Dynamic Sparsity in Machine Learning:

Routing Information through Neural Pathways

André Martins and Edoardo Ponti

NeurIPS Tutorial, Vancouver, Canada

December 10, 2024

Tutorial Information


Presenters: André F. T. Martins and Edoardo M. Ponti.



Panellists: Sara Hooker, Alessandro Sordoni, and Zita Marinho.

https://dynamic-sparsity.github.io/

Outline

What Is Dynamic Sparsity?

Broad definition: any kind of sparse computation or memory compression where the sparsity pattern is input-dependent or adaptive.

- Mostly used with neural networks, most work after 2015
- Predictions triggered by a few neurons only (sparse activations)
- Can induce neural pathways (ReLUs, sparse attention, sparse MoEs, ...)
- Often inspired by sparse modeling techniques (e.g., ℓ_1 regularization).

What Is Dynamic Sparsity?

Broad definition: any kind of sparse computation or memory compression where the sparsity pattern is input-dependent or adaptive.

- Mostly used with neural networks, most work after 2015
- Predictions triggered by a few neurons only (sparse activations)
- Can induce neural pathways (ReLUs, sparse attention, sparse MoEs, ...)
- Often inspired by sparse modeling techniques (e.g., ℓ_1 regularization).

Contrast with *static* sparsity, such as:

- Model pruning (sparsity pattern **fixed** after training)
- Sparse attention with **fixed** masks (not input-dependent)
- **Fixed** routing.

AI vs Human Brain

Current AI systems:

- $\sim$100B-1T parameters
- A single H100 GPU consumes 700W.

AI vs Human Brain

Current AI systems:

- ∼100B-1T parameters
- A single H100 GPU consumes 700W.

Human brain:

- ∼100B neurons, 200T synapses (“parameters”)
- Only 20W power consumption: **highly sparse** and **highly parallel**!

Why Do We Care about Dynamic Sparsity?

It can be useful for:

- Computational efficiency
- Interpretability (selection of input features or circuits)
- Function specialization (e.g. sparse mixture-of-experts)
- Compositional generalization (e.g. mixture-of-adapters)
- Handling of long context information.

Where Can We Use Dynamic Sparsity?

Can be applied to different parts of the model:

- In tokens (*e.g.*, sparse memories / KV cache, memory compression)
- In attention heads (*e.g.*, adaptively sparse transformers)
- In layers (*e.g.*, mixture-of-depths, early-exit, ...)
- In expert modules (*e.g.*, sparse mixture-of-experts, mixture-of-adapters) to select active components of a model
- In output labels (*e.g.*, to generate confidence sets)

What This Tutorial Is About

We aim to provide a unified perspective on dynamic sparsity, covering:

- Methods and principles to induce dynamic sparsity
- Structured forms of sparsity
- Applications to routing and conditional computation
- Sparsity for efficiency and memory reduction.

What This Tutorial Is Not About

- Static sparsity (although we'll refer to it occasionally)
- Model pruning
- Sparsity in representations (e.g. sparse autoencoders)
- Hardware supporting sparse operations
- Neuromorphic computing.

Some of these topics are better covered in other tutorials.

Related Tutorials

- **Deep Implicit Layers** by Zico Kolter, David Duvenaud, and Matt Johnson ▸ NeurIPS 2020
- **Sparsity in Deep Learning** by Torsten Hoefler and Dan Alistarh ▸ ICML 2021
- **Modular and Parameter-Efficient Fine-Tuning for NLP Models** by Sebastian Ruder, Jonas Pfeiffer, Ivan Vulić ▸ EMNLP 2022
- **Mixture-of-Experts in the Era of LLMs** by Tianlong Chen, Yu Cheng, Beidi Chen, Minjia Zhang, Mohit Bansal ▸ ICML 2024

Tutorial Slides and Materials

Check out our tutorial webpage:

https://dynamic-sparsity.github.io/

Jupyter notebooks for:

- Sparse transformations
- Sparse Hopfield networks
- Sparse mixture-of-experts
- Sparse memory
- Mixture-of-adapters

This webpage will stay live after the tutorial.

Outline

What We'll Cover in This Part

A set of transformations that induce **sparsity** and **structure**:

- All differentiable (efficient forward and backward propagation)
- Can be used at hidden (attention) or output layers (loss).

What We'll Cover in This Part

A set of transformations that induce **sparsity** and **structure**:

- All differentiable (efficient forward and backward propagation)
- Can be used at hidden (attention) or output layers (loss).

We denote the probability simplex by

$$\triangle = \{\boldsymbol{p} \geq 0 \ : \ 1^\top \boldsymbol{p} = 1\}.$$

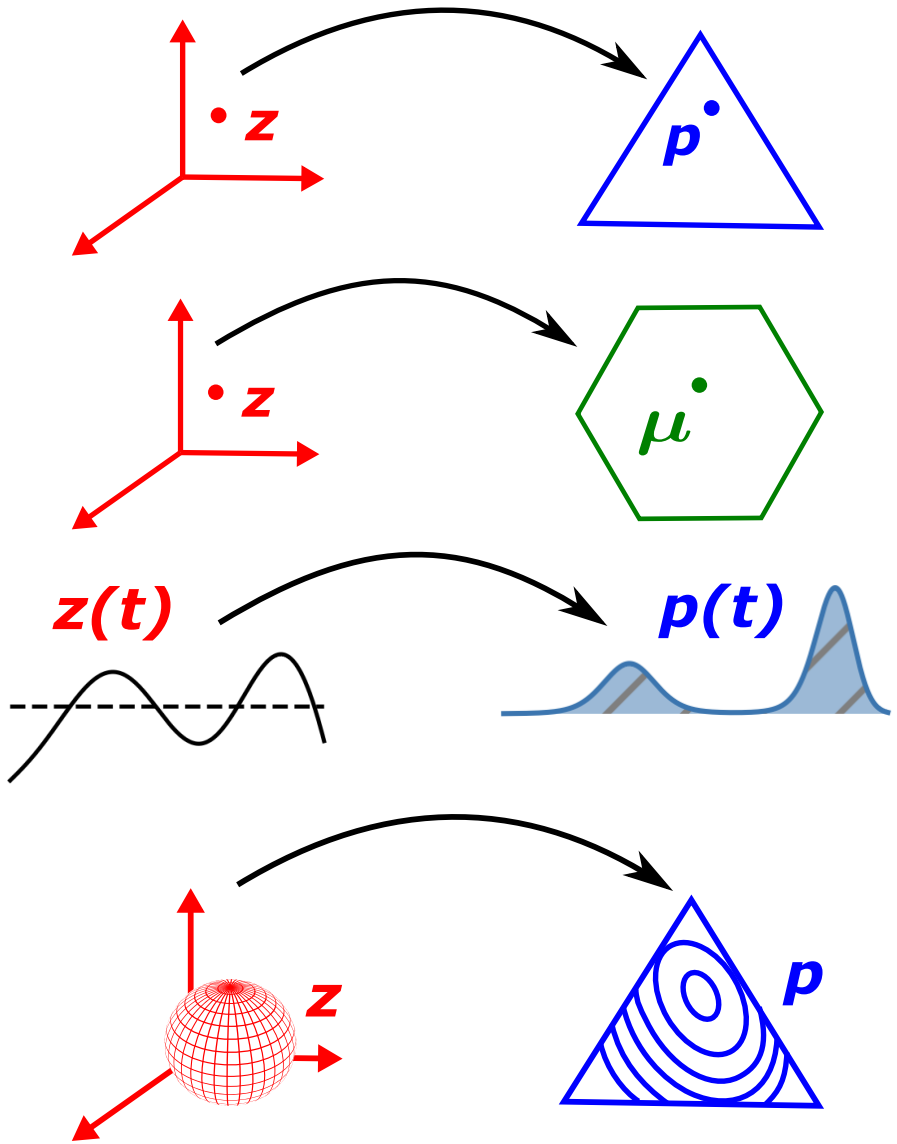
Set of vectors which are non-negative and normalized ("probability vectors").

Sparse

Structured

Continuous

Stochastic

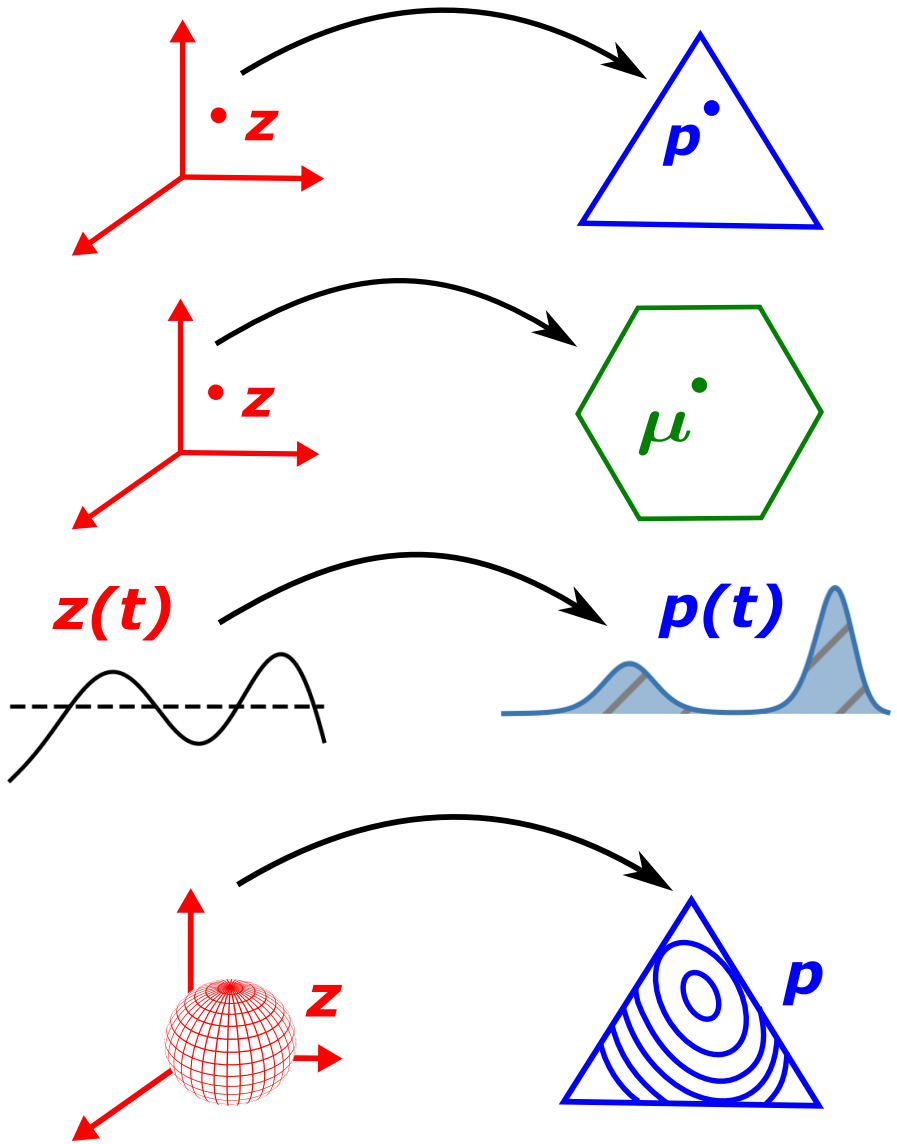


Sparse

Structured

Continuous

Stochastic



Outline

Rectified Linear Unit (Glorot et al., 2011, AISTATS)

The simplest sparse activation, $\text{relu}(\boldsymbol{z}) := [\boldsymbol{z}]_+ = \max(0, \boldsymbol{z})$.

In variational form,

$$\text{relu}(\boldsymbol{z}) = \arg\min_{\boldsymbol{y} \geq 0} \|\boldsymbol{y} - \boldsymbol{z}\|^2.$$

(Euclidean projection onto non-negative orthant.)

Non-negative, but not normalized.

Rectified Linear Unit (Glorot et al., 2011, AISTATS)

The simplest sparse activation, $\text{relu}(\boldsymbol{z}) := [\boldsymbol{z}]_+ = \max(0, \boldsymbol{z})$.

In variational form,

$$\text{relu}(\boldsymbol{z}) = \arg\min_{\boldsymbol{y} \geq 0} \|\boldsymbol{y} - \boldsymbol{z}\|^2.$$

(Euclidean projection onto non-negative orthant.)

Non-negative, but not normalized.

What are commonly used normalized transformations?

Softmax and Argmax

Softmax exponentiates and normalizes:

$$\text{softmax}(\boldsymbol{z}) = \frac{\exp(\boldsymbol{z})}{\sum_{k=1}^{K} \exp(z_k)}$$

- Normalized but not sparse: $\text{softmax}(\boldsymbol{z}) > 0, \forall \boldsymbol{z}$

Softmax and Argmax

Softmax exponentiates and normalizes:

$$\text{softmax}(\boldsymbol{z}) = \frac{\exp(\boldsymbol{z})}{\sum_{k=1}^{K} \exp(z_k)}$$

- Normalized but not sparse: $\text{softmax}(\boldsymbol{z}) > 0, \forall \boldsymbol{z}$

Argmax retrieves a **one-hot vector** for the highest scored index:

$$\begin{aligned} \text{argmax}(\boldsymbol{z}) &:= \arg\max_{\boldsymbol{p} \in \triangle} \boldsymbol{z}^\top \boldsymbol{p} \\ &= \lim_{\tau \to 0^+} \text{softmax}(\boldsymbol{z}/\tau) \qquad \text{(temperature trick)} \end{aligned}$$

- Sparse but zero gradient.

Softmax and Argmax

Softmax exponentiates and normalizes:

$$\text{softmax}(\boldsymbol{z}) = \frac{\exp(\boldsymbol{z})}{\sum_{k=1}^{K} \exp(z_k)}$$

- Normalized but not sparse: $\text{softmax}(\boldsymbol{z}) > 0, \forall \boldsymbol{z}$

Argmax retrieves a **one-hot vector** for the highest scored index:

$$\begin{aligned} \text{argmax}(\boldsymbol{z}) &:= \arg\max_{\boldsymbol{p} \in \triangle} \boldsymbol{z}^\top \boldsymbol{p} \\ &= \lim_{\tau \to 0^+} \text{softmax}(\boldsymbol{z}/\tau) \qquad \text{(temperature trick)} \end{aligned}$$

- Sparse but zero gradient.

Softmax with low temperature $\tau > 0$ is only *approximately* sparse and gradient decays exponentially fast.

Shortcomings of Softmax: Dispersion

`softmax` is not enough (for sharp out-of-distribution)

Petar Veličković
Google DeepMind

Christos Perivolaropoulos
Google DeepMind

Federico Barbero*
University of Oxford

Razvan Pascanu
Google DeepMind

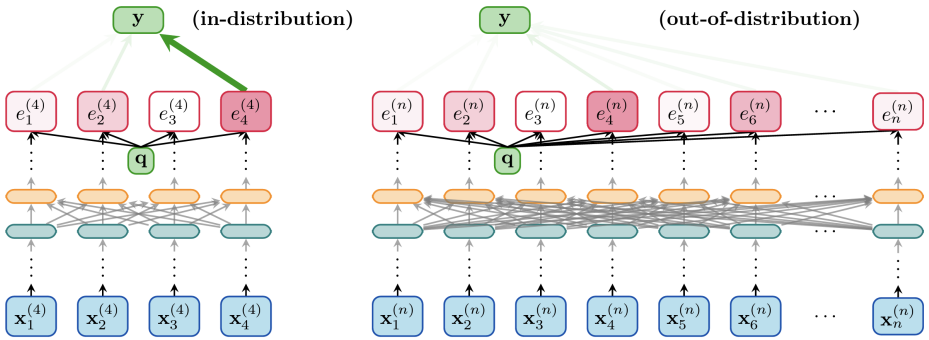
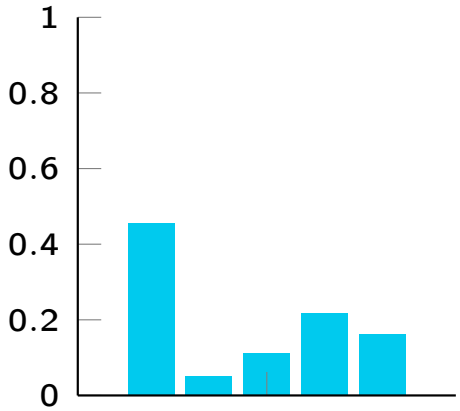


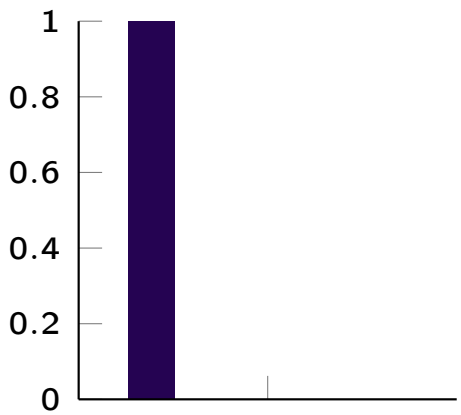
Figure 1: Illustration of Theorem 2.2, one of our key results. Assuming a tokenised input from a fixed vocabulary and a non-zero temperature, for every `softmax` attention head inside an architecture comprising only **MLPs** and **`softmax` self-attention layers**, it must hold that, given sufficiently many **tokens**, its **attention coefficients** will *disperse*, even if they were sharp for in-distribution instances.

Softmax has a dispersion effect when context gets bigger.

softmax($\boldsymbol{z}$)



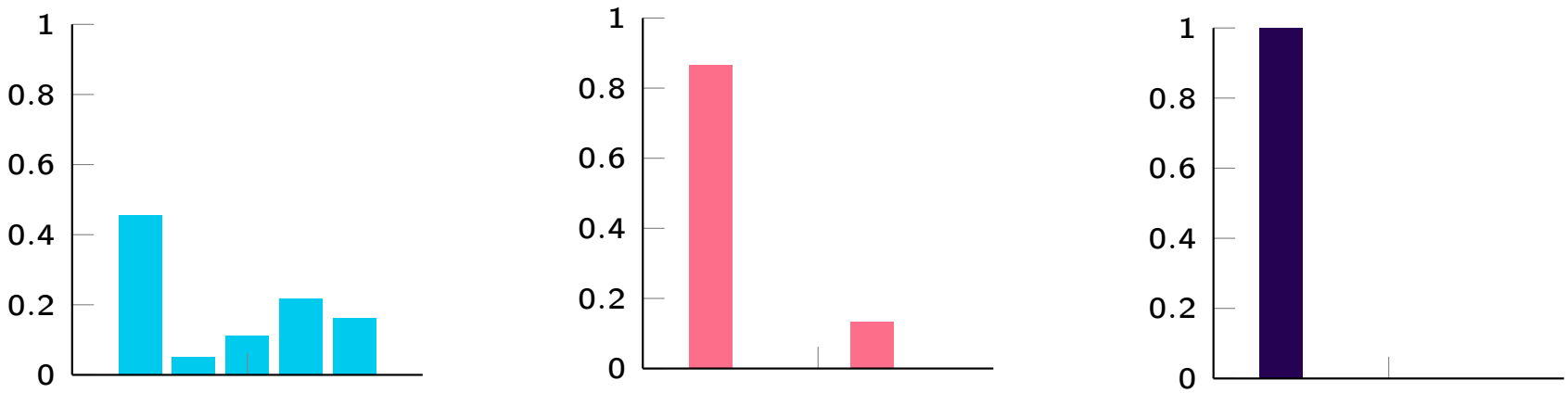
argmax($\boldsymbol{z}$)



(Same $\boldsymbol{z} = [1.0716, -1.1221, -0.3288, 0.3368, 0.0425]$)

- Argmax is an extreme case of sparsity, but it is **discontinuous**.
- Is there a **sparse** and **differentiable** alternative?

softmax($\boldsymbol{z}$) ? argmax($\boldsymbol{z}$)



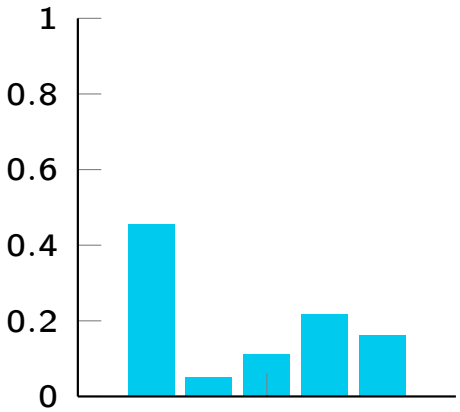
(Same $\boldsymbol{z} = [1.0716, -1.1221, -0.3288, 0.3368, 0.0425]$)

- Argmax is an extreme case of sparsity, but it is **discontinuous**.
- Is there a **sparse** and **differentiable** alternative?

One Possibility: top-k softmax

Take softmax, keep only the top-k probabilities, and renormalize.

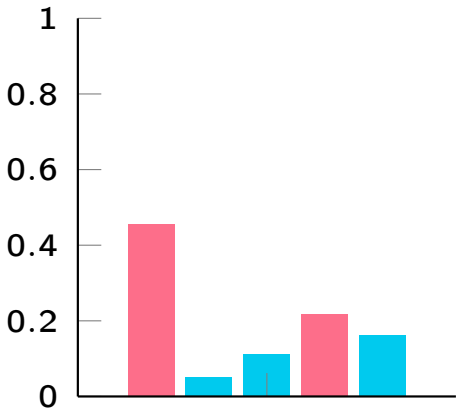
Commonly used for routing in sparse MoEs (we'll talk about this later).



One Possibility: top-k softmax

Take softmax, keep only the top-k probabilities, and renormalize.

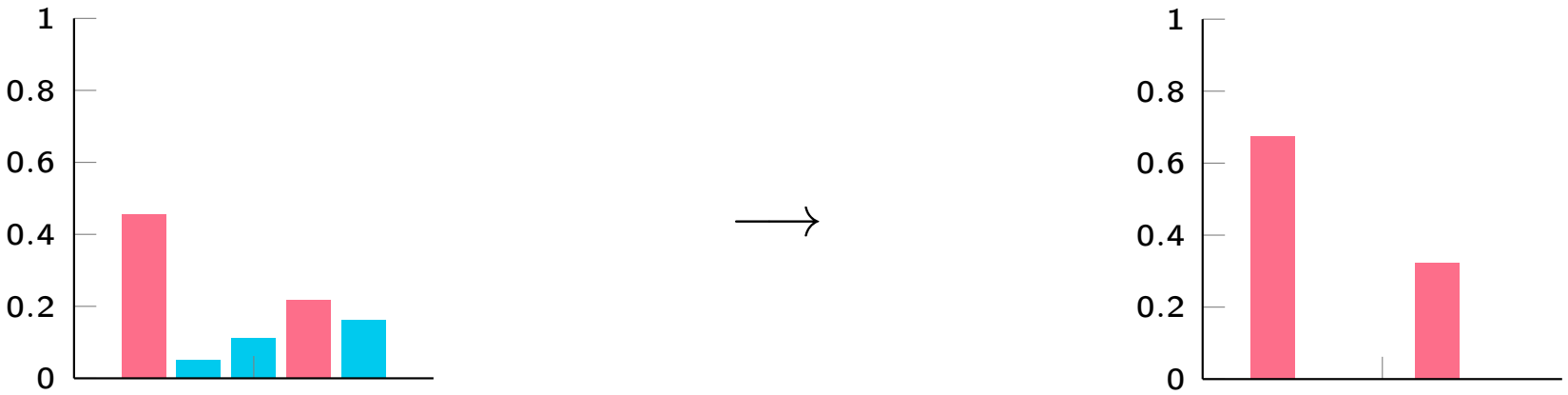
Commonly used for routing in sparse MoEs (we'll talk about this later).



One Possibility: top-k softmax

Take softmax, keep only the top-k probabilities, and renormalize.

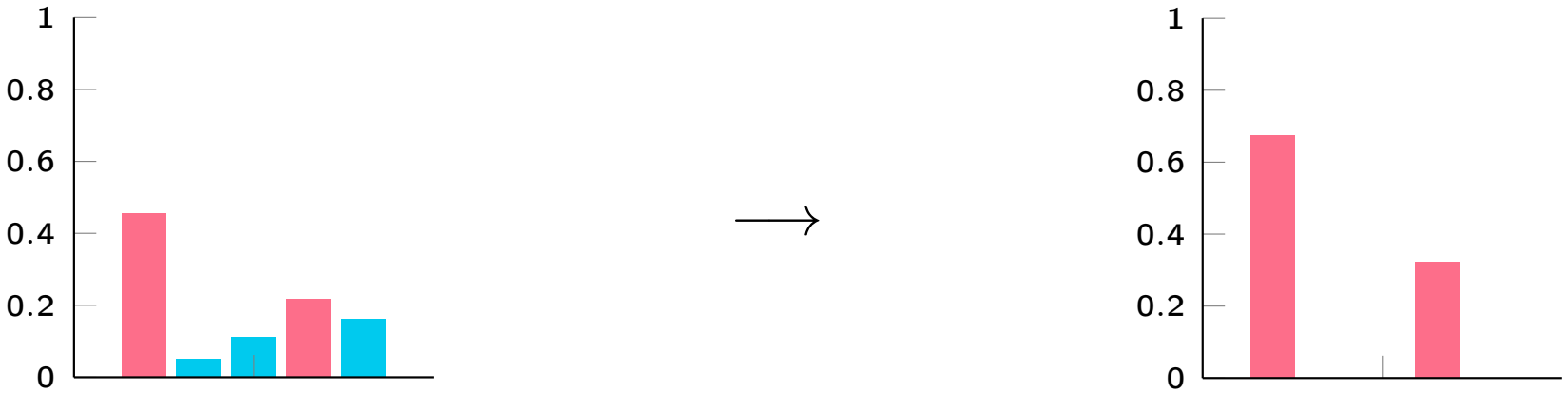
Commonly used for routing in sparse MoEs (we'll talk about this later).



One Possibility: top-k softmax

Take softmax, keep only the top-k probabilities, and renormalize.

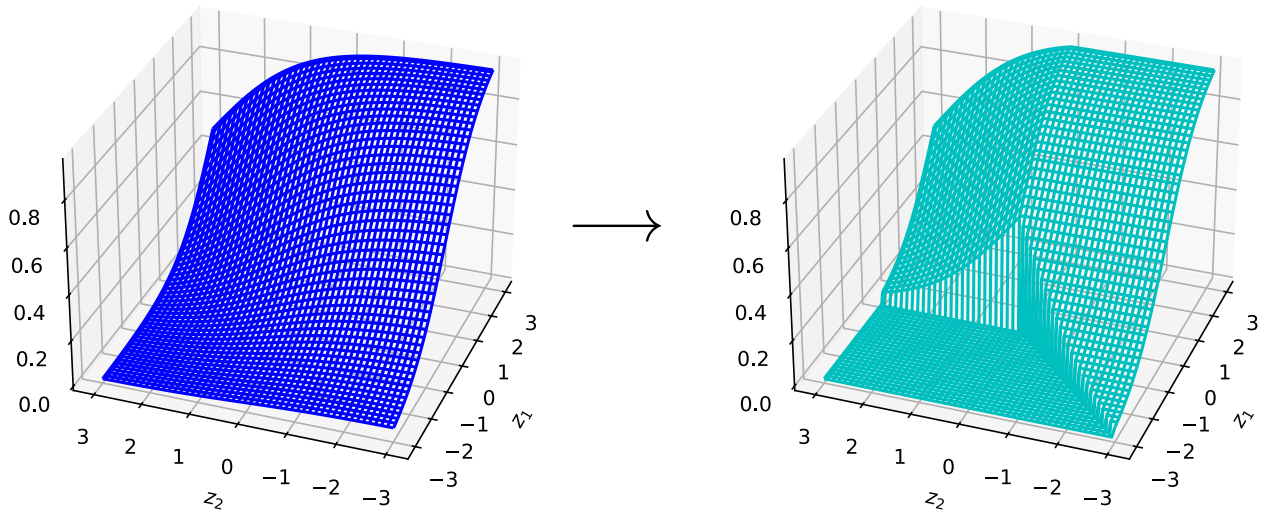
Commonly used for routing in sparse MoEs (we'll talk about this later).



Disadvantages:

- k is fixed.
- Discontinuities.

(example in 3D for $k = 2$)



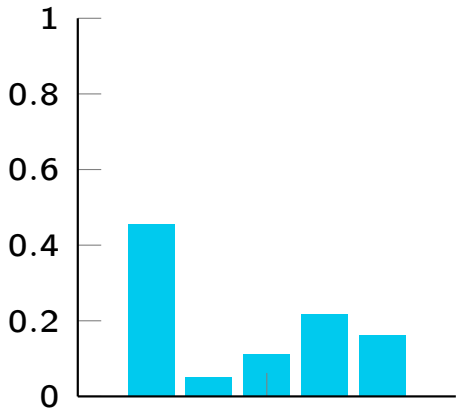
Another Possibility: Sparsemax (Martins and Astudillo, 2016, ICML)

Euclidean projection of $\boldsymbol{z}$ onto the probability simplex $\triangle$:

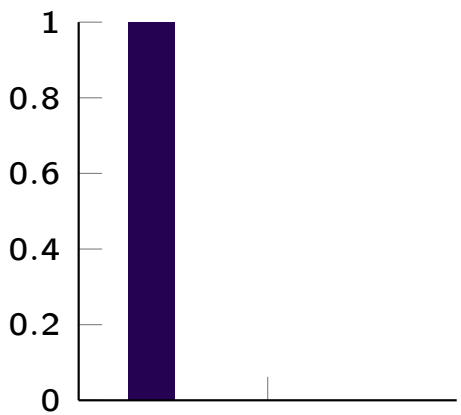
$$\begin{aligned} \text{sparsemax}(\boldsymbol{z}) &:= \arg\min_{\boldsymbol{p}\in\triangle} \|\boldsymbol{p}-\boldsymbol{z}\|^2 \\ &= \arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{z}^\top \boldsymbol{p} - \frac{1}{2}\|\boldsymbol{p}\|^2. \end{aligned}$$

- Likely to hit the boundary of the simplex, in which case $\text{sparsemax}(\boldsymbol{z})$ becomes sparse (hence the name)
- End-to-end differentiable
- Forward pass: $O(K \log K)$ or $O(K)$, **(almost) as fast as softmax**
- Backprop: sublinear, **better than softmax!**

softmax($\boldsymbol{z}$)

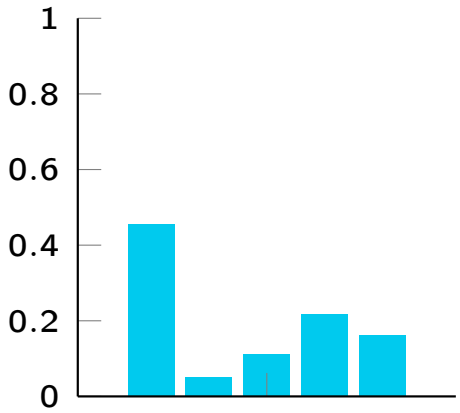


argmax($\boldsymbol{z}$)

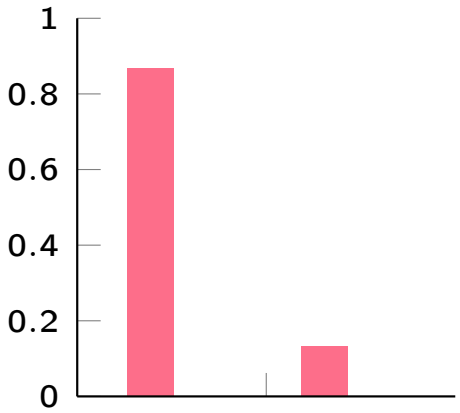


(Same $\boldsymbol{z} = [1.0716, -1.1221, -0.3288, 0.3368, 0.0425]$)

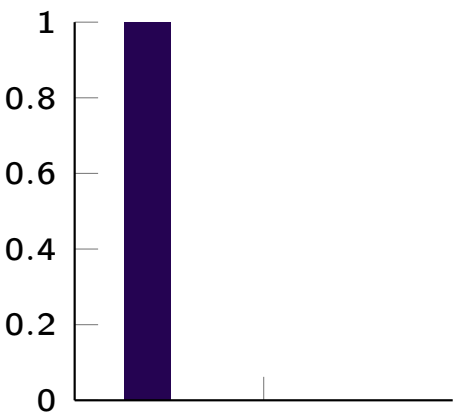
softmax($\boldsymbol{z}$)



sparsemax($\boldsymbol{z}$)



argmax($\boldsymbol{z}$)



(Same $\boldsymbol{z} = [1.0716, -1.1221, -0.3288, 0.3368, 0.0425]$)

Sparsemax as "Normalized" ReLU

ReLU in variational form is

$$\text{relu}(\boldsymbol{z}) = \arg\min_{\boldsymbol{y}\geq 0} \|\boldsymbol{y} - \boldsymbol{z}\|^2.$$

Non-negative, but not normalized.

