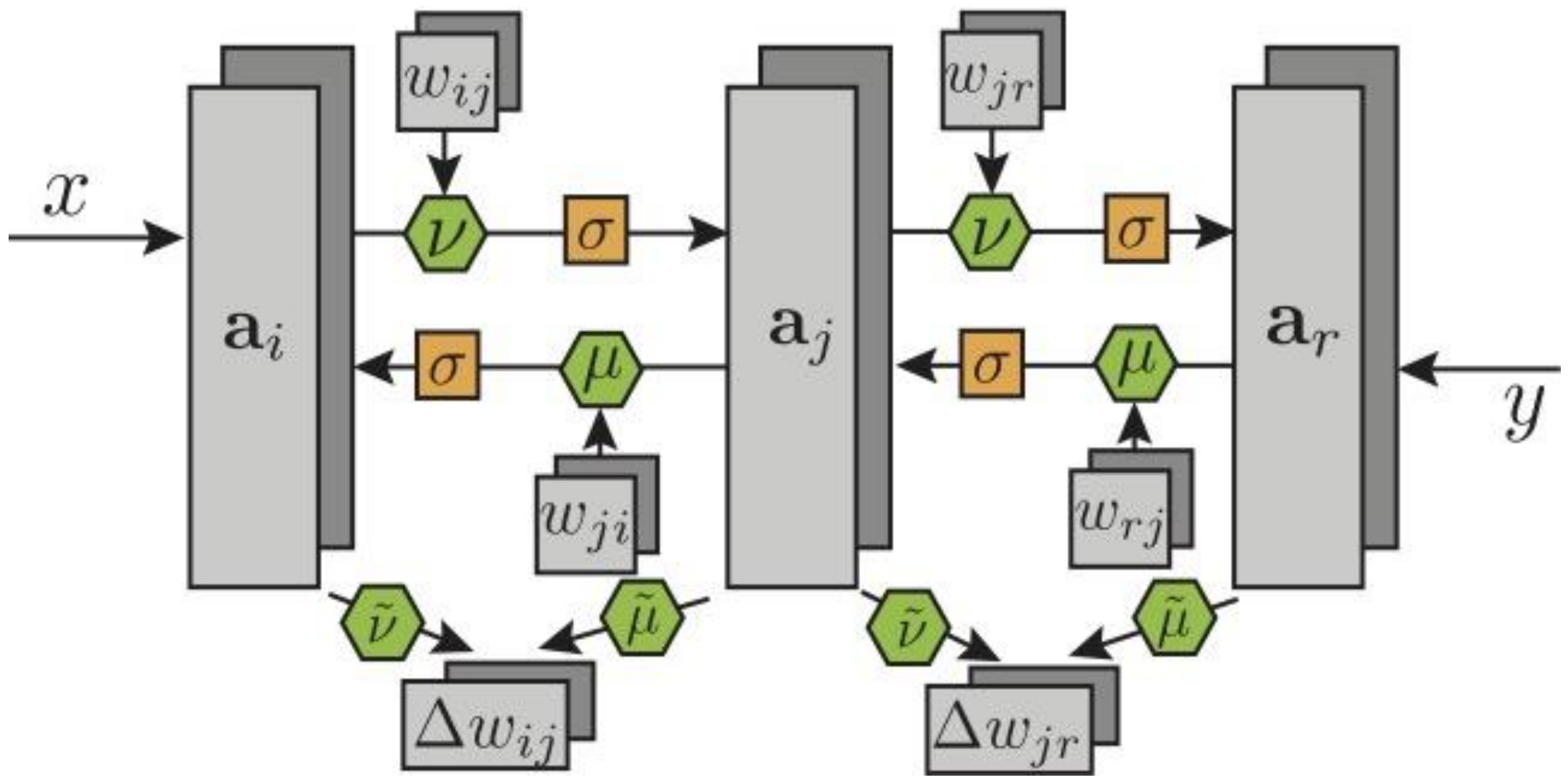


Forward pass: $a_j^c \leftarrow \sigma\left(f a_j^c + \eta \sum_{i,d} w_{ij}^c \nu^{cd} a_i^d\right)$

