

A Modern Self-Referential Weight Matrix That Learns to Modify Itself

Kazuki Irie



Imanol Schlag



Róbert Csordás



Jürgen Schmidhuber



IDSIA



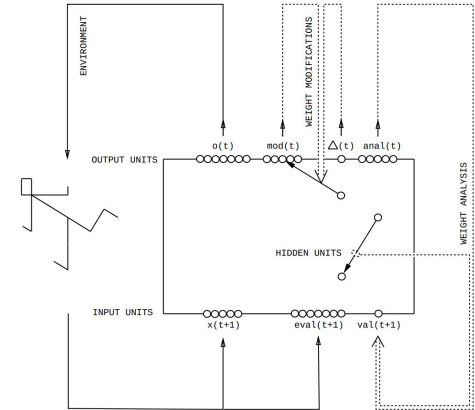
جامعة الملك عبد الله
للعلوم والتقنية
King Abdullah University of
Science and Technology

Can we build a *neural net* that learns to modify & improve *itself*? (*recursively*)

- Idea since Schmidhuber 1992 (30 years ago!)
‘Steps towards “self-referential” learning.’

“Original” **Self-Referential Weight Matrix**

- *RNN that learns its own weight change algorithm*
- *‘Address based’ self-modification*



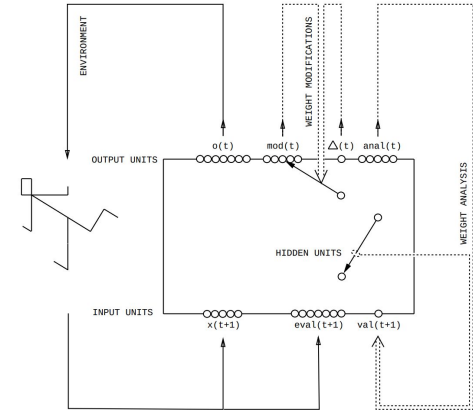
Can we build a *neural net* that learns to modify & improve *itself*? (*recursively*)

- Idea since Schmidhuber 1992 (30 years ago!)
‘Steps towards “self-referential” learning.’

“Original” Self-Referential Weight Matrix

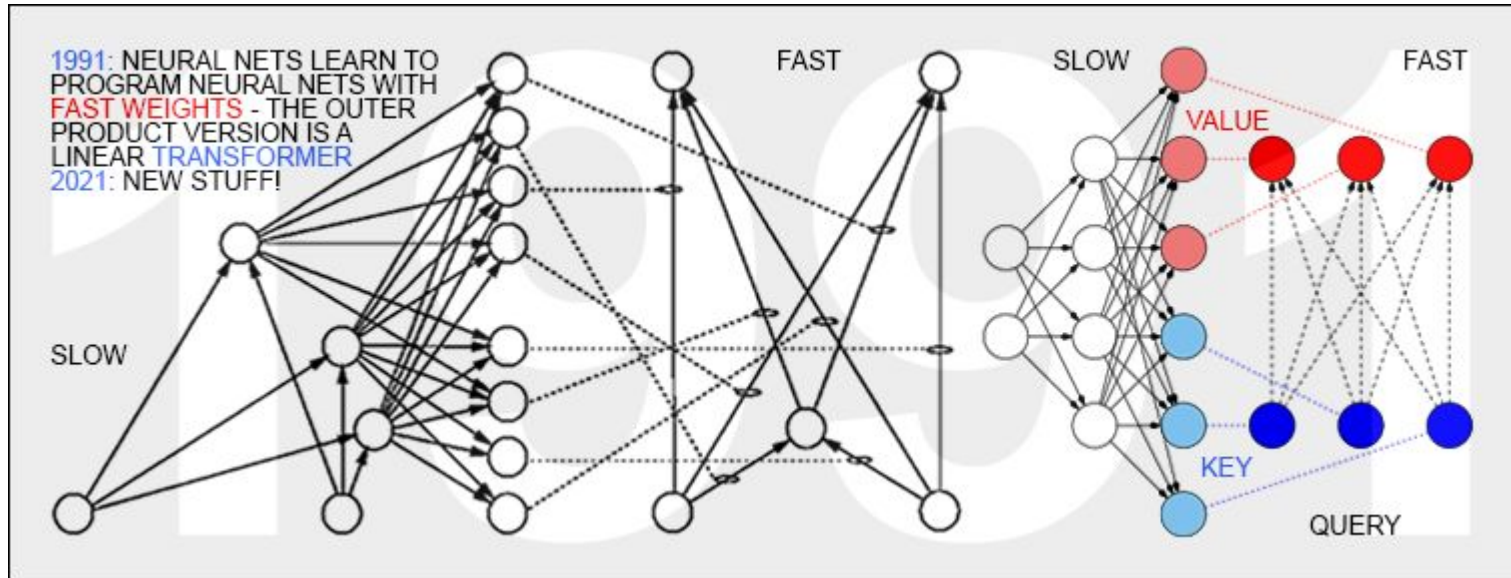
- *RNN that learns its own weight change algorithm*
 - *‘Address based’ self-modification*
-
- *What’s the most natural/scalable way to do this today?*