Sparsemax is

$$\text{sparsemax}(\boldsymbol{z}) = \arg\min_{\substack{\boldsymbol{y}\geq 0 \\ 1^\top \boldsymbol{y}=1}} \|\boldsymbol{y} - \boldsymbol{z}\|^2.$$

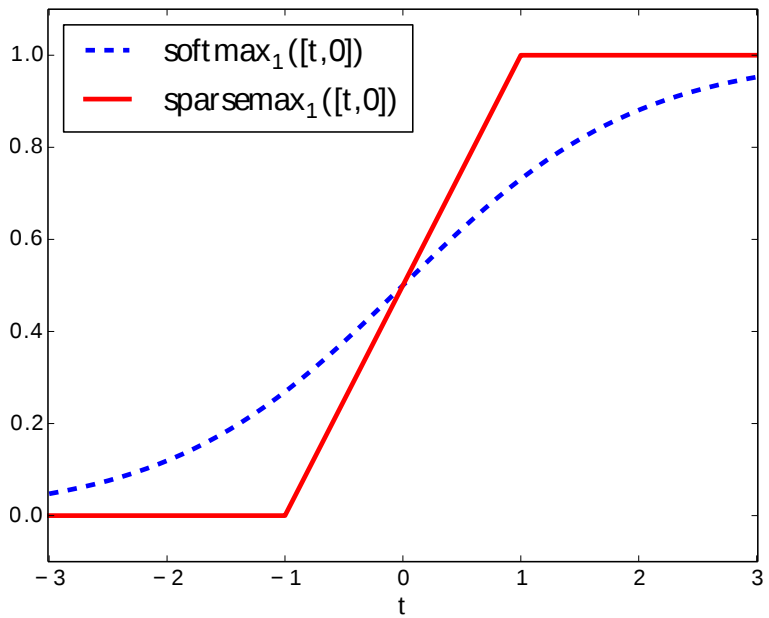
Adds normalization constraint.

Softmax vs sparsemax in 2D

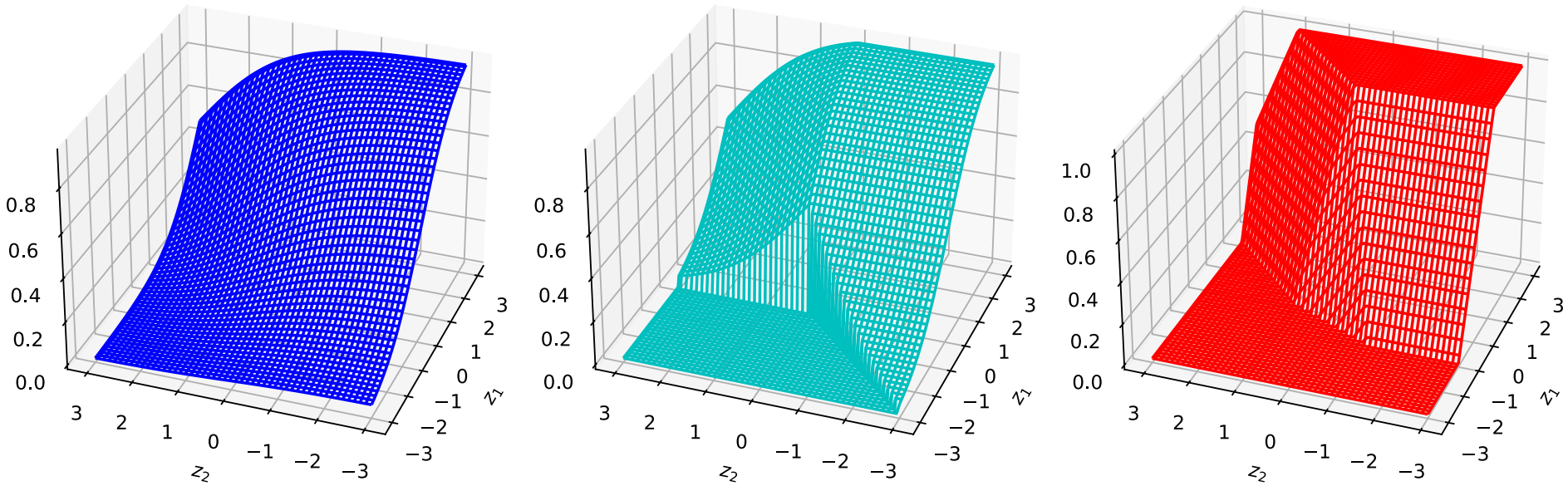
- Parametrize $\boldsymbol{z} = (t, 0)$
- The 2D softmax is the logistic (sigmoid) function:

$$\text{softmax}_1(\boldsymbol{z}) = (1 + \exp(-t))^{-1}$$

- The 2D sparsemax is the "hard" version of the sigmoid:

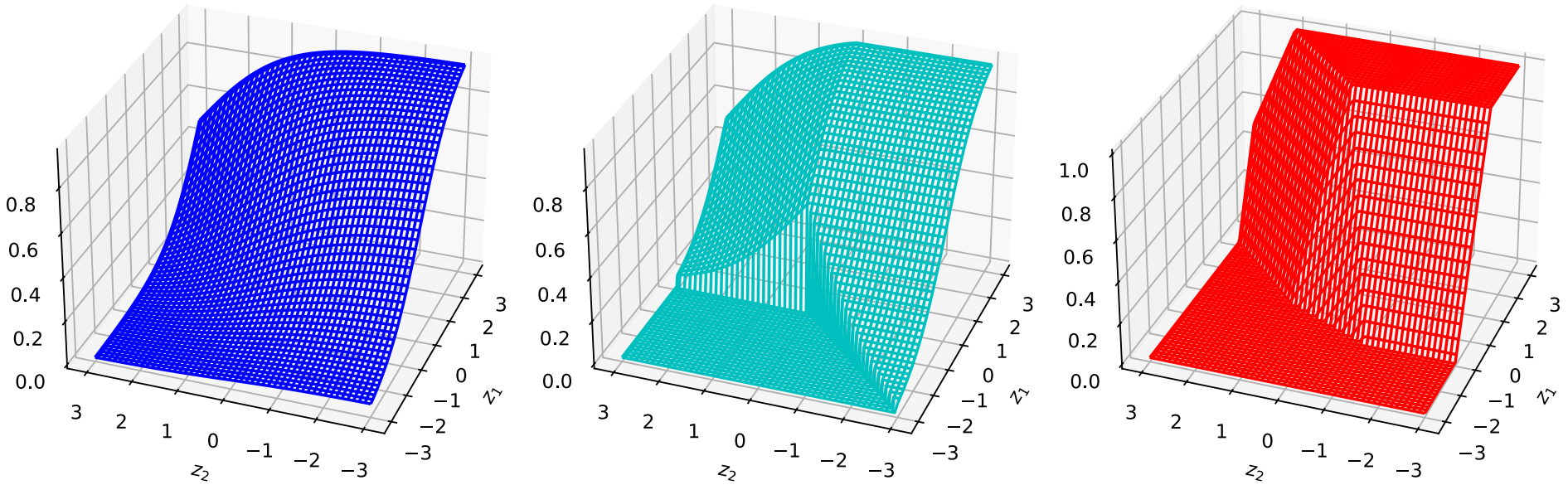


Softmax vs top-k softmax vs sparsemax in 3D



- Sparsemax does not fix k and it is not discontinuous.
- Sparsemax is piecewise linear, but asymptotically similar to softmax.

Softmax vs top-k softmax vs sparsemax in 3D



- Sparsemax does not fix k and it is not discontinuous.
- Sparsemax is piecewise linear, but asymptotically similar to softmax.

Are there other sparse transformations?

Ω-Regularized Argmax (Niculae and Blondel, 2017, NeurIPS)

For convex Ω, define the **Ω-regularized argmax transformation**:

$$\operatorname{argmax}_\Omega(\boldsymbol{z}) := \arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{z}^\top \boldsymbol{p} - \Omega(\boldsymbol{p})$$

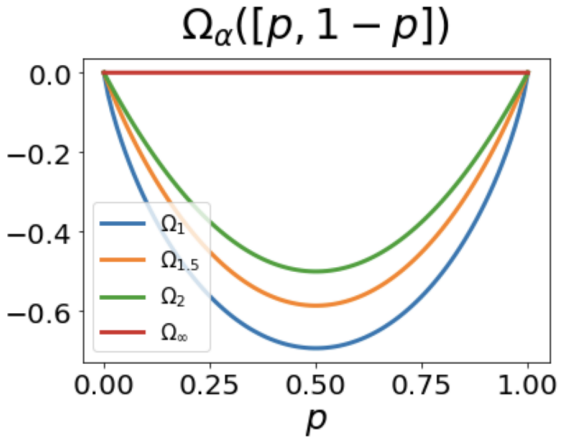
- **Argmax** corresponds to **no regularization**, $\Omega \equiv 0$
- **Softmax** amounts to **entropic regularization**, $\Omega(\boldsymbol{p}) = \sum_{i=1}^K p_i \log p_i$
- **Sparsemax** amounts to **ℓ_2-regularization**, $\Omega(\boldsymbol{p}) = \frac{1}{2}\|\boldsymbol{p}\|^2$

In general, we can let Ω be a generalized negative entropy (negentropy).

Entmax (Tsallis, 1988; Peters et al., 2019, ACL)

Tsallis α-negentropies ($\alpha \geq 0$):

$$\Omega_\alpha(\boldsymbol{p}) := \begin{cases} \frac{1}{\alpha(\alpha-1)}\left(\sum_{i=1}^K p_i^\alpha - 1\right) & \text{if } \alpha \neq 1 \\ \sum_{i=1}^K p_i \log p_i & \text{if } \alpha = 1. \end{cases}$$



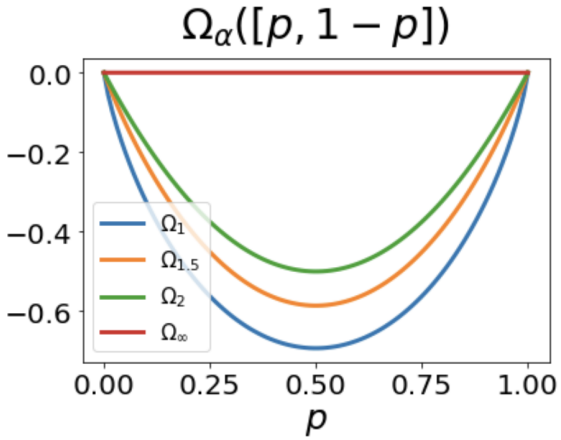
With $\text{argmax}_{\Omega_\alpha}$ we recover:

- **Softmax** for $\alpha = 1$, **sparsemax** for $\alpha = 2$, **argmax** for $\alpha \to \infty$.

Entmax (Tsallis, 1988; Peters et al., 2019, ACL)

Tsallis α-negentropies ($\alpha \geq 0$):

$$\Omega_\alpha(\boldsymbol{p}) := \begin{cases} \frac{1}{\alpha(\alpha-1)}\left(\sum_{i=1}^K p_i^\alpha - 1\right) & \text{if } \alpha \neq 1 \\ \sum_{i=1}^K p_i \log p_i & \text{if } \alpha = 1. \end{cases}$$



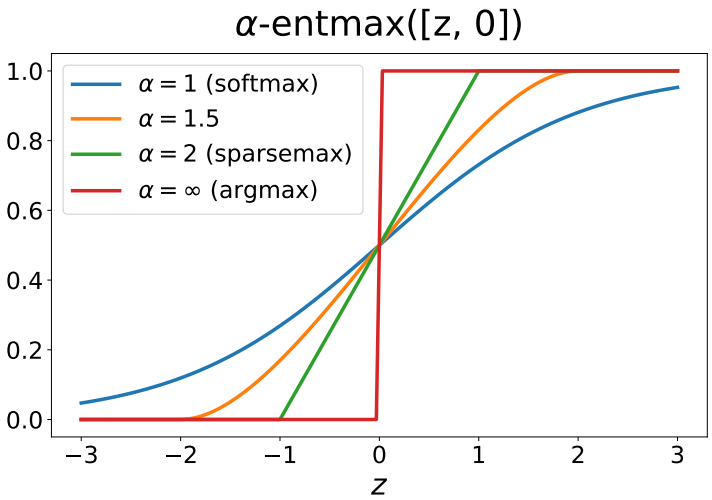
With $\text{argmax}_{\Omega_\alpha}$ we recover:

- **Softmax** for $\alpha = 1$, **sparsemax** for $\alpha = 2$, **argmax** for $\alpha \to \infty$.

Always sparse for $\alpha > 1$, sparsity increases with α.

- Forward pass for general α can be done with a bissection algorithm
- Faster algorithm for $\alpha = 1.5$
- Backward pass runs in sublinear time.

Entmax in 2D (Peters et al., 2019, ACL)



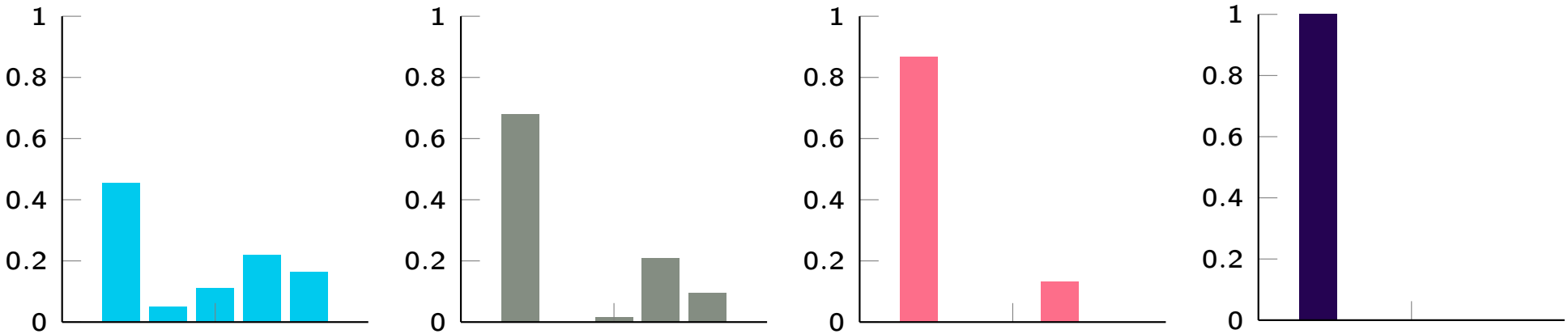

```
!pip install entmax

import torch
from torch.nn.functional import softmax
from entmax import sparsemax, entmax15, entmax_bisect

z = torch.tensor([-2.0, 0.0, 0.5])

print(softmax(z, dim=0))
print(entmax_bisect(z, alpha=1.2, dim=0))
print(entmax15(z, dim=0))
print(sparsemax(z, dim=0))
```

Check out the Jupyter notebook on our tutorial webpage!

Entmax Transformations (Peters et al., 2019, ACL)

$\alpha = 1$ | $\alpha = 1.5$ | $\alpha = 2$ | $\alpha = \infty$

softmax($\boldsymbol{z}$) | 1.5-entmax($\boldsymbol{z}$) | sparsemax($\boldsymbol{z}$) | argmax($\boldsymbol{z}$)

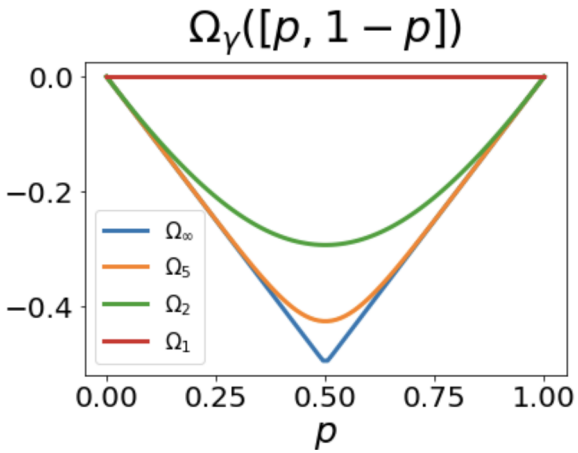


(Same $\boldsymbol{z} = [1.0716, -1.1221, -0.3288, 0.3368, 0.0425]$)

Normmax (Blondel et al., 2020, JMLR) (Santos et al., 2024, ICML)

Norm γ-negentropies ($\gamma \geq 1$):

$$\Omega_\gamma(\boldsymbol{p}) := \|\boldsymbol{p}\|_\gamma - 1 = \left(\sum_i p_i^\gamma\right)^{1/\gamma} - 1.$$

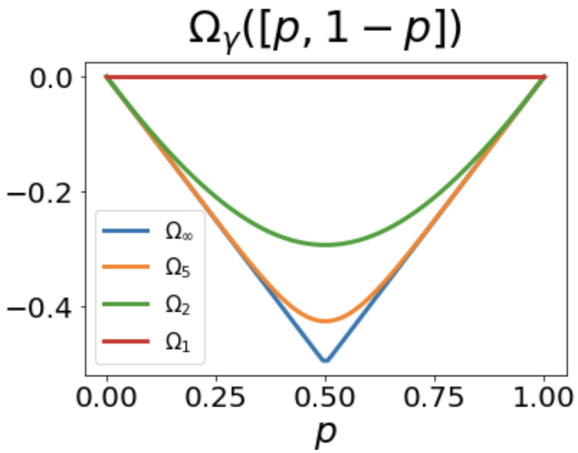


$\gamma \to +\infty \implies$ Berger-Parker dominance index.

Normmax (Blondel et al., 2020, JMLR) (Santos et al., 2024, ICML)

Norm γ-negentropies ($\gamma \geq 1$):

$$\Omega_\gamma(\boldsymbol{p}) := \|\boldsymbol{p}\|_\gamma - 1 = \left(\sum_i p_i^\gamma\right)^{1/\gamma} - 1.$$

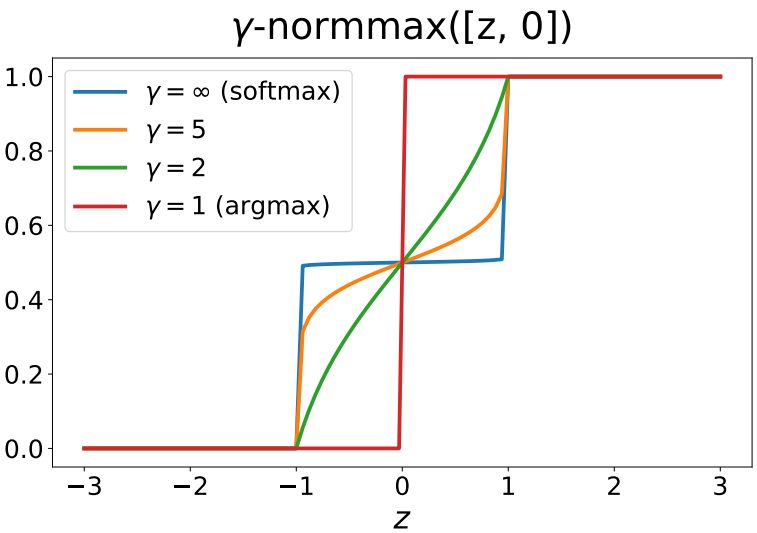


$\gamma \to +\infty \implies$ Berger-Parker dominance index.

What do we get with $\text{argmax}_{\Omega_\gamma}$?

- **Argmax** corresponds to $\gamma = 1$
- When $\gamma \to +\infty$, encourages **sparse** but **uniform** $\boldsymbol{p}$
- Forward pass for general γ can be done with a bissection algorithm
- Backward pass runs in sublinear time.

Normmax in 2D

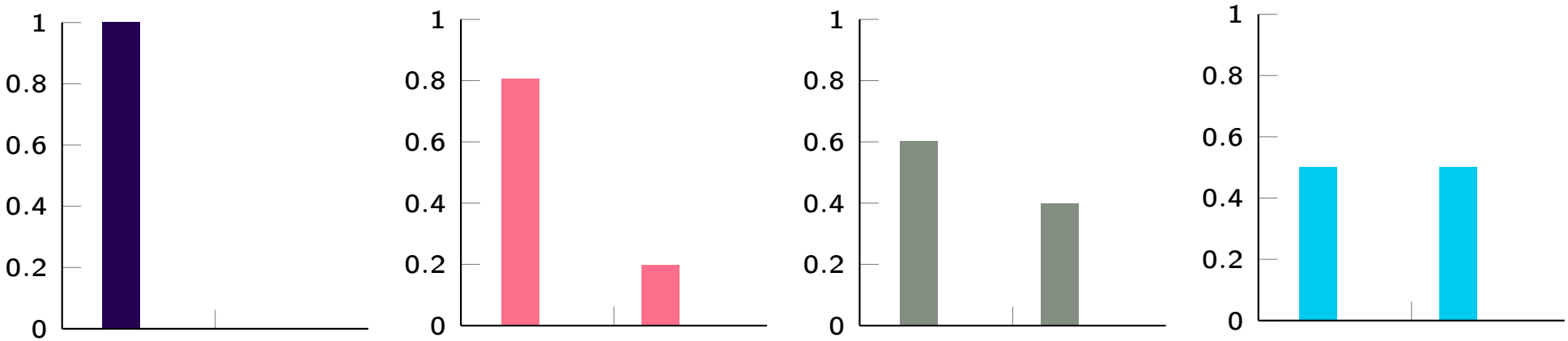



```
!pip install entmax

import torch
from torch.nn.functional import softmax
from entmax import normmax_bisect

z = torch.tensor([-2.0, 0.0, 0.5])

print(normmax_bisect(z, alpha=1, dim=0))
print(normmax_bisect(z, alpha=2, dim=0))
print(normmax_bisect(z, alpha=5, dim=0))
print(normmax_bisect(z, alpha=1000, dim=0))
```

Check out the Jupyter notebook on our tutorial webpage!

Normmax Transformations

$\gamma = 1$	$\gamma = 2$	$\gamma = 5$	$\gamma \to \infty$
argmax($\boldsymbol{z}$)	2-normmax($\boldsymbol{z}$)	5-normmax($\boldsymbol{z}$)	∞-normmax($\boldsymbol{z}$)



(Same $\boldsymbol{z} = [1.0716, -1.1221, -0.3288, 0.3368, 0.0425]$)

Where do we use these sparse transformations?

- Sparse attention mechanisms
- Sparse losses

Where do we use these sparse transformations?

- Sparse attention mechanisms
- Sparse losses

Example: Sparse Attention for Machine Translation

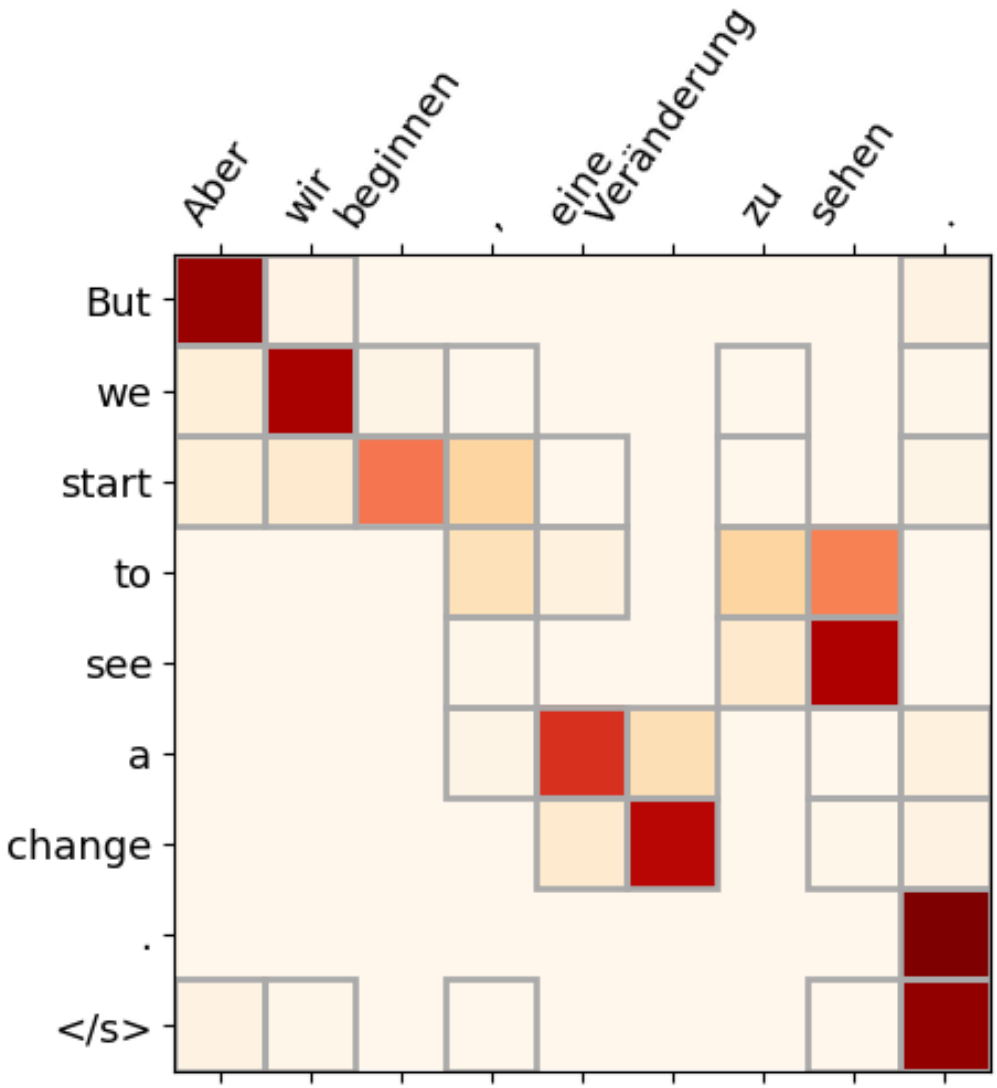
(Malaviya et al., 2018; Peters et al., 2019; Correia et al., 2019, ACL)

- Sparse alignments $\Longrightarrow$ better interpretability
- Other uses of sparse attention for XAI:

  (Treviso and Martins, 2020)

  (Pruthi et al., 2022, TACL)

  (Fernandes et al., 2022, NeurIPS)



Questions

For α-entmax:

- Which α to choose? Larger $\alpha \Longrightarrow$ higher propensity to sparsity
- What if we have many attention heads, and we don't know how sparse we want each one to be?
- Attention heads tend to have specialized functions **(Voita et al., 2019, ACL)**
- Adjusting sparsity per head adaptively can help them specialize

Can we learn α from data?

Questions

For α-entmax:

- Which α to choose? Larger $\alpha \Longrightarrow$ higher propensity to sparsity
- What if we have many attention heads, and we don't know how sparse we want each one to be?
- Attention heads tend to have specialized functions **(Voita et al., 2019, ACL)**
- Adjusting sparsity per head adaptively can help them specialize

Can we learn α from data? Yes!

Adaptively Sparse Transformers

(Correia et al., 2019, EMNLP)

Most attempts to sparsify transformers use static sparsity patterns (Child et al., 2019; Beltagy et al., 2020).

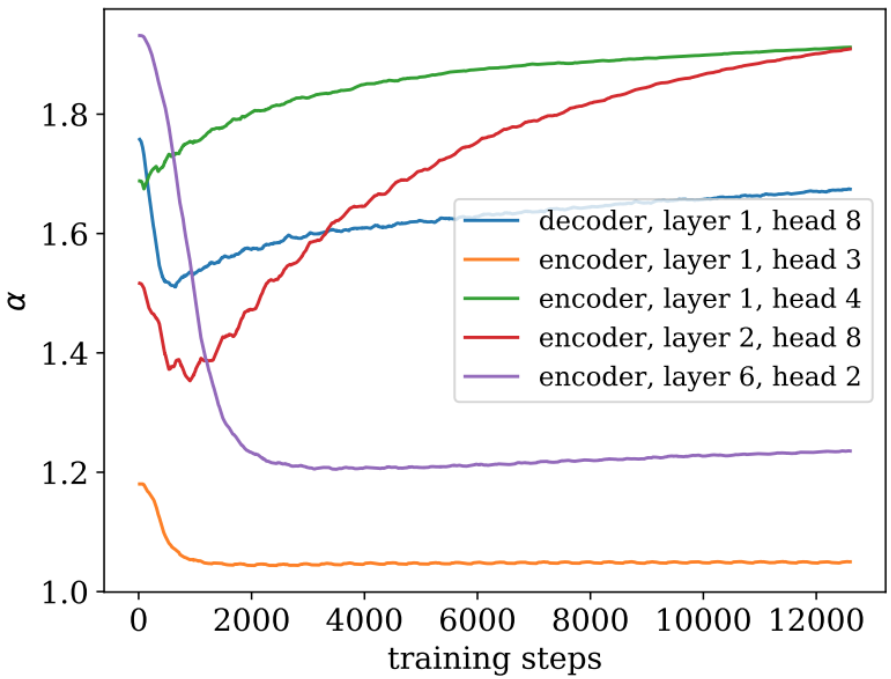
However we may benefit from dynamic (input-dependent) sparsity.

