MesaNet: Sequence Modelling by Locally Optimal Test Time Training

Johannes von Oswald & Nino Scherrer

Google Paradigms of Intelligence

ASAP Seminar
June 24th, 2025

Joint Work with an Amazing Team!!

Max Schlegel

Joao Sacramento

Songlin Yang

Seijin Kobayashi

Luca Versari

&

Alexander Meulemans
Alexander Mordvintsev
Angelika Steger
Andrey Zhmoginov
Charlotte Frenkel
Eyvind Niklasson
Guillaume Lajoie
Kaitlin Maile
Mark Sandler
Max Vladymyrov
Nolan Miller
Oliver Sieberling
Seijin Kobayashi
Songlin Yang
Razvan Pascanu
Ettore Randazzo
Yanick Schimpf
Nicolas Zucchet
Rif A. Saurous
Blaise Aguera y Arcas

Talk Outline

Part 1: The Origin of the MesaLayer

1. Analysis of transformers trained on synthetic sequence data
2. A toy model for in-context few-shot learning

Google 2024-10-16

Uncovering mesa-optimization algorithms in Transformers

Johannes von Oswald[a,b,*], Maximilian Schlegel[a,b,*], Alexander Meulemans[a,b], Seijin Kobayashi[a,b], Eyvind Niklasson[a], Nicolas Zucchet[b], Nino Scherrer[a], Nolan Miller[d], Mark Sandler[d], Blaise Agüera y Arcas[a], Max Vladymyrov[d], Razvan Pascanu[e] and João Sacramento[a,b,*]

[a]Google, Paradigms of Intelligence Team, [b]ETH Zürich, [d]Google Research, [e]Google DeepMind, [*]Contributed equally to this work.

https://arxiv.org/abs/2309.05858

Part 2: Scaling MesaNets

3. Efficient sequence modeling by stacking greedy local learners
4. Synthetic + 1B parameter language modeling experiments

MesaNet: Sequence Modeling by Locally Optimal Test-Time Training

Johannes von Oswald*, Nino Scherrer*,
Seijin Kobayashi, Luca Versari, Songlin Yang[1], Maximilian Schlegel, Kaitlin Maile,
Yanick Schimpf, Oliver Sieberling[2], Alexander Meulemans, Rif A. Saurous, Guillaume Lajoie,
Charlotte Frenkel, Razvan Pascanu[3], Blaise Agüera y Arcas , and João Sacramento

https://arxiv.org/abs/2506.05233

Talk Outline

Part 1: The Origin of the MesaLayer

1. Analysis of transformers trained on synthetic sequence data
2. A toy model for in-context few-shot learning

Google 2024-10-16

Uncovering mesa-optimization algorithms in Transformers

Johannes von Oswald[a,b,*], Maximilian Schlegel[a,b,*], Alexander Meulemans[a,b], Seijin Kobayashi[a,b], Eyvind Niklasson[a], Nicolas Zucchet[b], Nino Scherrer[a], Nolan Miller[d], Mark Sandler[d], Blaise Agüera y Arcas[a], Max Vladymyrov[d], Razvan Pascanu[e] and João Sacramento[a,b,*]

[a]Google, Paradigms of Intelligence Team, [b]ETH Zürich, [d]Google Research, [e]Google DeepMind, [*]Contributed equally to this work.

https://arxiv.org/abs/2309.05858

Part 2: Scaling MesaNets

3. Efficient sequence modeling by stacking greedy local learners
4. Synthetic + 1B parameter language modeling experiments

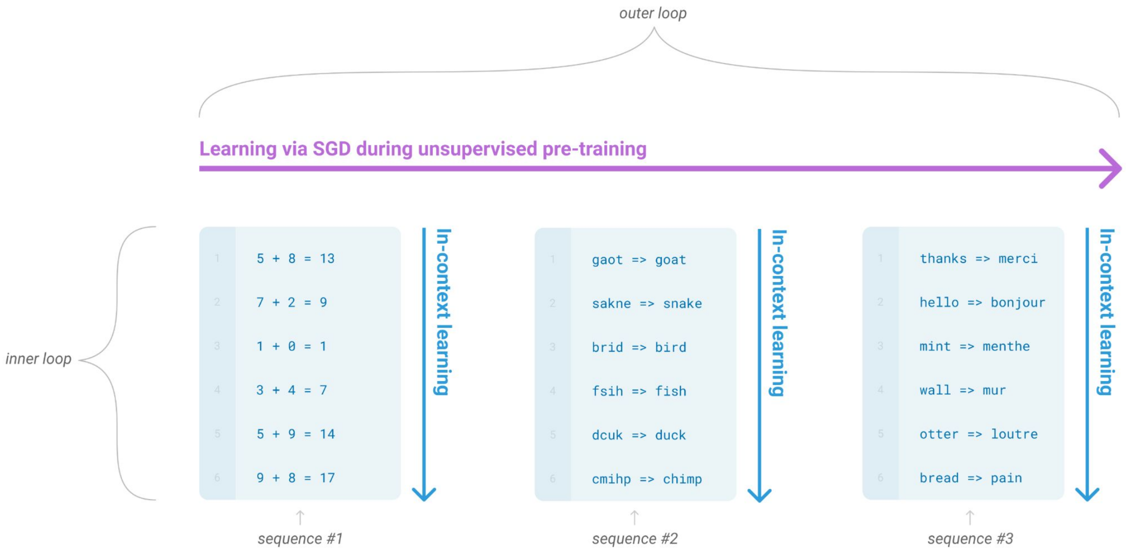
MesaNet: Sequence Modeling by Locally Optimal Test-Time Training

Johannes von Oswald[*], Nino Scherrer[*],
Seijin Kobayashi, Luca Versari, Songlin Yang[1], Maximilian Schlegel, Kaitlin Maile, Yanick Schimpf, Oliver Sieberling[2], Alexander Meulemans, Rif A. Saurous, Guillaume Lajoie, Charlotte Frenkel, Razvan Pascanu[3], Blaise Agüera y Arcas , and João Sacramento

https://arxiv.org/abs/2506.05233

MesaNet Origin | LLMs are few-shot learners

In-Context Learning (ICL)
⟶ no parameter updates



- **Understanding:** Get insights on in-context learning.
- **Building:** Use these insights to design new sequence models

Reference: Brown et al. (2020)

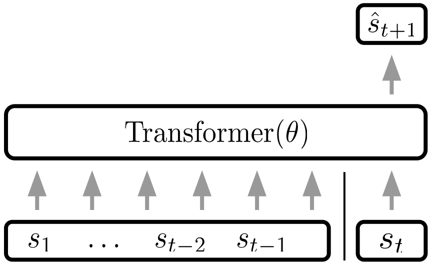
MesaNet Origin | An autoregressive toy model

- To generate a sequence, sample an initial state h_1 and run

$$h_t = Af(h_{t-1}) + \text{noise}$$
$$s_t = Ch_t + \text{noise}$$

$f = \text{MLP}$ or $f = \text{Id}$

- “The world is large”: every sequence comes from new A, C matrices

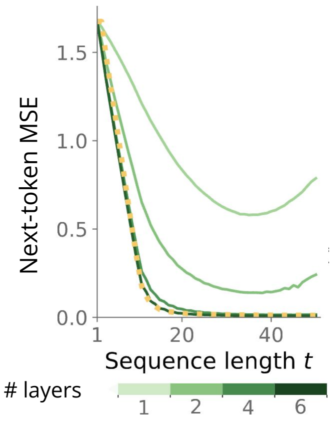


$$\min_{\theta} \mathbb{E}_{s \sim p(s)} \left[\frac{1}{2} \sum_{t=1}^{T-1} \| s_{t+1} - \hat{s}_{t+1}(s_{1:t}, \theta) \|^2 \right]$$

→ minimize next-input prediction error online by stochastic gradient descent

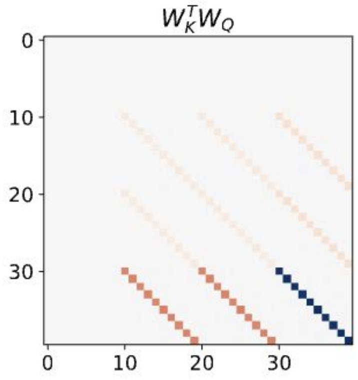
Reference: Vaswani et al. (2017)

MesaNet Origin | First inspection of the toy model



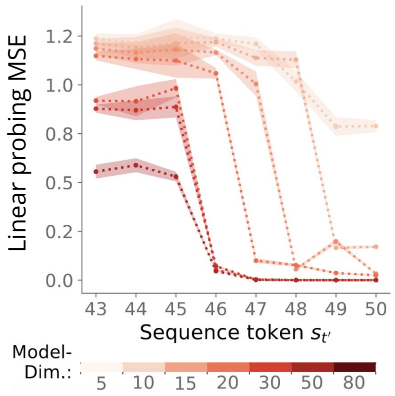
Predictions improve with sequence length and # layers

In-context learning according to Kaplan et al. (2020)



Some weights are sparse and highly structured

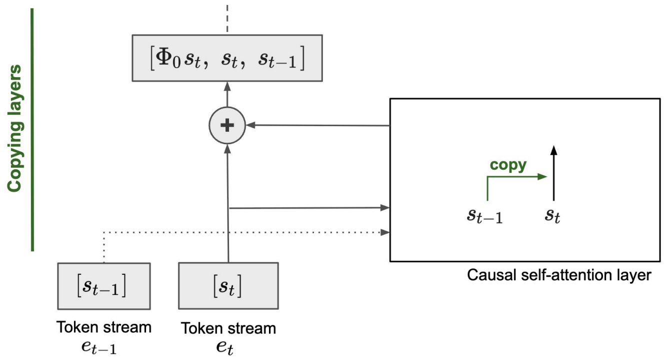
Suggestive of an algorithmic solution



Past inputs can be read out from the internal representation of current input

Robust phenomenon analyzed by Olsson et al. (2020) in natural language tasks
Co-occurs with sharp increase in in-context learning performance

MesaNet Origin | Reverse engineering trained transformers

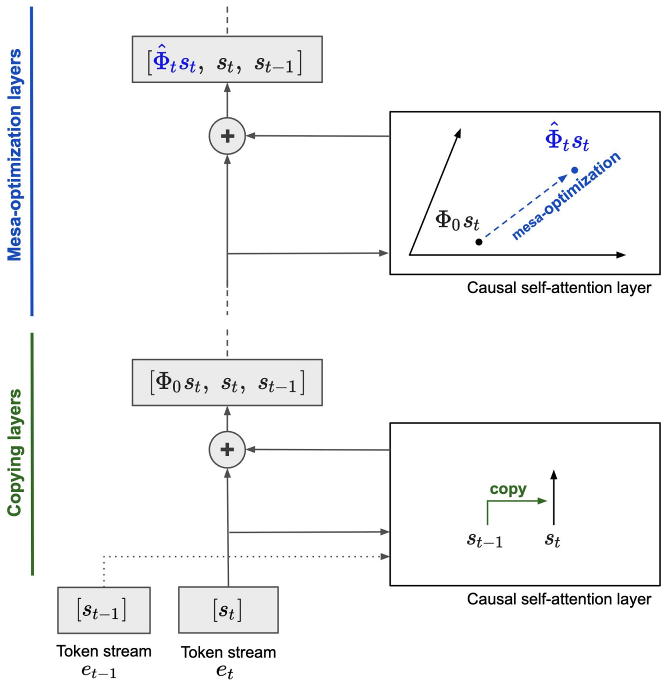


First layers:

- Apply nonlinear transform through MLP
- Bind *k* inputs into aggregate representation (*k* depends on degree of partial observability)

$$z_t^k = \begin{bmatrix} s_{t-k+1} \\ \vdots \\ s_t \end{bmatrix}$$

MesaNet Origin | Reverse engineering trained transformers



Subsequent attention layers:
Learn linear model Φ

$$\min_{\Phi} \sum_{t=1}^{T-1} \|z_{t+1}^k - \Phi z_t^k\|^2$$

using an efficient gradient-based **mesa-optimizer**

cf. Hubinger et al. (2021)

First layers:

- Apply nonlinear transform through MLP
- Bind *k* inputs into aggregate representation (*k* depends on degree of partial observability)

$$z_t^k = \begin{bmatrix} s_{t-k+1} \\ \vdots \\ s_t \end{bmatrix}$$

MesaNet Origin | Theoretical attention-based optimizer

We construct a multi-attention layer algorithm for minimizing the in-context loss:

$$L_t(\Phi) = \frac{1}{2} \sum_{t'=1}^{t-1} \| z_{t'+1} - \Phi \underbrace{H_t}_{\text{preconditioner}} z_{t'} \|^2 + \frac{\lambda}{2} \underbrace{\|\Phi\|_{\mathrm{F}}^2}_{\text{regularizer}}$$

- Attention can precondition inputs, which speeds up optimization. With sufficiently many attention layers the optimal preconditioning is approached:

$$H_t^* z_t = \left(Z_{t-1} Z_{t-1}^\top + \lambda I\right)^{-1} z_t$$

- Attention can take a gradient step $-\nabla L_t(\Phi)$ on the loss. A single step suffices for optimally preconditioned inputs.

MesaNet Origin | Theoretical attention-based optimizer

We construct a multi-attention layer algorithm for minimizing the in-context loss:

$$L_t(\Phi) = \frac{1}{2} \sum_{t'=1}^{t-1} \| z_{t'+1} - \Phi \underbrace{H_t}_{\text{preconditioner}} z_{t'} \|^2 + \frac{\lambda}{2} \underbrace{\| \Phi \|_{\mathrm{F}}^2}_{\text{regularizer}}$$

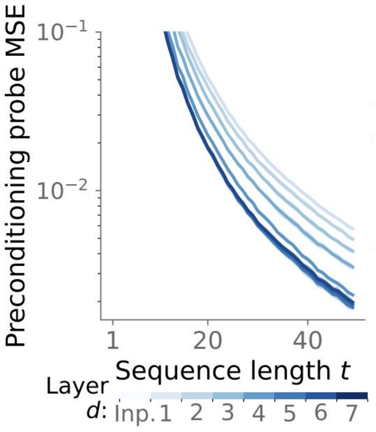
- Attention can precondition inputs, which speeds up optimization. With sufficiently many attention layers the optimal preconditioning is approached:

$$H_t^* z_t = \left(Z_{t-1} Z_{t-1}^\top + \lambda I \right)^{-1} z_t$$

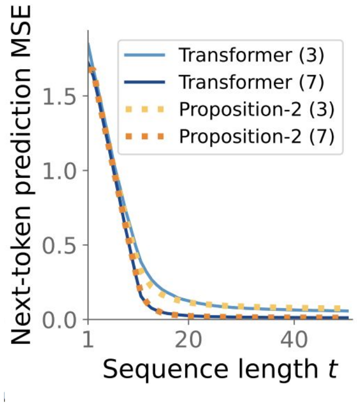
- Attention can take a gradient step $-\nabla L_t(\Phi)$ on the loss. A single step suffices for optimally preconditioned inputs.