Backward pass: $a_i^c \leftarrow \sigma\left(f a_i^c + \eta \sum_{j,d} w_{ji}^c \mu^{cd} a_j^d\right)$

Weights update: $w_{ij}^c \leftarrow \tilde{f} w_{ij}^c + \tilde{\eta} \sum_{e,d} a_i^e \tilde{\nu}^{ec} \cdot \tilde{\mu}^{cd} a_j^d$

Bidirectional Learning Update Rules



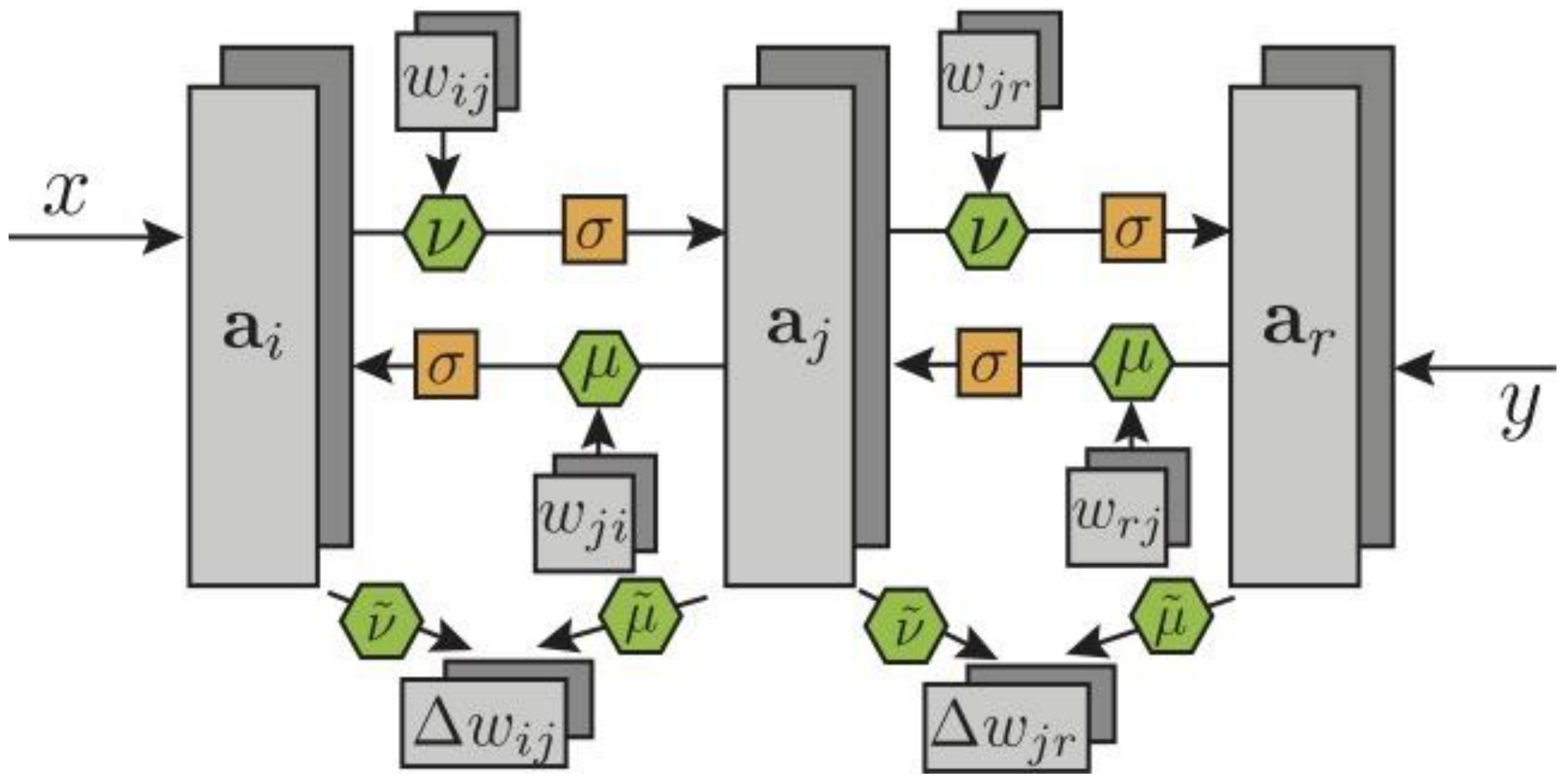
Forward pass: $a_j^c \leftarrow \sigma\left(f a_j^c + \eta \sum_{i,d} w_{ij}^c \nu^{cd} a_i^d\right)$