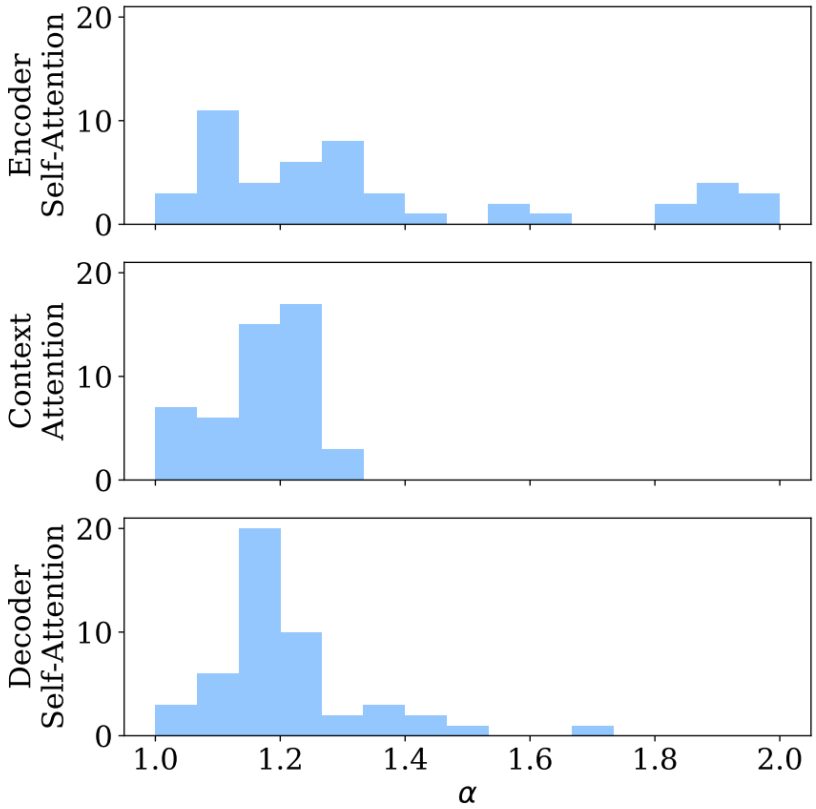
Example: replace softmax in attention heads by α-entmax!

Allows **non-contiguous** attention for each head, learned **adaptively**.

- Recall: α controls propensity to sparsity
- Learn each $\alpha \in [1, 2]$ **adaptively**!
- One α for each attention head and each layer.
- Can compute the gradient w.r.t. α!

Trajectories of α during MT Training (Correia et al., 2019, EMNLP)





Most heads become denser in the beginning, before converging.

Dense attention more beneficial while the network is still uncertain, becomes sparser as the network learns.

Where do we use these sparse transformations?

- Sparse attention mechanisms ✓
- Sparse losses

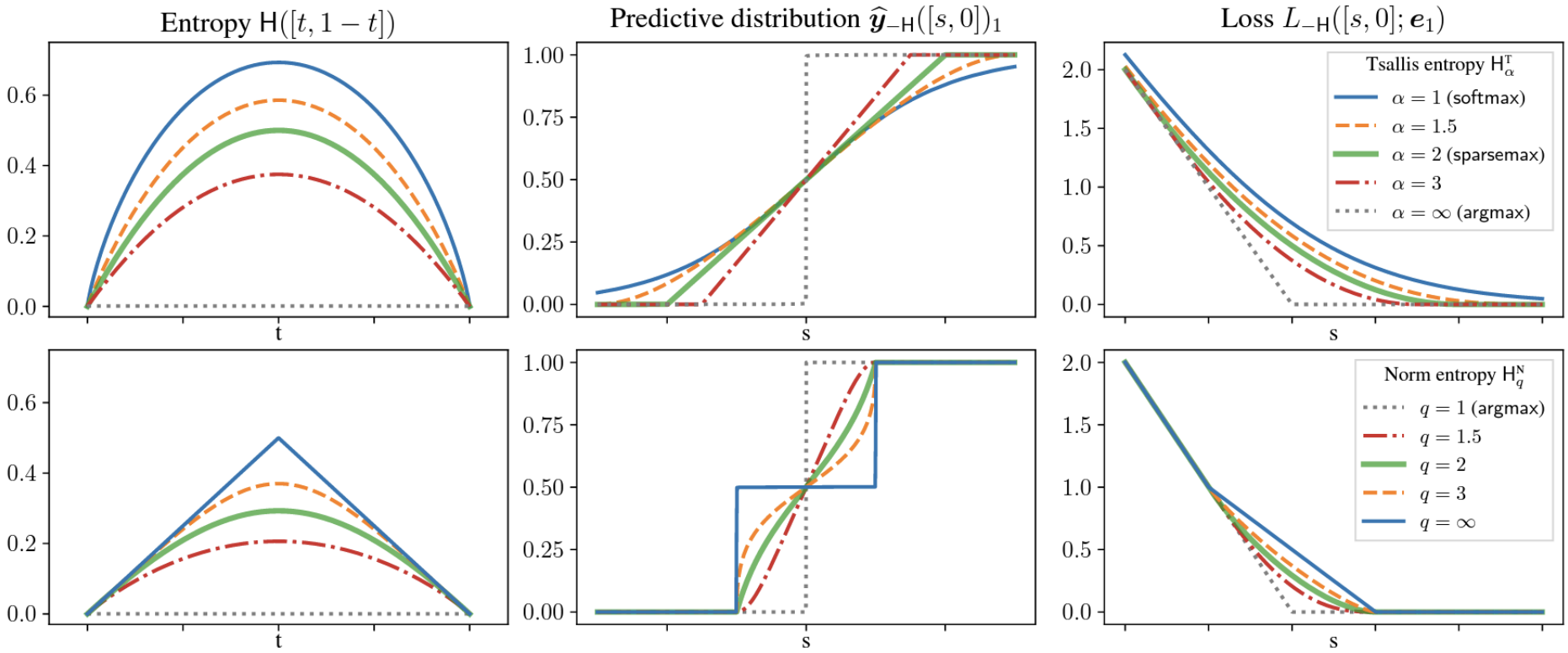
Where do we use these sparse transformations?

- Sparse attention mechanisms ✓
- Sparse losses

How about Sparsity in the Output?

- Entmax and normmax can also be used as a loss in the **output layer** (to replace logistic/cross-entropy loss)
- However, not expressed as a log-likelihood (which could lead to $\log(0)$ problems due to sparsity)
- **Fenchel-Young losses** offer a solution to this problem.

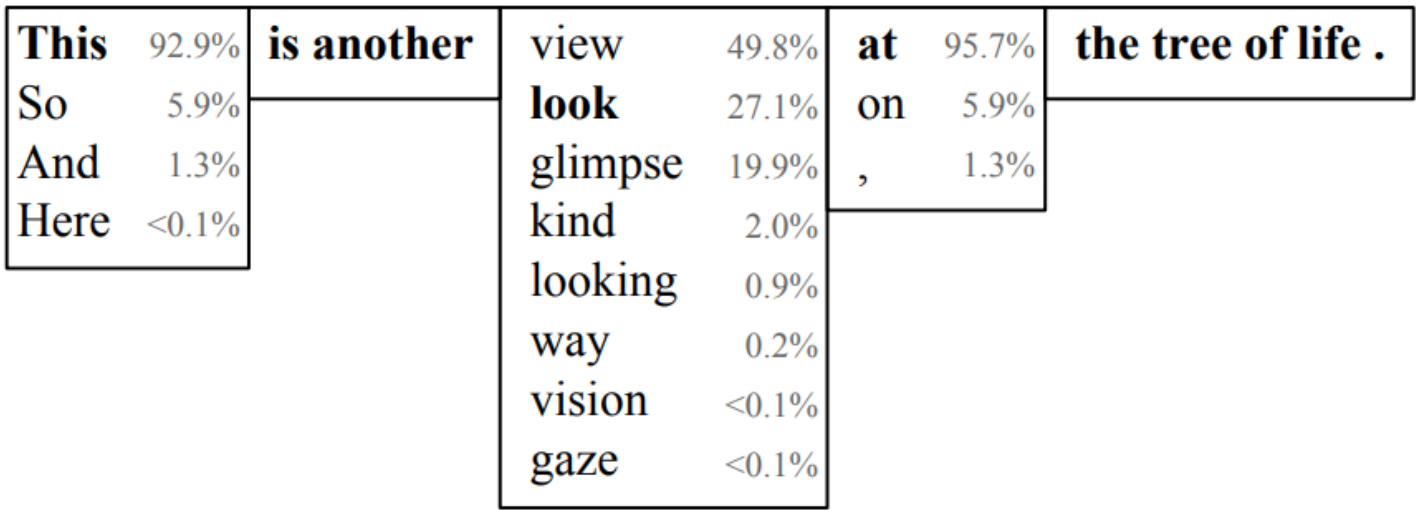
Fenchel-Young Losses (Blondel et al., 2020, JMLR)



- Entmax: for all $\alpha > 1$, transformations are **sparse** and losses have **margins**!
- Normmax: same for $\gamma \geq 1$
- The **margin size** is related to the **slope** of the entropy in the simplex corners! ($\frac{1}{\alpha - 1}$ for entmax losses; 1 for normmax losses.)

Example: Machine Translation

(Peters et al., 2019, ACL) (Peters and Martins, 2021, NAACL)



(Source: "Dies ist ein weiterer Blick auf den Baum des Lebens.")

- Only a few words get non-zero probability at each time step
- Auto-completion when several words in a row have probability 1
- Useful for predictive translation.

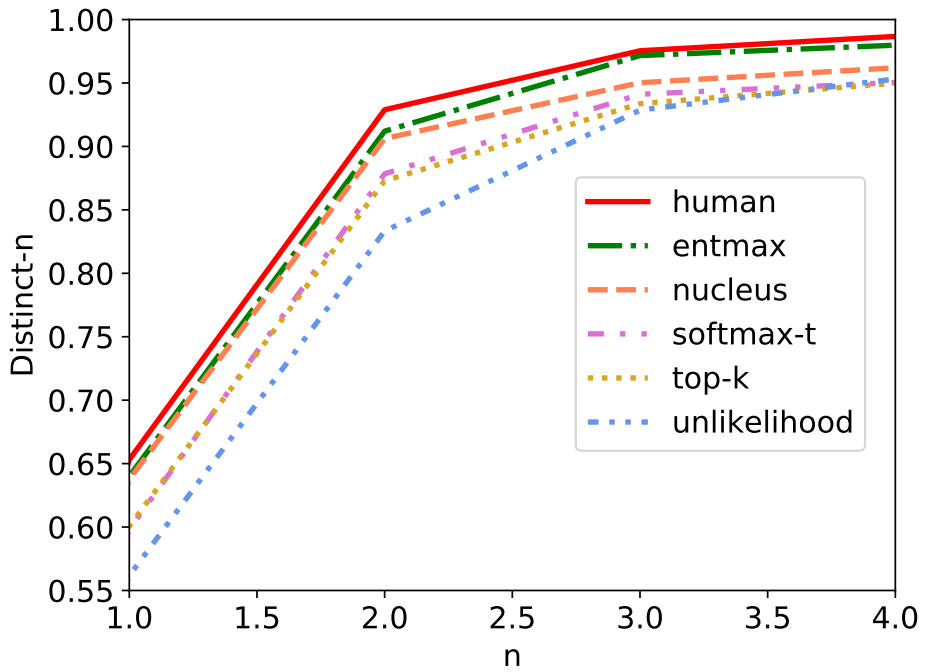
Entmax Sampling (Martins et al., 2020b, EMNLP)

Use the entmax loss for training language models.

At test time, sample from this sparse distribution.

No need for post-hoc truncation (top-k, nucleus, ...)

Better quality with less repetitions than other methods:

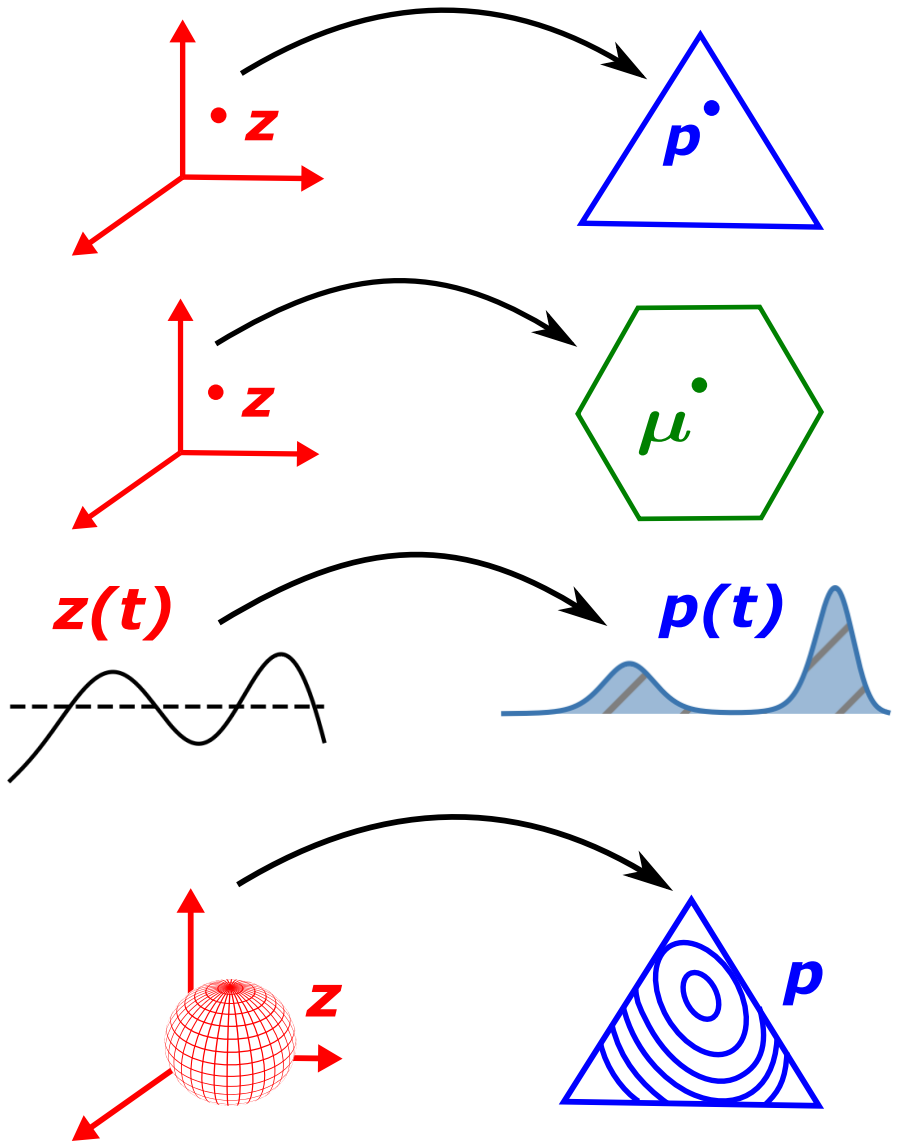


Sparse ✓

Structured

Continuous

Stochastic

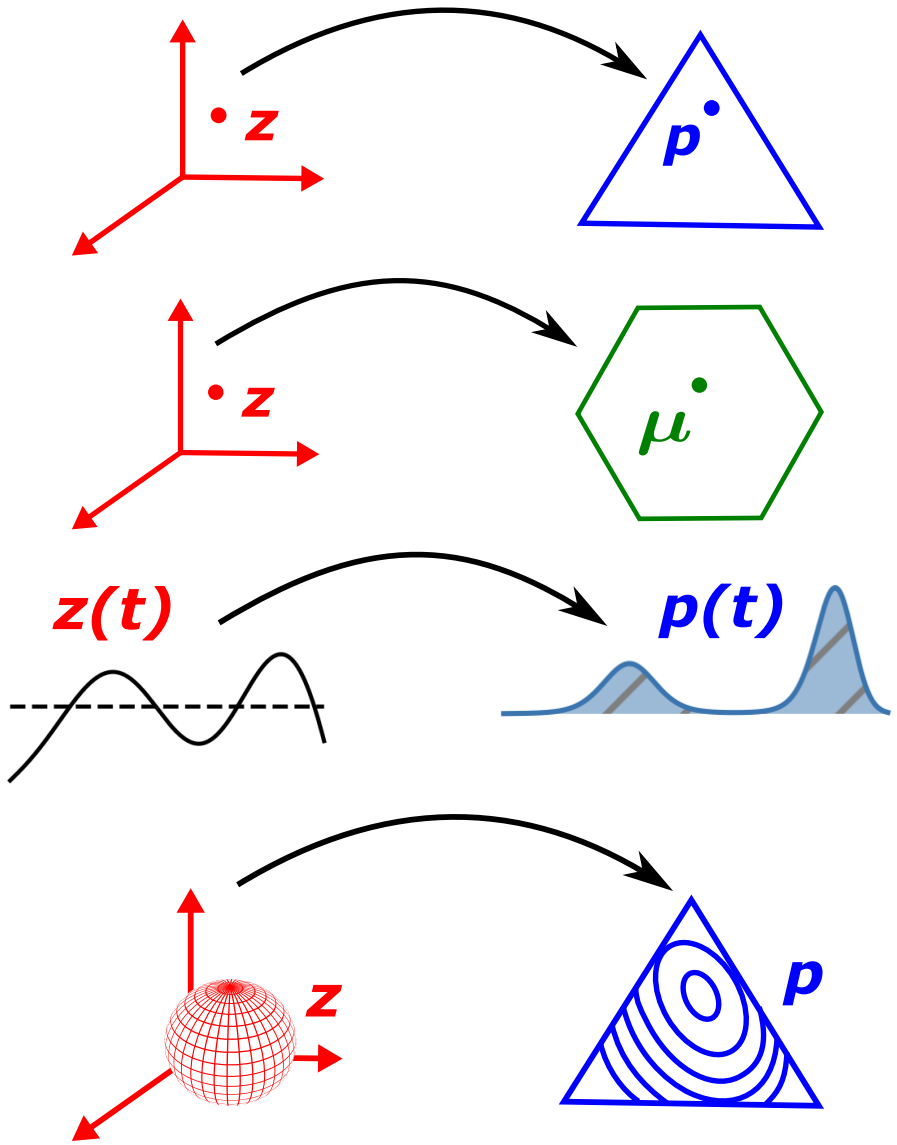


Sparse ✓

Structured

Continuous

Stochastic



Outline

Structured and Sparse Transformations

Two complementary ideas:

- Sparse modeling techniques (fused lasso, TV regularization, group lasso, etc.) can be adapted to induce structured dynamic sparsity
- Structured prediction techniques (combinatorial/graphical model algorithms, dynamic programming) can be sparsified.

Structured / Constrained Sparsity

Constrained softmax/sparsemax

(Martins and Kreutzer, 2017, EMNLP) (Malaviya et al., 2018, ACL)

- Allows placing a **budget** on how much attention a word can receive
- Useful to avoid repetitions and to model **fertility** in machine translation

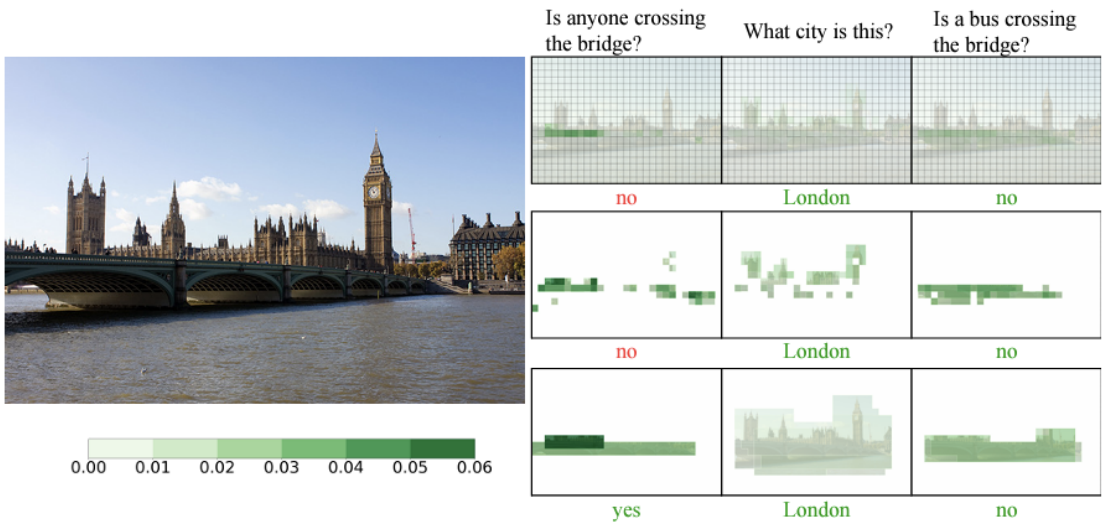
Fusedmax

(Niculae and Blondel, 2017, NeurIPS)

- Promotes contiguous selection (inspired by fused lasso)

TVmax (Martins et al., 2021)

- Promotes selection of compact regions in 2D images



Structured Prediction Techniques

Unstructured	Structured
argmax	MAP inference
softmax	Marginal inference
sparsemax	?

Structured Prediction Techniques

Unstructured	Structured
argmax	MAP inference
softmax	Marginal inference
sparsemax	?

Can we have differentiable and sparse structured prediction?

Differentiable Optimization as a Layer (Amos and Kolter, 2017, ICML)

We can integrate optimization problems as layers in neural nets

We saw examples before (sparsemax, entmax, normmax)

OptNet: a general architecture for QPs (sparsemax is a particular case)

Can incorporate structural constraints and output interdependencies

But how can we

- exploit the structure of the problem (e.g. factor graph, parse tree, matching structure, ...)?
- leverage existing combinatorial algorithms (e.g. dynamic programming)?

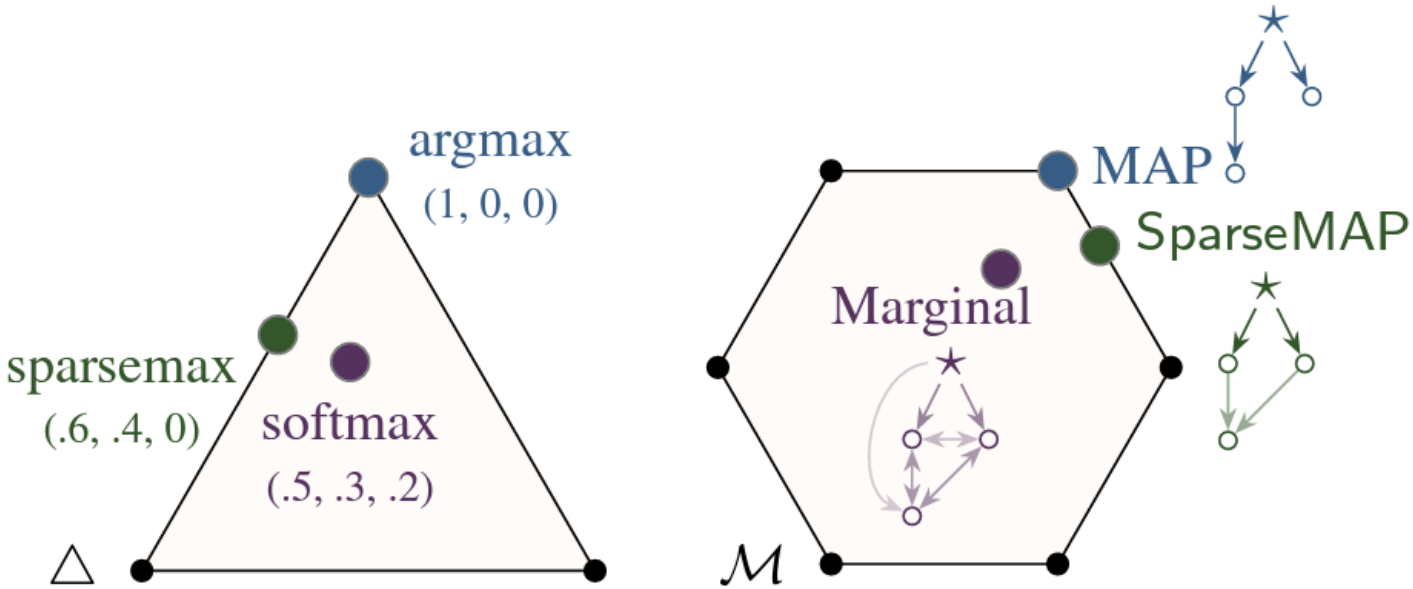
Structured Prediction Techniques

Unstructured	Structured
argmax	MAP inference
softmax	Marginal inference
sparsemax	?

Structured Prediction Techniques

Unstructured	Structured
argmax	MAP inference
softmax	Marginal inference
sparsemax	SparseMAP

SparseMAP (Niculae et al., 2018, ICML)



Projects onto the marginal polytope! (vertices correspond to structures)

Yields a **sparse combination of vertices**, hence it selects only a small number of structures (out of exponentially many)

Solvable as sequence of **MAP problems**; gradient comes for free.

Works both as output and hidden layer.

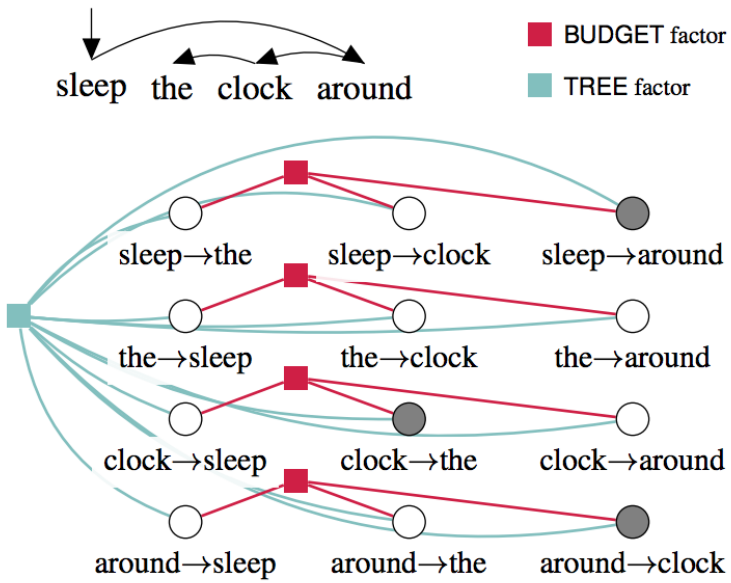
LP-SparseMAP (Niculae and Martins, 2020, ICML)

Extension to combinations of structured problems.

Can handle logic variables and constraints through a factor graph.

Returns **sparse** and **differentiable combination of structures**.

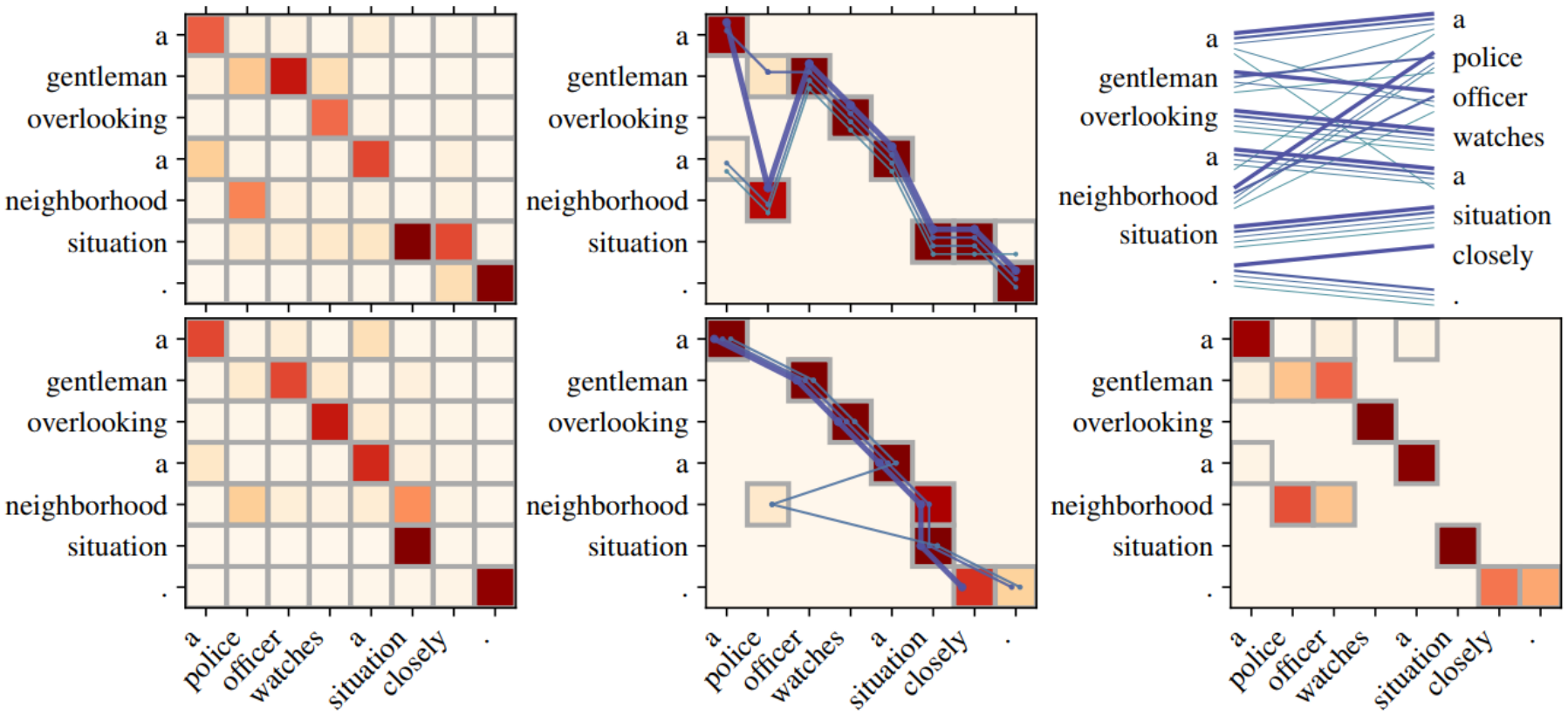
Efficient forward/backprop (requires only a MAP oracle).




```
fg = TorchFactorGraph()
u = fg.variable_from(arc_scores)
fg.add(DepTree(u))
for k in range(n):
    fg.add(Budget(u[:, k], budget=5))
fg.solve()
```

Example: Latent Structured Alignments in SNLI

(Niculae et al., 2018)



(a) softmax (b) sequence (c) matching

Example: Sparse and Structured Text Rationalization

(Guerreiro and Martins, 2021, EMNLP) (Treviso et al., 2023, ACL)

Use SparseMAP to generate rationales for model predictions.

- Budget constraint to ensure short rationales
- Sequential model to promote contiguous spans.

1 *an amber pour with hints of pink and yellow . fluffy head , good lacing .* smells of high citrus (gf , lemon) and some leafy flower plants . hops are in there somewhere . taste has the hops ; nice crispness and flavor medium body and great mouthfeel , leaves clean with enough taste residue to want more .

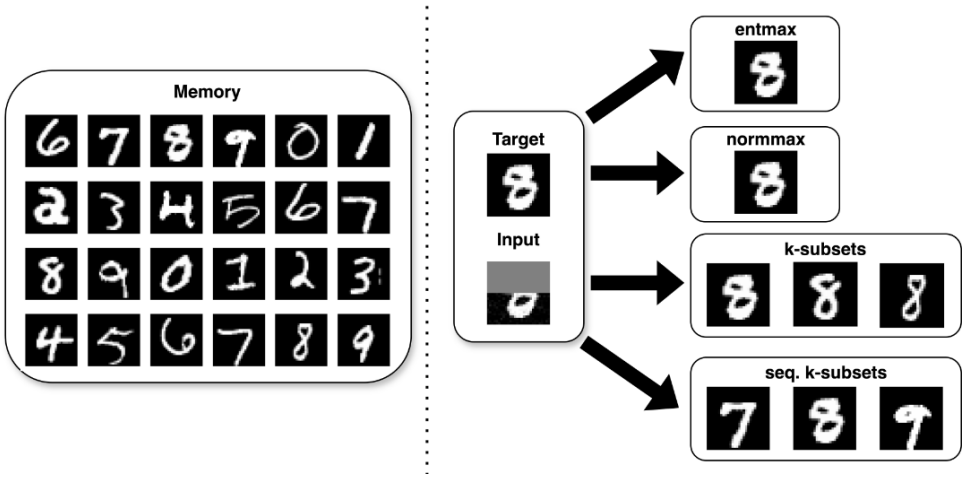
2 *hazy bright orange in color with a fluffy white head that quickly dissipates , leaving delicate lace . way too orange looking .* aroma is very mild wheat and subtle spice completely dominated by artificial orange . smells like tang . flavor ditto . tastes like an artificially flavored witte . there 's no way the orange flavor is authentic . mouthfeel is actually nice and creamy with that good wheaty quality . too bad it tastes like an orange soda .