Important insight for Mesa layer: Both these terms can be computed **in-parallel fashion** over the sequence length by attention layers. In particular, we discuss how to compute the inverse x vector product with (accelerated) Newton-Schulz.

MesaNet Origin | Evidence for theory

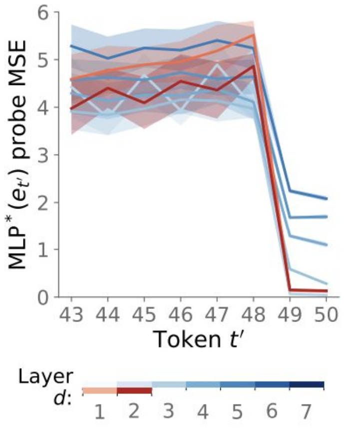


Preconditioning improves with sequence length and depth



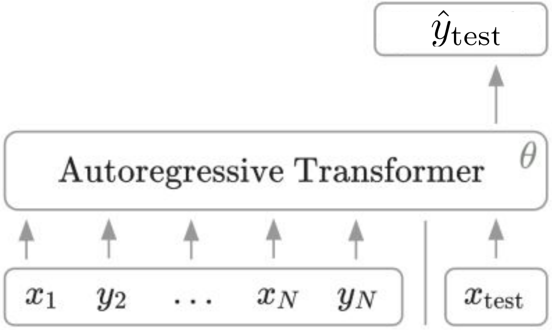
Theoretical model fits trained model

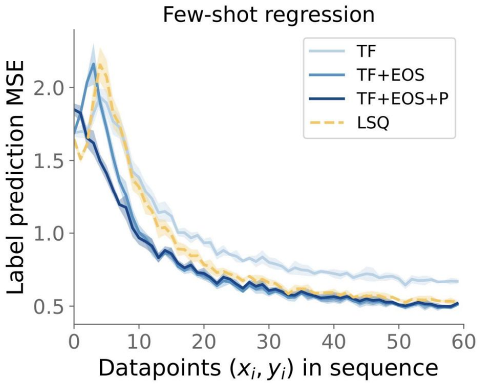
(Theoretical model hyperparameters tuned for best performance, not to fit transformer)



Early MLPs provide basis functions for nonlinear sequences

MesaNet Origin | Resulting transformer is a few-shot learner





- Evaluate **out-of-distribution on supervised regression task**
- Toy model for GPT-3 in-context learning findings of Brown et al. (2020)
- Toy model for **prompt fine-tuning** $[x_1, y_1, \mathrm{EOS}, x_2, y_2, \ldots, \mathrm{EOS}, x_N, y_N]$

finetune over many sequences to improve prediction of y_i

MesaNet Origin | Uncovering mesa-opt in transformers

- **Mechanistically interpretable model** of gradient-based ICL/mesa-optimization
- **Transformers develop in-context version of a classical machine learning algorithm**:
  Learn linear model after fixed nonlinear transform ⟶ good OOD generalization/transfer expected
- **Maximum likelihood estimation without explicitly inferring generator matrices**
  (cf. algorithms from data-driven control theory)
- **Large enough generator diversity is needed**
  ⟶ Raventós et al. (2023) analyze the transition to the algorithmic solution studied here

Talk Outline

Part 1: The Origin of the MesaLayer

1. Analysis of transformers trained on synthetic sequence data
2. A toy model for in-context few-shot learning

Google 2024-10-16

Uncovering mesa-optimization algorithms in Transformers

Johannes von Oswald[a,b,*], Maximilian Schlegel[a,b,*], Alexander Meulemans[a,b], Seijin Kobayashi[a,b], Eyvind Niklasson[a], Nicolas Zucchet[b], Nino Scherrer[a], Nolan Miller[d], Mark Sandler[d], Blaise Agüera y Arcas[a], Max Vladymyrov[d], Razvan Pascanu[e] and João Sacramento[a,b,*]
[a]Google, Paradigms of Intelligence Team, [b]ETH Zürich, [d]Google Research, [e]Google DeepMind, [*]Contributed equally to this work.

https://arxiv.org/abs/2309.05858

Part 2: Scaling MesaNets

3. Efficient sequence modeling by stacking greedy local learners
4. Synthetic + 1B parameter language modeling experiments

MesaNet: Sequence Modeling by Locally Optimal Test-Time Training

Johannes von Oswald[*], Nino Scherrer[*],
Seijin Kobayashi, Luca Versari, Songlin Yang[1], Maximilian Schlegel, Kaitlin Maile,
Yanick Schimpf, Oliver Sieberling[2], Alexander Meulemans, Rif A. Saurous, Guillaume Lajoie,
Charlotte Frenkel, Razvan Pascanu[3], Blaise Agüera y Arcas , and João Sacramento

https://arxiv.org/abs/2506.05233

MesaNet | Sequence modeling by stacking local learners

Our findings suggest a motif for designing neural sequence models:

- Sequence inputs specify layerwise objective functions;
- Each layer solves its own local learning problem;
- MLPs interconnect many such layers.

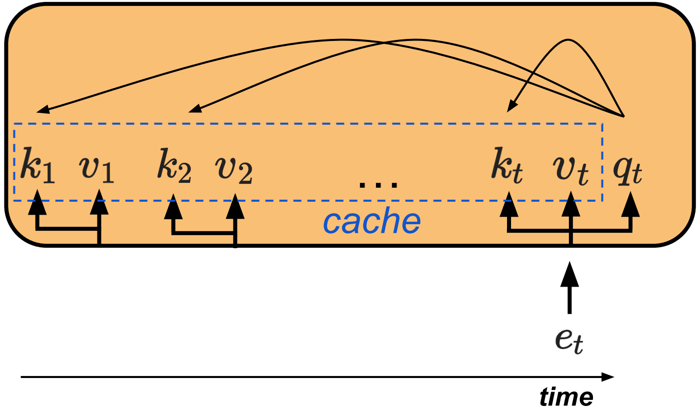
Bring local learning ideas explored in computational neuroscience to the fast 'in context' processing timescale

- Rely on backpropagation to discover how to usefully orchestrate local learners (cf. 'fast weight programmers', Schmidhuber 1992)

References: Schmidhuber (1992), Lillicrap et al. (2015), Nokland & Eidnes (2019), Veness et al. (2021), Hinton (2022)

MesaLayer | Softmax self-attention

$$V_t \,\mathrm{softmax}(K_t^\top q_t)$$



$$q_t = W_Q\, e_t$$ query

$$V_t = \begin{bmatrix} | & | & & | \\ W_V\, e_1 & W_V\, e_2 & \dots & W_V\, e_t \\ | & | & & | \end{bmatrix}$$ values

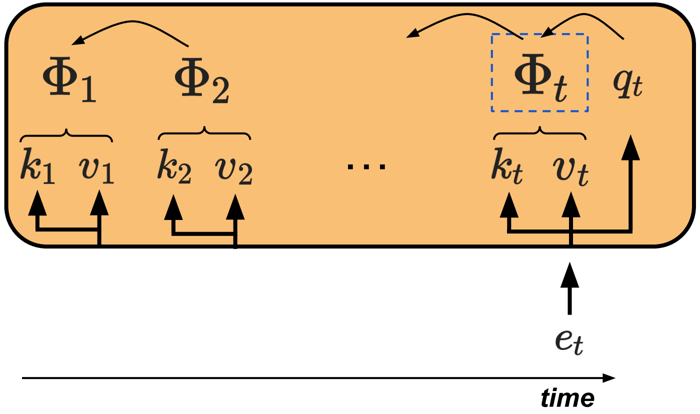
$$K_t = \begin{bmatrix} | & | & & | \\ W_K\, e_1 & W_K\, e_2 & \dots & W_K\, e_t \\ | & | & & | \end{bmatrix}$$ keys

Softmax self-attention stores the entire past

→ compute and memory cost per time step **grows linearly** with sequence length

MesaLayer | Efficient linear self-attention

$$V_t K_t^\top q_t = \left(\sum_{i=1}^{t} v_i k_i^\top \right) q_t = \Phi_t q_t$$



$q_t = W_Q \, e_t$ query

$V_t = \begin{bmatrix} | & | & & | \\ W_V \, e_1 & W_V \, e_2 & \dots & W_V \, e_t \\ | & | & & | \end{bmatrix}$ values

$K_t = \begin{bmatrix} | & | & & | \\ W_K \, e_1 & W_K \, e_2 & \dots & W_K \, e_t \\ | & | & & | \end{bmatrix}$ keys

Linear self-attention admits efficient implementation: recursively update state matrix Φ_t

$\longrightarrow$ compute and memory cost per time step is **constant**

References: Schmidhuber (1992), Katharopoulos et al. (2020)

MesaLayer | Associative memory view of attention

$$V_t K_t^\top q_t = \left(\sum_{i=1}^{t} v_i k_i^\top \right) q_t = \Phi_t q_t$$

- Learn linear key-value map using **local Hebbian rule**
- Store *t* associations in |*v*||*k*| weights
- Key idea behind many recent recurrent neural network alternatives to transformers (e.g., Mamba, xLSTM)

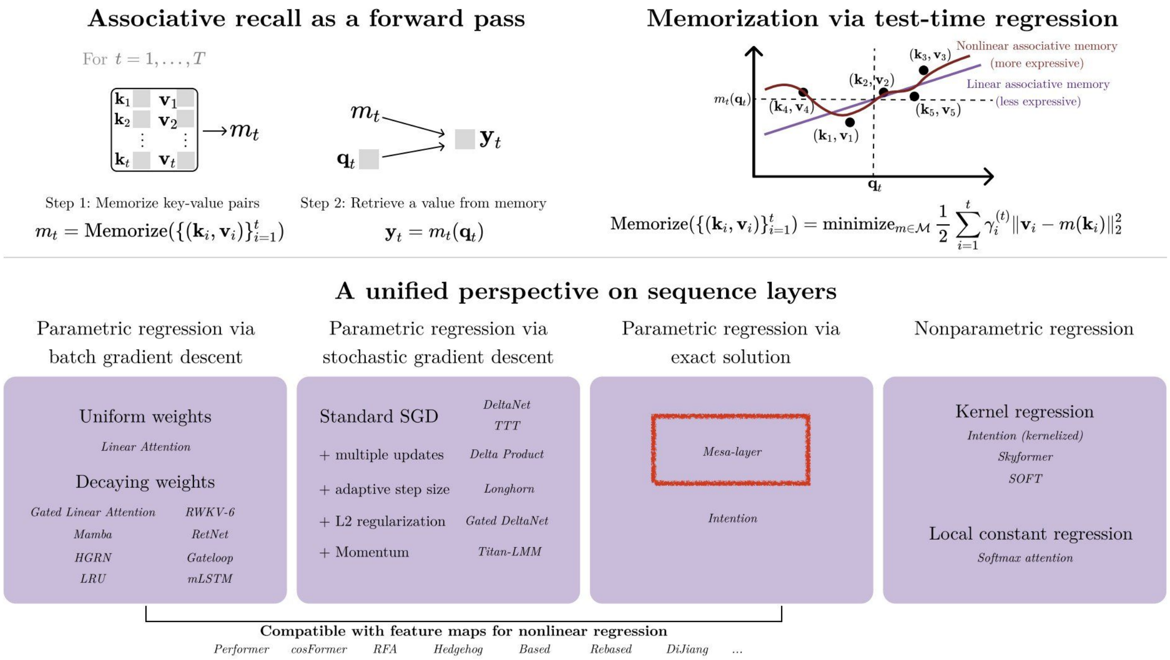
References: Hebb (1949), Oja (1982), Schlag et al. (2021), Gu & Dao (2023), Yang et al. (2023), Beck et al. (2024)

$$V_t \,\mathrm{softmax}(K_t^\top q_t)$$

- Soft nearest-neighbor memory retrieval using dot-product similarity kernel
- Store *t* key-value pairs explicitly

References: Vaswani et al. (2017), Krotov & Hopfield (2021), Ramsauer et al. (2021)

MesaLayer | Sequence layers as local learning



"MesaNet vs. DeltaNet is roughly analogous to Recursive Least Squares vs. Least Mean Squares in the signal processing literature—making MesaNet the more powerful of the two" - Songlin Yang

References: Wang et al. (2024), von Oswald et al. (2023)

MesaLayer | Current RNN overview

Model	Recurrence	Memory read-out
LinearAttention	$\Phi_t = \Phi_{t-1} + v_t \mathbf{k}_t^\top$	$o_t = \Phi_t q_t$
+Kernel	$\Phi_t = \Phi_{t-1} + v_t \phi(\mathbf{k}_t)^\top$	$o_t = \Phi_t \phi(q_t)$
+Normalization	$\Phi_t = \Phi_{t-1} + v_t \phi(\mathbf{k}_t)^\top, \quad \mathbf{z}_t = \mathbf{z}_{t-1} + \phi(\mathbf{k}_t)$	$o_t = \Phi_t \phi(q_t) / (\mathbf{z}_t^\top \phi(q_t))$
DeltaNet	$\Phi_t = \Phi_{t-1}(\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top) + \beta_t v_t \mathbf{k}_t^\top$	$o_t = \Phi_t q_t$
RetNet	$\Phi_t = \gamma \Phi_{t-1} + v_t \mathbf{k}_t^\top$	$o_t = \Phi_t q_t$
Mamba	$\Phi_t = \Phi_{t-1} \odot \exp(-\alpha_t \mathbf{1}^\top) \odot \exp(\mathbf{A}) + (\alpha_t \odot v_t) \mathbf{k}_t^\top$	$o_t = \Phi_t q_t + d \odot v_t$
GLA	$\Phi_t = \Phi_{t-1} \odot (\mathbf{1} \alpha_t^\top) + v_t \mathbf{k}_t^\top, \quad \Phi_t' = \Phi_{t-1} \mathrm{Diag}(\alpha_t) + v_t \mathbf{k}_t^\top$	$o_t = \Phi_t' q_t$
RWKV − 6	$\Phi_t = \Phi_{t-1} \mathrm{Diag}(\alpha_t) + v_t \mathbf{k}_t^\top$	$o_t = (\Phi_{t-1} + (d \odot v_t) \mathbf{k}_t^\top) q_t$
HGRN − 2	$\Phi_t = \Phi_{t-1} \mathrm{Diag}(\alpha_t) + v_t (\mathbf{1} - \alpha_t)^\top$	$o_t = \Phi_t q_t$
mLSTM	$\Phi_t = f_t \Phi_{t-1} + i_t v_t \mathbf{k}_t^\top, \quad \mathbf{z}_t = f_t \mathbf{z}_{t-1} + i_t \mathbf{k}_t$	$o_t = \Phi_t q_t / \max\{\mathbf{1}, \lvert \mathbf{z}_t^\top q_t \rvert\}$
Mamba − 2	$\Phi_t = \gamma \Phi_{t-1} + v_t \mathbf{k}_t^\top$	$o_t = \Phi_t q_t$
GatedDeltaNet	$\Phi_t = \Phi_{t-1} \odot (\alpha_t (\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)) + \beta_t v_t \mathbf{k}_t^\top$	$o_t = \Phi_t q_t$
** *Mesa* **	$G_t = \gamma_t G_{t-1} + \beta_t v_t k_t^\top, H_t = \gamma_t H_{t-1} + \beta_t k_t k_t^\top$	$o_t = G_t \mathrm{linsolve}(H_t + \Lambda, q_t)$