- Weight change mechanism used in recent works on **Fast Weight Programmers**
- **Simple/practical** form of **recursive self-modification**



Fast Weight Programmers ('Fast Weight Controllers'; Schmidhuber 1991)

A **slow** net with **slow weights** (trained by gradient descent) generates **changes of fast weights** of a **fast** net



Outer-product based version is a **linear Transformer** (ICML 2020/2021)

Example of Fast Weight Programmers:

DeltaNet (Schlag et al. [ICML 2021](#))

Slow Net

$$\mathbf{k}_t, \mathbf{v}_t, \mathbf{q}_t, \beta_t = \mathbf{W}_{\text{slow}} \mathbf{x}_t$$

$$\bar{\mathbf{v}}_t = \mathbf{W}_{t-1} \phi(\mathbf{k}_t)$$

$$\mathbf{W}_t = \mathbf{W}_{t-1} + \sigma(\beta_t) (\mathbf{v}_t - \bar{\mathbf{v}}_t) \otimes \phi(\mathbf{k}_t)$$

$$\mathbf{y}_t = \mathbf{W}_t \phi(\mathbf{q}_t)$$

Follow-up ([NeurIPS 2021](#)): "*Going Beyond Linear Transformers with Recurrent Fast Weight Programmers*" (e.g. [Delta-Delta-Net](#))

Example of Fast Weight Programmers:

DeltaNet (Schlag et al. [ICML 2021](#))

Slow Net

$$\mathbf{k}_t, \mathbf{v}_t, \mathbf{q}_t, \beta_t = \mathbf{W}_{\text{slow}} \mathbf{x}_t \quad \text{Update Rule / Programming Instruction}$$

$$\bar{\mathbf{v}}_t = \mathbf{W}_{t-1} \phi(\mathbf{k}_t)$$
$$\mathbf{W}_t = \mathbf{W}_{t-1} + \sigma(\beta_t) (\mathbf{v}_t - \bar{\mathbf{v}}_t) \otimes \phi(\mathbf{k}_t)$$

$$\mathbf{y}_t = \mathbf{W}_t \phi(\mathbf{q}_t)$$

Outer product

Self generated learning rate *Currently stored value to be replaced*

Follow-up ([NeurIPS 2021](#)): "Going Beyond Linear Transformers with Recurrent Fast Weight Programmers" (e.g. [Delta-Delta-Net](#))

Example of Fast Weight Programmers:

DeltaNet (Schlag et al. [ICML 2021](#))

Slow Net

$$\mathbf{k}_t, \mathbf{v}_t, \mathbf{q}_t, \beta_t = \mathbf{W}_{\text{slow}} \mathbf{x}_t \quad \text{Update Rule / Programming Instruction}$$

$$\bar{\mathbf{v}}_t = \mathbf{W}_{t-1} \phi(\mathbf{k}_t)$$
$$\mathbf{W}_t = \mathbf{W}_{t-1} + \sigma(\beta_t)(\mathbf{v}_t - \bar{\mathbf{v}}_t) \otimes \phi(\mathbf{k}_t)$$

$$\mathbf{y}_t = \mathbf{W}_t \phi(\mathbf{q}_t) \quad \text{Fast Net}$$

Outer product

Self generated learning rate *Currently stored value to be replaced*

Follow-up ([NeurIPS 2021](#)): "Going Beyond Linear Transformers with Recurrent Fast Weight Programmers" (e.g. [Delta-Delta-Net](#))