Backward pass: $a_i^c \leftarrow \sigma\left(f a_i^c + \eta \sum_{j,d} w_{ji}^c \mu^{cd} a_j^d\right)$

Weights update: $w_{ij}^c \leftarrow \tilde{f} w_{ij}^c + \tilde{\eta} \sum_{e,d} a_i^e \tilde{\nu}^{ec} \cdot \tilde{\mu}^{cd} a_j^d$

Multistate synapses.

Bidirectional Learning Update Rules



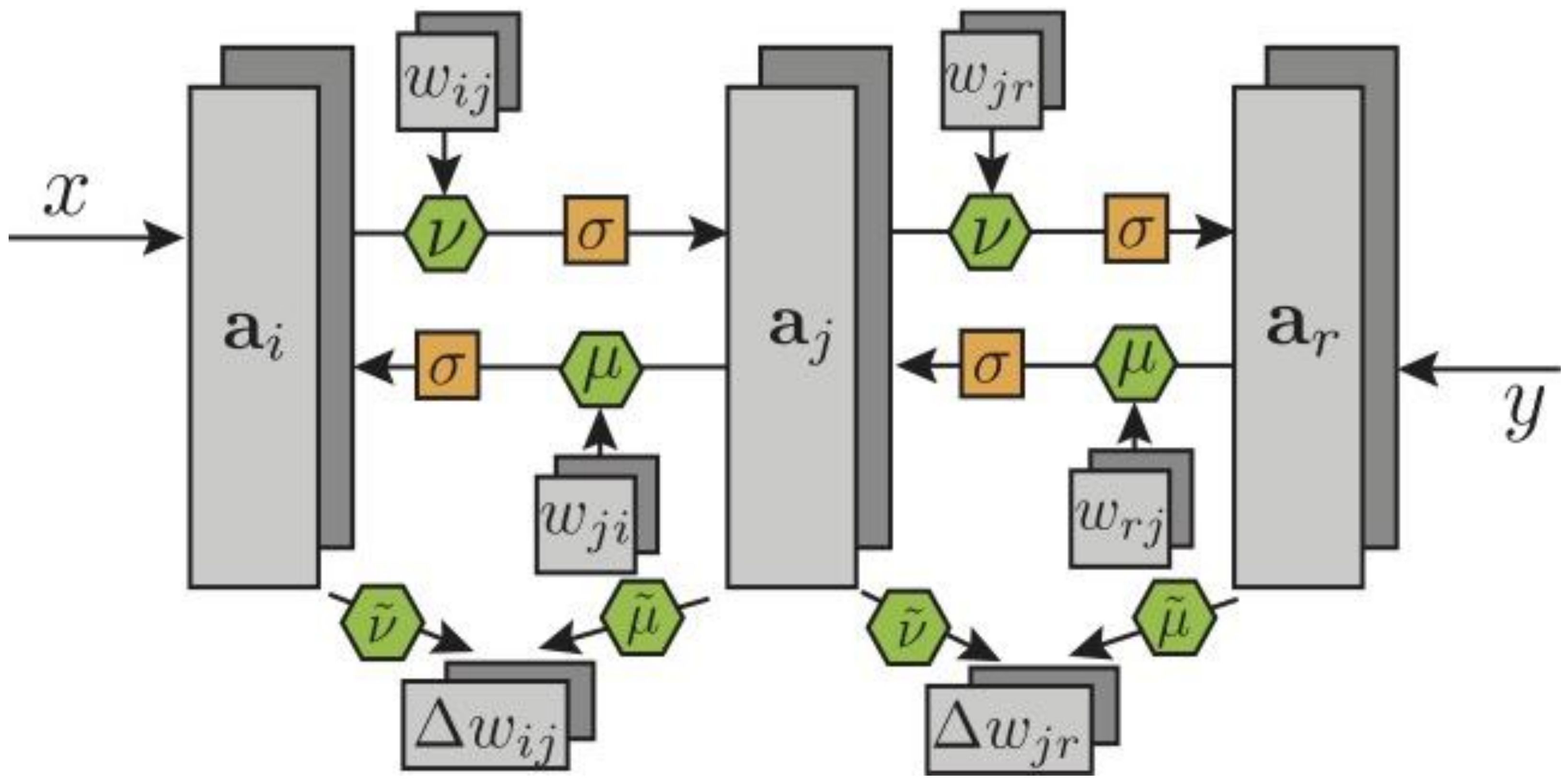
Forward pass: $$a_j^c \leftarrow \sigma\Big(f a_j^c + \eta \sum_{i,d} w_{ij}^c \nu^{cd} a_i^d\Big)$$