MesaLayer | Attention as locally-optimal learning

Local Learning Objective:

$$\Phi_t = \arg\min_{\Phi} \frac{1}{2} \sum_{t'=1}^{t} \gamma_{t'} \|v_{t'} - \Phi k_{t'}\|^2 + \frac{\lambda}{2} \|\Phi\|_{\mathrm{F}}^2$$

Derive a sequence modeling layer from a learning objective function

- **Hardwire the recipe discovered by our toy trained transformers**
- Improve generalization and memory capacity of Hebbian learning
- Value v_t plays the role of an internally generated target for local learning
- Forget factor $\gamma_t \in \mathbb{R}$ determines how strongly an association is learned/retained

MesaLayer | The Mesa layer: Locally Optimal Test-Time Training

Many of the previous layers can be motivated by online GD on a squared regression loss.
Mesa layer takes this to an extreme:

local learning objective

$$\Phi_t = \arg\min_{\Phi} \frac{1}{2} \sum_{t'=1}^{t} \gamma_{t'} \|v_{t'} - \Phi k_{t'}\|^2 + \frac{\lambda}{2} \|\Phi\|_{\mathrm{F}}^2$$

internal state variables

$$G_t = \underbrace{\gamma_t G_{t-1} + \beta_t v_t k_t^\top}_{\text{linear attention}}, \qquad H_t = \gamma_t H_{t-1} + \beta_t k_t k_t^\top$$

layer output
(predict using learned model)

$$\Phi_t q_t = G_t \underbrace{\mathrm{linsolve}(H_t + \lambda I, q_t)}_{\substack{\text{can use solver of choice} \\ \text{here: conjugate gradient method}}}$$ 😰

MesaLayer | Chunkwise parallel training

Softmax

$$\sum_{i=1}^{t} v_i \frac{e^{k_i^\top q_t}}{\sum_{j=1}^{t} e^{k_j^\top q_t}} \longrightarrow V\text{softmax}(D \odot K^\top Q)$$

Linearized Softmax

$$\sum_{i=1}^{t} v_i \frac{\phi(k_i)^\top \phi(q_t)}{\sum_{j=1}^{t} \phi(k_j)^\top \phi(q_t)} \longrightarrow V(D \odot K^\top Q)$$

Mesa

$$G_t \text{linsolve}(H_t + \Lambda, q_t) = \sum_{i=1}^{t} v_i k_i^\top q_t^* \longrightarrow V(D \odot K^\top Q^*)$$

with

$$q_t^* = \text{linsolve}(H_t + \Lambda, q_t) \longrightarrow$$

Approximate with the the conjugate gradient method in parallel!

MesaLayer | Conjugate Gradient Method in parallel

Algorithm 1 Rank-One Update Conjugate Gradient Method

procedure RANKONECONJUGATEGRADIENT($H_{t-1}, \gamma_t, k_t, q_t, \epsilon, k_{\max}$)
  Input: Symmetric positive-definite matrix $H_{t-1} \in \mathbb{R}^{n \times n}$, forget strength $\gamma_t \in (0, 1)$, key $k_t \in \mathbb{R}^n$, query $q_t \in \mathbb{R}^n$, tolerance $\epsilon > 0$, maximum iterations $k_{\max}$.
  Output: Approximate solution x.

  $k \leftarrow 0$
  $x \leftarrow q_t \cdot \operatorname{diag}(H_{t-1} + \Lambda_t)^{-1}$ ▷ Initial guess $x \in \mathbb{R}^n$
  $\boxed{r \leftarrow q_t - (H_{t-1}\gamma_t + k_t k_t^\top + \Lambda_t)x}$ ▷ Initial residual r
  $p \leftarrow r$ ▷ Initial search direction p
  $\delta_{old} \leftarrow r^T r$ ▷ Squared norm of the initial residual
  $\delta_0 \leftarrow \delta_{old}$ ▷ Store initial squared norm for relative tolerance

  while $k < k_{\max}$ **do** ▷ Loop until max iterations reached
    $\boxed{q \leftarrow (H_{t-1}\gamma_t + k_t k_t^\top + \Lambda_t)p}$ ▷ Matrix-vector product $(H_{t-1}\gamma_t + k_t k_t^\top + \Lambda_t)p$
    $\alpha \leftarrow \frac{\delta_{old}}{p^T q}$ ▷ Step length α
    $x \leftarrow x + \alpha p$ ▷ Update solution x

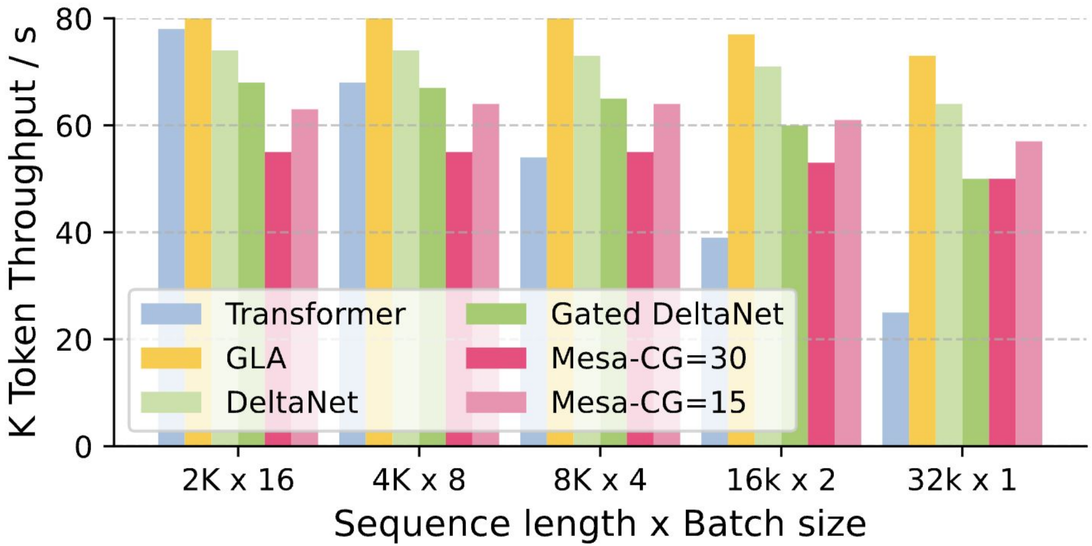
$$(H_t + \Lambda_t)p = H_t p + \Lambda_t \cdot p = \boxed{\sum_{i=1}^{t} \zeta_{ti} k_i k_i^\top p} + \Lambda_t \cdot p$$

= GLA computation

Attention: Numerics are problematic here

MesaNet | Runtime on H100

Mesa layer consists of running many steps of GLA to approximate $q_t^* = \mathrm{linsolve}(H_t + \Lambda, q_t)$ for many t in parallel and finally a last GLA step to compute $V(D \odot K^\top Q^*)$



[Check code here]

MesaLayer | Dynamic Test-Time Compute Allocation

Inside the conjugate gradient method:

$\delta_{old} \leftarrow r^T r$ ▷ Squared norm of the initial residual
$\delta_0 \leftarrow \delta_{old}$ ▷ Store initial squared norm for relative tolerance

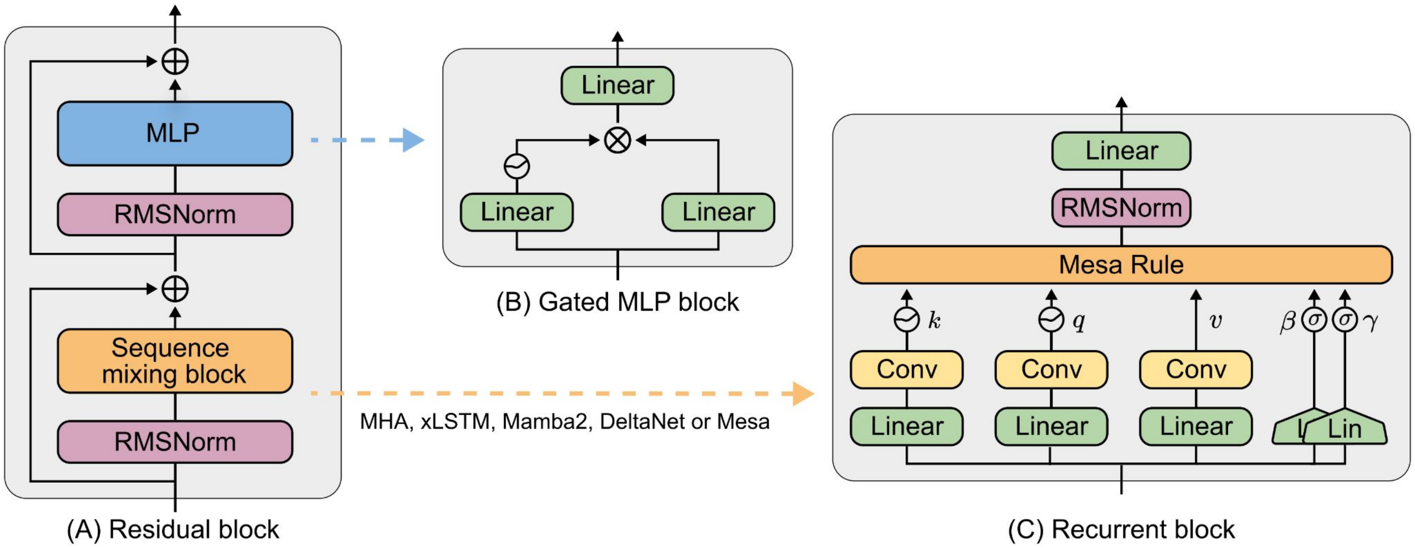
while $k < k_{\max}$ **do** ▷ Loop until max iterations reached
    $q \leftarrow (H_{t-1}\gamma_t + k_t k_t^\top + \Lambda_t)p$ ▷ Matrix-vector product $(H_{t-1}\gamma_t + k_t k_t^\top + \Lambda_t)p$
    $\alpha \leftarrow \frac{\delta_{old}}{p^T q}$ ▷ Step length α
    $x \leftarrow x + \alpha p$ ▷ Update solution x
    $r \leftarrow r - \alpha q$ ▷ Update residual r
    $\delta_{new} \leftarrow r^T r$ ▷ Squared norm of the new residual, δ_{new}

    if $\sqrt{\delta_{new}} \leq \epsilon\sqrt{\delta_0}$ **then** ▷ Check relative convergence: $||r_{k+1}|| \leq \epsilon||r_0||$
        break ▷ Converged
    end if

    $\beta \leftarrow \frac{\delta_{new}}{\delta_{old}}$ ▷ Improvement factor β
    $p \leftarrow r + \beta p$ ▷ Update search direction p

→ Mesa layer dynamically allocates test-time flops, in a data dependent manner

MesaNet | Model overview



Talk Outline

Part 1: The Origin of the MesaLayer

1. Analysis of transformers trained on synthetic sequence data
2. A toy model for in-context few-shot learning

Part 2: Scaling MesaNets

3. Efficient sequence modeling by stacking greedy local learners
4. Synthetic + 1B parameter language modeling experiments

Google 2024-10-16

Uncovering mesa-optimization algorithms in Transformers

Johannes von Oswald[a,b,*], Maximilian Schlegel[a,b,*], Alexander Meulemans[a,b], Seijin Kobayashi[a,b], Eyvind Niklasson[a], Nicolas Zucchet[b], Nino Scherrer[a], Nolan Miller[d], Mark Sandler[d], Blaise Agüera y Arcas[a], Max Vladymyrov[d], Razvan Pascanu[e] and João Sacramento[a,b,*]
[a]Google, Paradigms of Intelligence Team, [b]ETH Zürich, [d]Google Research, [e]Google DeepMind, [*]Contributed equally to this work.

https://arxiv.org/abs/2309.05858

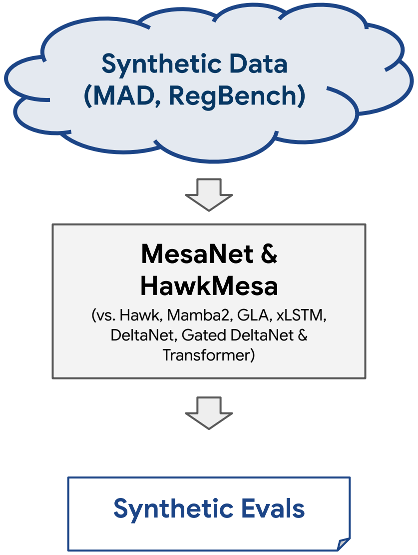
MesaNet: Sequence Modeling by Locally Optimal Test-Time Training

Johannes von Oswald[*], Nino Scherrer[*],
Seijin Kobayashi, Luca Versari, Songlin Yang[1], Maximilian Schlegel, Kaitlin Maile,
Yanick Schimpf, Oliver Sieberling[2], Alexander Meulemans, Rif A. Saurous, Guillaume Lajoie,
Charlotte Frenkel, Razvan Pascanu[3], Blaise Agüera y Arcas , and João Sacramento

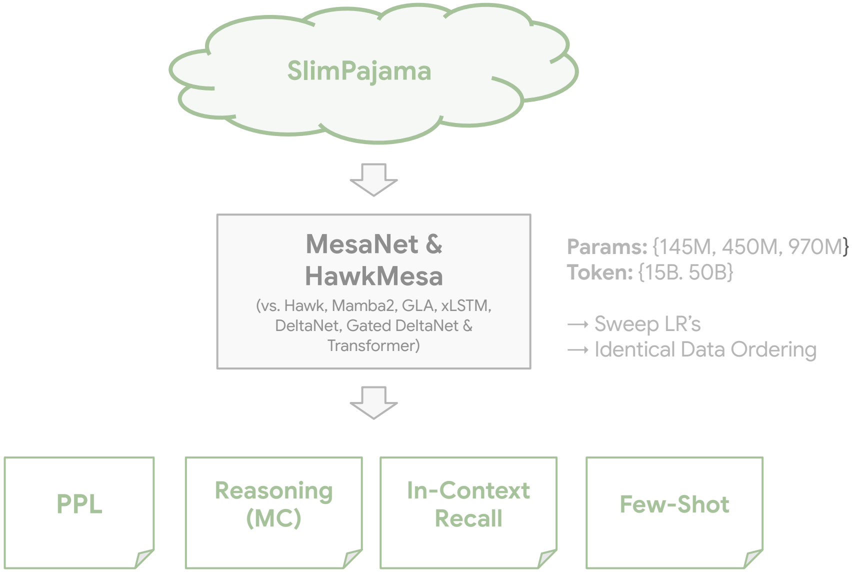
https://arxiv.org/abs/2506.05233

MesaNet | Evaluation Outlook

MesaNet in a **Synthetic World**



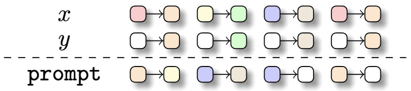
MesaNet in a **Language World**



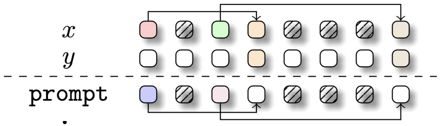
MesaNet | Synthetic Benchmark: MAD

Synthetic Benchmarks for Architecture Design

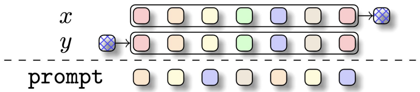
- In-Context Recall (Clean, Noisy, Fuzzy)