Example: Sparse and Structured Associative Memories

(Hu et al., 2023, NeurIPS) (Wu et al., 2024, ICLR) (Santos et al., 2024, ICML)

- Neural associative memories (Amari, 1972; Nakano, 1972; Hopfield, 1982)
- By plugging sparse transformations in Hopfield-like energies we obtain exact retrieval with exponential memory capacity!
- With structured transformations (k-subsets, sequential k-subsets, ...), we can retrieve pattern associations instead of individual patterns.

Check out our Jupyter notebook!



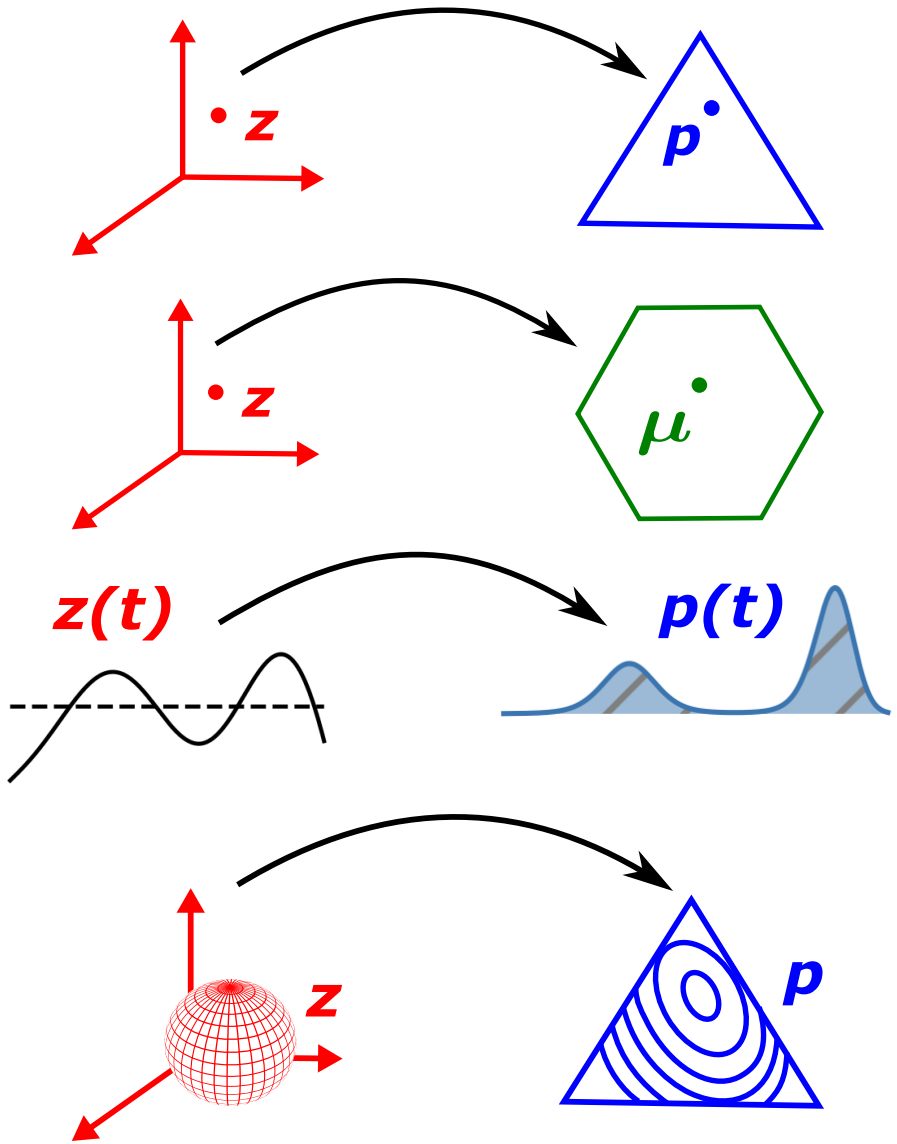
(Santos et al., 2024, ICML)

Sparse ✓

Structured ✓

Continuous

Stochastic

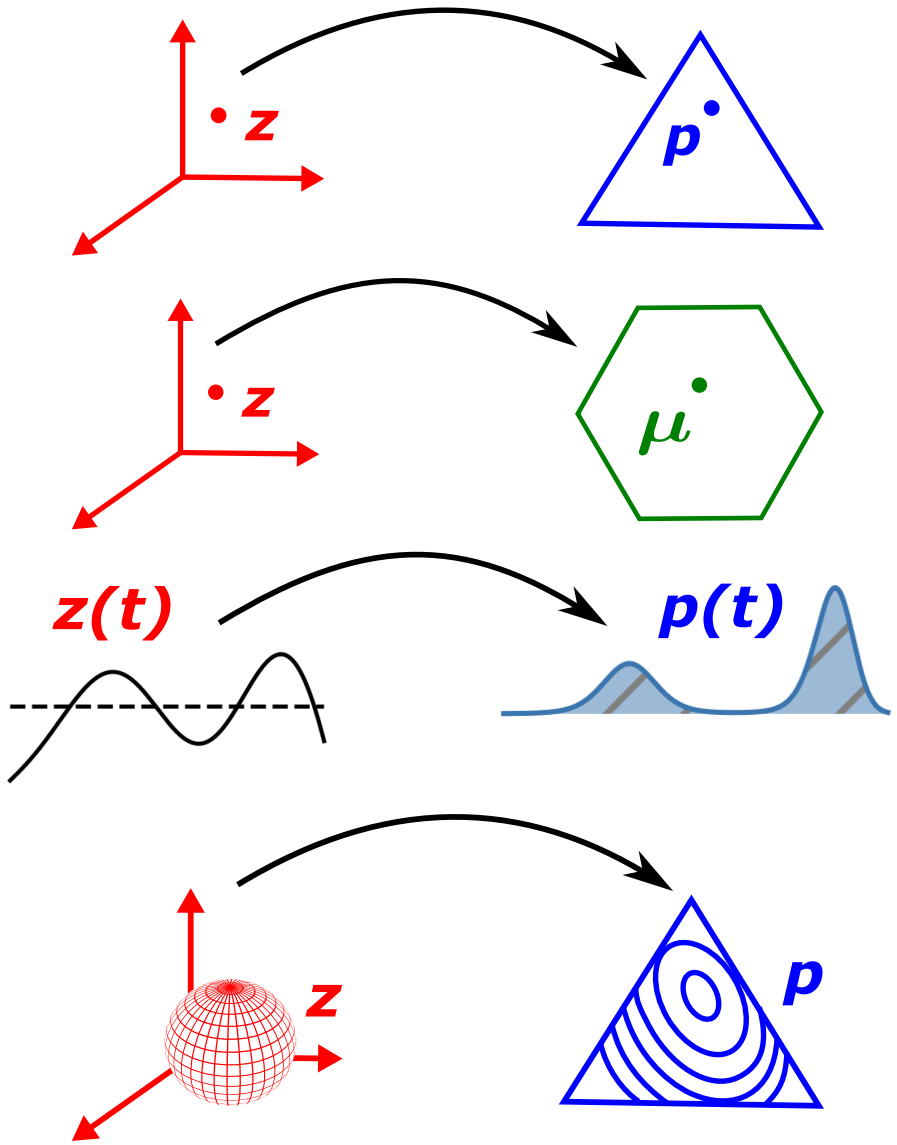


Sparse ✓

Structured ✓

Continuous

Stochastic



Outline

Sparse and Continuous Transformations

- So far: attention mechanisms over a **finite set** (words, pixel regions)
- Can we generalize attention to *continuous domains*?
- Useful for modalities like speech, images, video, or continuous-time signals.
- Useful for modeling continuous-time memories.

From Discrete to Continuous Attention

(Martins et al., 2020a, 2022a, NeurIPS, JMLR)

Discrete attention

Finite set $S = \{1, \ldots, L\}$

Three ingredients:

- Score vector $\boldsymbol{z} \in \mathbb{R}^L$
- Value matrix $V \in \mathbb{R}^{D \times L}$
- Transformation from $\boldsymbol{z}$ to probability vector $\boldsymbol{p} \in \triangle^L$

Output:

- Weighted average $V\boldsymbol{p} \in \mathbb{R}^D$

From Discrete to Continuous Attention

(Martins et al., 2020a, 2022a, NeurIPS, JMLR)

Discrete attention

Finite set $S = \{1, \ldots, L\}$

Three ingredients:

- Score vector $\boldsymbol{z} \in \mathbb{R}^L$
- Value matrix $V \in \mathbb{R}^{D \times L}$
- Transformation from $\boldsymbol{z}$ to probability vector $\boldsymbol{p} \in \triangle^L$

Output:

- Weighted average $V\boldsymbol{p} \in \mathbb{R}^D$

Continuous attention

Measure space S (*e.g.* continuous)

From Discrete to Continuous Attention

(Martins et al., 2020a, 2022a, NeurIPS, JMLR)

Discrete attention

Finite set $S = \{1, \ldots, L\}$

Three ingredients:

- Score vector $\boldsymbol{z} \in \mathbb{R}^L$
- Value matrix $V \in \mathbb{R}^{D \times L}$
- Transformation from $\boldsymbol{z}$ to probability vector $\boldsymbol{p} \in \triangle^L$

Output:

- Weighted average $V\boldsymbol{p} \in \mathbb{R}^D$

Continuous attention

Measure space S (*e.g.* continuous)

Three ingredients:

- Score *function* $z : S \to \mathbb{R}$

From Discrete to Continuous Attention

(Martins et al., 2020a, 2022a, NeurIPS, JMLR)

Discrete attention

Finite set $S = \{1, \ldots, L\}$

Three ingredients:

- Score vector $\boldsymbol{z} \in \mathbb{R}^L$
- Value matrix $V \in \mathbb{R}^{D \times L}$
- Transformation from $\boldsymbol{z}$ to probability vector $\boldsymbol{p} \in \triangle^L$

Output:

- Weighted average $V\boldsymbol{p} \in \mathbb{R}^D$

Continuous attention

Measure space S (*e.g.* continuous)

Three ingredients:

- Score *function* $z : S \to \mathbb{R}$ ✓
- Value *function* $V : S \to \mathbb{R}^D$

From Discrete to Continuous Attention

(Martins et al., 2020a, 2022a, NeurIPS, JMLR)

Discrete attention

Finite set $S = \{1, \ldots, L\}$

Three ingredients:

- Score vector $\boldsymbol{z} \in \mathbb{R}^L$
- Value matrix $V \in \mathbb{R}^{D \times L}$
- Transformation from $\boldsymbol{z}$ to probability vector $\boldsymbol{p} \in \triangle^L$

Output:

- Weighted average $V\boldsymbol{p} \in \mathbb{R}^D$

Continuous attention

Measure space S (*e.g.* continuous)

Three ingredients:

- Score *function* $z : S \to \mathbb{R}$ ✓
- Value *function* $V : S \to \mathbb{R}^D$ ✓
- Transformation from z to *density* $p : S \to \mathbb{R}_+$, $\int_S p = 1$

From Discrete to Continuous Attention

(Martins et al., 2020a, 2022a, NeurIPS, JMLR)

Discrete attention

Finite set $S = \{1, \ldots, L\}$

Three ingredients:

- Score vector $\boldsymbol{z} \in \mathbb{R}^L$
- Value matrix $V \in \mathbb{R}^{D \times L}$
- Transformation from $\boldsymbol{z}$ to probability vector $\boldsymbol{p} \in \triangle^L$

Output:

- Weighted average $V\boldsymbol{p} \in \mathbb{R}^D$

Continuous attention

Measure space S (*e.g.* continuous)

Three ingredients:

- Score *function* $z : S \to \mathbb{R}$ ✓
- Value *function* $V : S \to \mathbb{R}^D$ ✓
- Transformation from z to *density* $p : S \to \mathbb{R}_+$, $\int_S p = 1$ ✓

Output:

- $\mathbb{E}_p[V(t)] = \int_S p(t)V(t) \in \mathbb{R}^D$

From Discrete to Continuous Attention

(Martins et al., 2020a, 2022a, NeurIPS, JMLR)

Discrete attention

Finite set $S = \{1, \ldots, L\}$

Three ingredients:

- Score vector $\boldsymbol{z} \in \mathbb{R}^L$
- Value matrix $V \in \mathbb{R}^{D \times L}$
- Transformation from $\boldsymbol{z}$ to probability vector $\boldsymbol{p} \in \triangle^L$

Output:

- Weighted average $V\boldsymbol{p} \in \mathbb{R}^D$

Continuous attention

Measure space S (*e.g.* continuous)

Three ingredients:

- Score *function* $z : S \to \mathbb{R}$ ✓
- Value *function* $V : S \to \mathbb{R}^D$ ✓
- Transformation from z to *density* $p : S \to \mathbb{R}_+$, $\int_S p = 1$ ✓

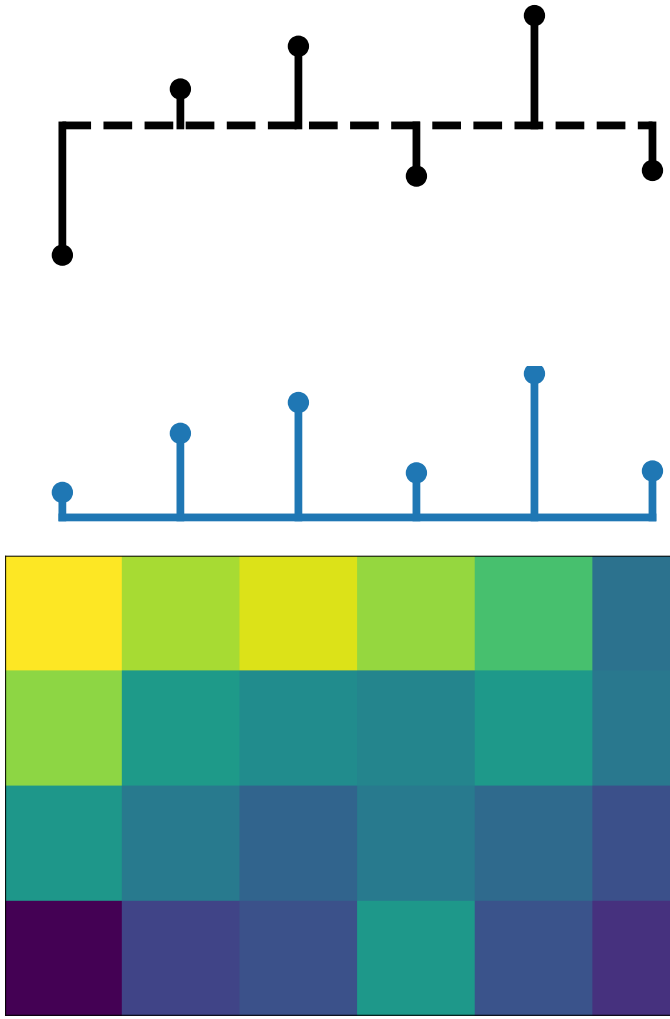
Output:

- $\mathbb{E}_p[V(t)] = \int_S p(t)V(t) \in \mathbb{R}^D$

Instead of tokens $\longrightarrow$ use basis function representations.
Useful to smooth and compress memory.

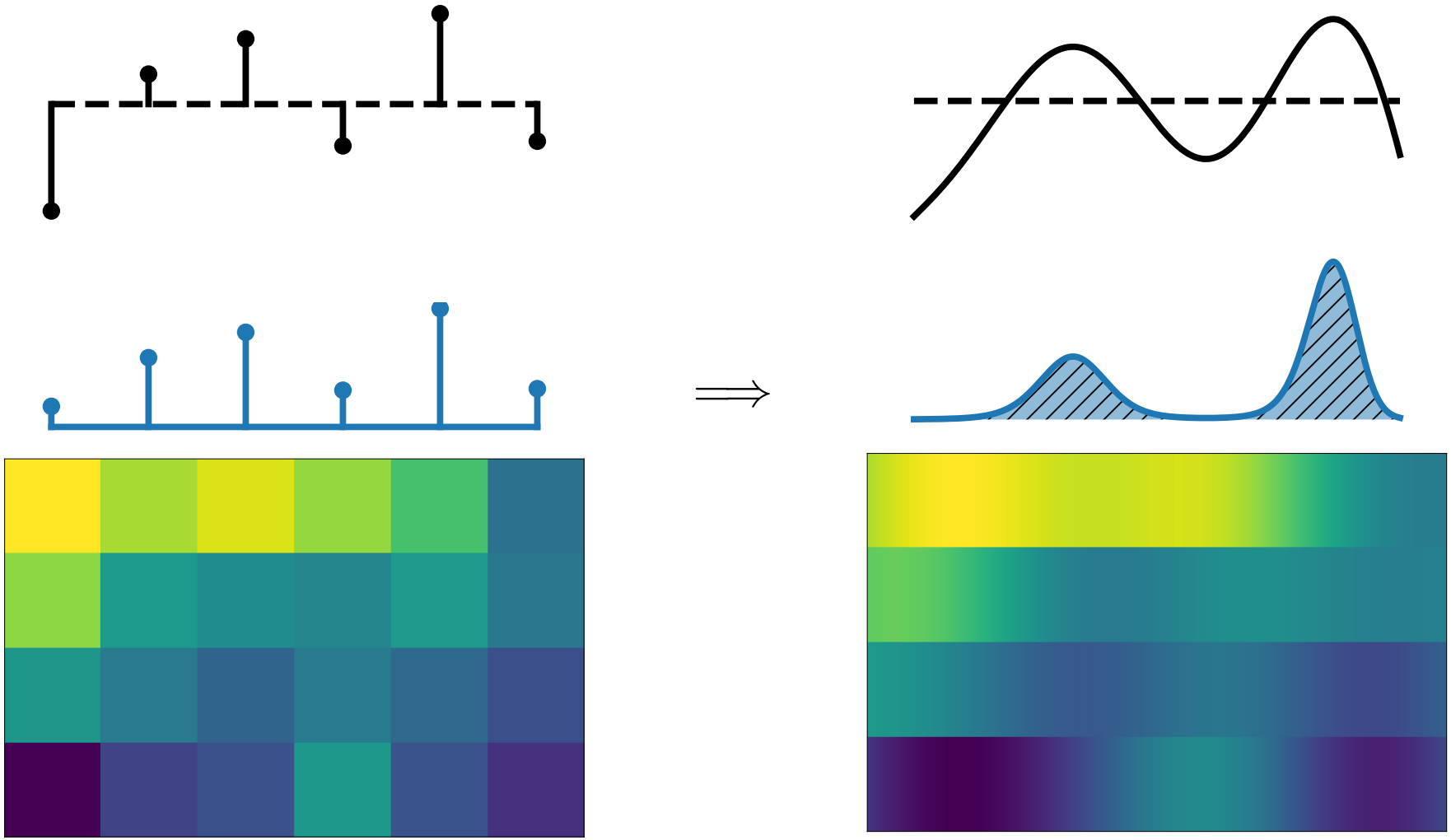
From Discrete to Continuous Attention

(Martins et al., 2020a, 2022a, NeurIPS, JMLR)

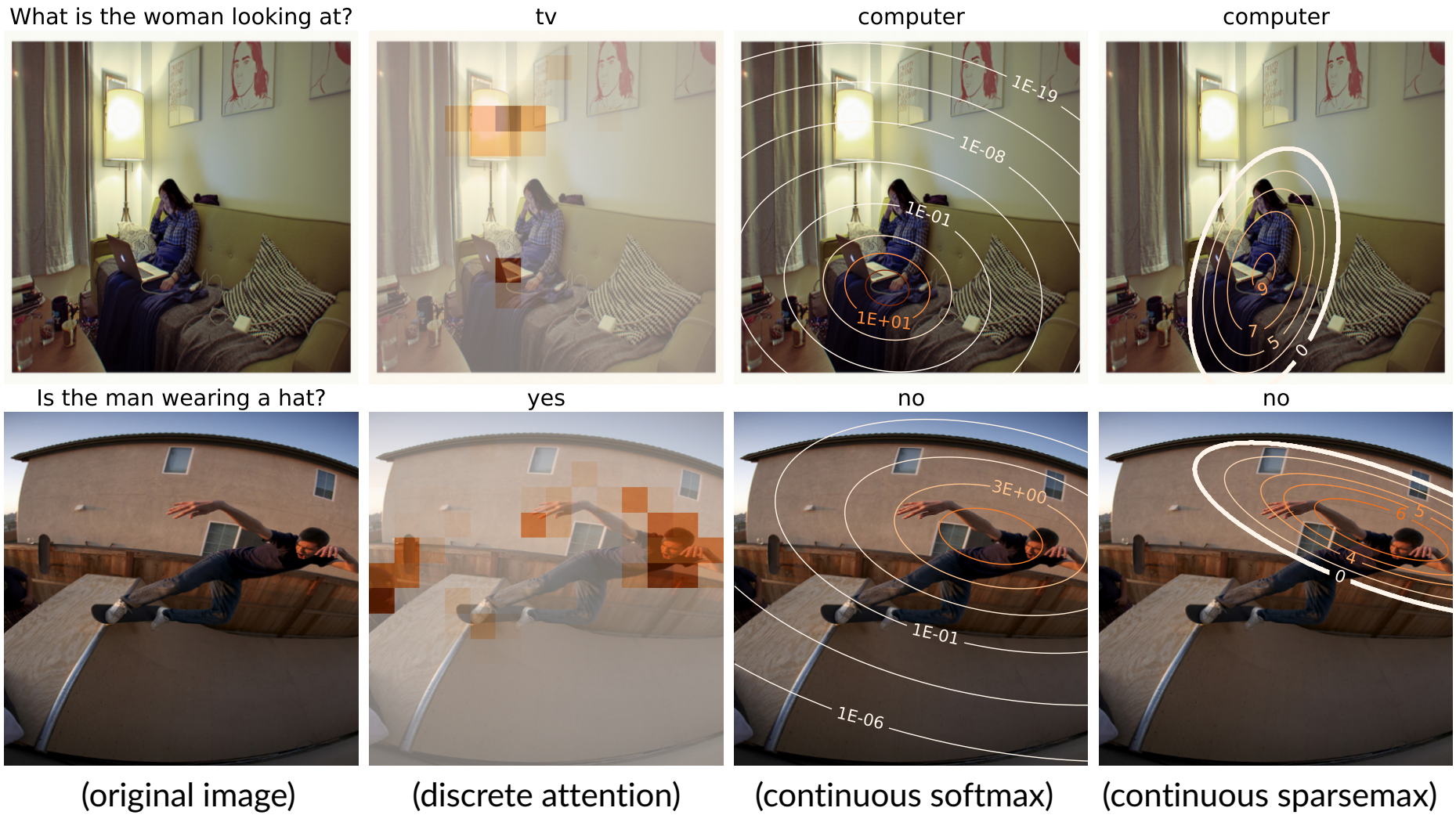


From Discrete to Continuous Attention

(Martins et al., 2020a, 2022a, NeurIPS, JMLR)



Example: Visual Question Answering

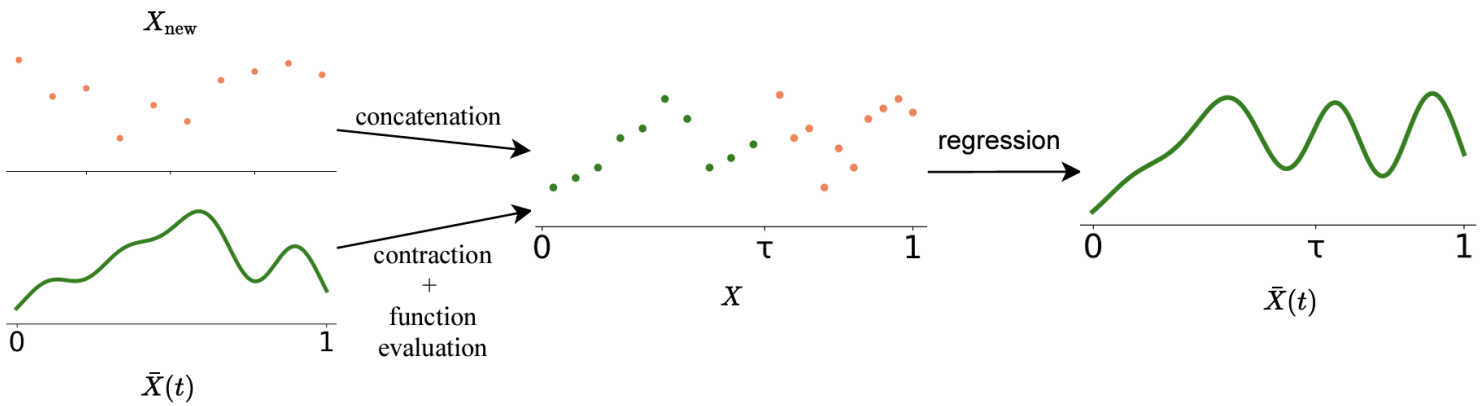


(Martins et al., 2020a, NeurIPS)

Example: ∞-former (Martins et al., 2022b, ACL)

Append a continuous-time long-term memory in $[0, 1]$ via a basis function representation.

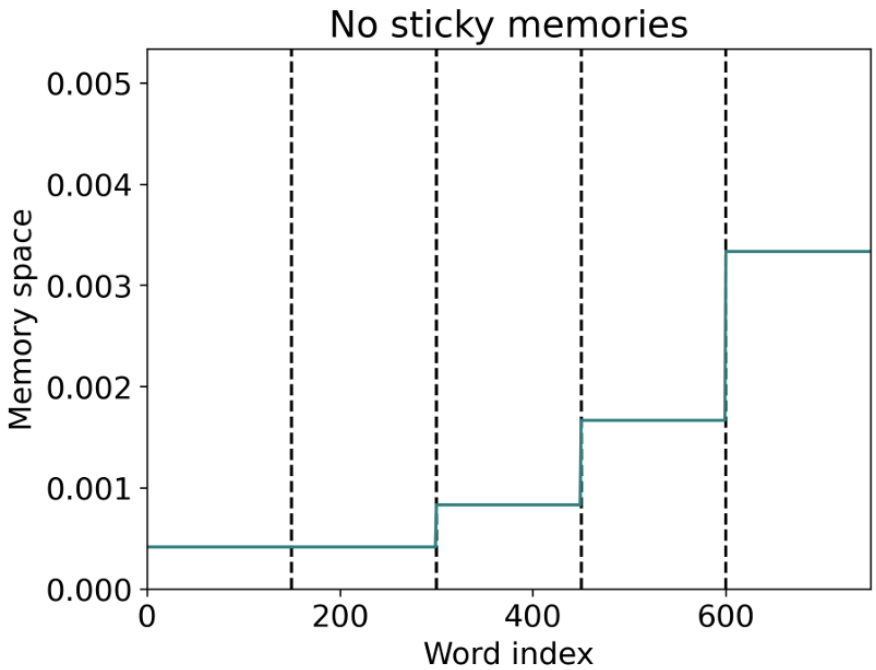
After processing a chunk of data, update the long-term memory:

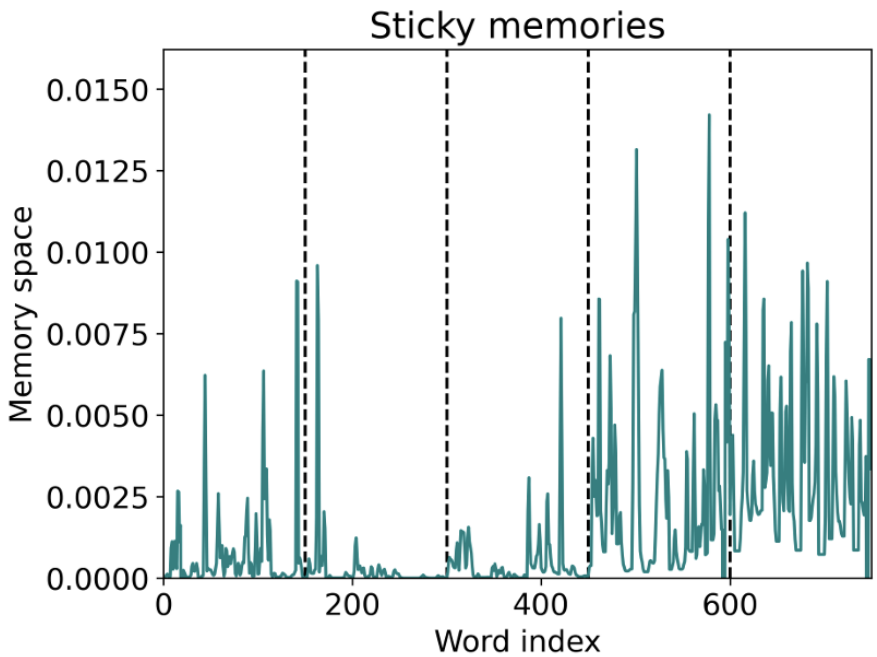


Unbounded memory: Never delete information, just compress it.

Sticky Memories (Martins et al., 2022b, ACL)

Memories attended more often become sticky (larger slice in [0,1]).





Example: Sparsifying Action Spaces in RL

Sparsemax/entmax have also been applied extensively to RL.