Backward pass: $$a_i^c \leftarrow \sigma\Big(f a_i^c + \eta \sum_{j,d} w_{ji}^c \mu^{cd} a_j^d\Big)$$

Weights update: $$w_{ij}^c \leftarrow \tilde{f} w_{ij}^c + \tilde{\eta} \sum_{e,d} a_i^e \tilde{\nu}^{ec} \cdot \tilde{\mu}^{cd} a_j^d$$

Meta-learned *transform* matrices for interaction between the neuron and synapse states.

Bidirectional Learning Update Rules



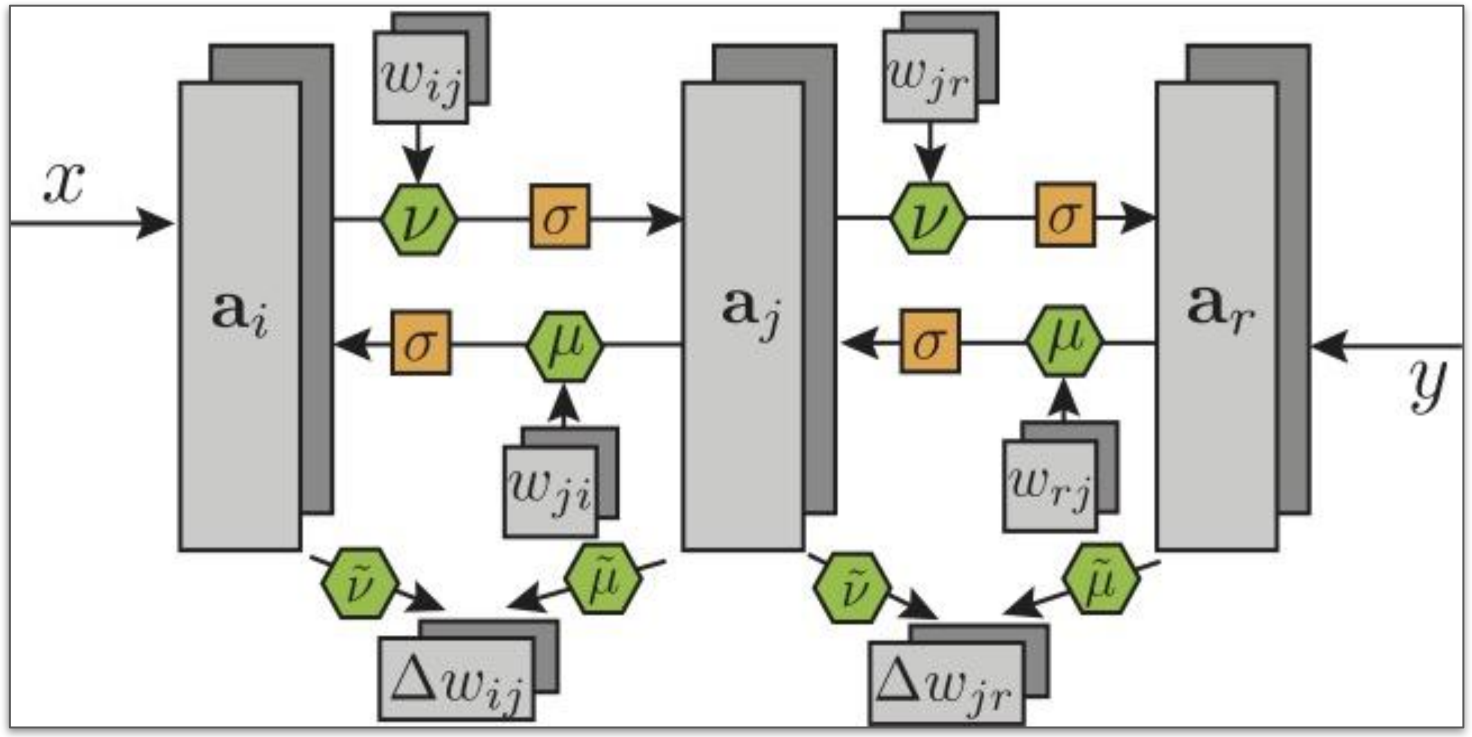
Forward pass: $a_j^c \leftarrow \sigma\Big(f a_j^c + \eta \sum_{i,d} w_{ij}^c \nu^{cd} a_i^d\Big)$