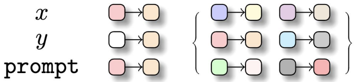
- Selective Copying



- Compression



- Memorization



	IC & Noisy Recall	Fuzzy Recall	Memorize Train Data	Selective Copy	Compress	Avg.
Mamba2	100	51.2	42.0	95.4	41.3	66.0
GLA	100	39.0	82.5	96.1	42.3	72.0
xLSTM	100	47.6	79.8	95.4	43.4	73.2
DeltaNet	100	55.5	40.8	98.8	43.3	67.7
Gated DeltaNet	100	32.7	81.7	95.7	45.0	71.0
Hawk	93.0	13.6	91.3	77.0	47.7	64.5
MesaNet	100	58.5	77.2	99.2	45.4	76.1
Hawk-MesaNet	100	30.2	85.6	99.6	52.3	73.5
Transformer	100	48.6	84.7	96.0	49.5	75.8

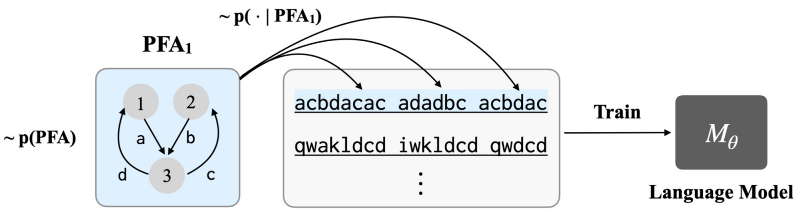
Table 1: Performance (% Accuracy ↑) on the MAD benchmark [73]. The MesaNet performs strongly compared to other RNNs and matches the transformer.

Eval Setup: 2-Layer models / Sweep over optimization params / Report best

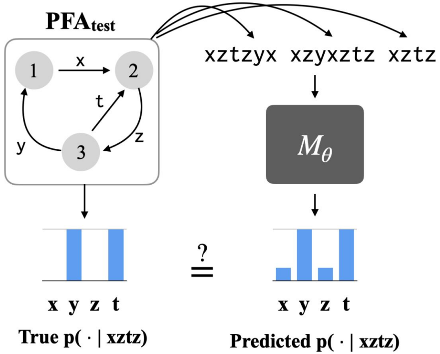
Reference: Poli et al, Mechanistic Design and Scaling of Hybrid Architectures, 2024

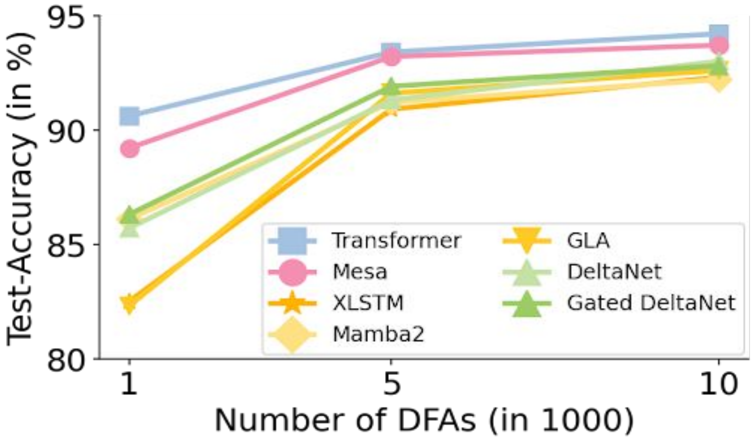
MesaNet | Synthetic Benchmark: RegBench

(1) Training



(2) Evaluation





Model Setup: Best performance across a sweep over

Hyper Parameter	Search
Embedding dimension	[64, 128, 256, 512, 1024]
Number of layers	[1, 2, 4, 8, 12]
Number of heads	[1, 2, 4]

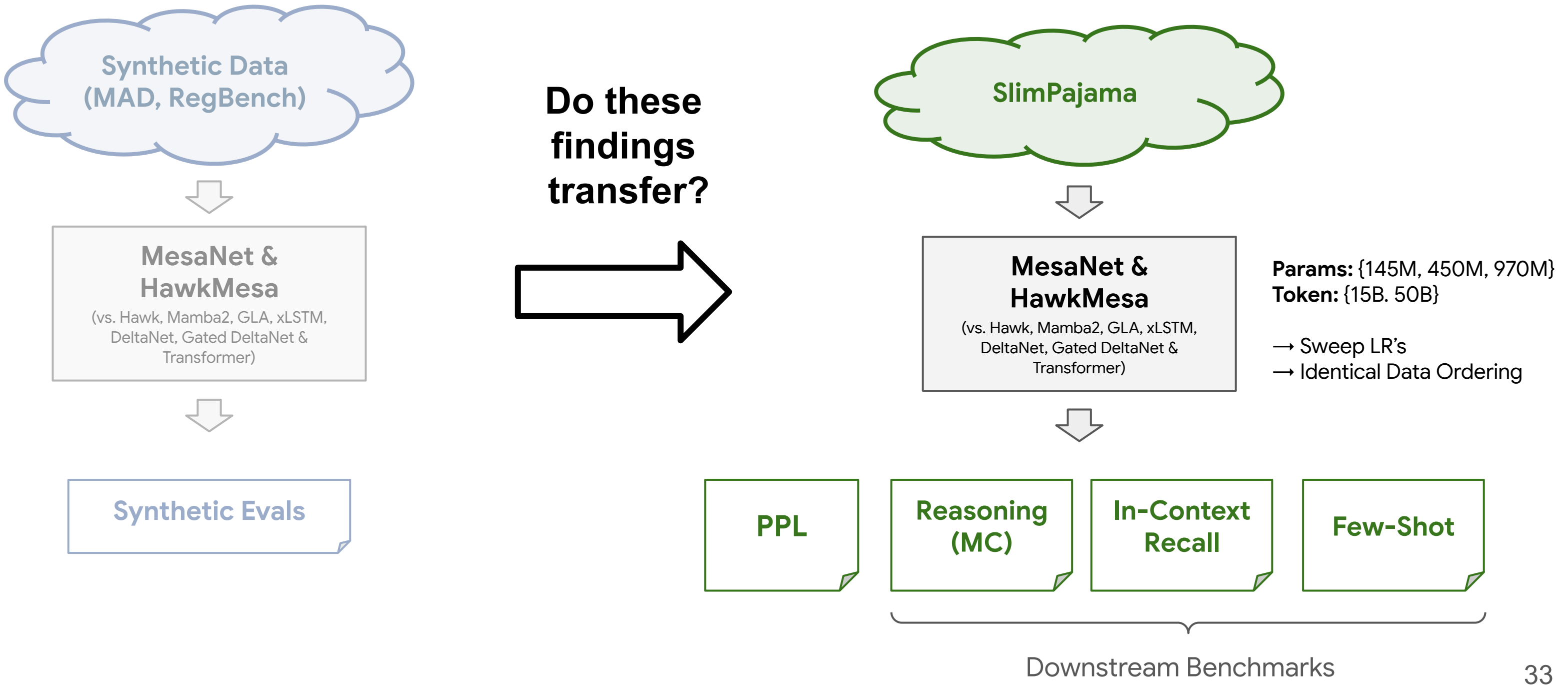
+ Optimization Hyperparameters

Reference: Akyürek et al., In-Context Language Learning: Architectures and Algorithms, 2024

MesaNet | Evaluation Outlook

MesaNet in a **Synthetic World**

MesaNet in a **Language World**

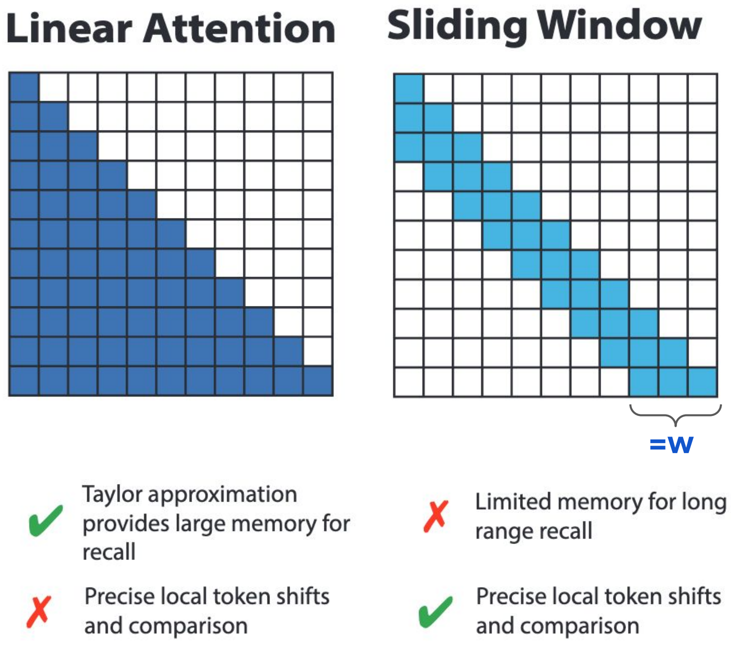


MesaNet | Sliding-Window-Attention (SWA) as a Control

- We know that linear layers suffer at in-context recall if the context is large enough
  Q: How long is the attention span of these models?

- **SWA with varying window sizes w** as a control
  - Perfect recall within window
  - Constant per-token memory and compute cost

- **Q:** How much window size do SWA models need to be competitive on PPL and benchmarks?



(Figure from Arora et al, 2024)

MesaNet | Perplexity (PPL) is all that counts!

		15B Tokens							50B Tokens						
		SLIM ppl ↓	LMB. ppl ↓	WIKI. ppl ↓	PG19 ppl ↓	GOV. ppl ↓	QASP. ppl ↓	AVG ppl ↓	SLIM ppl ↓	LMB. ppl ↓	WIKI. ppl ↓	PG19 ppl ↓	GOV. ppl ↓	QASP. ppl ↓	AVG ppl ↓
145M	- Hawk	19.73	38.94	23.06	19.87	19.23	29.66	25.08	18.34	37.43	21.25	18.49	18.17	27.83	23.59
	- Mamba2	18.29	40.34	20.86	19.17	17.03	23.71	23.23	17.05	38.22	19.24	17.87	15.90	22.10	21.73
	- GLA	17.37	37.96	19.57	18.11	15.86	22.37	21.87	16.30	36.20	18.43	16.90	15.02	20.91	20.62
	- xLSTM	17.35	37.97	19.57	18.12	15.88	22.50	21.90	16.20	36.19	18.31	16.97	14.91	20.85	20.57
	- DeltaNet	17.26	38.18	19.29	17.93	15.67	21.75	21.68	16.17	36.55	18.08	16.78	14.81	20.53	20.49
	- Gated-DeltaNet	17.12	37.62	19.18	17.77	15.55	22.13	21.56	16.05	35.80	18.04	16.79	14.77	20.67	20.35
	- Mesa	17.02	37.64	19.10	17.72	15.44	21.87	21.47	16.05	36.17	17.96	16.60	14.72	20.57	20.34
	- Hawk-Mesa	16.81	37.20	18.87	17.14	15.29	21.62	21.15	15.82	35.51	17.70	16.19	14.55	20.38	20.02
	- Transformer	16.95	38.69	18.65	17.47	15.00	20.80	21.26	15.81	36.54	17.35	16.25	14.04	19.33	19.89
400M	- Hawk	14.40	31.54	16.12	14.23	13.67	19.85	18.30	12.87	29.44	14.30	12.71	12.24	17.54	16.52
	- Mamba2	14.45	33.38	15.99	14.80	13.27	18.36	18.37	13.07	31.05	14.28	13.28	12.10	16.37	16.69
	- GLA	13.69	31.64	15.01	13.89	12.36	17.08	17.28	12.61	29.93	13.73	12.75	11.52	15.77	16.05
	- xLSTM	13.71	31.70	14.95	13.88	12.28	17.10	17.27	12.56	29.79	13.60	12.72	11.49	15.72	15.98
	- DeltaNet	13.80	31.98	15.07	14.01	12.51	17.20	17.43	12.59	30.00	13.68	12.70	11.49	15.57	16.00
	- Gated-DeltaNet	13.48	31.40	14.71	13.59	12.16	16.64	17.00	12.44	29.57	13.45	12.52	11.31	15.42	15.79
	- Mesa	13.44	31.38	14.65	13.51	12.02	16.56	16.93	12.34	29.57	13.36	12.40	11.15	15.19	15.67
	- Hawk-Mesa	13.37	31.10	14.55	13.32	12.07	16.68	16.85	12.30	29.38	13.33	12.30	11.28	15.32	15.65
	- SWA-4	23.36	38.65	29.29	23.51	26.94	48.24	31.66	19.32	33.76	23.43	19.35	21.50	35.41	25.46
	- SWA-64	15.98	32.97	18.89	16.31	15.20	23.08	20.40	14.04	30.51	16.35	14.19	13.25	19.37	17.95
	- SWA-256	14.69	32.64	16.99	15.04	13.42	19.36	18.69	13.23	30.36	14.94	13.38	12.08	17.09	16.85
	- SWA-1024	13.95	32.63	15.40	14.09	12.36	17.05	17.58	12.52	30.13	13.71	12.56	11.12	15.26	15.88
	- Transformer	13.64	32.25	14.71	13.73	12.06	16.51	17.15	12.40	30.10	13.23	12.42	10.96	14.84	15.66
1B	- Hawk	12.71	28.72	13.95	12.44	11.90	17.30	16.17	11.24	26.67	12.23	10.93	10.63	14.89	14.43
	- Mamba2	12.78	30.30	13.97	12.92	11.68	15.97	16.27	11.39	28.02	12.23	11.42	10.42	14.02	14.58
	- GLA	12.28	29.13	13.29	12.35	11.08	15.20	15.55	10.99	26.98	11.77	10.95	9.99	13.52	14.03
	- xLSTM	12.38	29.21	13.43	12.40	11.16	15.33	15.65	11.01	26.93	11.81	10.94	10.00	13.55	14.04
	- DeltaNet	12.23	29.13	13.20	12.28	11.04	15.11	15.50	11.01	27.08	11.73	11.00	10.02	13.44	14.05
	- Gated-DeltaNet	12.06	28.67	13.00	12.05	10.85	14.86	15.25	10.89	26.79	11.58	10.81	9.88	13.28	13.87
	- Mesa	12.02	28.57	12.92	11.96	10.76	14.76	15.17	10.83	26.78	11.49	10.71	9.80	13.13	13.79
	- Hawk-Mesa	11.91	28.45	12.79	11.83	10.72	14.60	15.05	10.78	26.59	11.53	10.60	9.79	13.20	13.75
	- SWA-4	20.27	34.66	24.56	20.33	22.98	40.37	27.20	16.46	29.93	19.42	16.42	17.86	29.15	21.54
	- SWA-64	14.08	30.01	16.47	14.33	13.34	19.78	18.00	12.37	27.76	14.14	12.51	11.56	16.77	15.85
	- SWA-256	12.98	29.63	14.76	13.18	11.82	16.82	16.53	11.60	27.39	12.89	11.71	10.58	14.69	14.81
	- SWA-1024	12.33	29.65	13.47	12.35	10.92	14.93	15.61	11.00	27.22	11.78	10.92	9.79	13.11	13.97
	- Transformer	12.16	29.55	12.90	12.10	10.68	14.47	15.31	10.86	27.16	11.42	10.74	9.69	12.86	13.79

(not meant to be readable)

MesaNet | Perplexity (PPL) is all that counts!