Both for discrete and continuous action spaces:

- Sparse MDPs with sparsemax (Lee et al., 2018) (Chow et al., 2018, ICML)
- Sparse regularized actor-critic (Lee et al., 2019) (Yang et al., 2019, NeurIPS)
- q-Munchausen Reinforcement Learning (Zhu et al., 2023, NeurIPS)
- Continuous actions (Zhu et al., 2024)

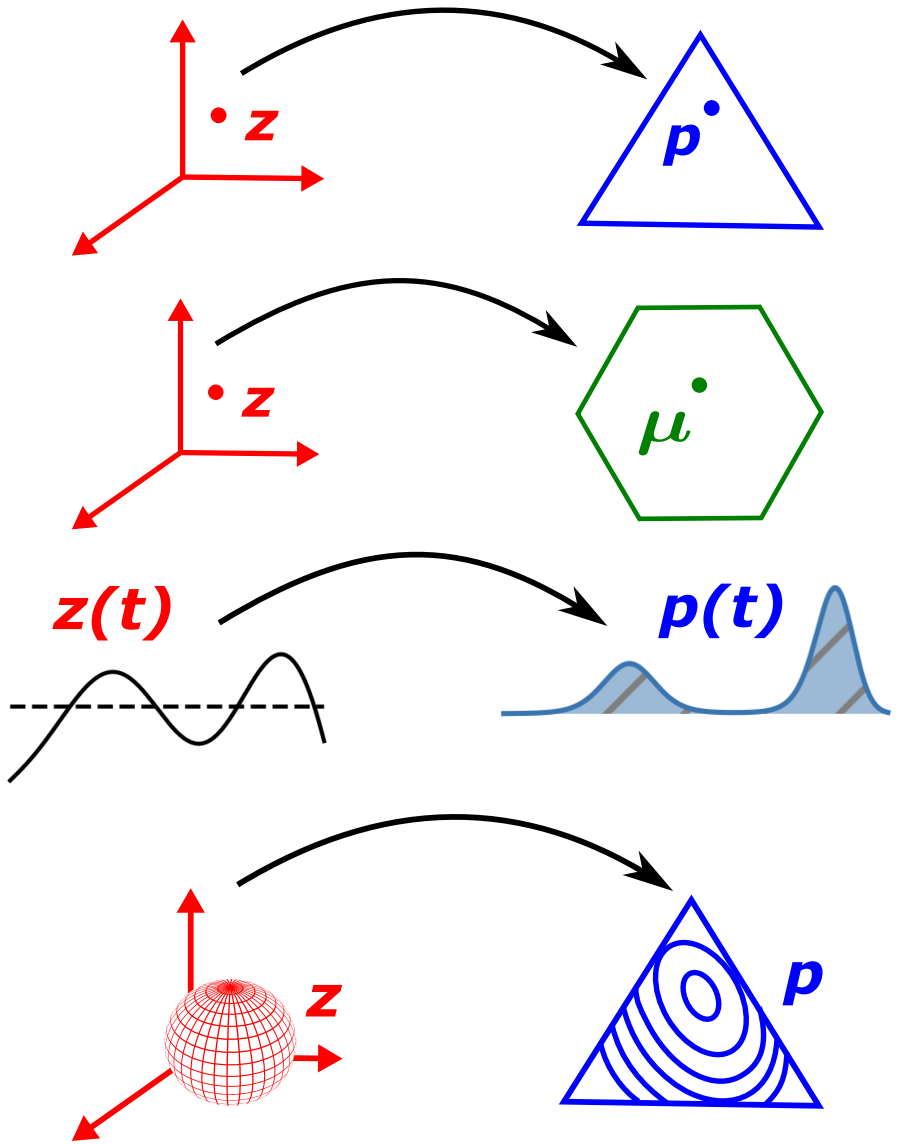
There's a rich body of literature on this topic!

Sparse ✓

Structured ✓

Continuous ✓

Stochastic

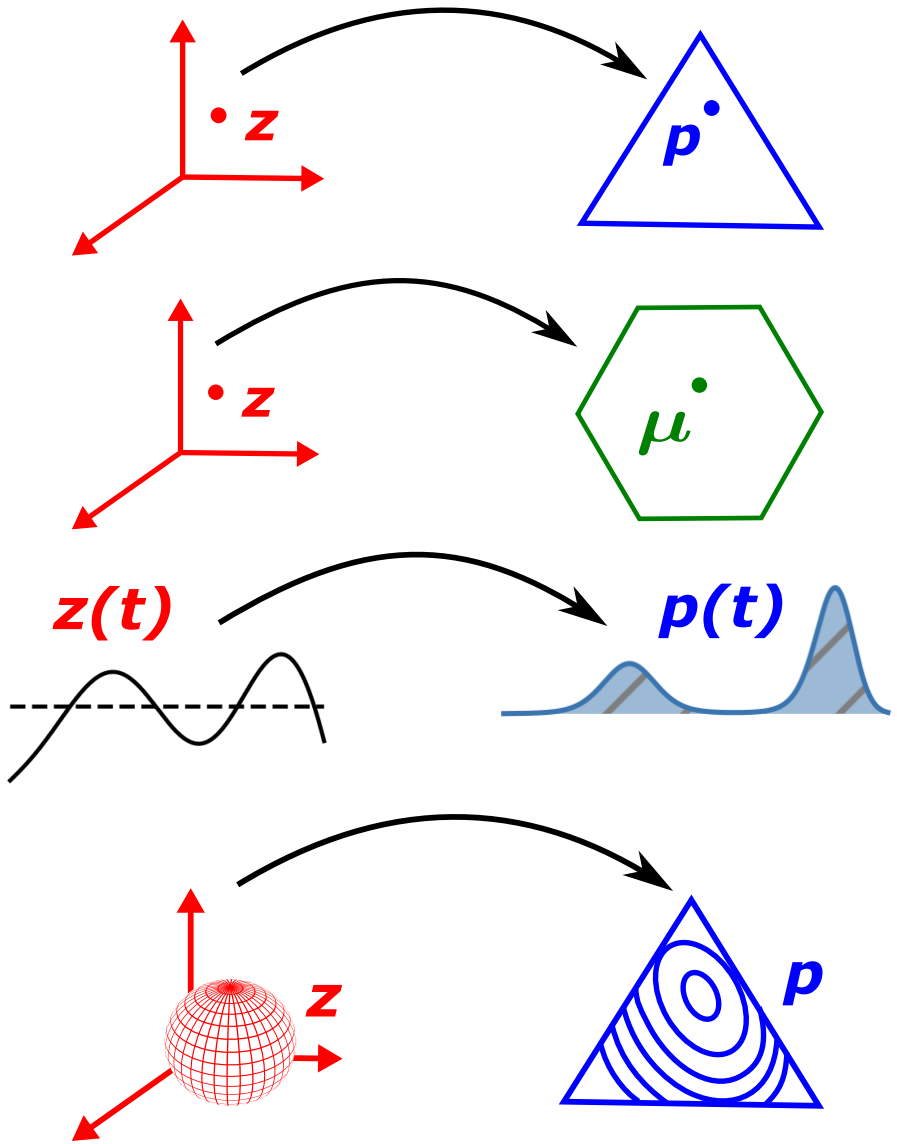


Sparse ✓

Structured ✓

Continuous ✓

Stochastic



Outline

So Far...

We talked about sparse, structured, and continuous-time transformations.

But all are *deterministic*.

What if we want a latent probabilistic model? Need randomness.

Example: Gumbel softmax is often used as a *continuous* proxy for a (*discrete*) categorical latent variable **(Maddison et al., 2017; Jang et al., 2017)**

Other examples:

- Perturb-and-MAP **(Papandreou and Yuille, 2011; Corro and Titov, 2019)**
- Implicit MLE **(Niepert et al., 2021)**
- Differentiable perturbed optimizers **(Berthet et al., 2020)**
- Stochastic softmax tricks **(Paulus et al., 2020)**

Can we combine sparse transformations with stochasticity?

Densities over $\triangle$

We denote by relint($\triangle$) $:= \{\boldsymbol{p} \in \triangle : \boldsymbol{p} > 0\}$ the relative interior of $\triangle$.

Common densities over $\triangle$:

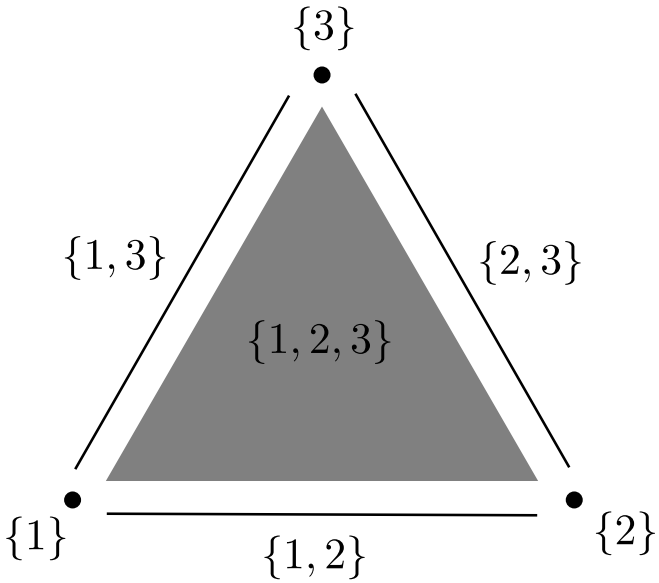
- Dirichlet distribution (conjugate prior of categorical)
- Logistic-Normal (a.k.a. Gaussian-Softmax)
- Concrete (a.k.a. Gumbel-Softmax)

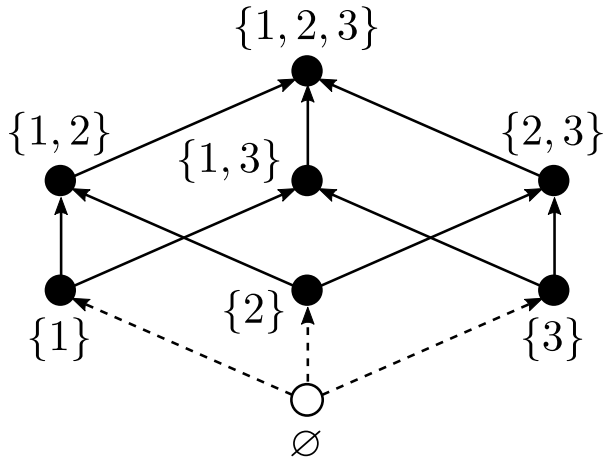
All these place probability mass to relint($\triangle$) only.

No probability mass on the boundary (i.e., no stochastic *sparsity*).

Face Stratification of $\triangle$ (Farinhas et al., 2022, ICLR)

$\triangle$ can be decomposed as a "direct sum" of faces which form a lattice:





0-faces are vertices, 1-faces are edges, ..., the $(K-1)$-face is relint($\triangle$).

We want a density over $\triangle$ assigning probability mass to all the faces.

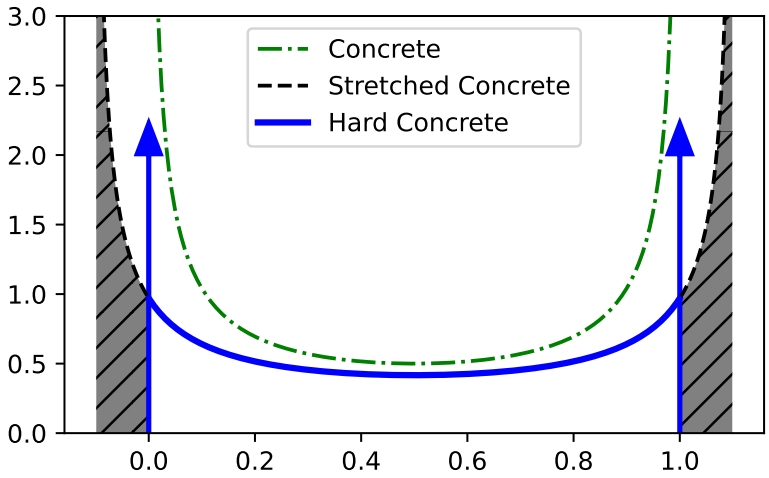
This is called a mixed density.

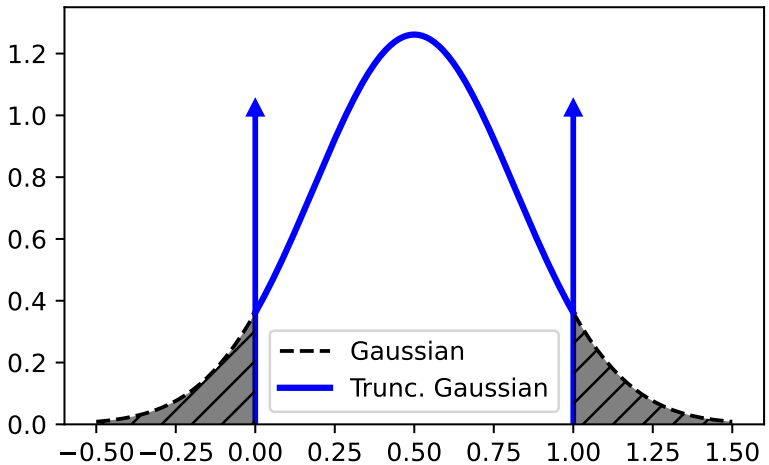
Binary Mixed Densities ($K = 2$)

When $K = 2$, the simplex is isomorphic to unit interval, $\triangle_1 \simeq [0, 1]$.

They can be constructed by truncating continuous densities:

- Binary Hard Concrete
  (Louizos et al., 2018)
- Hard Kumaraswamy
  (Bastings et al., 2019)
- Truncated Gaussian
  (Hinton and Ghahramani, 1997; Palmer et al., 2017)





What about $K > 2$?

K-Dimensional Mixed Densities (Farinhas et al., 2022, ICLR)

K-D Hard Concrete: "stretch-and-project"

$$Y \sim \text{HardConcrete}(\boldsymbol{z}, \lambda, \tau) \quad \Leftrightarrow \quad \begin{array}{l} Y' \sim \text{Concrete}(\boldsymbol{z}, \lambda) \\ Y = \text{sparsemax}(\tau Y'), \quad \text{with } \tau \geq 1. \end{array}$$

- The larger τ, the higher the tendency to hit a low-dimensional face and induce sparsity.

K-Dimensional Mixed Densities (Farinhas et al., 2022, ICLR)

K-D Hard Concrete: "stretch-and-project"

$$Y \sim \text{HardConcrete}(\boldsymbol{z}, \lambda, \tau) \quad \Leftrightarrow \quad \begin{array}{l} Y' \sim \text{Concrete}(\boldsymbol{z}, \lambda) \\ Y = \text{sparsemax}(\tau Y'), \quad \text{with } \tau \geq 1. \end{array}$$

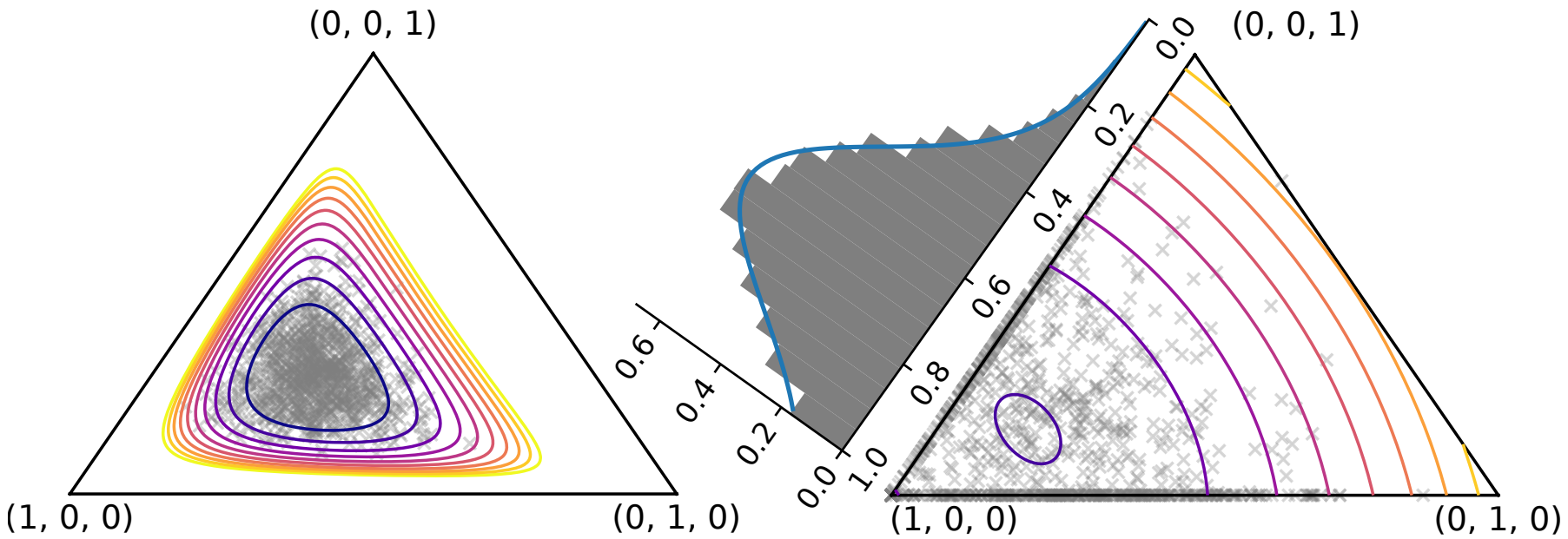
- The larger τ, the higher the tendency to hit a low-dimensional face and induce sparsity.

K-D Gaussian-Sparsemax: sample from a Gaussian and project.

$$Y \sim \text{GaussianSparsemax}(\boldsymbol{z}, \Sigma) \quad \Leftrightarrow \quad \begin{array}{l} N \sim \mathcal{N}(0, 1) \\ Y = \text{sparsemax}(\boldsymbol{z} + \Sigma^{1/2} N). \end{array}$$

- Sparsemax counterpart of the Logistic-Normal.

Logistic-Normal vs Gaussian-Sparsemax (Farinhas et al., 2022, ICLR)



Logistic-Normal (left) assigns zero probability to all faces but relint($\triangle$)

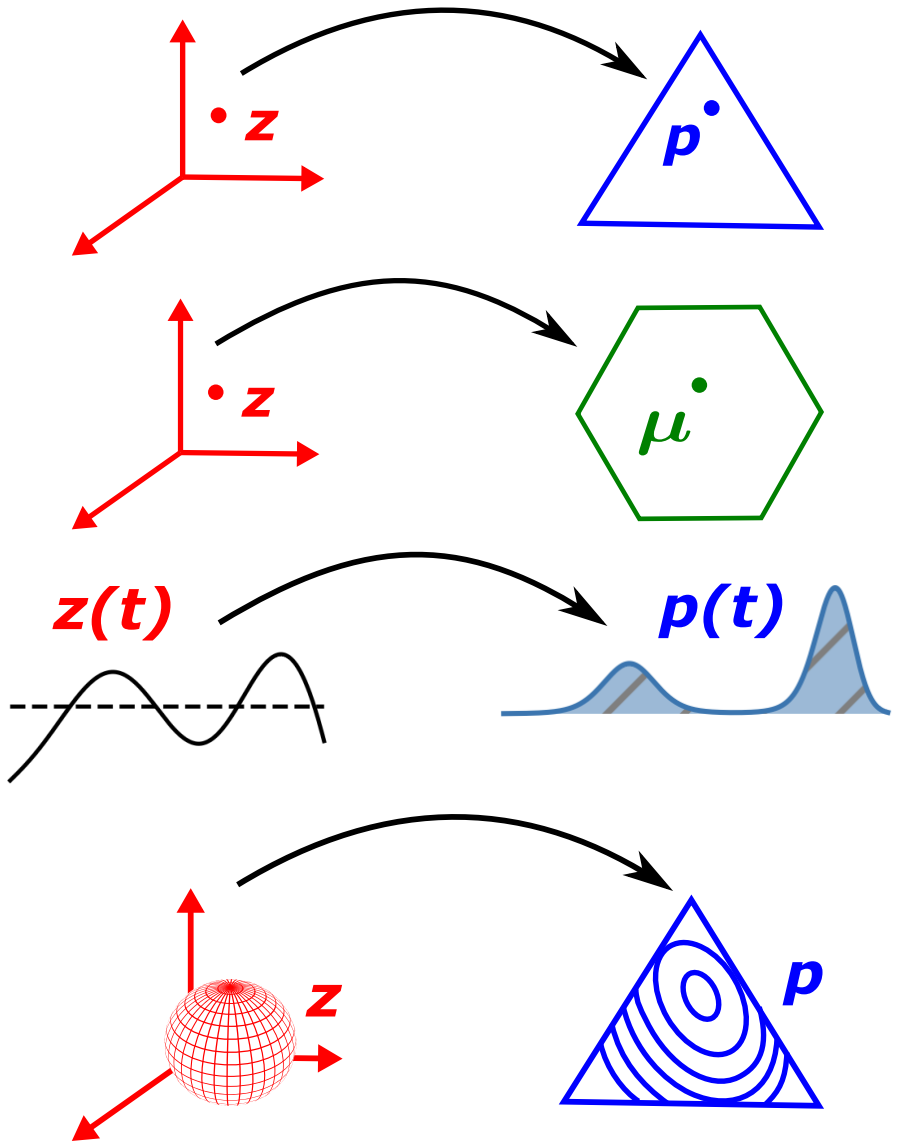
Gaussian-Sparsemax (right) is a mixed distribution: it assigns probability to the *full* simplex, including its boundary.

Sparse ✓

Structured ✓

Continuous ✓

Stochastic ✓



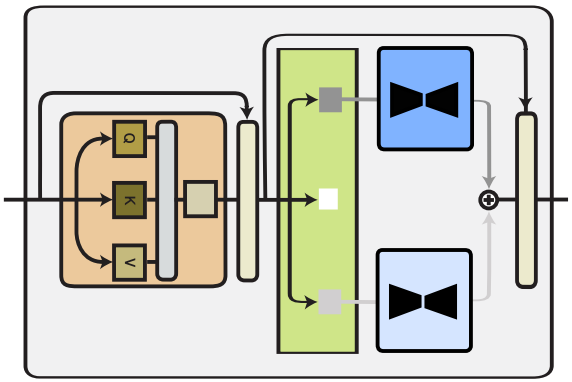
Summary of Part I

We covered:

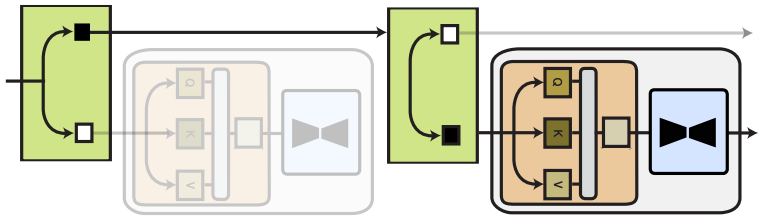
- Transformations that handle **sparsity**, **constraints**, and **structure**
- All are differentiable and their gradients are efficient to compute
- Can work on discrete domains on in continuous domains
- Can be deterministic or stochastic
- Can be used as hidden layers or as output layers
- Many applications (interpretability, translation, generation, associative memories, ...)

Outline

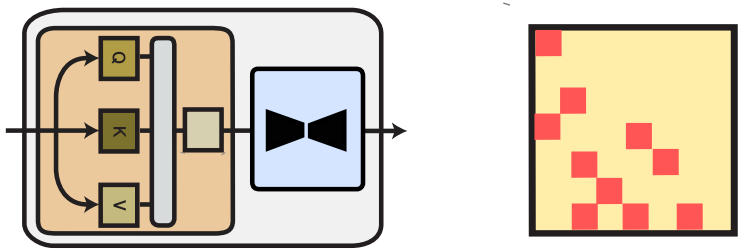
Experts and Adapters



Depth



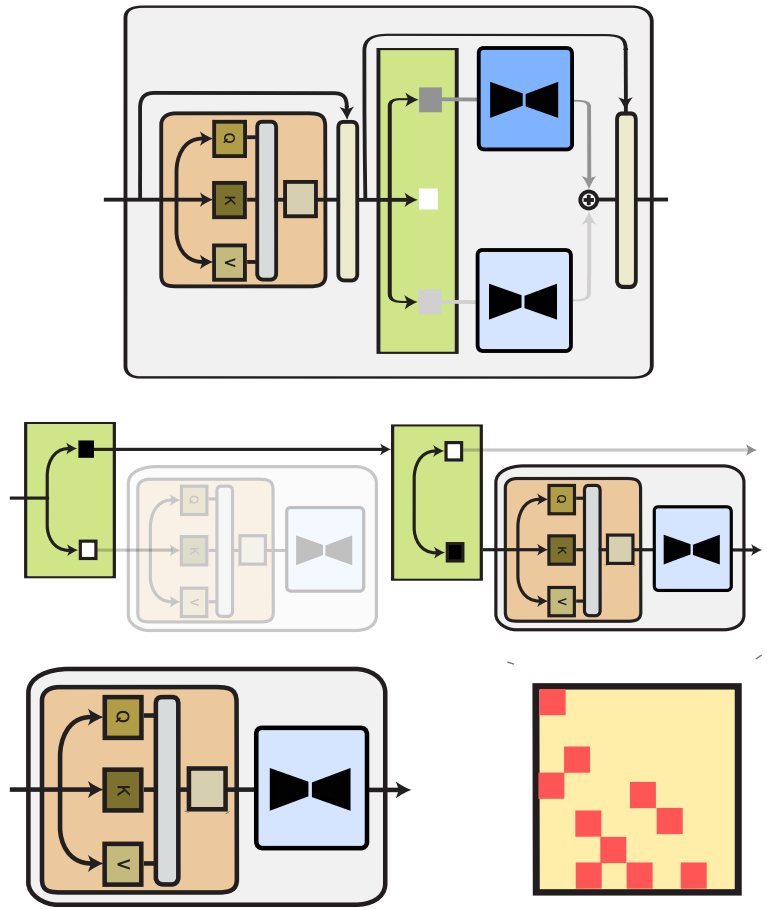
Attention



Experts and Adapters

Depth

Attention



Outline

Why *Sparse* MoEs?

- For the original MoEs, Jacobs et al. (1991a) listed these advantages:
  - learning speed,
  - better generalisation,
  - more interpretable representations,
  - reduced hardware requirements.
- Core idea for sparse MoEs (Shazeer et al., 2017):
  - increase the total parameters of foundation models while retaining efficiency $\rightarrow$ better scaling laws (Kaplan et al., 2020)
  - Still, high memory footprint!

Mixtures of Experts (MoEs) (Jacobs et al., 1991b)

Given input $\boldsymbol{x}$ and experts $\mathcal{E} = \{f_1, \ldots, f_n\}$, a gating function (or router) returns the 'importance' of each expert:

$$p(\mathcal{E} \mid \boldsymbol{x}) = g(\boldsymbol{x}) = \text{softmax}[\text{MLP}(\boldsymbol{x})]$$

The output is the linear combination of the outputs of each expert:

$$\hat{y} = \sum_{i=1}^{|\mathcal{E}|} g(\boldsymbol{x})_i f_i(\boldsymbol{x})$$

MoEs (continued)

This encourages competition between experts, leading to specialisation.

Why? Let's take a look at the derivative of an MSE loss ℓ wrt $g(\boldsymbol{x})_i$:

$$\frac{\partial \ell}{\partial g(\boldsymbol{x})_i} = (\hat{y} - y) g(\boldsymbol{x})_i \left(f_i(\boldsymbol{x}) - \sum_j f_j(\boldsymbol{x}) g(\boldsymbol{x})_j \right)$$

The sign also depends on the margin between the output expert i and the others combined.

Another perspective: implicitly partitioning the data

In the limit where $p(\mathcal{E} \mid \boldsymbol{x})$ is deterministic, each $\mathcal{E}_i$ is trained on the dataset $\mathcal{D} \triangleq \{\boldsymbol{x} \mid g(\boldsymbol{x})_i = 1\}$. Probabilistic routing defines a soft partition.

Deep MoEs (Eigen et al., 2013)

Next innovation: instead of experts being whole models, stack multiple layers of MoEs

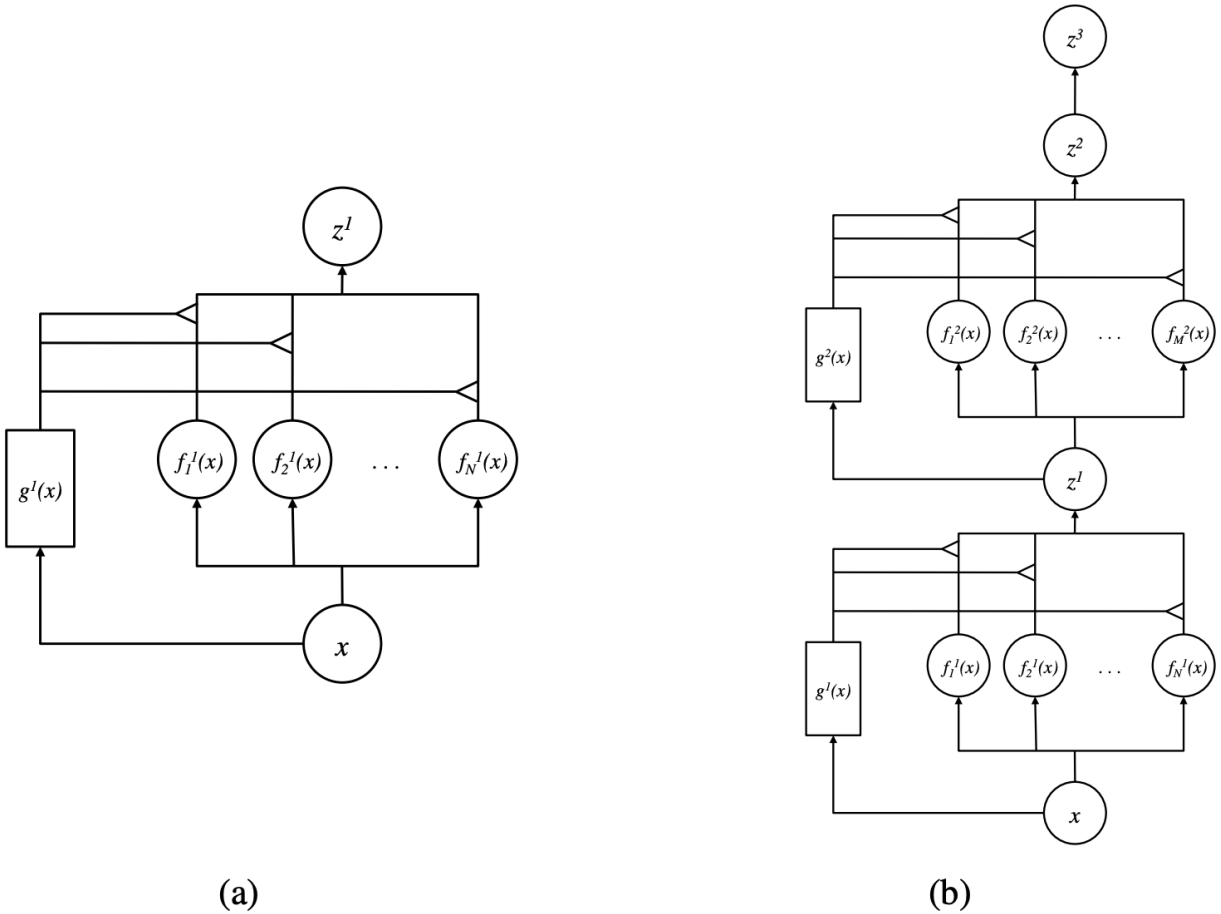
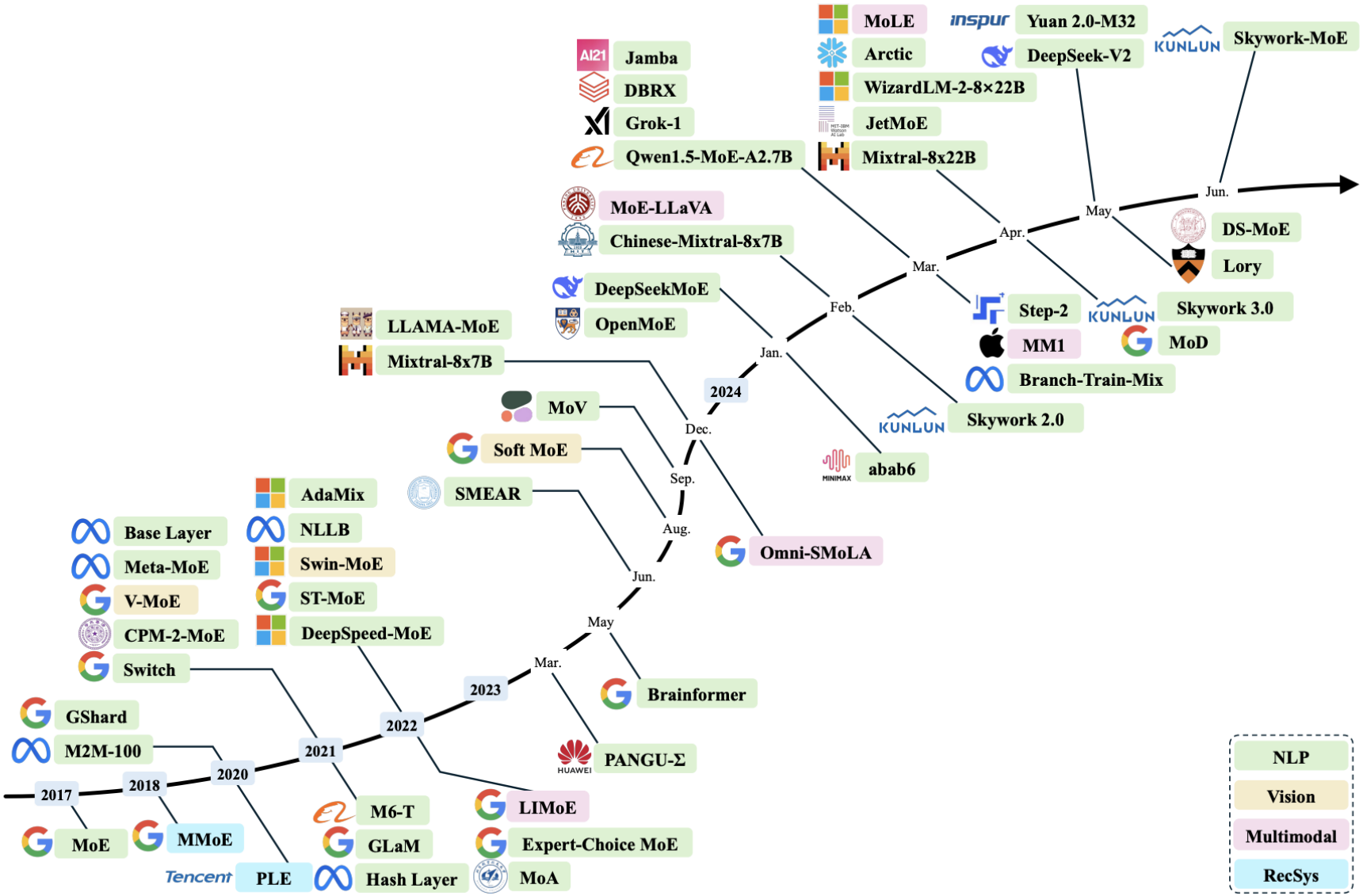


Figure 1: (a) Mixture of Experts; (b) Deep Mixture of Experts with two layers.