A “Modern” Self-Referential Weight Matrix

Input $x_t \in \mathbb{R}^{d_{in}}$ Output $y_t \in \mathbb{R}^{d_{out}}$

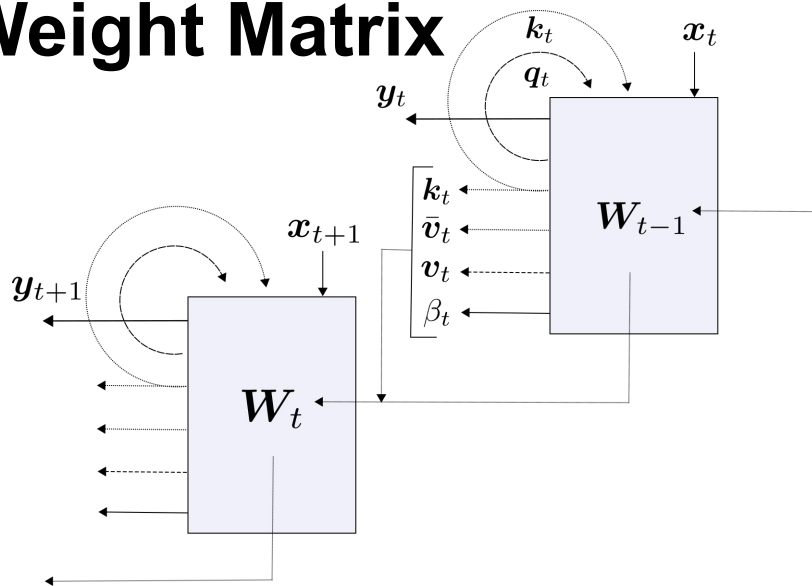
Weight Matrix $W_{t-1} \in \mathbb{R}^{(d_{out}+2*d_{in}+1) \times d_{in}}$

$$y_t, k_t, q_t, \beta_t = W_{t-1} \phi(x_t)$$

$$\bar{v}_t = W_{t-1} \phi(k_t)$$

$$v_t = W_{t-1} \phi(q_t)$$

$$W_t = W_{t-1} + \sigma(\beta_t)(v_t - \bar{v}_t) \otimes \phi(k_t)$$



- Self-modifications based on **key/value associations**
- Learns to train itself (at runtime) using self-generated training patterns/learning rates (**recursively self-modifying programs**)
- *Practical note: use **multiple heads**, i.e., each layer contains multiple SRWMs*

A “Modern” Self-Referential Weight Matrix

Input $x_t \in \mathbb{R}^{d_{in}}$ Output $y_t \in \mathbb{R}^{d_{out}}$

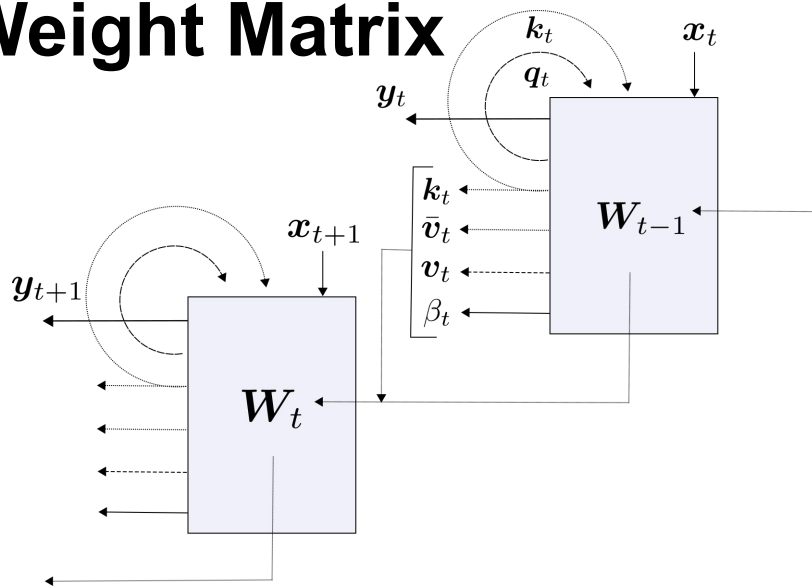
Weight Matrix $W_{t-1} \in \mathbb{R}^{(d_{out}+2*d_{in}+1) \times d_{in}}$

$$y_t, k_t, q_t, \beta_t = W_{t-1} \phi(x_t)$$

$$\bar{v}_t = W_{t-1} \phi(k_t)$$

$$v_t = W_{t-1} \phi(q_t)$$

$$W_t = W_{t-1} + \sigma(\beta_t)(v_t - \bar{v}_t) \otimes \phi(k_t)$$



- Self-modifications based on **key/value associations**
- Learns to train itself (at runtime) using self-generated training patterns/learning rates (**recursively self-modifying programs**)
- *Practical note: use **multiple heads**, i.e., each layer contains multiple SRWMs*