		15B Tokens							50B Tokens						
		SLIM ppl ↓	LMB. ppl ↓	WIKI. ppl ↓	PG19 ppl ↓	GOV. ppl ↓	QASP. ppl ↓	AVG ppl ↓	SLIM ppl ↓	LMB. ppl ↓	WIKI. ppl ↓	PG19 ppl ↓	GOV. ppl ↓	QASP. ppl ↓	AVG ppl ↓
145M	- Hawk	19.73	38.94	23.06	19.87	19.23	29.66	25.08	18.34	37.43	21.25	18.49	18.17	27.83	23.59
	- Mamba2	18.29	40.34	20.86	19.17	17.03	23.71	23.23	17.05	38.22	19.24	17.87	15.90	22.10	21.73
	- GLA	17.37	37.96	19.57	18.11	15.86	22.37	21.87	16.30	36.20	18.43	16.90	15.02	20.91	20.62
	- xLSTM	17.35	37.97	19.57	18.12	15.88	22.50	21.90	16.20	36.19	18.31	16.97	14.91	20.85	20.57
	- DeltaNet	17.26	38.18	19.29	17.93	15.67	21.75	21.68	16.17	36.55	18.08	16.78	14.81	20.53	20.49
	- Gated-DeltaNet	17.12	37.62	19.18	17.77	15.55	22.13	21.56	16.05	35.80	18.04	16.79	14.77	20.67	20.35
	- Mesa	17.02	37.64	19.10	17.72	15.44	21.87	21.47	16.05	36.17	17.96	16.60	14.72	20.57	20.34
	- Hawk-Mesa	16.81	37.20	18.87	17.14	15.29	21.62	21.15	15.82	35.51	17.70	16.19	14.55	20.38	20.02
	- Transformer	16.95	38.69	18.65	17.47	15.00	20.80	21.26	15.81	36.54	17.35	16.25	14.04	19.33	19.89
400M	- Hawk	14.40	31.54	16.12	14.23	13.67	19.85	18.30	12.87	29.44	14.30	12.71	12.24	17.54	16.52
	- Mamba2	14.45	33.38	15.99	14.80	13.27	18.36	18.37	13.07	31.05	14.28	13.28	12.10	16.37	16.69
	- GLA	13.69	31.64	15.01	13.89	12.36	17.08	17.28	12.61	29.93	13.73	12.75	11.52	15.77	16.05
	- xLSTM	13.71	31.70	14.95	13.88	12.28	17.10	17.27	12.56	29.79	13.60	12.72	11.49	15.72	15.98
	- DeltaNet	13.80	31.98	15.07	14.01	12.51	17.20	17.43	12.59	30.00	13.68	12.70	11.49	15.57	16.00
	- Gated-DeltaNet	13.48	31.40	14.71	13.59	12.16	16.64	17.00	12.44	29.57	13.45	12.52	11.31	15.42	15.79
	- Mesa	13.44	31.38	14.65	13.51	12.02	16.56	16.93	12.34	29.57	13.36	12.40	11.15	15.19	15.67
	- Hawk-Mesa	13.37	31.10	14.55	13.32	12.07	16.68	16.85	12.30	29.38	13.33	12.30	11.28	15.32	15.65
	- SWA-4	23.36	38.65	29.29	23.51	26.94	48.24	31.66	19.32	33.76	23.43	19.35	21.50	35.41	25.46
	- SWA-64	15.98	32.97	18.89	16.31	15.20	23.08	20.40	14.04	30.51	16.35	14.19	13.25	19.37	17.95
	- SWA-256	14.69	32.64	16.99	15.04	13.42	19.36	18.69	13.23	30.36	14.94	13.38	12.08	17.09	16.85
	- SWA-1024	13.95	32.63	15.40	14.09	12.36	17.05	17.58	12.52	30.13	13.71	12.56	11.12	15.26	15.88
	- Transformer	13.64	32.25	14.71	13.73	12.06	16.51	17.15	12.40	30.10	13.23	12.42	10.96	14.84	15.66
1B	- Hawk	12.71	28.72	13.95	12.44	11.90	17.30	16.17	11.24	26.67	12.23	10.93	10.63	14.89	14.43
	- Mamba2	12.78	30.30	13.97	12.92	11.68	15.97	16.27	11.39	28.02	12.23	11.42	10.42	14.02	14.58
	- GLA	12.28	29.13	13.29	12.35	11.08	15.20	15.55	10.99	26.98	11.77	10.95	9.99	13.52	14.03
	- xLSTM	12.38	29.21	13.43	12.40	11.16	15.33	15.65	11.01	26.93	11.81	10.94	10.00	13.55	14.04
	- DeltaNet	12.23	29.13	13.20	12.28	11.04	15.11	15.50	11.01	27.08	11.73	11.00	10.02	13.44	14.05
	- Gated-DeltaNet	12.06	28.67	13.00	12.05	10.85	14.86	15.25	10.89	26.79	11.58	10.81	9.88	13.28	13.87
	- Mesa	12.02	28.57	12.92	11.96	10.76	14.76	15.17	10.83	26.78	11.49	10.71	9.80	13.13	13.79
	- Hawk-Mesa	11.91	28.45	12.79	11.83	10.72	14.60	15.05	10.78	26.59	11.53	10.60	9.79	13.20	13.75
	- SWA-4	20.27	34.66	24.56	20.33	22.98	40.37	27.20	16.46	29.93	19.42	16.42	17.86	29.15	21.54
	- SWA-64	14.08	30.01	16.47	14.33	13.34	19.78	18.00	12.37	27.76	14.14	12.51	11.56	16.77	15.85
	- SWA-256	12.98	29.63	14.76	13.18	11.82	16.82	16.53	11.60	27.39	12.89	11.71	10.58	14.69	14.81
	- SWA-1024	12.33	29.65	13.47	12.35	10.92	14.93	15.61	11.00	27.22	11.78	10.92	9.79	13.11	13.97
	- Transformer	12.16	29.55	12.90	12.10	10.68	14.47	15.31	10.86	27.16	11.42	10.74	9.69	12.86	13.79

(not meant to be readable)

1B models trained on 50B tokens

	SLIM ppl ↓	LMB. ppl ↓	WIKI. ppl ↓	PG19 ppl ↓	GOV. ppl ↓	QASP. ppl ↓	AVG
- Hawk	11,24	26,67	12,23	10,93	10,63	14,89	14.43
- Mamba2	11,39	28,02	12,23	11,42	10,42	14,02	14.58
- GLA	10,99	29,77	11,77	10,95	9,99	13,52	14.03
- xLSTM	11,01	26,93	11,81	10,94	10,00	13,55	14.03
- DeltaNet	11,01	27,08	11,73	11,00	10,02	13,44	14.05
- Gated DeltaNet	10,89	26,79	11,58	10,81	9,88	13,28	13.87
- Mesa	10,83	26,78	11,49	10,71	9,80	13,13	13.79
- Hawk-Mesa	10,78	26,59	11,53	10,60	9,79	13,20	13.75
- SWA-4	16,46	29,93	19,42	16,42	17,86	29,15	21.54
- SWA-64	12,37	27,76	14,14	12,51	11,56	16,77	15.85
- SWA-1024	11,00	27,22	11,78	10,92	9,79	13,11	13.97
- Transformer	10,86	27,16	11,42	10,74	9,69	12,86	13.79

- Trends hold across model sizes & number of training tokens!

MesaNet | Perplexity (PPL) is all that counts!

		15B Tokens							50B Tokens						
		SLIM ppl ↓	LMB. ppl ↓	WIKI. ppl ↓	PG19 ppl ↓	GOV. ppl ↓	QASP. ppl ↓	AVG ppl ↓	SLIM ppl ↓	LMB. ppl ↓	WIKI. ppl ↓	PG19 ppl ↓	GOV. ppl ↓	QASP. ppl ↓	AVG ppl ↓
145M	- Hawk	19.73	38.94	23.06	19.87	19.23	29.66	25.08	18.34	37.43	21.25	18.49	18.17	27.83	23.59
	- Mamba2	18.29	40.34	20.86	19.17	17.03	23.71	23.23	17.05	38.22	19.24	17.87	15.90	22.10	21.73
	- GLA	17.37	37.96	19.57	18.11	15.86	22.37	21.87	16.30	36.20	18.43	16.90	15.02	20.91	20.62
	- xLSTM	17.35	37.97	19.57	18.12	15.88	22.50	21.90	16.20	36.19	18.31	16.97	14.91	20.85	20.57
	- DeltaNet	17.26	38.18	19.29	17.93	15.67	21.75	21.68	16.17	36.55	18.08	16.78	14.81	20.53	20.49
	- Gated-DeltaNet	17.12	37.62	19.18	17.77	15.55	22.13	21.56	16.05	35.80	18.04	16.79	14.77	20.67	20.35
	- Mesa	17.02	37.64	19.10	17.72	15.44	21.87	21.47	16.05	36.17	17.96	16.60	14.72	20.57	20.34
	- Hawk-Mesa	16.81	37.20	18.87	17.14	15.29	21.62	21.15	15.82	35.51	17.70	16.19	14.55	20.38	20.02
	- Transformer	16.95	38.69	18.65	17.47	15.00	20.80	21.26	15.81	36.54	17.35	16.25	14.04	19.33	19.89
400M	- Hawk	14.40	31.54	16.12	14.23	13.67	19.85	18.30	12.87	29.44	14.30	12.71	12.24	17.54	16.52
	- Mamba2	14.45	33.38	15.99	14.80	13.27	18.36	18.37	13.07	31.05	14.28	13.28	12.10	16.37	16.69
	- GLA	13.69	31.64	15.01	13.89	12.36	17.08	17.28	12.61	29.93	13.73	12.75	11.52	15.77	16.05
	- xLSTM	13.71	31.70	14.95	13.88	12.28	17.10	17.27	12.56	29.79	13.60	12.72	11.49	15.72	15.98
	- DeltaNet	13.80	31.98	15.07	14.01	12.51	17.20	17.43	12.59	30.00	13.68	12.70	11.49	15.57	16.00
	- Gated-DeltaNet	13.48	31.40	14.71	13.59	12.16	16.64	17.00	12.44	29.57	13.45	12.52	11.31	15.42	15.79
	- Mesa	13.44	31.38	14.65	13.51	12.02	16.56	16.93	12.34	29.57	13.36	12.40	11.15	15.19	15.67
	- Hawk-Mesa	13.37	31.10	14.55	13.32	12.07	16.68	16.85	12.30	29.38	13.33	12.30	11.28	15.32	15.65
	- SWA-4	23.36	38.65	29.29	23.51	26.94	48.24	31.66	19.32	33.76	23.43	19.35	21.50	35.41	25.46
	- SWA-64	15.98	32.97	18.89	16.31	15.20	23.08	20.40	14.04	30.51	16.35	14.19	13.25	19.37	17.95
	- SWA-256	14.69	32.64	16.99	15.04	13.42	19.36	18.69	13.23	30.36	14.94	13.38	12.08	17.09	16.85
	- SWA-1024	13.95	32.63	15.40	14.09	12.36	17.05	17.58	12.52	30.13	13.71	12.56	11.12	15.26	15.88
	- Transformer	13.64	32.25	14.71	13.73	12.06	16.51	17.15	12.40	30.10	13.23	12.42	10.96	14.84	15.66
1B	- Hawk	12.71	28.72	13.95	12.44	11.90	17.30	16.17	11.24	26.67	12.23	10.93	10.63	14.89	14.43
	- Mamba2	12.78	30.30	13.97	12.92	11.68	15.97	16.27	11.39	28.02	12.23	11.42	10.42	14.02	14.58
	- GLA	12.28	29.13	13.29	12.35	11.08	15.20	15.55	10.99	26.98	11.77	10.95	9.99	13.52	14.03
	- xLSTM	12.38	29.21	13.43	12.40	11.16	15.33	15.65	11.01	26.93	11.81	10.94	10.00	13.55	14.04
	- DeltaNet	12.23	29.13	13.20	12.28	11.04	15.11	15.50	11.01	27.08	11.73	11.00	10.02	13.44	14.05
	- Gated-DeltaNet	12.06	28.67	13.00	12.05	10.85	14.86	15.25	10.89	26.79	11.58	10.81	9.88	13.28	13.87
	- Mesa	12.02	28.57	12.92	11.96	10.76	14.76	15.17	10.83	26.78	11.49	10.71	9.80	13.13	13.79
	- Hawk-Mesa	11.91	28.45	12.79	11.83	10.72	14.60	15.05	10.78	26.59	11.53	10.60	9.79	13.20	13.75
	- SWA-4	20.27	34.66	24.56	20.33	22.98	40.37	27.20	16.46	29.93	19.42	16.42	17.86	29.15	21.54
	- SWA-64	14.08	30.01	16.47	14.33	13.34	19.78	18.00	12.37	27.76	14.14	12.51	11.56	16.77	15.85
	- SWA-256	12.98	29.63	14.76	13.18	11.82	16.82	16.53	11.60	27.39	12.89	11.71	10.58	14.69	14.81
	- SWA-1024	12.33	29.65	13.47	12.35	10.92	14.93	15.61	11.00	27.22	11.78	10.92	9.79	13.11	13.97
	- Transformer	12.16	29.55	12.90	12.10	10.68	14.47	15.31	10.86	27.16	11.42	10.74	9.69	12.86	13.79

(not meant to be readable)