Backward pass: $a_i^c \leftarrow \sigma\Big(f a_i^c + \eta \sum_{j,d} w_{ji}^c \mu^{cd} a_j^d\Big)$

Weights update: $w_{ij}^c \leftarrow \tilde{f} w_{ij}^c + \tilde{\eta} \sum_{e,d} a_i^e \tilde{\nu}^{ec} \cdot \tilde{\mu}^{cd} a_j^d$

Meta-learned `keep` and `update` parameters for synapse update.

Bidirectional Learning Update Rules



Forward pass: $a_j^c \leftarrow \sigma\left(f a_j^c + \eta \sum_{i,d} w_{ij}^c \nu^{cd} a_i^d\right)$

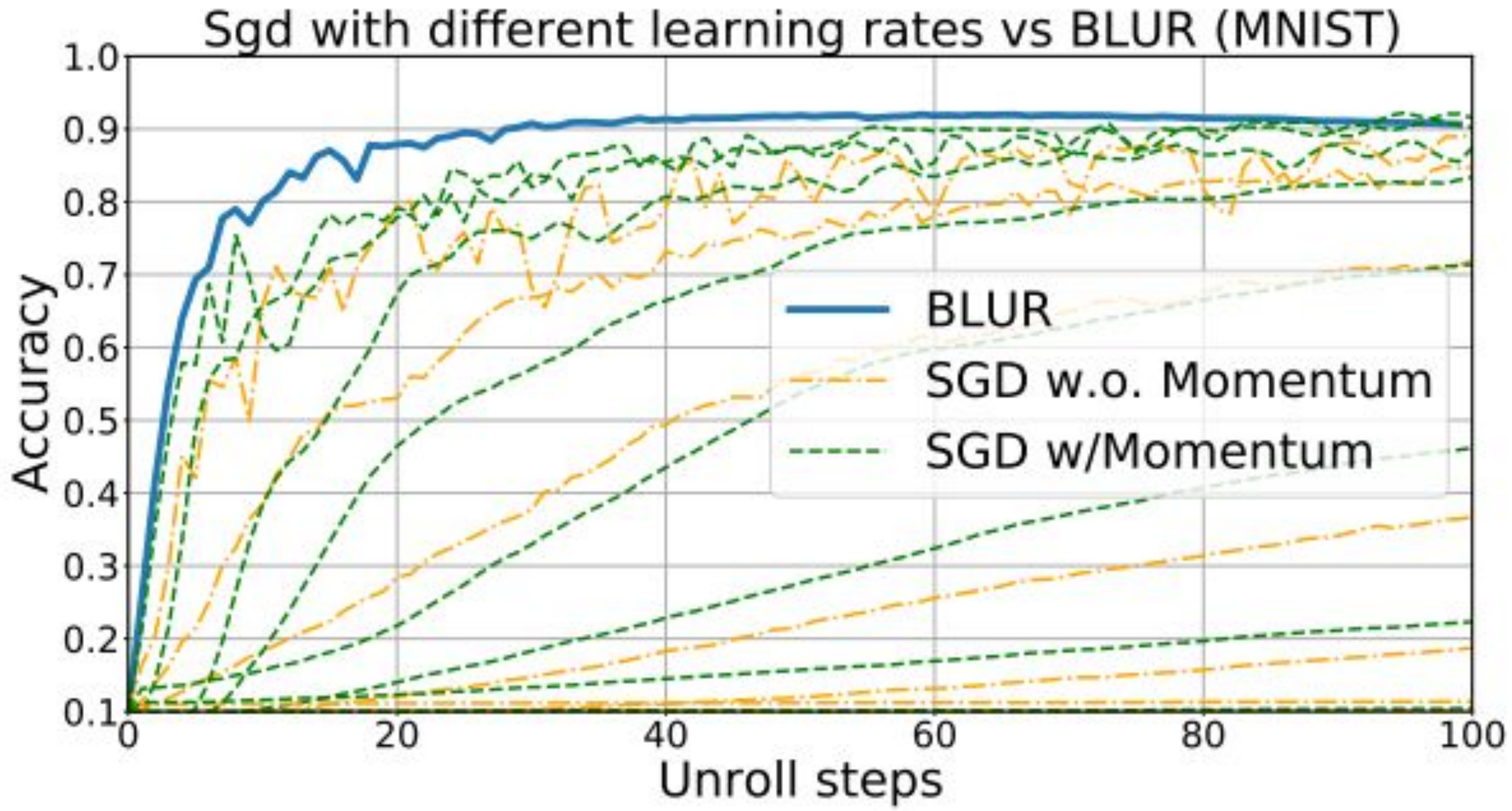
Backward pass: $a_i^c \leftarrow \sigma\left(f a_i^c + \eta \sum_{j,d} w_{ji}^c \mu^{cd} a_j^d\right)$

Weights update: $w_{ij}^c \leftarrow \tilde{f} w_{ij}^c + \tilde{\eta} \sum_{e,d} a_i^e \tilde{\nu}^{ec} \cdot \tilde{\mu}^{cd} a_j^d$