History of Sparse MoEs (Cai et al., 2024)



Sparse MoE LM (Shazeer et al., 2017)

Several key innovations:

- Introduce MoE layers in LSTM LMs, with token-level routing
- Top-k routing for conditional computation (ϵ is random noise):

$$p(\mathcal{E} \mid \boldsymbol{x}) = \text{softmax}(\boldsymbol{z}) \quad z_i = \begin{cases} g(\boldsymbol{x})_i & \text{if } i \in \arg\text{top-}k[g(\boldsymbol{x}) + \epsilon] \\ -\infty & \text{else} \end{cases}$$

- Load balancing loss to optimise expert usage during training

Challenges of Sparse MoE LM

1

- Shrinking batch problem: each expert observes only $\frac{kb}{n}$ tokens among the b tokens in a batch by selecting k out of n experts.
- Solution: expand the batch size with mixed parallelism: d devices serve
  - as *data-parallel* replicas for standard / routing layers;
  - as *expert-parallel* shards for experts (note: this is different from *model parallelism* of FFNs!)

  $\rightarrow$ each expert observes d more tokens

2

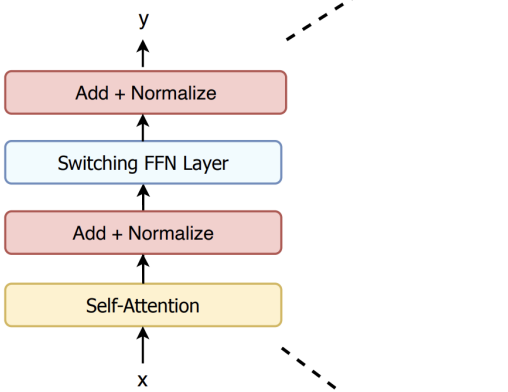
- Module collapse: rich-gets-richer effect for few experts. Underutilisation of resources due to uneven routing.
- Solution: auxiliary losses based on importance and load

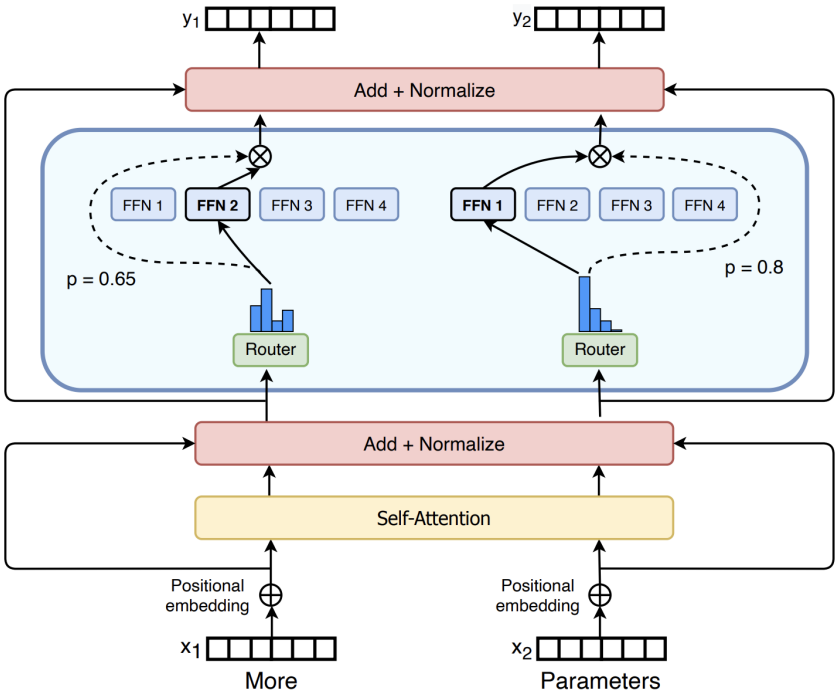
Switch Transformer (Fedus et al., 2022)

Building on G-Shard (Lepikhin et al., 2021), experts are now FFNs in Transformers and routing is top-1.

Tensor sizes are static but routing choices are dynamic $\rightarrow$ trade-off expert underusing (padding) and overusing (token dropping) via expert capacity.

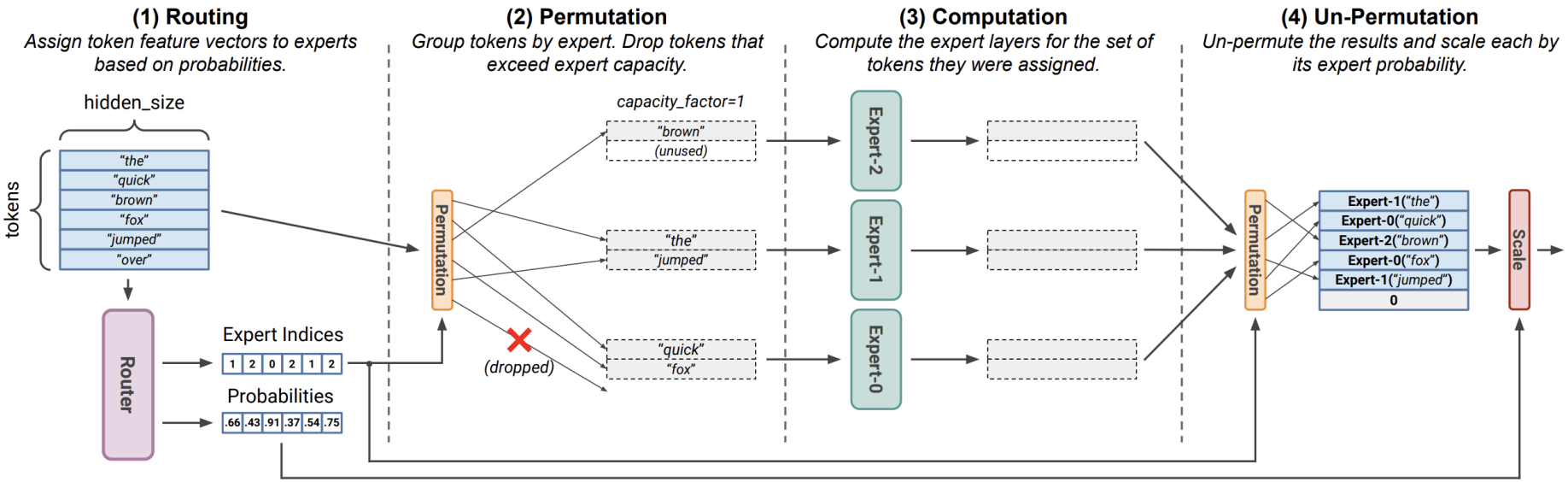
$$\text{expert capacity} = \frac{\text{tokens per batch}}{\text{number of experts}} \times \text{capacity factor}$$





Jupyter notebook on our tutorial webpage!

Sparse MoE Workflow



How to ensure that computation is not wasted and information is not discarded?

Two directions:

1. balancing losses
2. block-sparse matrix multiplication (Gale et al., 2023)

Balancing Losses

Aux loss (Lepikhin et al., 2021; Fedus et al., 2022):

$$\ell_A = \frac{1}{|\mathcal{E}|} \sum_{j=1}^{|\mathcal{E}|} \frac{c_j}{b} \left(\frac{1}{b} \sum_{i=1}^{b} g(x_i)_j \right)$$

where c_j is the number of tokens assigned to expert $\mathcal{E}_j$ and $g(x_i)$ the router scores. Simplifies Shazeer et al. (2017)'s losses.

Router z loss (Zoph et al., 2022):

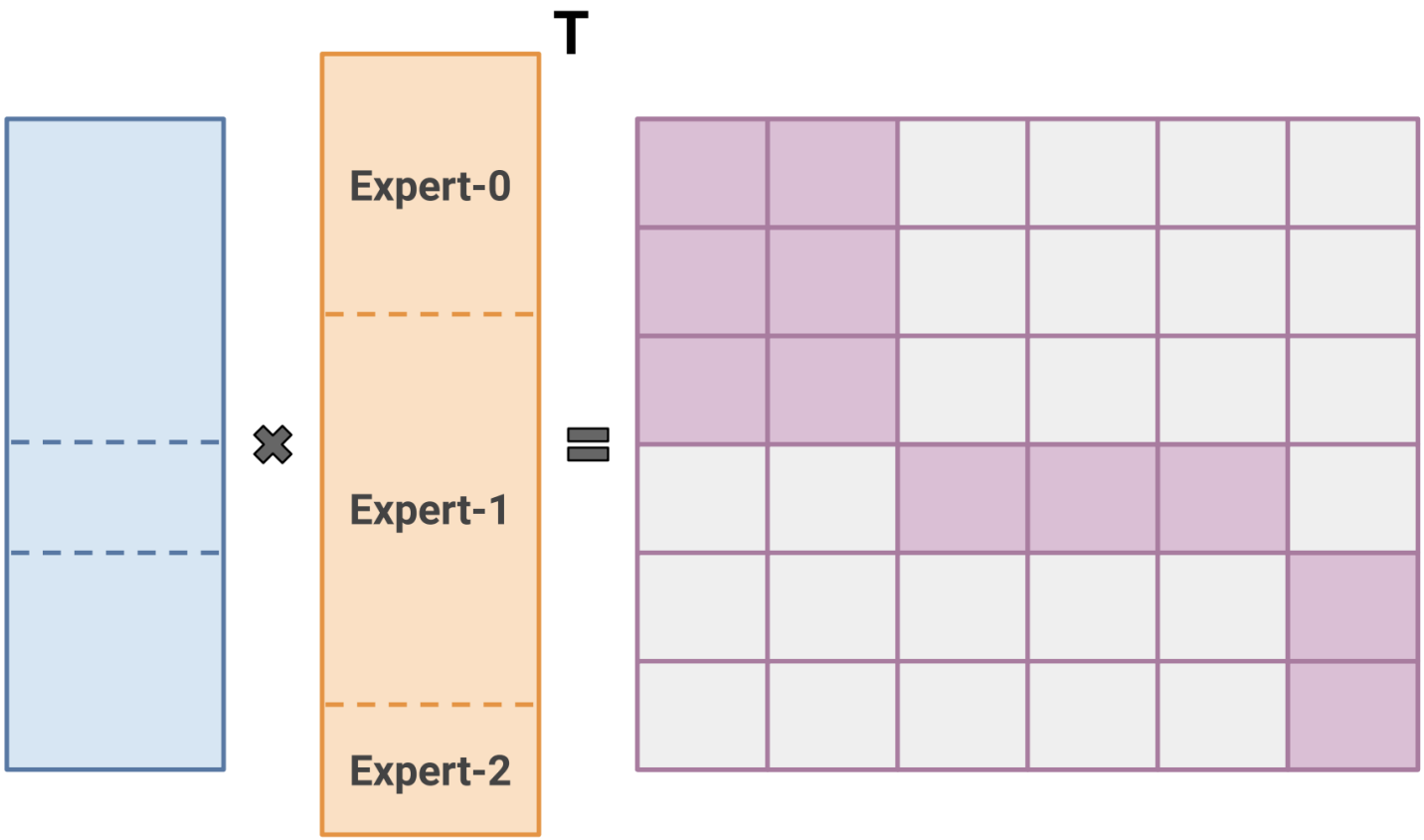
$$\ell_Z = \frac{1}{b} \sum_{i=1}^{b} \left(\log \sum_{j=1}^{|\mathcal{E}|} \exp[g(x_i)_j] \right)^2$$

Intuition: avoid extreme router logits for increased stability

Overcoming the Hardware Lottery

Hardware lottery (Hooker, 2021): model design is due to its compatibility with existing hardware, rather than its inherent superiority.

Megablocks (Gale et al., 2023): block sparse matrix multiplication makes load-balancing unnecessary $\rightarrow$ MoE is more expressive



Will Sparse MoE become the 'default' LLM architecture?

Bullish

- Instruction tuning better than dense (Shen et al., 2024)
- Scaling laws better than dense (Ludziejewski et al., 2024)

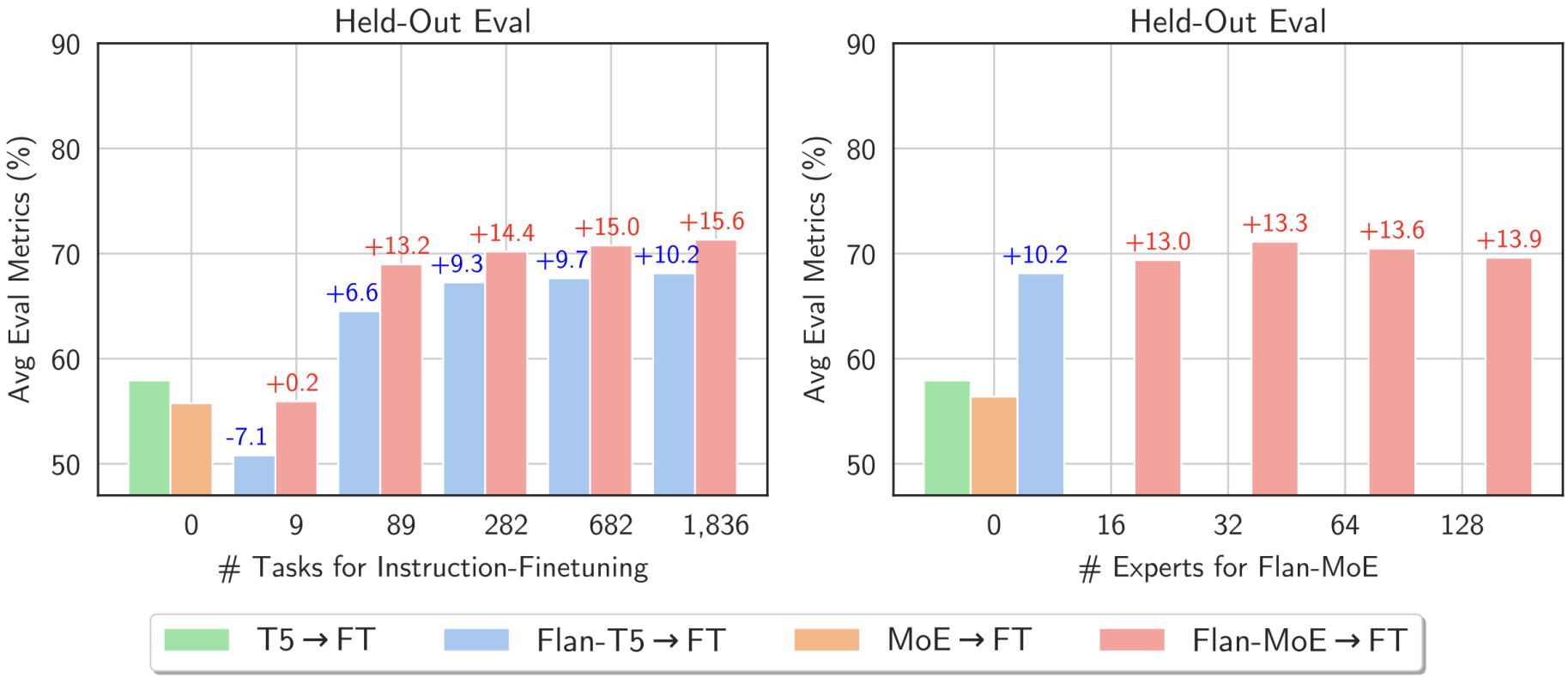
Bearish

- Cost–Accuracy–Performance trade-off (Fu et al., 2024)
- Limited expert specialisation (Muennighoff et al., 2024; Jiang et al., 2024a)
- Discrepancy in factual vs reasoning tasks (Jelassi et al., 2024)

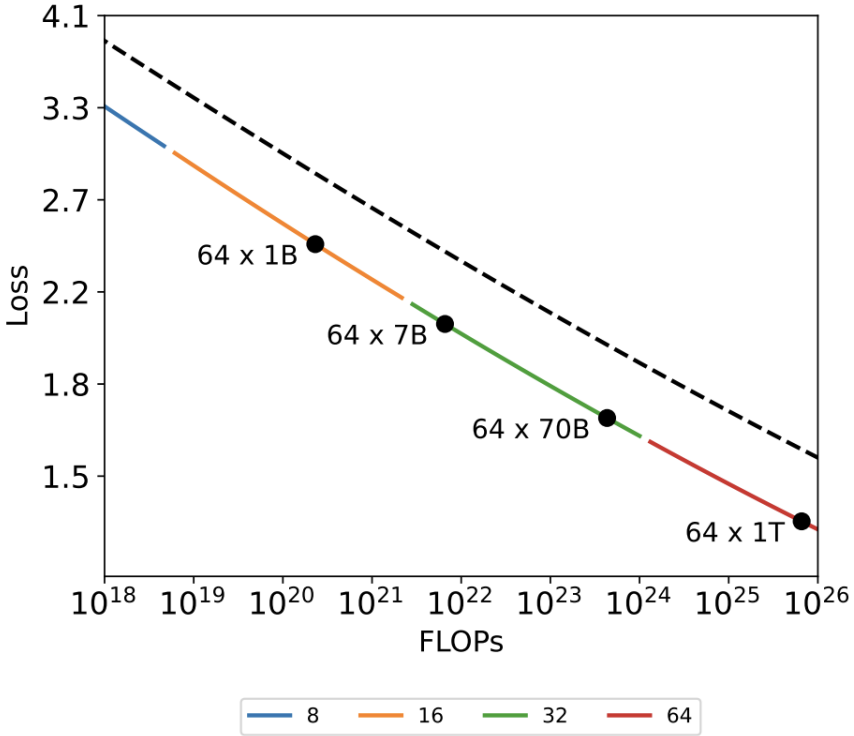
Instruction Tuning Sparse MoEs (Shen et al., 2024)

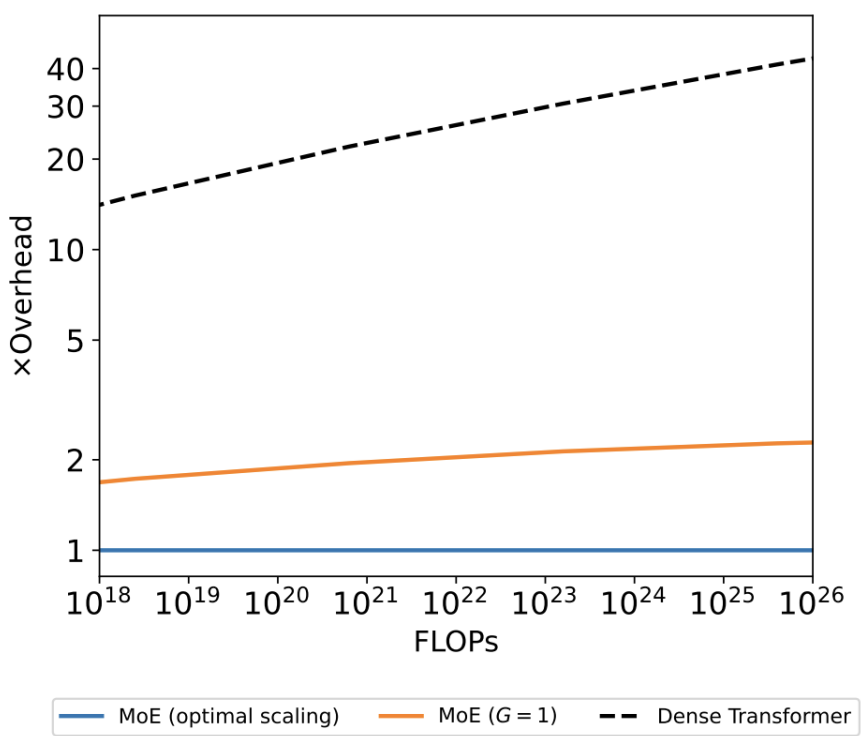
Sparse MoEs were believed to be harder to fine-tune (Artetxe et al., 2022)

However, *instruction* tuning is more beneficial than in dense models!



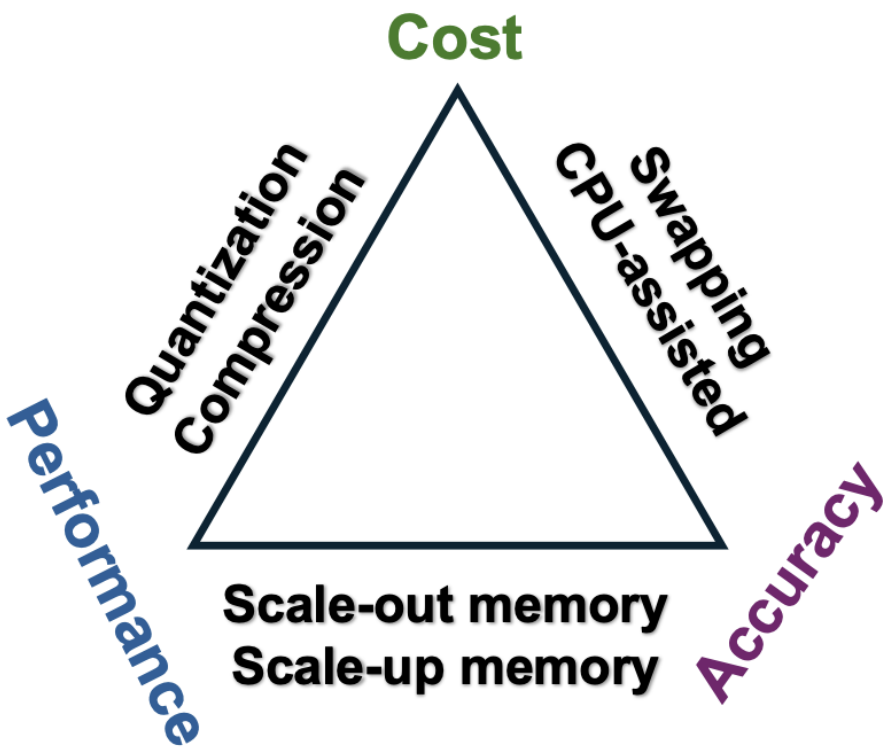
Scaling Laws for MoEs (Ludziejewski et al., 2024)





- Sparse MoEs offer a better performance–efficiency Pareto front compared to dense models
- The gap widens as we scale model size and data
- Optimal *granularity* G (ratio of d_{FFN} to d_{expert}) varies

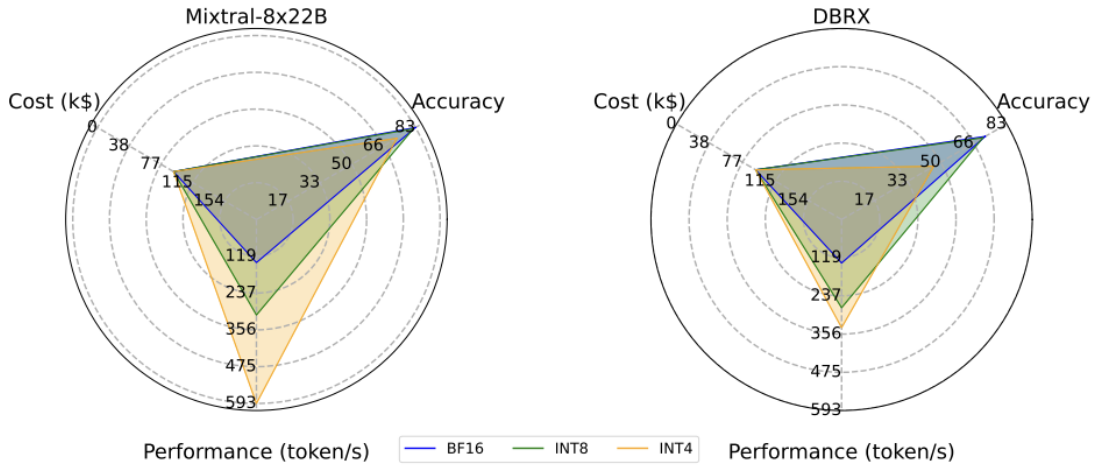
MoE-CAP: A Benchmark for Cost, Accuracy, and Performance in Sparse MoEs (Fu et al., 2024)



- New sparsity-aware metrics for performance (=efficiency): Memory Bandwidth Utilization and Model FLOPS Utilization
- New cost metric aware of heterogeneous computing resources, e.g., off-load experts from HBM to DRAM and SSD and/or compute to CPU.
- Accuracy benchmarking on MMLU, GSM8K, Arena-Hard.

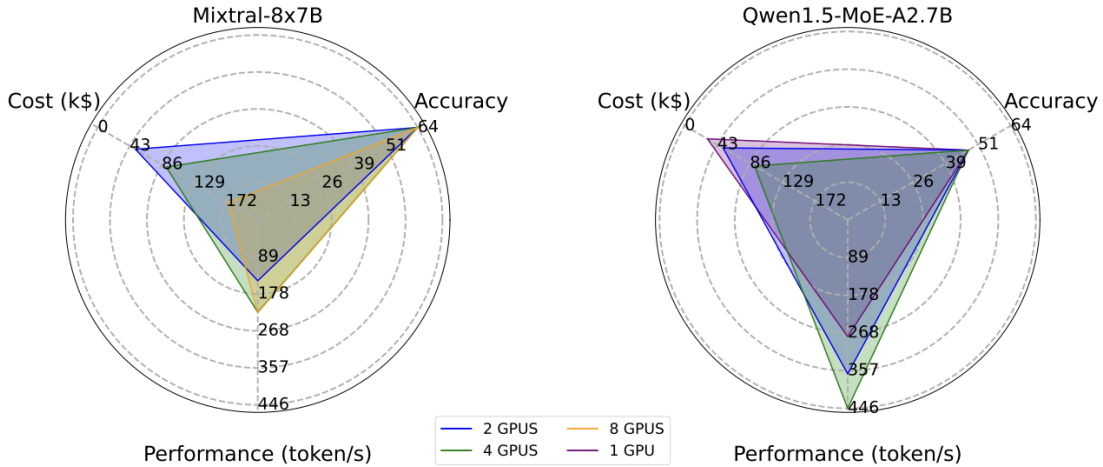
https://huggingface.co/spaces/sparse-generative-ai/open-moe-llm-leaderboard

CAP Trade-offs



Quantisation

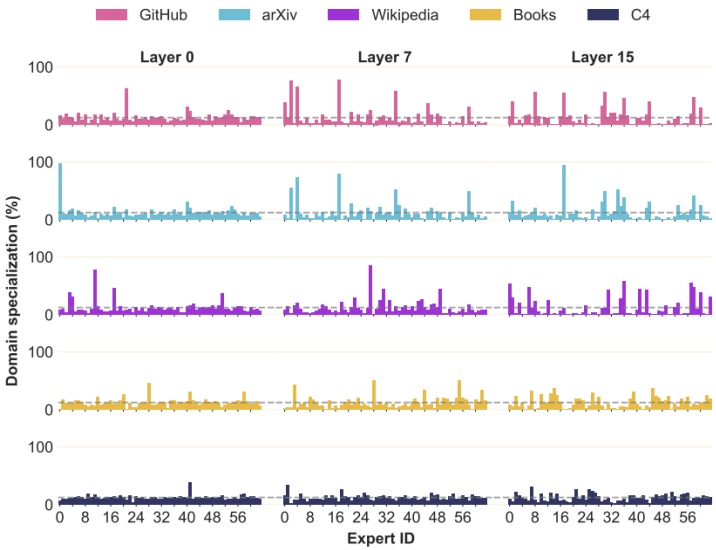
Figure 4: Comparison of CA and CP systems on GSM8K dataset under CAP view.

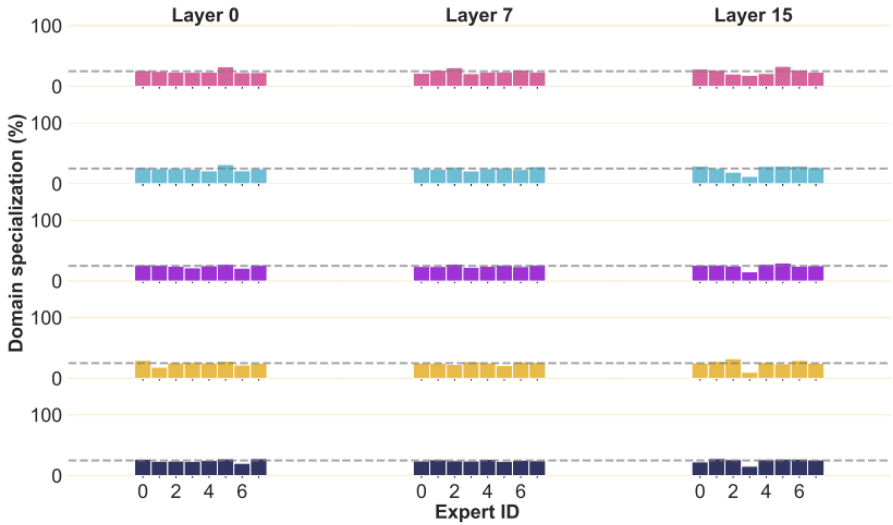


Parallelisation

Figure 5: Comparison of PA and AC systems on GSM8K under CAP method.