1B models trained on 50B tokens

	SLIM ppl ↓	LMB. ppl ↓	WIKI. ppl ↓	PG19 ppl ↓	GOV. ppl ↓	QASP. ppl ↓	AVG
- Hawk	11,24	26,67	12,23	10,93	10,63	14,89	14.43
- Mamba2	11,39	28,02	12,23	11,42	10,42	14,02	14.58
- GLA	10,99	29,77	11,77	10,95	9,99	13,52	14.03
- xLSTM	11,01	26,93	11,81	10,94	10,00	13,55	14.03
- DeltaNet	11,01	27,08	11,73	11,00	10,02	13,44	14.05
- Gated DeltaNet	10,89	26,79	11,58	10,81	9,88	13,28	13.87
- Mesa	10,83	26,78	11,49	10,71	9,80	13,13	13.79
- Hawk-Mesa	10,78	26,59	11,53	10,60	9,79	13,20	13.75
- SWA-4	16,46	29,93	19,42	16,42	17,86	29,15	21.54
- SWA-64	12,37	27,76	14,14	12,51	11,56	16,77	15.85
- SWA-1024	11,00	27,22	11,78	10,92	9,79	13,11	13.97
- Transformer	10,86	27,16	11,42	10,74	9,69	12,86	13.79

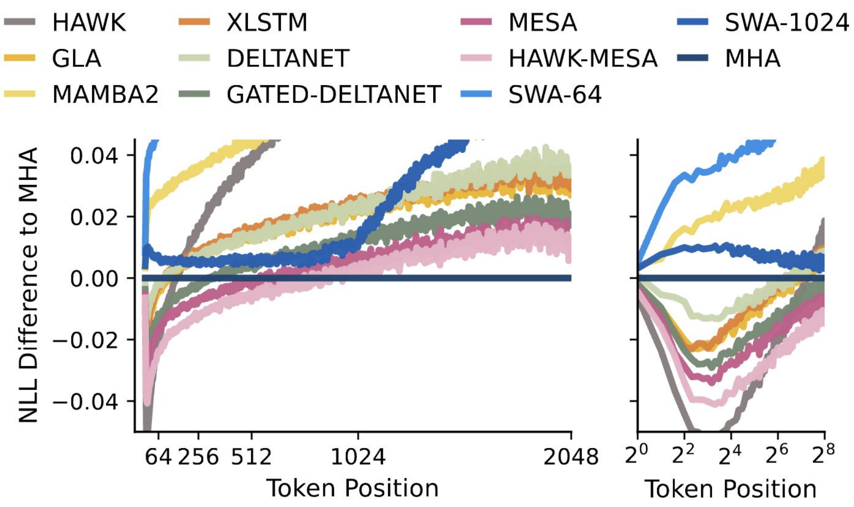
- Trends hold across model sizes & number of training tokens!
- SWA-1024 is a very competitive baseline (especially in the 50B token regime)
  Q: How much should we trust PPL when comparing different architectures?

MesaNet | Aggregated PPL masks important model differences

- Conditioning on token position + taking the delta w.r.t. Transformers reveals interesting differences

- Recurrent models are better than transformers **early-in-the sequence**
  ⟶ Majority of models: up to 256 tokens
  ⟶ Mesa & Hawk-Mesa: beyond 512 tokens

- Only Hawk beats Mesa early in the sequence, but deteriorates already around 64 tokens
  ⟶ Motivated Hybridization of Hawk & Mesa



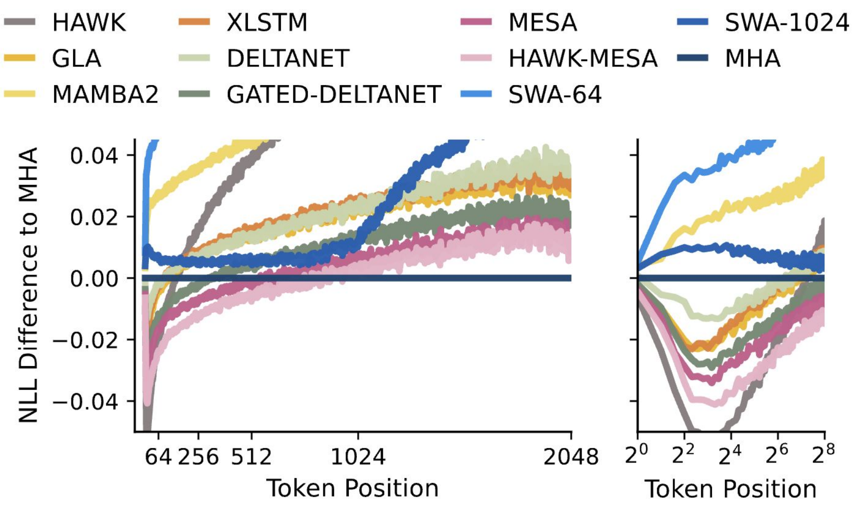
(1B models trained on 50B tokens)

MesaNet | Aggregated PPL masks important model differences

- Conditioning on token position + taking the delta w.r.t. Transformers reveals interesting differences

- Recurrent models are better than transformers **early-in-the sequence**
  → Majority of models: up to 256 tokens
  → Mesa & Hawk-Mesa: beyond 512 tokens

- Only Hawk beats Mesa early in the sequence, but deteriorates already around 64 tokens
  → Motivated Hybridization of Hawk & Mesa



(1B models trained on 50B tokens)

Take-Away Never solely look at aggregated PPL scores when comparing different model families.

MesaNet | Advantage window gets smaller with more tokens

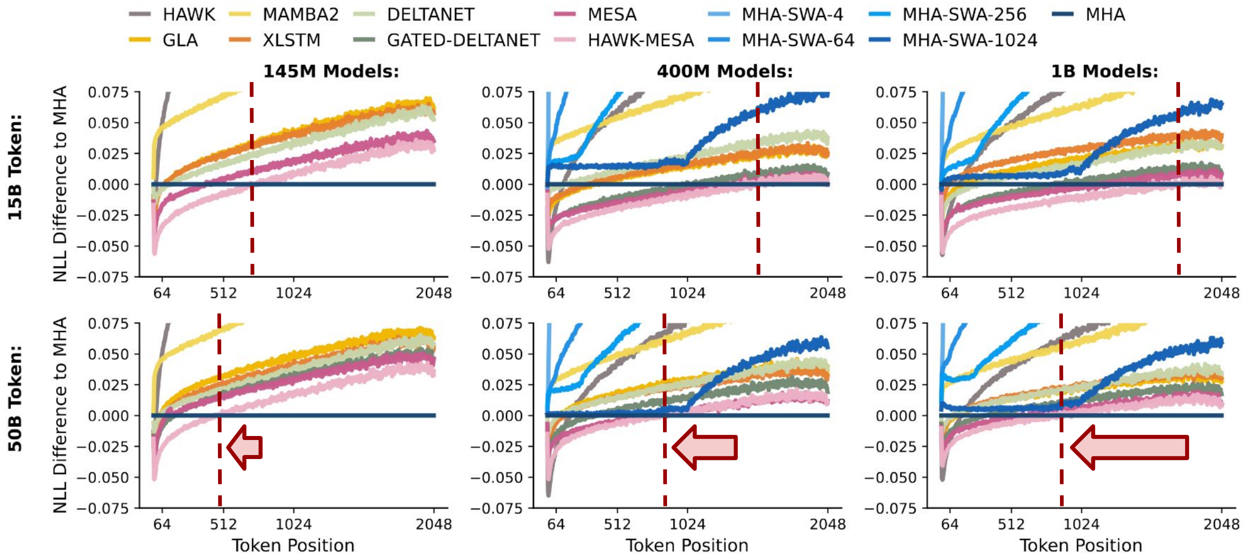


Figure 9: NLL Difference (per token-position) $\Delta\mathtt{NLL}_k^{\text{model}}$ relative to a Transformer on SlimPajama Validaton Dataset. Most recurrent models demonstrate superior language modelling abilities early in a sequence relative to the transformer baseline, across all settings. However, beyond a certain token position, transformers surpass the performance of all recurrent models.

→ The advantage window of linear models w.r.t. Transformers becomes smaller with more tokens

MesaNet | How well do recurrent models work at long context?

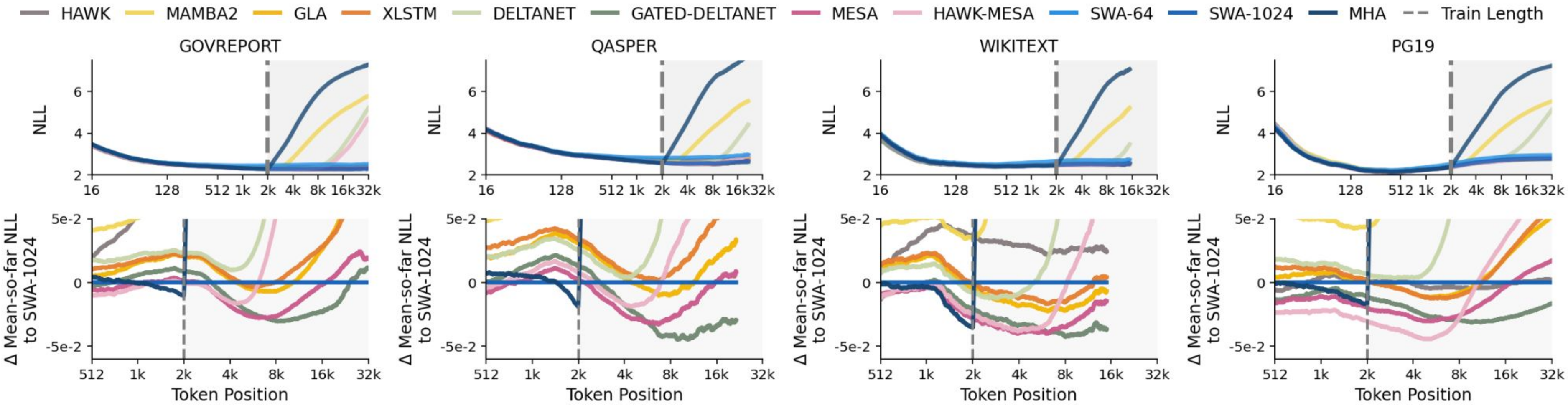
		15B Tokens					50B Tokens				
		WIKI. ppl ↓	PG19 ppl ↓	GOV. ppl ↓	QASP. ppl ↓	AVG ppl ↓	WIKI. ppl ↓	PG19 ppl ↓	GOV. ppl ↓	QASP. ppl ↓	AVG ppl ↓
145M	- Hawk	23.80	24.23	19.64	30.09	24.44	21.90	22.63	18.54	28.10	22.79
	- Mamba2	24.28	27.31	20.07	27.51	24.79	24.13	27.85	22.56	29.17	25.93
	- GLA	20.07	22.14	15.68	21.38	19.82	18.83	20.70	14.73	19.95	18.55
	- xLSTM	20.04	22.13	15.56	21.43	19.79	18.68	20.67	14.61	19.89	18.46
	- DeltaNet	19.85	22.05	15.47	20.85	19.55	18.66	20.64	14.64	19.76	18.42
	- Gated-DeltaNet	19.64	21.75	15.23	21.03	19.41	18.46	20.47	14.45	19.63	18.25
	- Mesa	19.52	21.60	15.10	20.78	19.25	18.38	20.25	14.42	19.52	18.14
	- Hawk-Mesa	19.33	20.86	15.03	20.69	18.98	18.15	19.72	14.31	19.48	17.91
	- Transformer	27.68	34.18	23.59	30.77	29.06	52.12	65.58	47.93	59.37	56.25
400M	- Hawk	16.61	17.35	13.80	19.73	16.87	14.70	15.45	12.33	17.35	14.96
	- Mamba2	18.31	20.59	15.33	20.59	18.70	17.94	20.75	16.07	20.48	18.81
	- GLA	15.31	16.84	12.08	16.20	15.11	14.05	15.43	11.26	14.95	13.92
	- xLSTM	15.31	16.82	11.98	16.18	15.07	13.90	15.39	11.22	14.87	13.85
	- DeltaNet	15.49	17.07	12.27	16.37	15.30	14.09	15.50	11.35	14.86	13.95
	- Gated-DeltaNet	14.99	16.46	11.84	15.73	14.76	13.75	15.13	11.04	14.60	13.63
	- Mesa	15.02	16.41	11.73	15.72	14.72	13.67	14.98	10.87	14.36	13.47
	- Hawk-Mesa	14.90	16.15	11.82	15.86	14.68	13.67	14.83	11.05	14.54	13.52
	- SWA-4	30.09	29.68	28.80	50.69	34.82	24.31	24.55	22.88	37.16	27.23
	- SWA-64	19.58	20.23	15.65	23.38	19.71	16.93	17.48	13.55	19.44	16.85
	- SWA-256	17.54	18.41	13.59	19.29	17.21	15.47	16.44	12.19	16.88	15.25
	- SWA-1024	15.90	17.28	12.32	16.58	15.52	14.22	15.41	11.27	14.92	13.95
	- Transformer	33.17	46.81	34.34	41.51	38.96	74.74	130.23	122.52	142.67	117.54
1B	- Hawk	14.37	15.11	12.01	17.10	14.65	12.59	13.25	10.67	14.68	12.80
	- Mamba2	15.90	18.03	13.33	17.85	16.28	17.56	20.90	16.28	19.98	18.68
	- GLA	13.56	14.90	10.81	14.37	13.41	12.05	13.15	9.77	12.80	11.94
	- xLSTM	13.71	14.98	10.88	14.54	13.53	12.11	13.15	9.79	12.86	11.98
	- DeltaNet	13.55	14.90	10.82	14.30	13.39	12.11	13.32	9.84	12.79	12.02
	- Gated DelaNet	13.26	14.50	10.56	14.01	13.08	11.86	12.98	9.62	12.54	11.75
	- Mesa	13.21	14.43	10.50	13.93	13.02	11.78	12.90	9.57	12.43	11.67
	- Hawk-Mesa	13.08	14.27	10.49	13.85	12.92	11.81	12.72	9.60	12.53	11.66
	- SWA-4	25.40	25.64	24.58	42.51	29.53	20.17	20.71	18.99	30.44	22.58
	- SWA-64	17.05	17.70	13.74	20.02	17.13	14.66	15.34	11.81	16.84	14.66
	- SWA-256	15.25	16.11	11.98	16.71	15.01	13.33	14.24	10.65	14.49	13.18
	- SWA-1024	13.89	15.03	10.84	14.45	13.56	12.20	13.27	9.75	12.71	11.98
	- Transformer	24.40	31.60	24.06	30.51	27.64	46.14	64.04	57.04	74.80	60.50

Table 9: PPL at a Maximum Sequence Length of 4k.