A “Modern” Self-Referential Weight Matrix

Input $x_t \in \mathbb{R}^{d_{in}}$ Output $y_t \in \mathbb{R}^{d_{out}}$

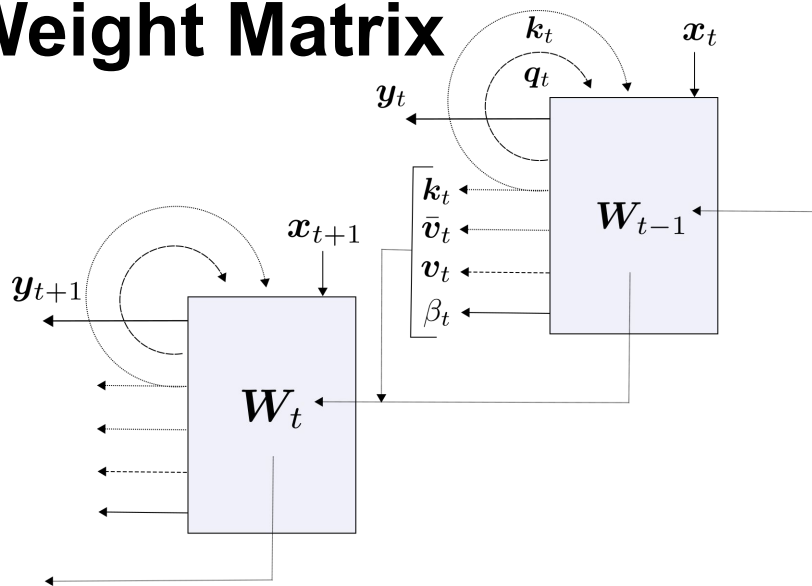
Weight Matrix $W_{t-1} \in \mathbb{R}^{(d_{out}+2*d_{in}+1) \times d_{in}}$

$$y_t, k_t, q_t, \beta_t = W_{t-1} \phi(x_t)$$

$$\bar{v}_t = W_{t-1} \phi(k_t)$$

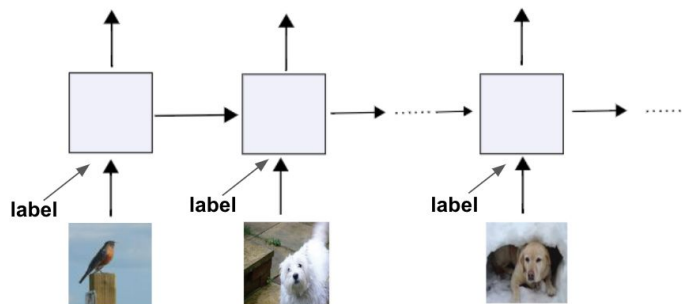
$$v_t = W_{t-1} \phi(q_t)$$

$$W_t = W_{t-1} + \sigma(\beta_t)(v_t - \bar{v}_t) \otimes \phi(k_t)$$

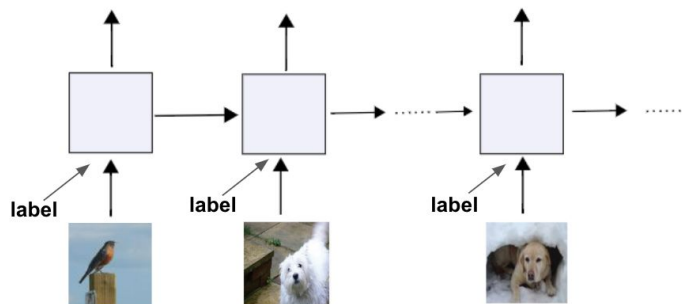


- Self-modifications based on **key/value associations**
- Learns to train itself (at runtime) using self-generated training patterns/learning rates (**recursively self-modifying programs**)
- *Practical note: use **multiple heads**, i.e., each layer contains multiple SRWMs*

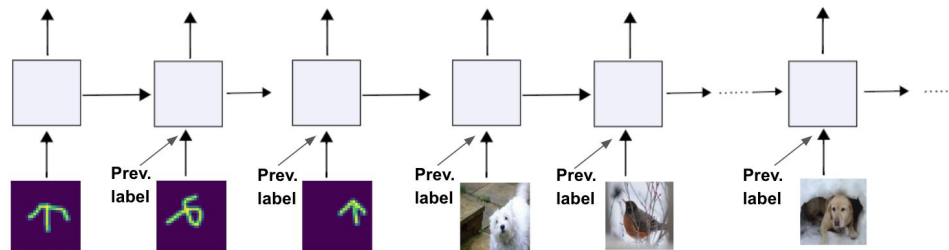
Sanity check on the **standard** few-shot image classification (**Mini-ImageNet**)



Sanity check on the **standard** few-shot image classification (**Mini-ImageNet**)