Do Experts Specialise? OLMoE has some answers.





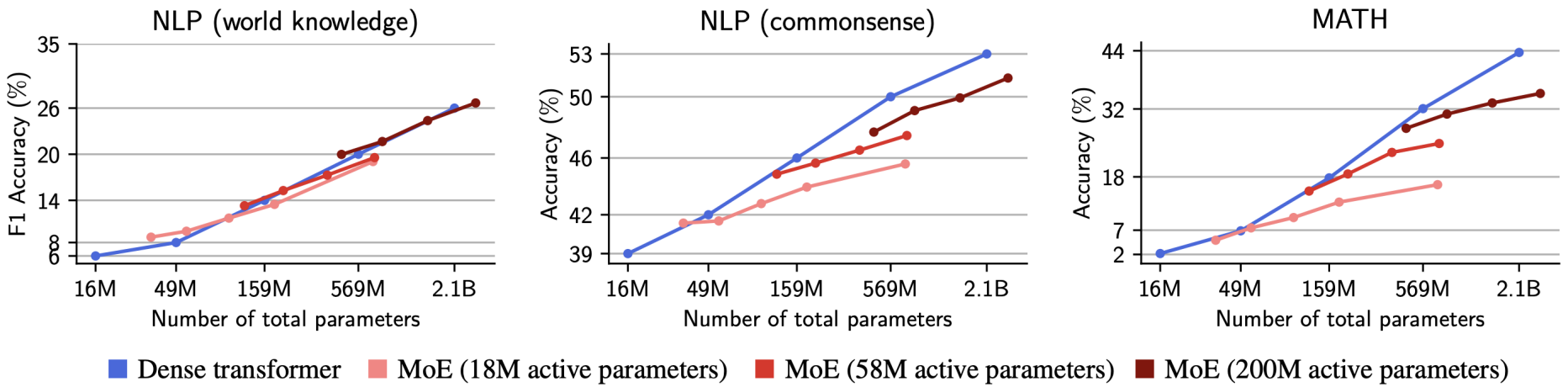
Comparing OLMoE (**left**) (Muennighoff et al., 2024) and Mixtral (**right**) (Jiang et al., 2024a), it looks like specialisation may depend on the expert inventory size or on upcycling (here, from dense Mistral)

Also: few strong co-activation among experts in one layer (no redundancy?)

Vocabulary specialization in later layers (but based on *output* token id)

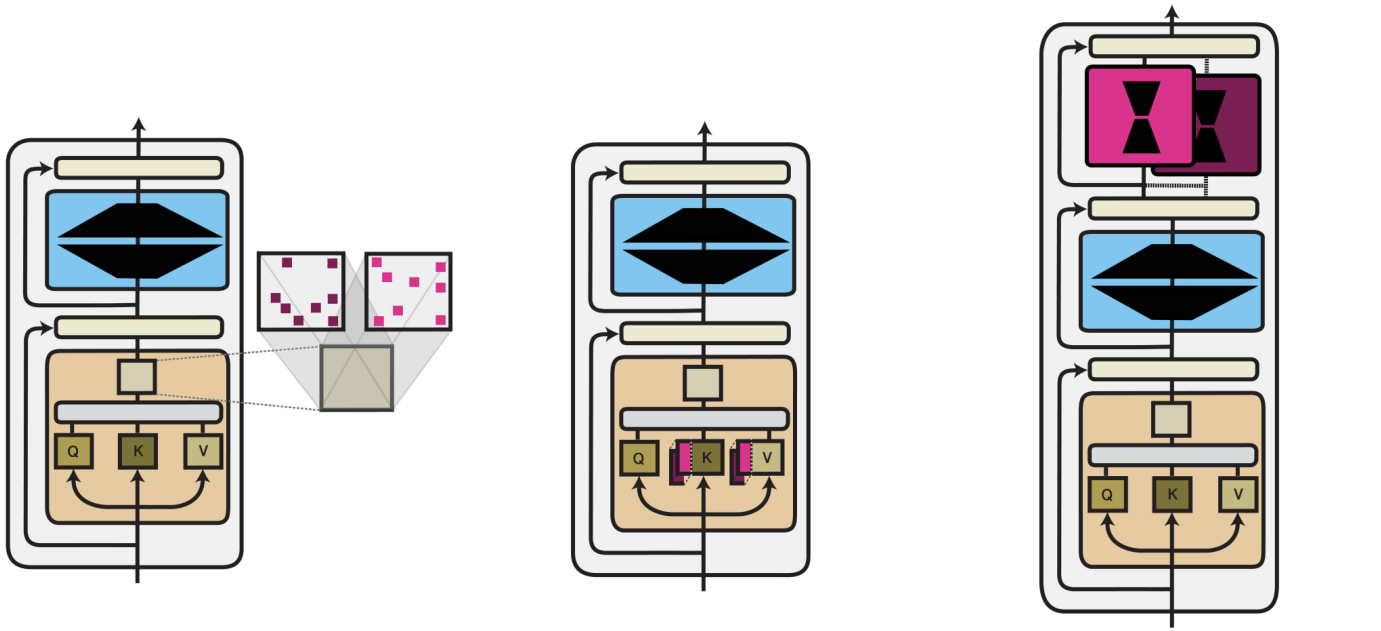
Mixture of parrots (Jelassi et al., 2024)

Sparse MoEs benefit factual tasks (by having larger parameter capacity to memorise information), but not necessarily reasoning tasks



Outline

Adapters and Composition (Pfeiffer et al., 2023)

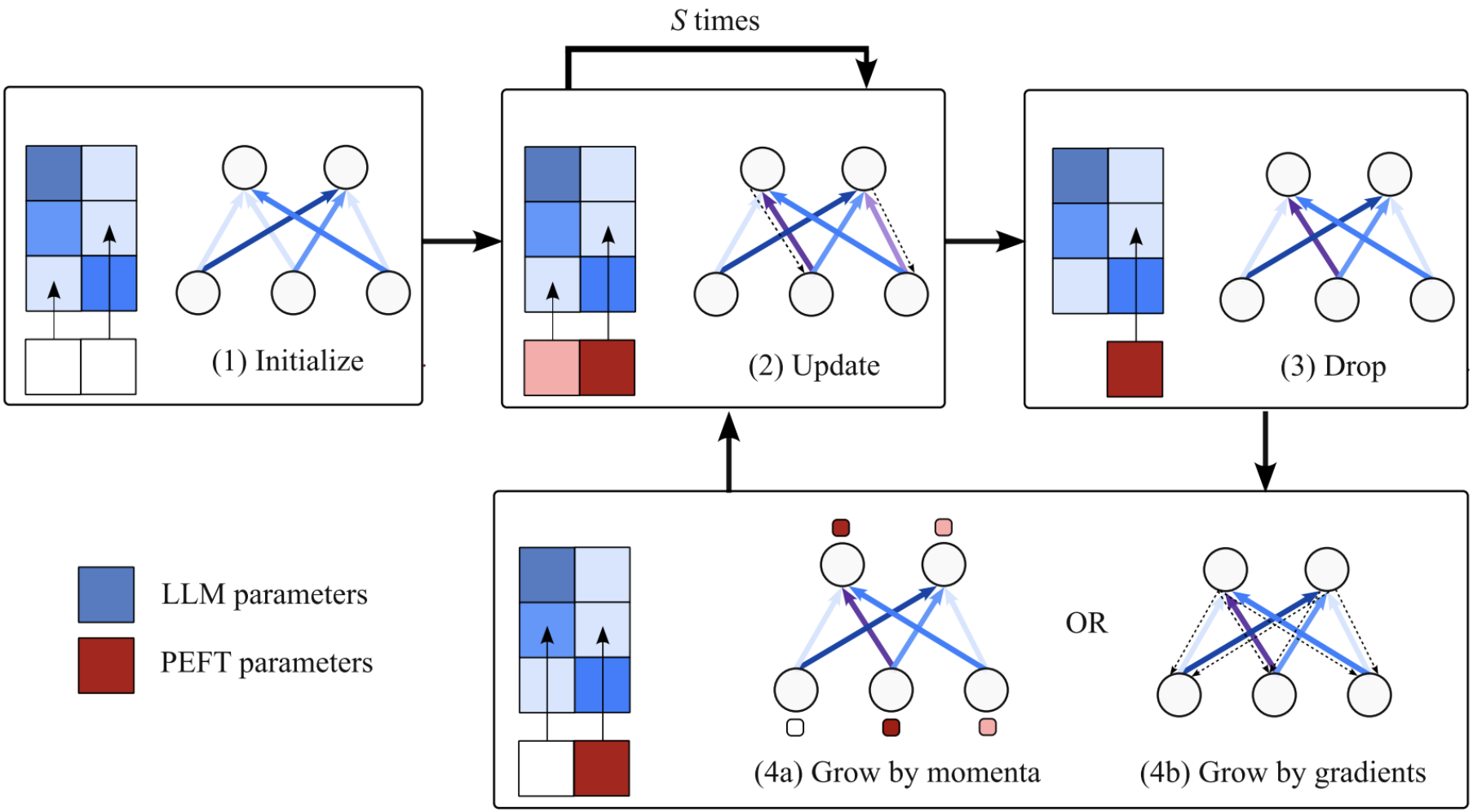


(a) Parameter Composition (b) Input Composition (c) Function Composition

Adapters are parameter-efficient fine-tunings of a backbone model.

They can be trainable tokens, sparse or low-rank weights, extra layers, ...

Example of Adapter: Subnetworks (Ansell et al., 2022, 2024)



Prototypical Architectures Vis-à-vis

Mixtures of Experts

- Token-level routing
- Pre-trained from scratch
- Goal: better scaling

Mixtures of Adapters

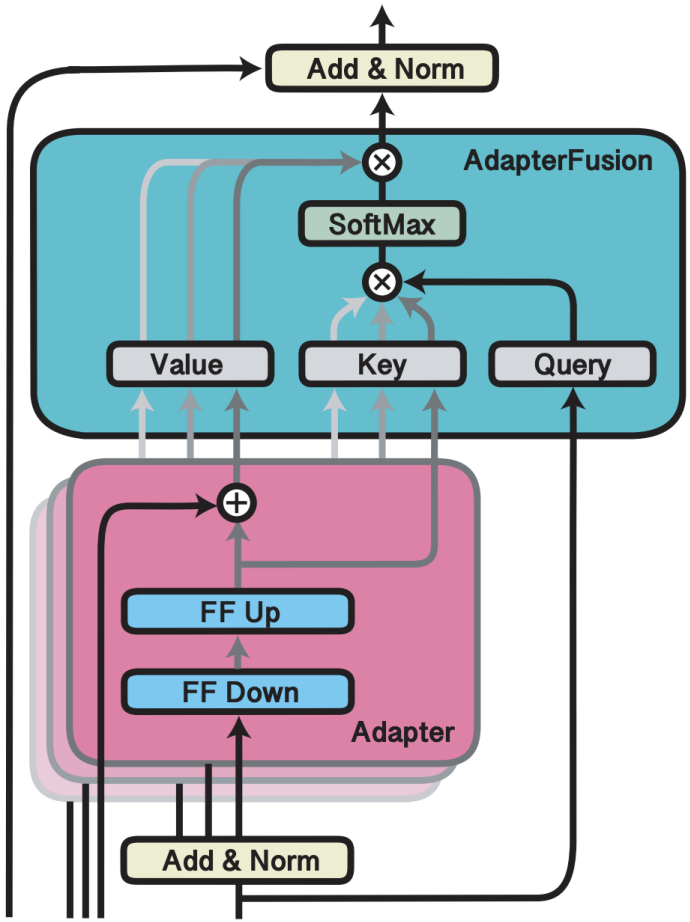
- Task/language/domain-level routing
- 'Modularised' post-hoc
- Goal: better generalisation

Online collections of models like Huggingface Hub are *latent massively multitask models* we can fuse!

AdapterFusion (Pfeiffer et al., 2021)

Adapters can be trained separately and then a routing function can be learned *post hoc*

This leads to compositional generalisation (e.g. recombining language-specific and tasks-specific knowledge for cross-lingual transfer)

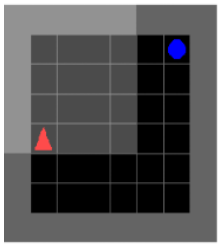


compared to	Fus. w/ ST-A ST-A	Fus. w/ ST-A MT-A	Fus. w/ MT-A ST-A	Fus. w/ MT-A MT-A
MNLI	→	↗	↘	↗
QQP	→	↗	→	↗
SST	↗	→	↗	↗
Winogrande	↘	↗	↘	↗
IMDB	↗	↗	↘	→
HellaSwag	→	↗	↘	↗
SocialIQA	↗	↗	→	↗
CosmosQA	↗	↘	↗	↗
SciTail	→	↗	↗	→
Argument	→	↗	↘	↗
CSQA	↗	↗	↘	↗
BoolQ	↗	↘	↗	↗
MRPC	↗	↗	↘	↗
SICK	↗	↘	↗	↗
RTE	↗	↘	↗	↗
CB	↗	↗	↗	↗
Improved	10/16	11/16	7/16	14/16

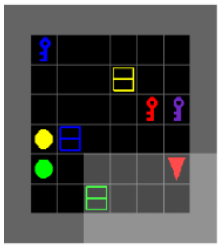
Polytropon (Ponti et al., 2023)

What if the skills a task is composed of are unknown / latent?

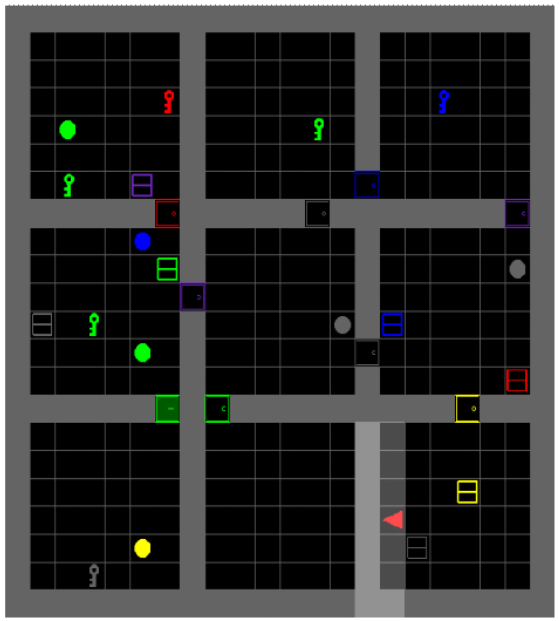
Jointly learn adapters *and* router on a mixture of tasks to discover them.



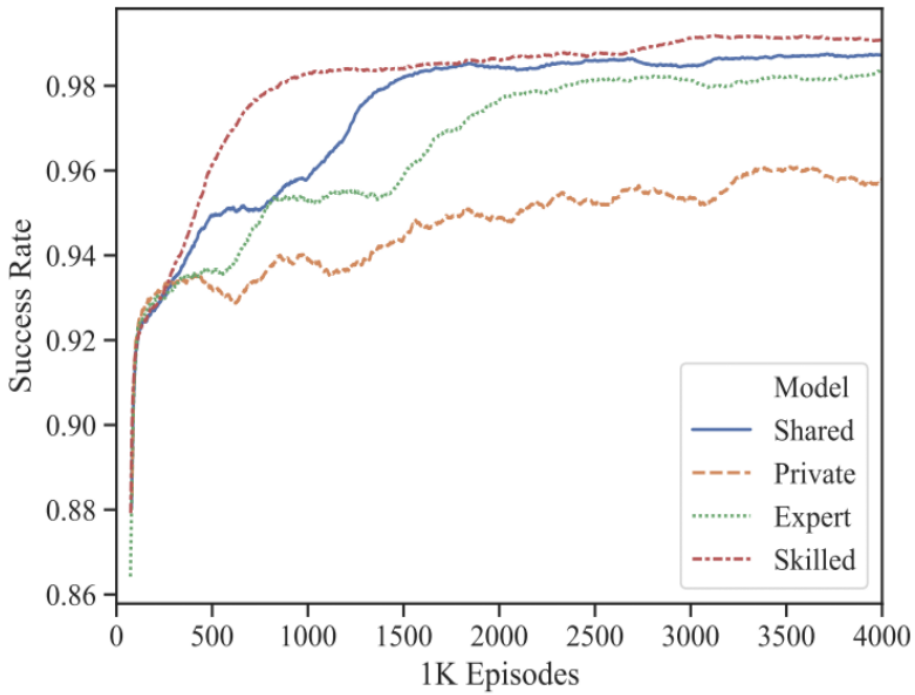
(a) GoToObj: "go to the blue ball"



(b) PutNextLocal: "put the blue key next to the green ball"



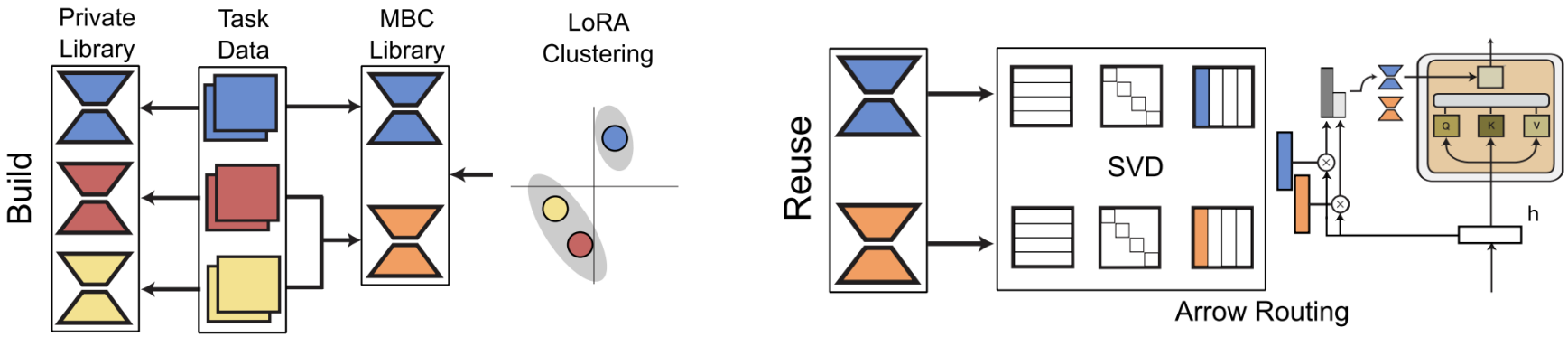
(c) BossLevel: "pick up the grey box behind you, then go to the grey key and open a door". Note that the green door near the bottom left needs to be unlocked with a green key, but this is not explicitly stated in the instruction.



Building and Reusing a Library of Adapters (Ostapenko et al., 2024)

How can we build a library of adapters such that they are more composable? $\rightarrow$ Model-based clustering

How can we reuse the library in a zero-shot setting, where we cannot fine-tune the router? $\rightarrow$ Arrow routing



Interpreting Soft Partitions of Tasks

Routers become a binary encoding of a task identity (note that T tasks can be encoded in $\log_2 S$ skills in the worst-case scenario)

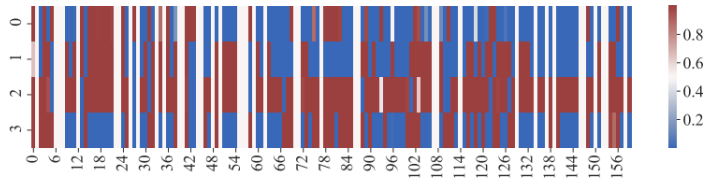


Figure 5: Posterior over Z in SKILLED for $|\mathcal{S}| = 4$.

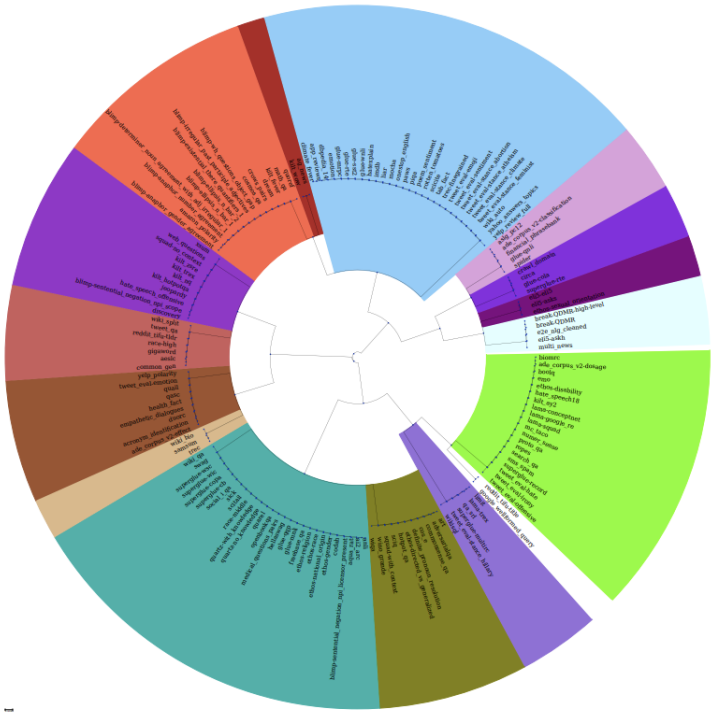
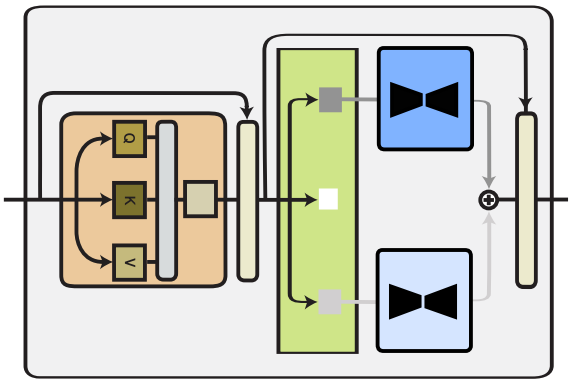
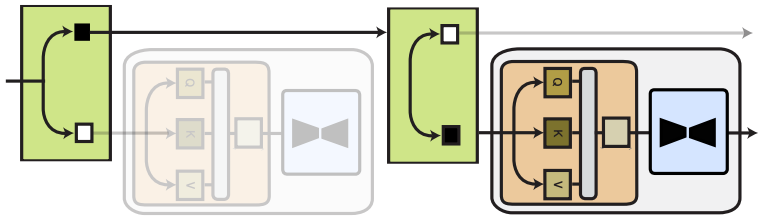


Figure 6: Task partitions for $|\mathcal{S}| = 4$, which corresponds to $2^{|\mathcal{S}|} = 16$ possible subsets of skills.

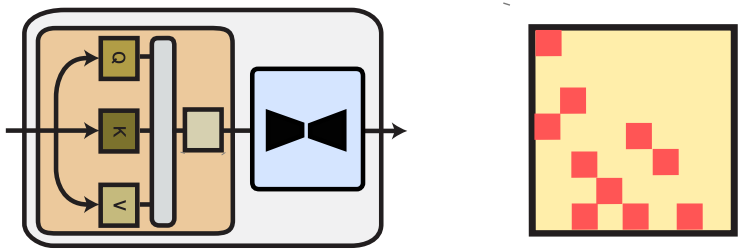
Experts and Adapters ✓



Depth



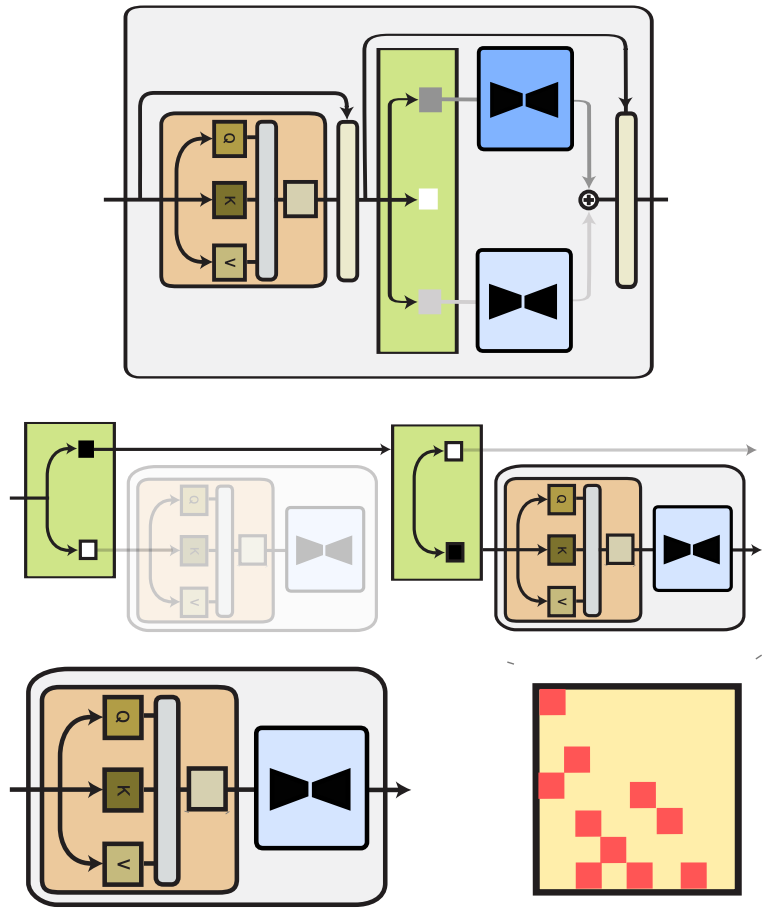
Attention



Experts and Adapters ✓

Depth

Attention



Outline

Conditional computation (Bengio et al., 2013)

Exploration of several estimators for 'stochastic neurons' (e.g., for gating) which result in sparse representations for conditional computation.

For $y = \sigma(x)$, $\hat{y} = \text{round}(y)$, $\mathcal{L} = f(\hat{y})$:

- REINFORCE: $\nabla_x \mathcal{L} \approx \mathcal{L} \cdot (\hat{y} - \sigma(x))$
- straight-through estimator: $\frac{\delta y}{\delta \hat{y}} \approx 1$, so $\nabla_x \mathcal{L} \approx \frac{\delta \mathcal{L}}{\delta \hat{y}} \frac{\delta y}{\delta x}$

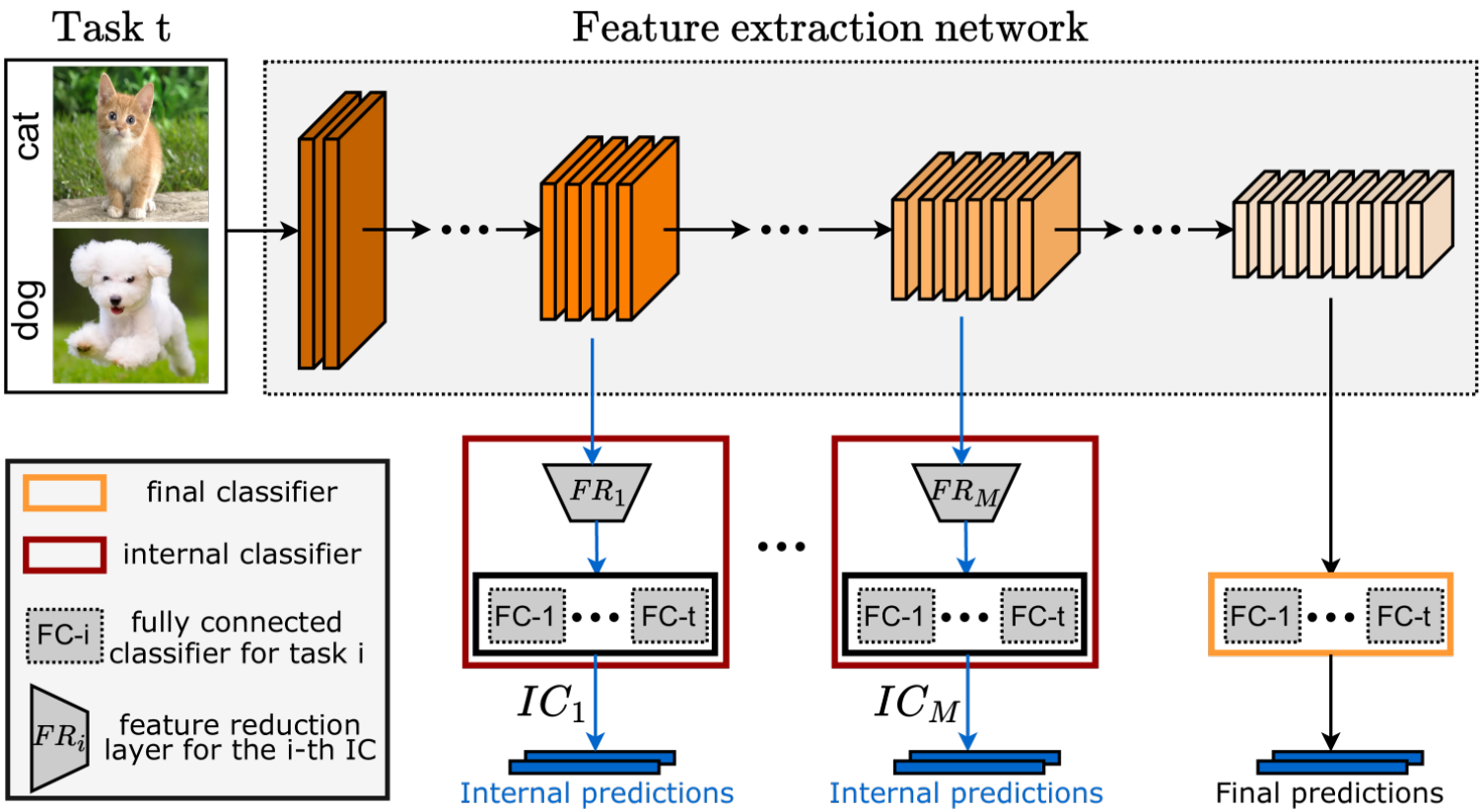
This results in a dynamic computational graph

Some extensions from Bengio et al. (2015): block sparsity and auxiliary losses for sparsity priors

BranchyNet (Teerapittayanon et al., 2016)

Train on a weighted sum of losses for all layers

Entropy as a heuristic for early exit



Spatially Adaptive Computation Time (Figurnov et al., 2017)

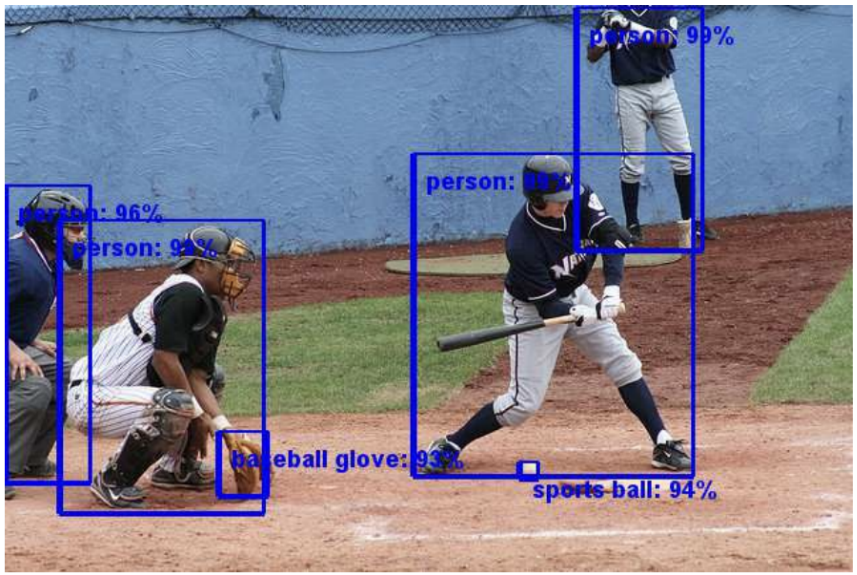
Repurpose Adaptive Computation Time (ACT) (Graves, 2016) to model depth instead of recurrence length.