1B models trained on 50B tokens

	WIKI. ppl ↓	PG19 ppl ↓	GOV. ppl ↓	QASP. ppl ↓	AVG ppl ↓
- Hawk	12.59	13.25	10.67	14.68	12.80
- Mamba2	17.56	20.90	16.28	19.98	18.68
- GLA	12.05	13.15	9.77	12.80	11.94
- xLSTM	12.11	13.15	9.79	12.86	11.98
- DeltaNet	12.11	13.32	9.84	12.79	12.02
- Gated DelaNet	11.86	12.98	9.62	12.54	11.75
- Mesa	11.78	12.90	9.57	12.43	11.67
- Hawk-Mesa	11.81	12.72	9.60	12.53	11.66
- SWA-4	20.17	20.71	18.99	30.44	22.58
- SWA-64	14.66	15.34	11.81	16.84	14.66
- SWA-256	13.33	14.24	10.65	14.49	13.18
- SWA-1024	12.20	13.27	9.75	12.71	11.98
- Transformer	46.14	64.04	57.04	74.80	60.50

MesaNet | How well do recurrent models work at long context?



→ Gated-DeltaNet outperform MesaNet beyond 4k tokens

→ **But:** SWA-1024 achieves competitive scores

MesaNet | Three types of downstream benchmarks

- Zero-Shot Reasoning
- In-context Recall
- Few-Shot Learning
  - Word Scrambling Tasks
  - Translation

MesaNet | How well perform SWA models?

- Zero-Shot Reasoning
- In-context Recall
- Few-Shot Learning
  - Word Scrambling Tasks
  - Translation

Model	LMB. acc ↑	Hella. acc ↑	RACE-M acc ↑	RACE-H acc ↑	AVG	PIQA acc ↑	Wino acc ↑	ARC-E acc ↑	ARC-C acc ↑	SIQA acc ↑	BOOLQ acc ↑	OBQA acc ↑	SC. acc ↑	AVG
400M Parameters / 15B Tokens														
- SWA-4	4,62	34,97	25,97	25,93	22,87	66,81	49,33	43,81	24,23	39,82	57,31	30,00	63,78	46,89
- SWA-16	27,11	37,20	28,18	28,04	30,13	67,63	52,64	43,52	23,81	39,71	54,89	27,60	65,82	46,95
- SWA-64	38,54	39,35	32,87	30,24	35,25	68,93	52,17	44,40	22,87	39,76	58,56	29,20	64,99	47,61
- SWA-256	40,52	40,44	34,25	31,48	36,67	69,21	50,67	43,35	24,91	40,89	56,82	30,20	66,90	47,87
- SWA-1024	41,43	40,90	37,57	34,26	38,54	67,90	52,80	44,49	22,61	40,58	60,37	30,20	66,58	48,19
- SWA-1536	41,01	40,63	35,64	33,49	37,69	68,50	52,88	43,35	23,81	39,25	56,06	29,00	66,65	47,44
- Transformer	41,12	41,27	37,29	34,45	38,53	68,23	51,07	44,28	24,57	40,23	58,10	28,40	66,58	47,68
400M Parameters / 50B Tokens														
- SWA-4	18,28	39,02	29,56	27,66	28,63	67,85	51,93	44,49	24,83	39,71	58,23	32,40	66,14	48,20
- SWA-16	35,03	41,52	29,01	28,33	33,47	68,99	52,72	45,88	24,32	39,56	57,40	33,00	67,54	48,68
- SWA-64	42,34	44,14	34,53	31,67	38,17	69,53	53,75	45,24	24,74	40,28	56,45	31,60	68,49	48,76
- SWA-256	43,86	45,31	36,46	35,79	40,36	70,24	52,33	45,79	23,98	40,23	57,00	32,40	68,94	48,86
- SWA-1024	45,08	46,43	38,95	34,74	41,30	69,64	52,25	45,71	25,00	40,07	57,92	32,20	67,92	48,84
- SWA-1536	46,56	46,57	37,02	34,74	41,22	70,02	53,83	46,17	25,60	40,48	53,12	33,80	70,15	49,15
- Transformer	44,96	46,30	41,44	35,89	42,15	69,91	52,64	45,96	24,06	40,48	57,31	30,40	69,64	48,80

MesaNet | How well perform SWA models?

- Zero-Shot Reasoning
- In-context Recall
- Few-Shot Learning
  - Word Scrambling Tasks
  - Translation

Model	LMB. acc ↑	Hella. acc ↑	RACE-M acc ↑	RACE-H acc ↑	AVG	PIQA acc ↑	Wino acc ↑	ARC-E acc ↑	ARC-C acc ↑	SIQA acc ↑	BOOLQ acc ↑	OBQA acc ↑	SC. acc ↑	AVG
400M Parameters / 15B Tokens														
- SWA-4	4,62	34,97	25,97	25,93	22,87	66,81	49,33	43,81	24,23	39,82	57,31	30,00	63,78	46,89
- SWA-16	27,11	37,20	28,18	28,04	30,13	67,63	52,64	43,52	23,81	39,71	54,89	27,60	65,82	46,95
- SWA-64	38,54	39,35	32,87	30,24	35,25	68,93	52,17	44,40	22,87	39,76	58,56	29,20	64,99	47,61
- SWA-256	40,52	40,44	34,25	31,48	36,67	69,21	50,67	43,35	24,91	40,89	56,82	30,20	66,90	47,87
- SWA-1024	41,43	40,90	37,57	34,26	38,54	67,90	52,80	44,49	22,61	40,58	60,37	30,20	66,58	48,19
- SWA-1536	41,01	40,63	35,64	33,49	37,69	68,50	52,88	43,35	23,81	39,25	56,06	29,00	66,65	47,44
- Transformer	41,12	41,27	37,29	34,45	38,53	68,23	51,07	44,28	24,57	40,23	58,10	28,40	66,58	47,68
400M Parameters / 50B Tokens														
- SWA-4	18,28	39,02	29,56	27,66	28,63	67,85	51,93	44,49	24,83	39,71	58,23	32,40	66,14	48,20
- SWA-16	35,03	41,52	29,01	28,33	33,47	68,99	52,72	45,88	24,32	39,56	57,40	33,00	67,54	48,68
- SWA-64	42,34	44,14	34,53	31,67	38,17	69,53	53,75	45,24	24,74	40,28	56,45	31,60	68,49	48,76
- SWA-256	43,86	45,31	36,46	35,79	40,36	70,24	52,33	45,79	23,98	40,23	57,00	32,40	68,94	48,86
- SWA-1024	45,08	46,43	38,95	34,74	41,30	69,64	52,25	45,71	25,00	40,07	57,92	32,20	67,92	48,84
- SWA-1536	46,56	46,57	37,02	34,74	41,22	70,02	53,83	46,17	25,60	40,48	53,12	33,80	70,15	49,15
- Transformer	44,96	46,30	41,44	35,89	42,15	69,91	52,64	45,96	24,06	40,48	57,31	30,40	69,64	48,80

Global Benchmarks
→ longer context ranges are of benefit

Local Benchmarks
→ solvable through local heuristics
→ too hard and hence noisy signals

MesaNet | How well perform SWA models?

- Zero-Shot Reasoning
- In-context Recall
- Few-Shot Learning
  - Word Scrambling Tasks
  - Translation

		15B Tokens							50B Tokens						
		SWDE acc ↑	SQUAD acc ↑	FDA acc ↑	TQA acc ↑	NQ acc ↑	DROP acc ↑	AVG acc ↑	SWDE acc ↑	SQUAD acc ↑	FDA acc ↑	TQA acc ↑	NQ acc ↑	DROP	AVG
400M Models:	- SWA-4	7,38	5,60	0,18	14,51	3,52	9,15	6,72	10,98	7,77	0,45	21,27	5,16	13,13	9,79
	- SWA-16	9,63	10,82	0,27	24,88	4,88	15,33	10,97	13,05	18,30	1,09	33,35	6,59	17,35	14,95
	- SWA-64	13,14	26,74	10,07	39,34	5,23	19,12	18,94	19,17	38,44	11,43	48,76	7,25	23,96	24,84
	- SWA-256	21,69	40,92	12,25	50,95	6,87	23,67	26,06	30,96	42,19	14,70	56,16	10,10	24,20	29,72
	- SWA-1024	54,91	43,06	17,79	52,67	10,86	26,45	34,29	60,04	46,82	22,60	58,06	13,84	27,89	38,21
	- Transformer	77,50	37,13	79,13	53,08	16,57	26,59	48,33	79,66	36,93	75,86	58,95	18,94	29,37	49,95
1B Models:	- SWA-4	9,00	6,53	0,27	17,06	4,40	11,60	8,14	13,05	10,66	0,27	26,54	7,10	13,61	11,87
	- SWA-16	9,54	15,25	0,27	29,15	6,46	16,44	12,85	16,74	23,76	2,09	39,28	8,46	18,59	18,15
	- SWA-64	16,74	30,56	16,61	44,55	7,19	20,46	22,69	22,32	39,85	12,70	51,90	9,63	23,91	26,72
	- SWA-256	25,74	45,34	17,79	56,10	8,81	26,45	30,04	35,82	46,45	17,33	59,77	12,54	27,46	33,23
	- SWA-1024	60,76	40,65	24,23	56,99	11,88	27,65	37,03	63,73	47,65	26,68	61,43	15,52	30,04	40,84
	- Transformer	79,21	42,76	77,04	56,99	18,69	29,47	50,69	83,35	46,92	70,96	63,21	21,79	27,41	52,27
	- Random	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0

Table 13: Reference Scores of SWA Models on In-Context Recall Benchmarks. The pattern of best scores (highlightreded) is very consistent across the evaluated settings. As expected, we see increasing performance with increasing sizes of attention windows. Except on SQUAD, the transformer commonly attains the best scores.

⟶ Scores generally improve with an increasing attention window w

⟶ But relevant information is not distributed equally across context length
e.g., FDA: Most relevant information is part of the header

MesaNet | How well perform SWA models?

- Zero-Shot Reasoning
- In-context Recall
- Few-Shot Learning
  - Word Scrambling Tasks
  - Translation

		gpt3/cycle_letters_in_word 0-shot	1-shot	10-shot	100-shot	gpt3/mid_word_2_anagrams 0-shot	1-shot	10-shot	100-shot
400M Models	- SWA-4	0.0	0.4±0.3	0.8±0.3	0.8±0.2	0.0	0.3±0.3	0.9±0.3	0.9±0.3
	- SWA-64	0.1	2.5±1.5	4.6±1.1	4.7±0.9	0.1	1.2±0.5	2.7±0.2	2.7±0.1
	- SWA-1024	0.3	2.5±1.6	6.1±0.9	7.7±0.5	0.8	1.2±0.8	2.9±0.4	3.1±0.3
	- Transformer	0.4	2.4±1.8	6.7±1.2	8.5±0.4	0.5	1.4±0.7	3.3±0.4	3.6±0.2
1B Models	- SWA-4	0.1	1.1±0.9	1.5±0.7	2.0±0.8	0.2	0.6±0.5	1.4±0.3	1.4±0.3
	- SWA-64	1.3	3.5±1.8	6.3±1.3	7.8±0.6	1.0	2.4±0.7	3.8±0.3	4.0±0.3
	- SWA-1024	0.1	3.4±1.8	7.5±1.3	9.0±0.5	0.1	1.9±0.9	4.3±0.4	4.3±0.2
	- Transformer	0.0	3.0±2.2	6.8±1.7	9.2±0.6	0.1	2.4±0.6	4.2±0.4	4.7±0.2

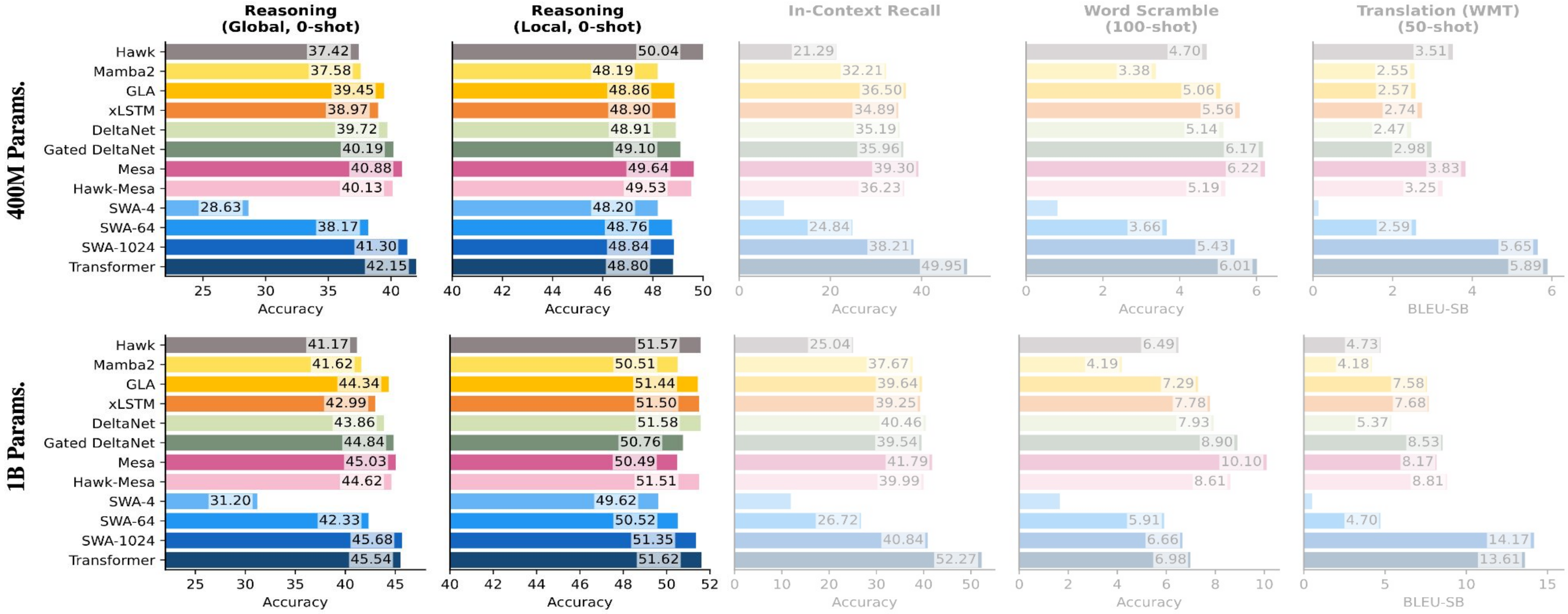
Table 16: Few-Shot Performance (Accuracy ± Std.) on GPT-3 Word Scrambling Tasks [17]

		WMT14 FR-EN 0	1	5	10	50	WMT16 DE-EN 0	1	5	10	50	WMT16 RO-EN 0	1	5	10	50
400M Models:	- SWA-4	0,34	0,13	0,14	0,13	0,12	0,25	0,19	0,26	0,21	0,26	0,29	0,10	0,06	0,07	0,05
	- SWA-64	1,35	3,82	4,46	4,94	4,92	1,45	2,17	1,66	2,09	1,57	1,18	1,10	1,38	0,88	1,29
	- SWA-1024	4,09	4,55	8,49	7,77	9,16	3,09	3,66	4,57	5,14	5,11	1,96	0,55	1,82	2,99	2,67
	- Transformer	2,61	8,27	8,77	8,92	9,63	2,04	3,13	5,73	5,34	5,49	1,94	1,02	1,29	2,23	2,56
1B Models:	- SWA-4	0,54	0,72	0,72	0,72	0,74	0,49	0,75	0,89	0,87	0,72	0,22	0,12	0,14	0,11	0,06
	- SWA-64	5,58	6,69	2,92	8,43	7,61	4,09	5,27	4,69	4,05	3,45	2,26	1,68	1,85	3,12	3,05
	- SWA-1024	8,75	16,65	18,09	18,70	19,83	5,99	10,85	14,58	14,91	14,30	3,36	4,19	10,14	10,05	8,38
	- Transformer	8,30	18,49	17,81	17,70	19,14	6,10	13,06	11,99	13,99	13,85	3,54	5,92	7,11	7,35	7,82

Table 17: Performance Scores (in BLEU-sb) on three Translation Tasks on Models Trained on 50B Tokens.