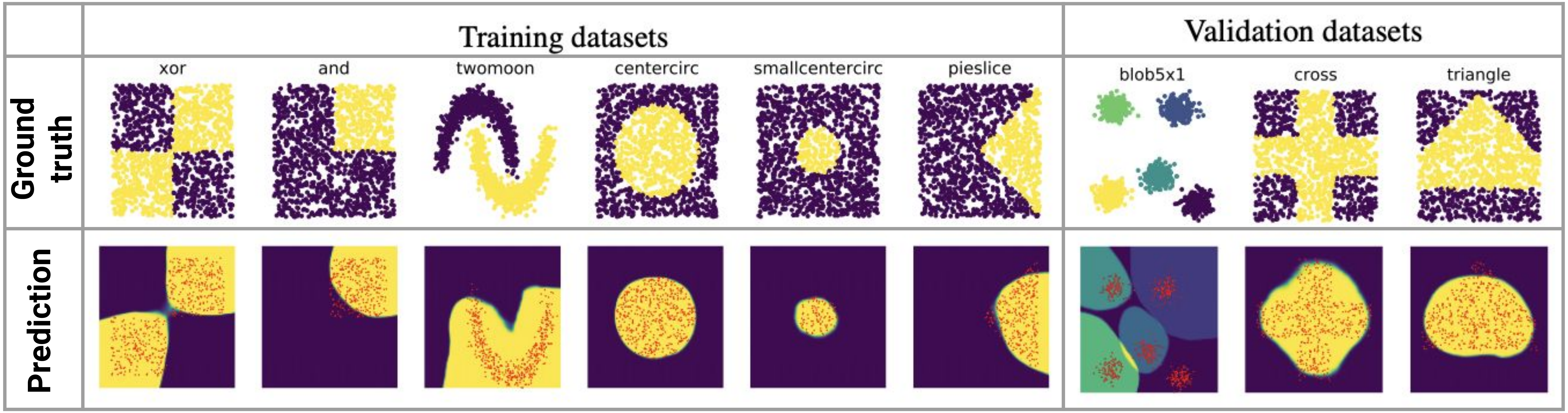
States	- `k` neuron states. - `k` synapse states (possibly asymmetric).
Feedback	Passed directly to the final layer.
Forward pass	- All states are updated via learnable transform matrix. - Same activation functions for each state. - `Keep` and `update` are learned parameters.
Backward pass	- All states are updated via learnable transform matrix. - Same activation functions for each state. - `Keep` and `update` are learned parameters.
Synapse update	- All states from presynaptic and postsynaptic are mixed together. - `Keep` and `update` are learned parameters.

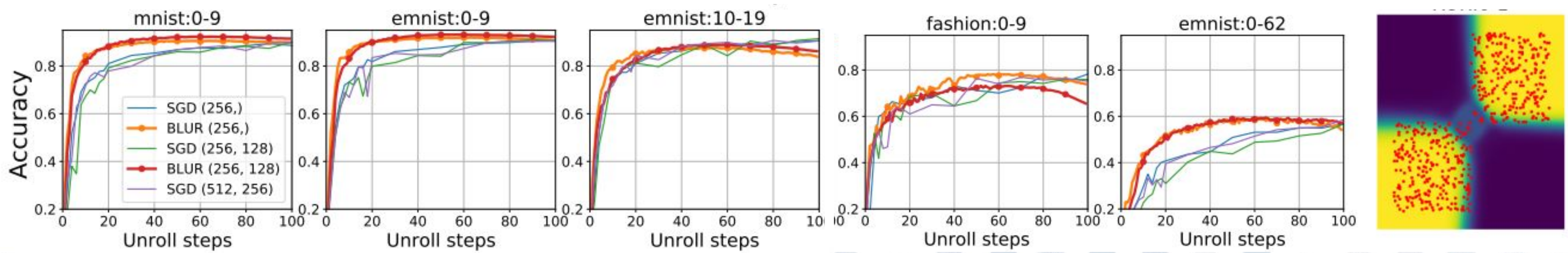
Meta-learning BLUR

- **Prediction:** use first state of the last layer of a given unroll.
- **Train:** compute loss at a given unroll number.
- **Test:** run validation batch through learned synapses and meta-parameters.
- **Optimize:** using SGD or CMA-ES.



Meta-generalization of BLUR





Conclusions

- BLUR defines a **meta-learned synapse update rules** that has very mild assumptions on the inner-loop:
    - **no loss functions**,
    - **no explicit gradients**.
- Trained with SGD or CMA for a **fix number of unrolls**.
- BLUR generalizes SGD:
    - **multistate neurons**,
    - **asymmetric**, **multistate** synapses,
    - **backward activations and normalization**.
- BLUR is **meta-generalizable**:
    - unseen datasets (MNIST → Fashion),
    - novel input data sizes (10x10 → 28x28),
    - novel architectures (deeper → shallower).