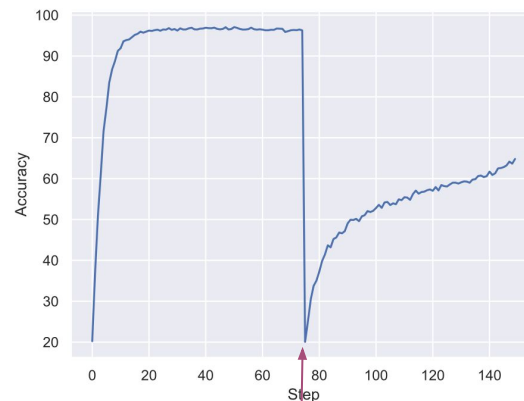


Sequential **Multi-task** Adaptation (**Mini-ImageNet + Omniglot**)



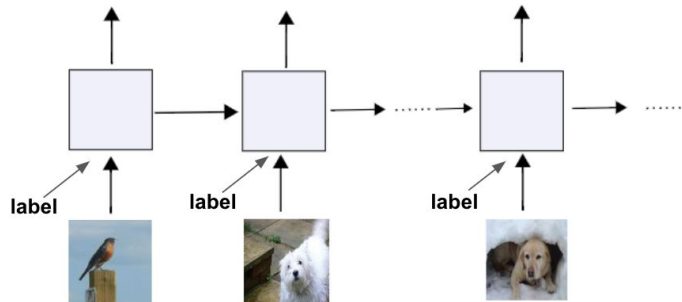
“Omniglot” dataset

“Mini-ImageNet” dataset

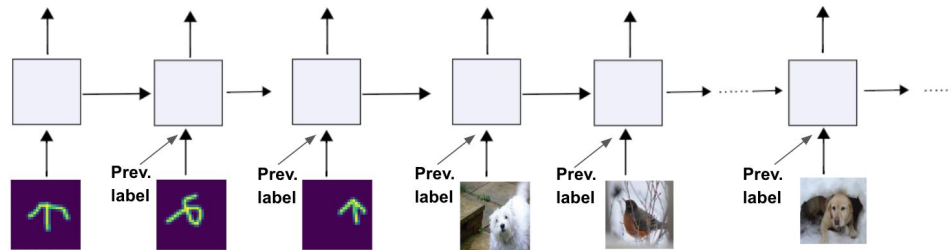


Dataset changes

Sanity check on the **standard** few-shot image classification (**Mini-ImageNet**)



Sequential **Multi-task** Adaptation (**Mini-ImageNet + Omniglot**)



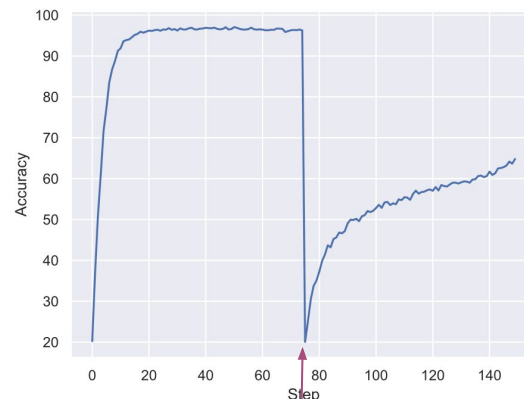
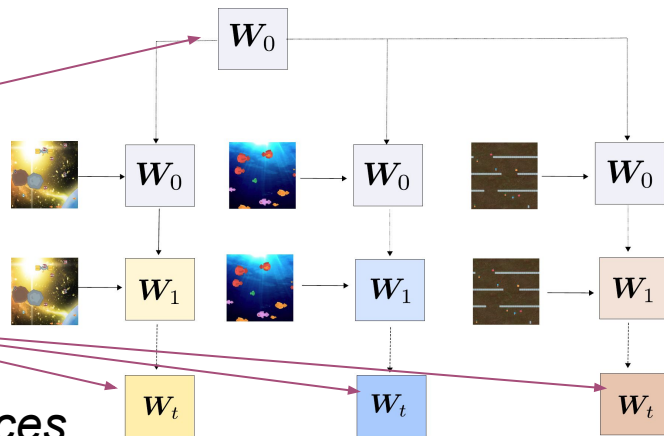
“Omniglot” dataset

“Mini-ImageNet” dataset

Multi-task Reinforcement Learning (**ProcGen**)

Common initial weight matrix

Self-modified task/episode specific weight matrices



Dataset changes

Implementation

- Simple equations, but require custom **memory efficient** backward implementation
- All our code is **public**



Code: [IDSIA/modern-srwm](https://github.com/IDSIA/modern-srwm)

Thank you for your attention.

Hope to see you at the poster!

Please also check (**tomorrow/Deep Learning**):
“The Dual Form of Neural Networks Revisited...”