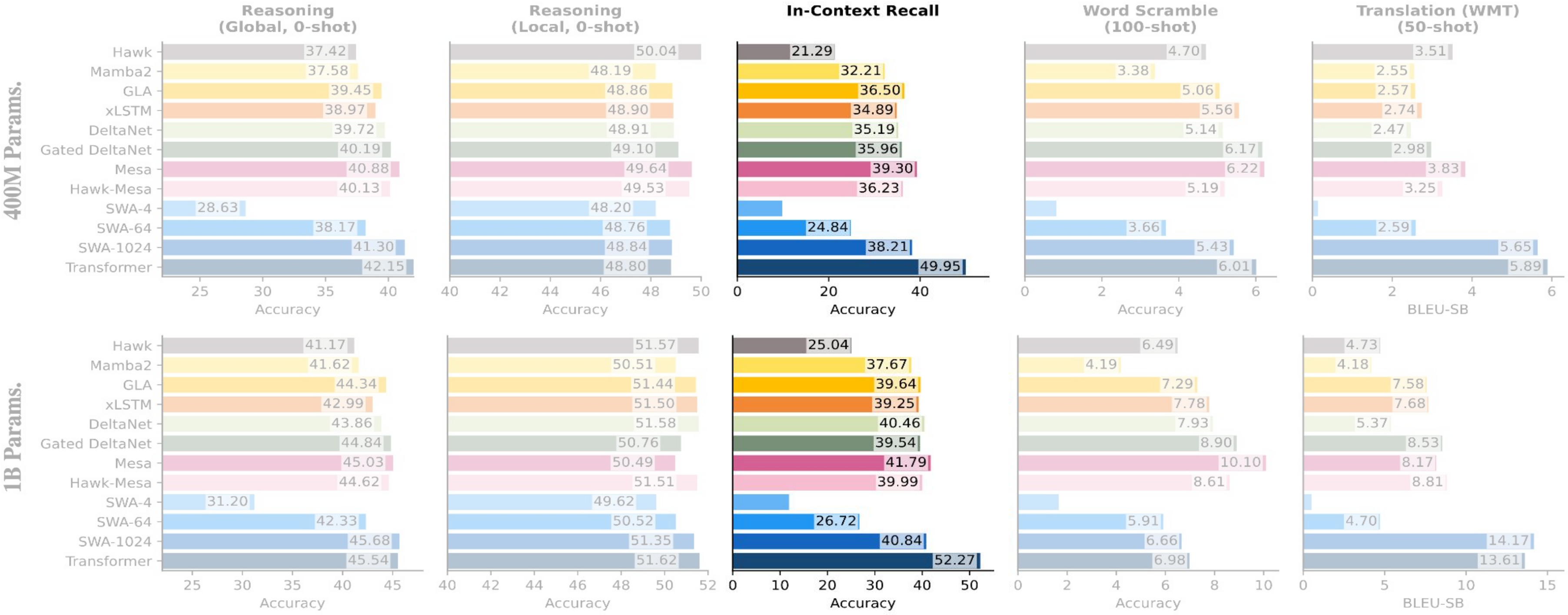
→ Scores improve with increasing an attention window w

MesaNet | Downstream Benchmarks: Zero-Shot Reasoning



→ **Global Benchmarks:** MesaNet & Transformer >= other RNNS
→ **Local Benchmark:** All models perform very similar – Hawk is surprisingly strong here

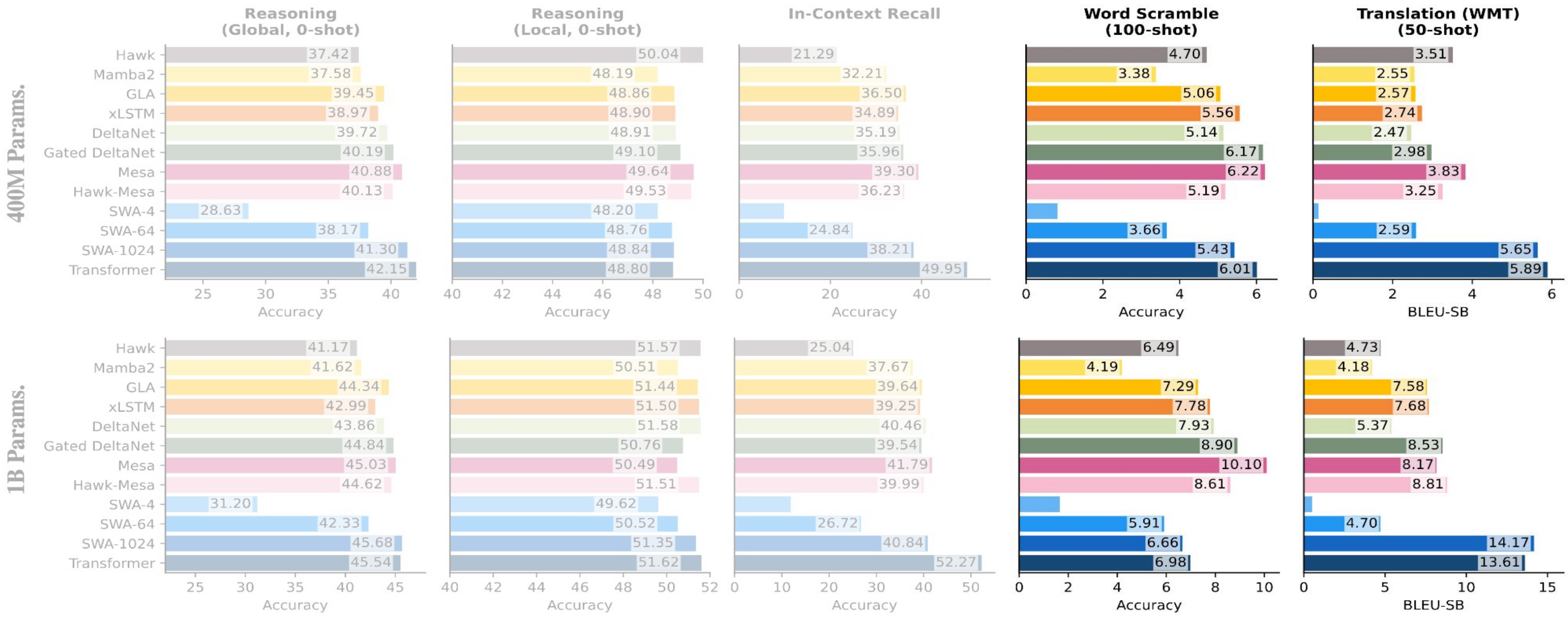
MesaNet | Downstream Benchmarks: In-Context Recall



→ Transformer >> MesaNet > other RNNs

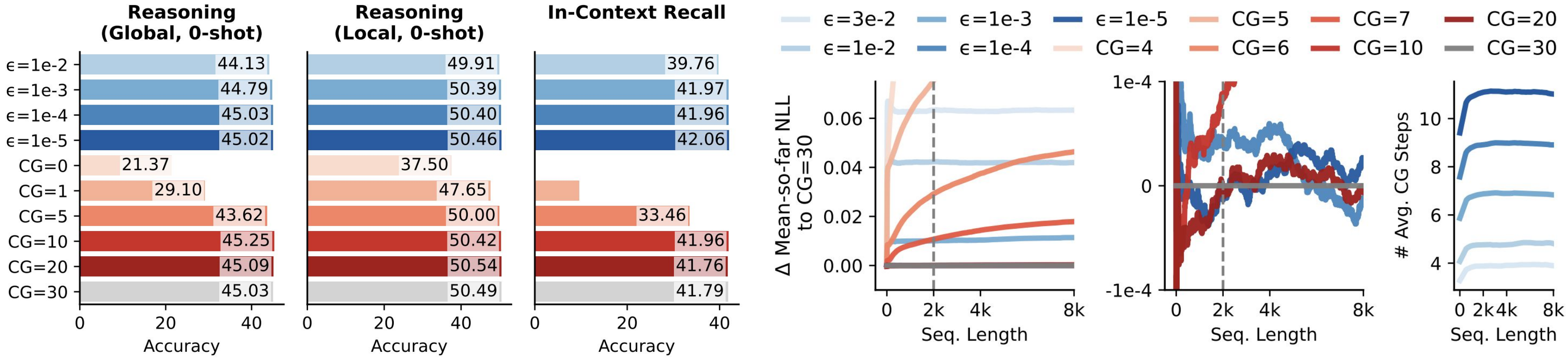
→ MesaNet exceed SWA-1024

MesaNet | Downstream Benchmarks: Few-Shot Learning



→ **Word Scramble:** MesaNet > Transformer >= Other RNNs
→ **Translation:** Transformer >> MesaNet >= Other RNNs

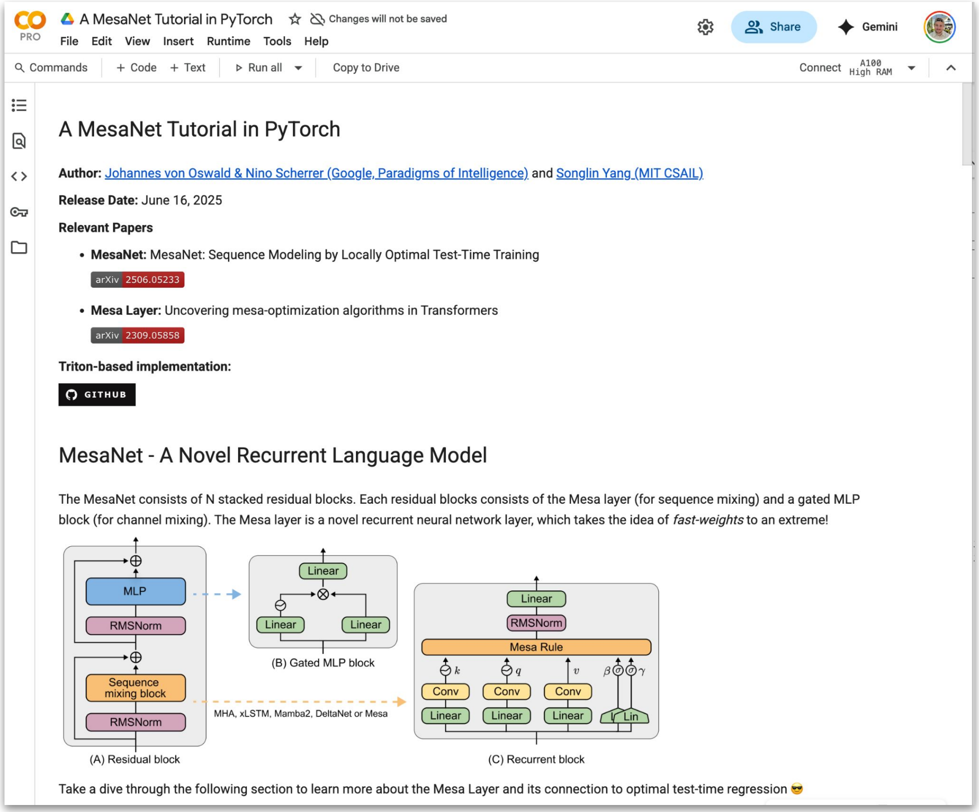
MesaNet | Inference Optimizations → Downstream Degradation



Two ways to reduce compute during inference:

1) Uniformly decrease CG steps
2) Apply a higher stopping criterion to end the CG method early

MesaNet | Colab Tutorial (GPU-based)



A MesaNet Tutorial in PyTorch
File Edit View Insert Runtime Tools Help
Changes will not be saved
Share Gemini
Commands + Code + Text Run all Copy to Drive
Connect A100 High RAM

A MesaNet Tutorial in PyTorch

Author: Johannes von Oswald & Nino Scherrer (Google, Paradigms of Intelligence) and Songlin Yang (MIT CSAIL)

Release Date: June 16, 2025

Relevant Papers

- **MesaNet:** MesaNet: Sequence Modeling by Locally Optimal Test-Time Training
  arXiv 2506.05233
- **Mesa Layer:** Uncovering mesa-optimization algorithms in Transformers
  arXiv 2309.05858

Triton-based implementation:

GITHUB

MesaNet - A Novel Recurrent Language Model

The MesaNet consists of N stacked residual blocks. Each residual blocks consists of the Mesa layer (for sequence mixing) and a gated MLP block (for channel mixing). The Mesa layer is a novel recurrent neural network layer, which takes the idea of *fast-weights* to an extreme!

(A) Residual block (B) Gated MLP block (C) Recurrent block

Take a dive through the following section to learn more about the Mesa Layer and its connection to optimal test-time regression 😎

Try it out :)

Thank you! Floor is Open for Questions :)

Google 2024-10-16

Uncovering mesa-optimization algorithms in Transformers

Johannes von Oswald[a,b,*], Maximilian Schlegel[a,b,*], Alexander Meulemans[a,b], Seijin Kobayashi[a,b], Eyvind Niklasson[a], Nicolas Zucchet[b], Nino Scherrer[a], Nolan Miller[d], Mark Sandler[d], Blaise Agüera y Arcas[a], Max Vladymyrov[d], Razvan Pascanu[e] and João Sacramento[a,b,*]
[a]Google, Paradigms of Intelligence Team, [b]ETH Zürich, [d]Google Research, [e]Google DeepMind, [*]Contributed equally to this work.

https://arxiv.org/abs/2309.05858

MesaNet: Sequence Modeling by Locally Optimal Test-Time Training

Johannes von Oswald[*], Nino Scherrer[*],
Seijin Kobayashi, Luca Versari, Songlin Yang[1], Maximilian Schlegel, Kaitlin Maile,
Yanick Schimpf, Oliver Sieberling[2], Alexander Meulemans, Rif A. Saurous, Guillaume Lajoie,
Charlotte Frenkel, Razvan Pascanu[3], Blaise Agüera y Arcas , and João Sacramento

https://arxiv.org/abs/2506.05233

Feel free to reach out:

Johannes von Oswald: jvoswald@google.com
Nino Scherrer: scherrernino@google.com