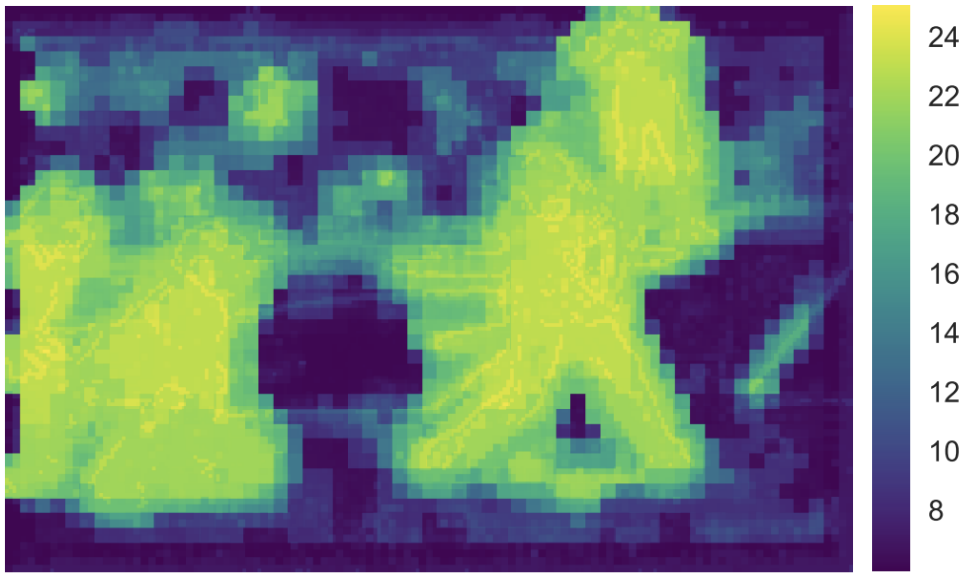
Different depths for different regions of the image

Predict a halting score $h(l) = [0, 1]$ for each layer: skip layers after n once $\sum_{l=1}^{n} h(l) > 1$.

To train halting modules, minimise the 'ponder cost' $\rho = 1 - \sum_{l=1}^{n} h(l)$.

Note that $\frac{\delta \rho}{\delta h(l)} = -\mathbb{1}_{l<n}$



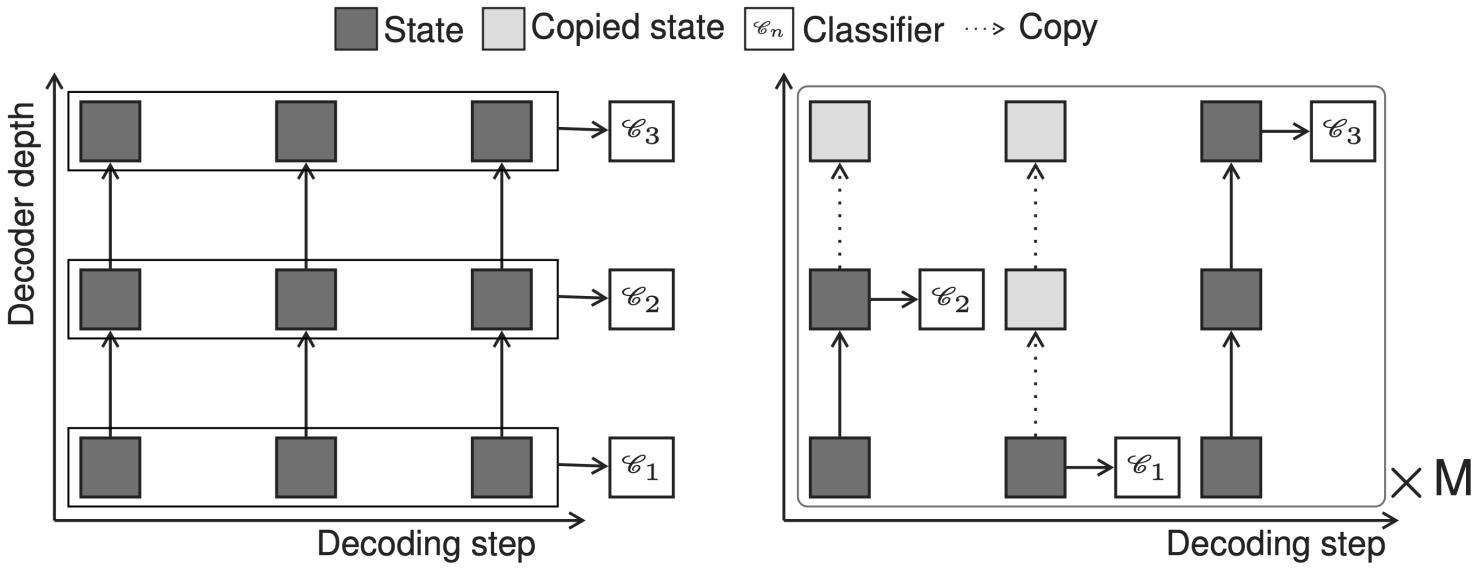


Depth-adaptive Transformer (Elbayad et al., 2020)

Decide depth token-wise: stop if $p(n)_t > \tau$, e.g. Geometric $p(n)_t = h(n)_t \sum_{l=1}^{n}(1 - h(l)_t)$

Halting module trained via cross-entropy wrt an oracle distribution (based on likelihood or correctness)

Auto-regressivity issue: How to deal with non-computed previous hidden states in self-attention? Aligned (as if present) vs mixed (copy over based on sampled exits) training: both are fine! (As also validated in CALM)



(a) Aligned training

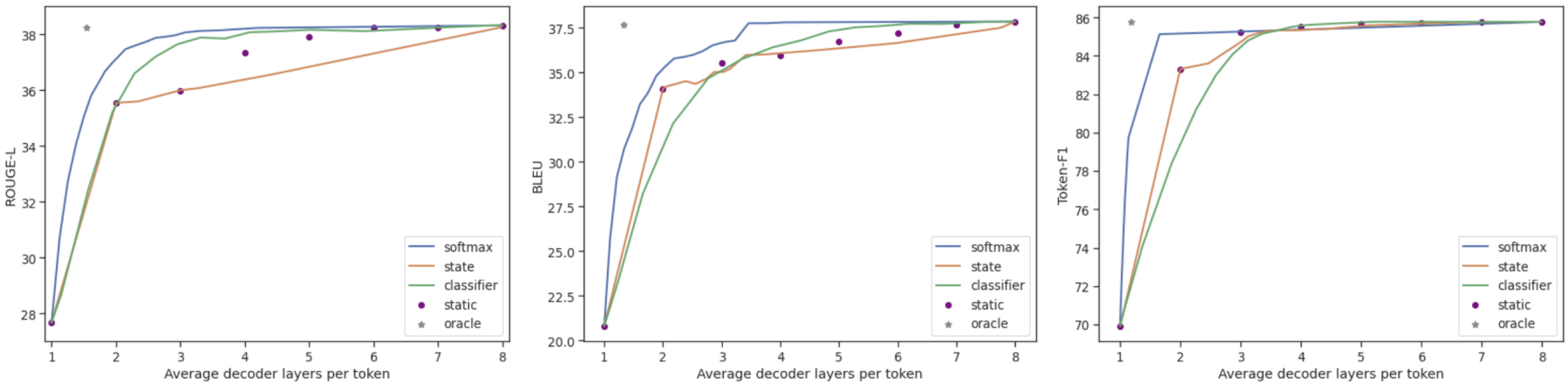
(b) Mixed training

Confident Adaptive Language Models

Generation becomes more resilient to perturbation as time step increases $\rightarrow$ decaying early-exit threshold

Strategies considered:

- uncertainty of each layer prediction: extra FLOPS but parallelisable
- saturation in hidden states of consecutive layers: fast and parameter-free
- parametric halting classifier, trained post-hoc via consistency oracle



(a) CNN/DM (b) WMT (c) SQuAD

Principled strategy for early exiting so that early-exit models:

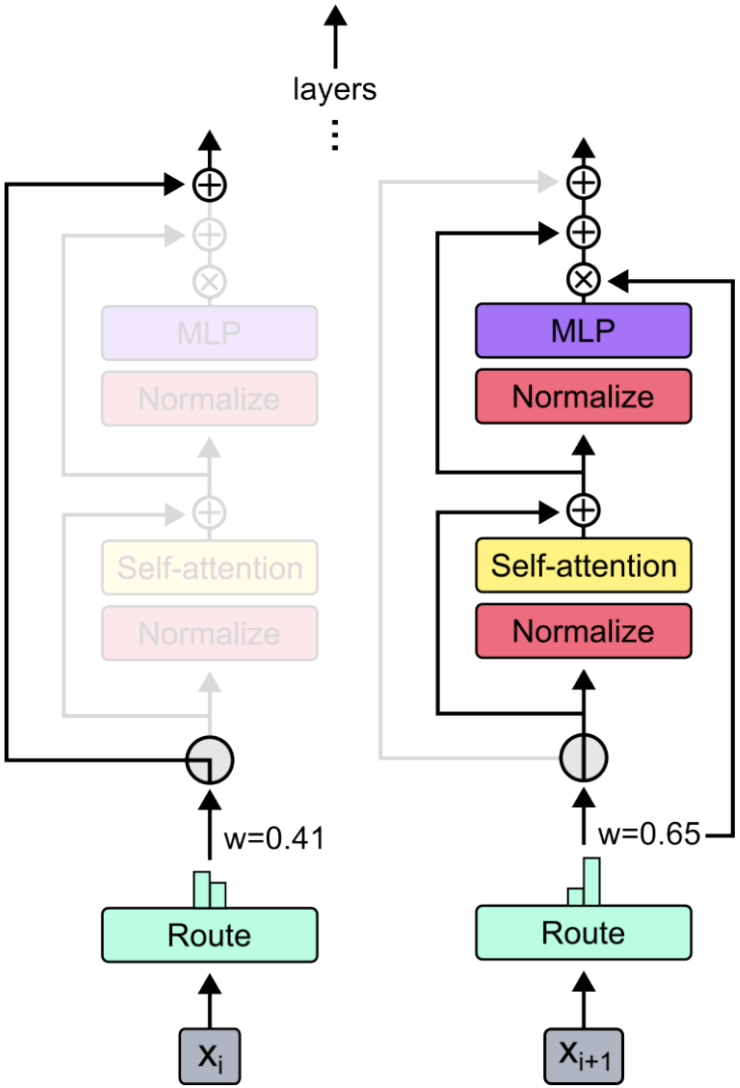
$p(\mathbb{E}[d(\hat{\boldsymbol{y}}_{\text{early}}, \hat{\boldsymbol{y}}_{\text{full}})] \leq \delta \mid \boldsymbol{x}) \geq 1 - \epsilon$ (match outputs of full models)

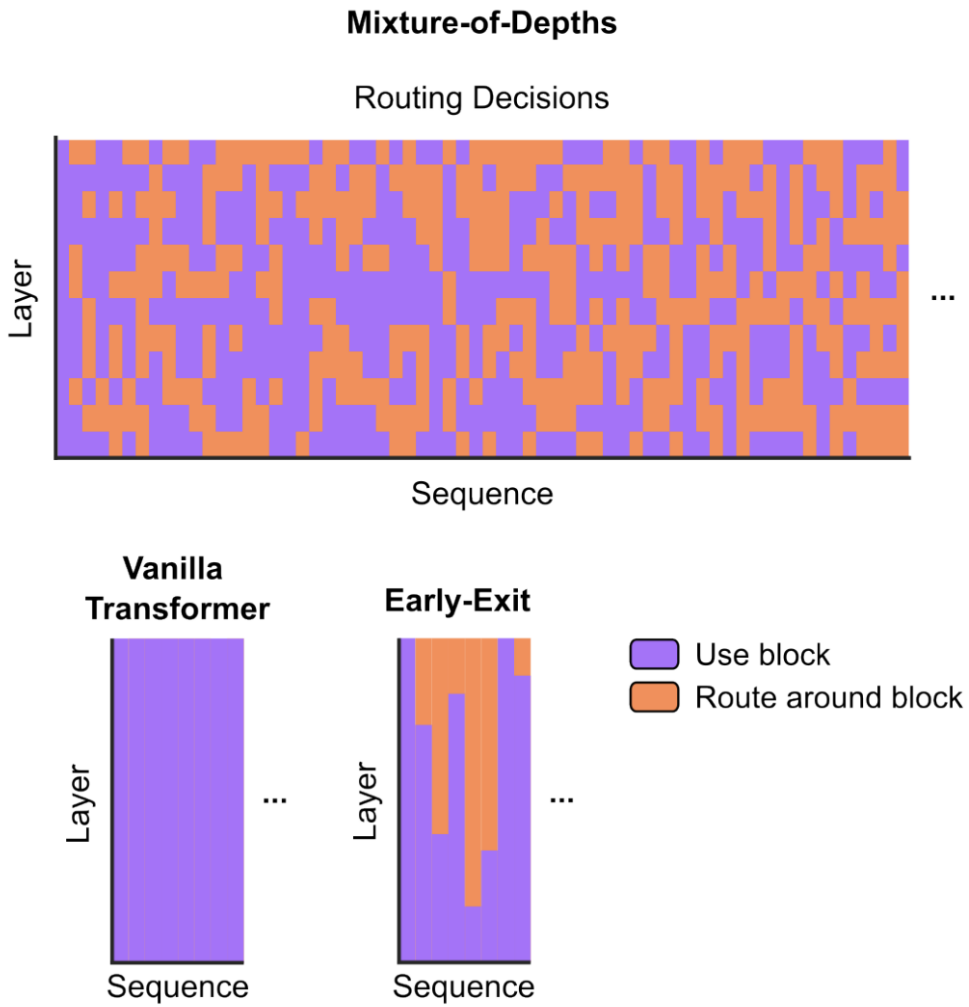
$p(\mathbb{E}[\mathcal{L}(\hat{\boldsymbol{y}}_{\text{early}}, \boldsymbol{y}) - \mathcal{L}(\hat{\boldsymbol{y}}_{\text{full}}, \boldsymbol{y})] \leq \delta \mid \boldsymbol{x}) \geq 1 - \epsilon$ (match loss of full models)

How to pick threshold τ that satisfies δ and ϵ? Hard because non-monotonic.

Learn Then Test (LTT) framework: cast hyper-parameter selection as a multiple testing problem based on a calibration set of inputs–outputs

Mixture of Depths (Raposo et al., 2024)





Mixture of Depths

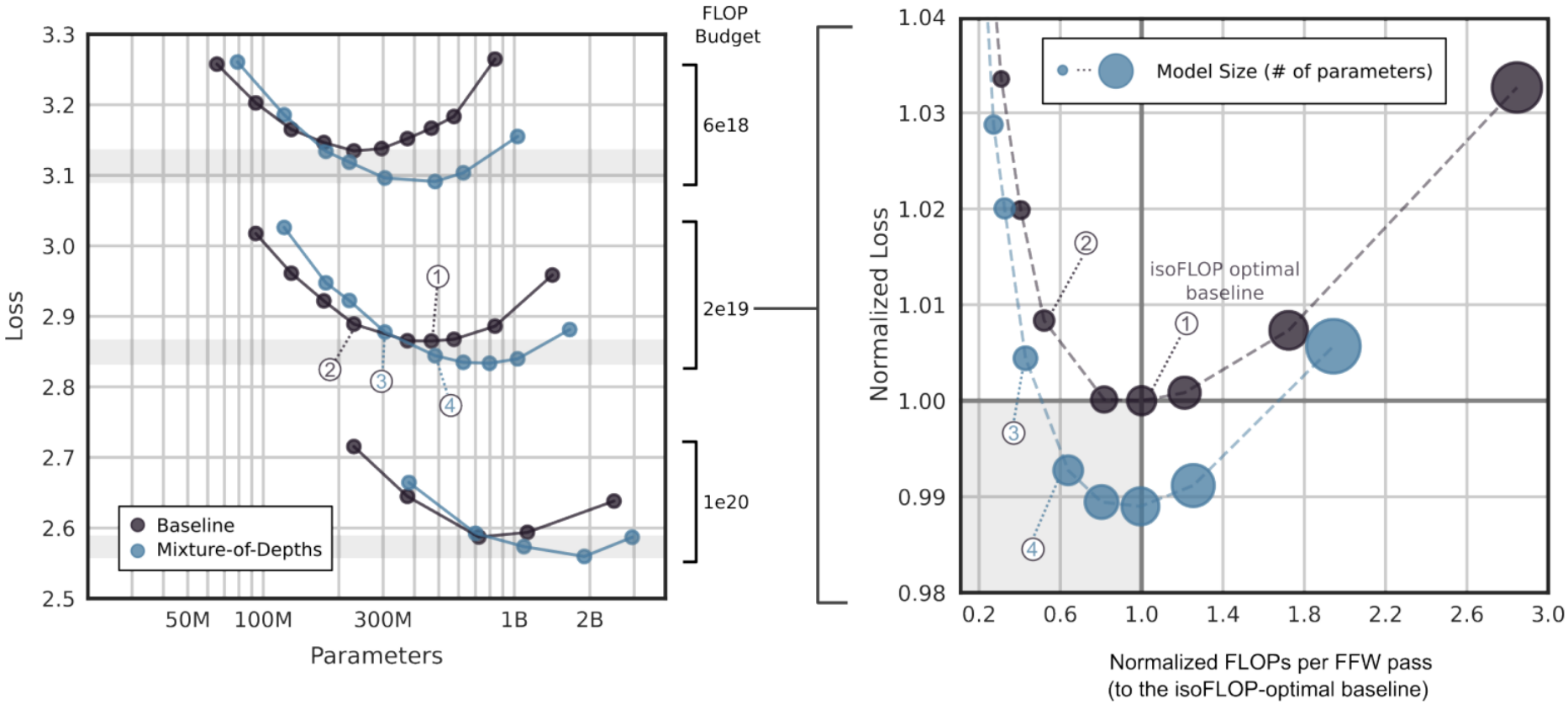
Not just for MLPs but also self-attention

Capping the number of tokens per layer via top-k learned routing. Token chooses path or path chooses token (better for load balancing, but may result in over/under-processing)

How to do this during generation? Top-k is non-autoregressive. Two solutions: auxiliary autoregressive router or auxiliary loss to quantise scores

Static computation graphs: computing resources needed are fixed but the choice of tokens is dynamic $\rightarrow$ the hardware lottery strikes again!

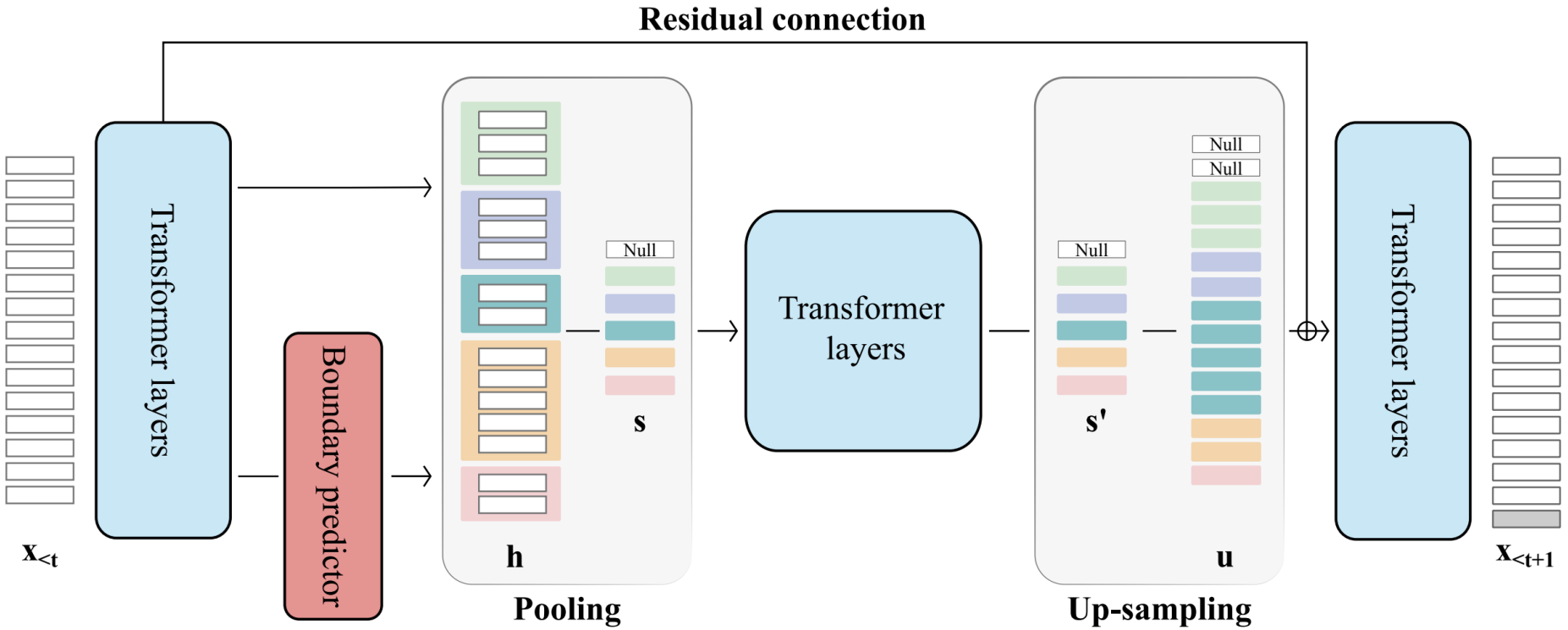
IsoFLOP Analysis



Some MoD variants are both faster to step and better performing than the isoFLOP optimal baseline.

Don't skip, merge! (Nawrot et al., 2023)

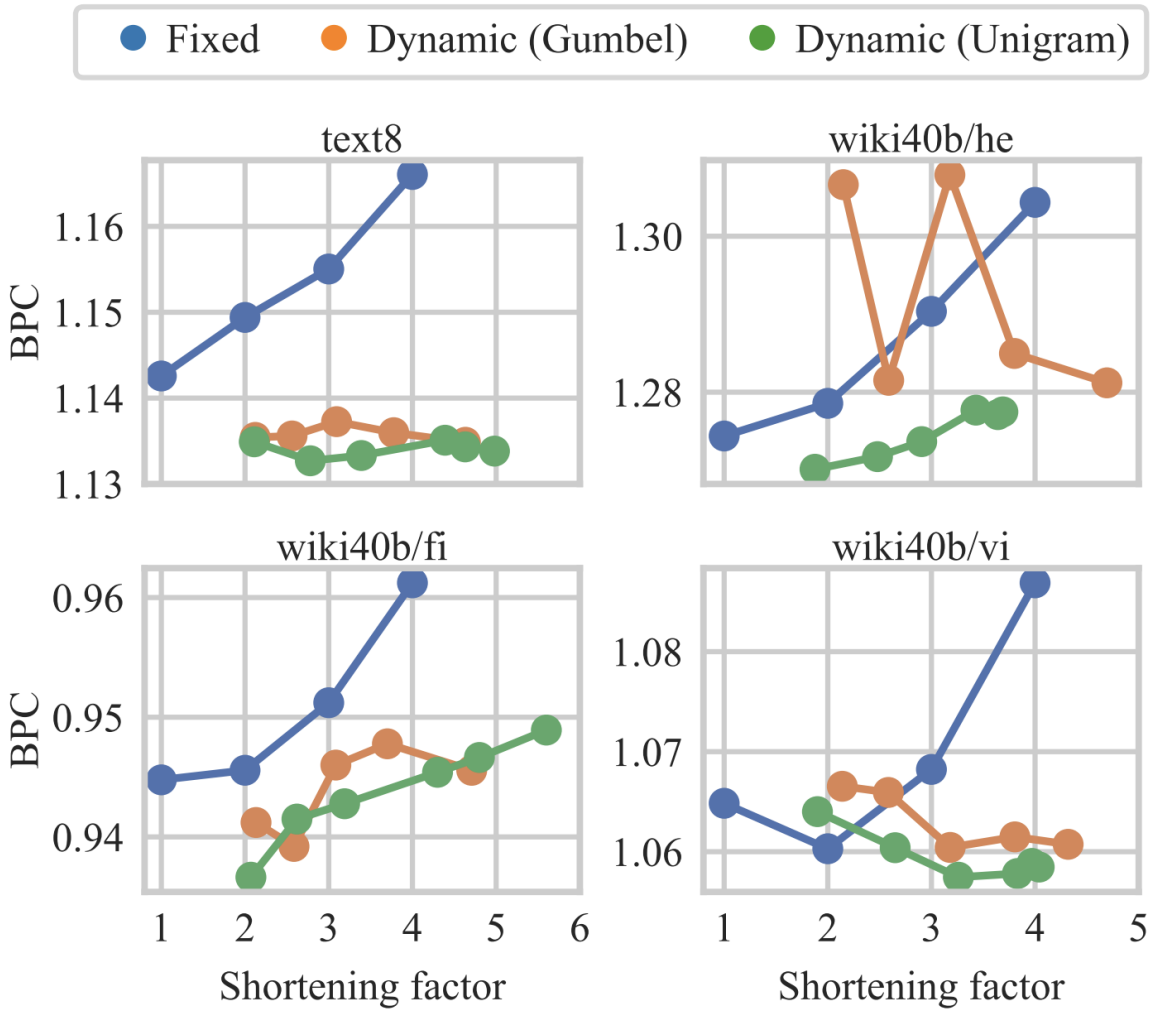
Instead of skipping layers for certain tokens, merge contiguous segments of tokens: dynamic token pooling reduces attention complexity by compression ratio squared.



Efficient also during training! Autoregressivity is preserved with special tokens (Null in the image).

Multilingual language modelling

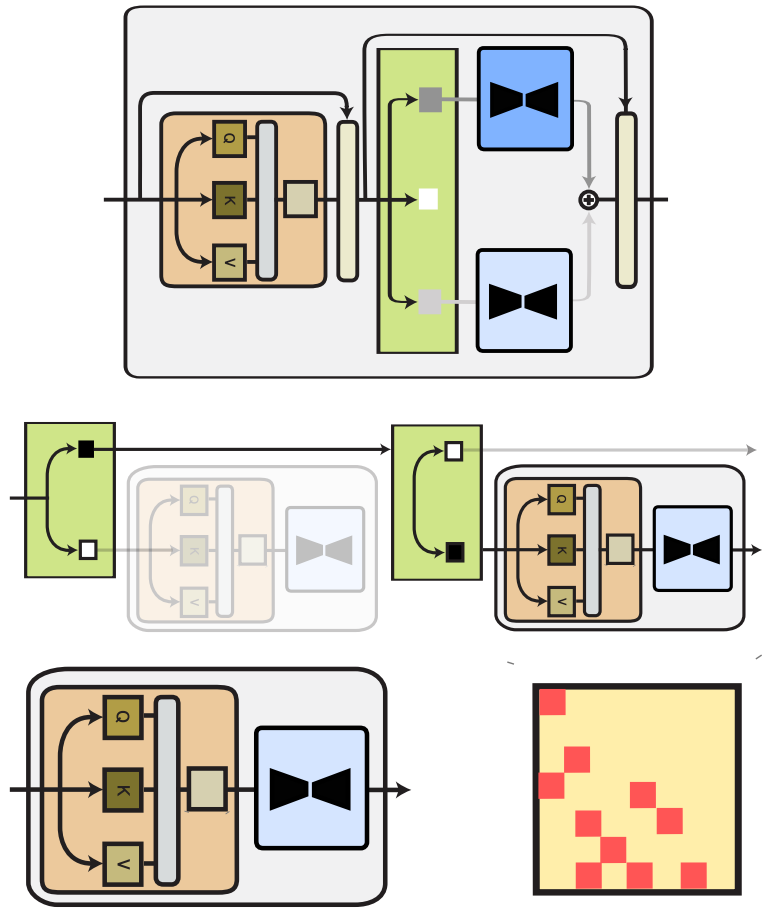
The boundary predictor can be learned end-to-end via stochastic reparameterisation, or via supervision from Unigram tokenizer or entropy spikes.



Experts and Adapters ✓

Depth ✓

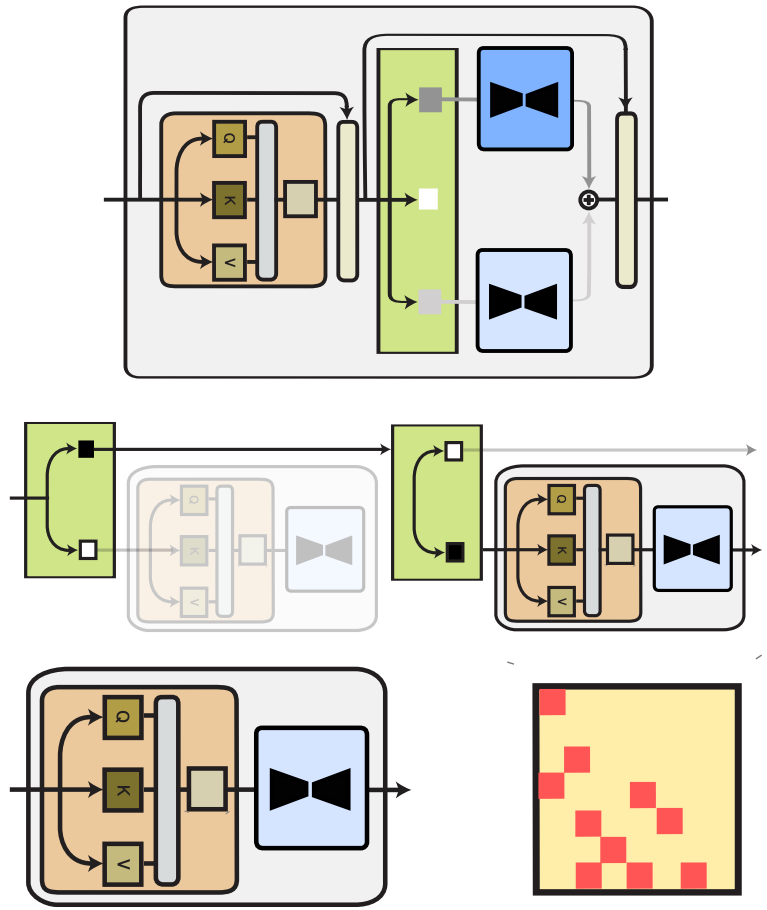
Attention



Experts and Adapters ✓

Depth ✓

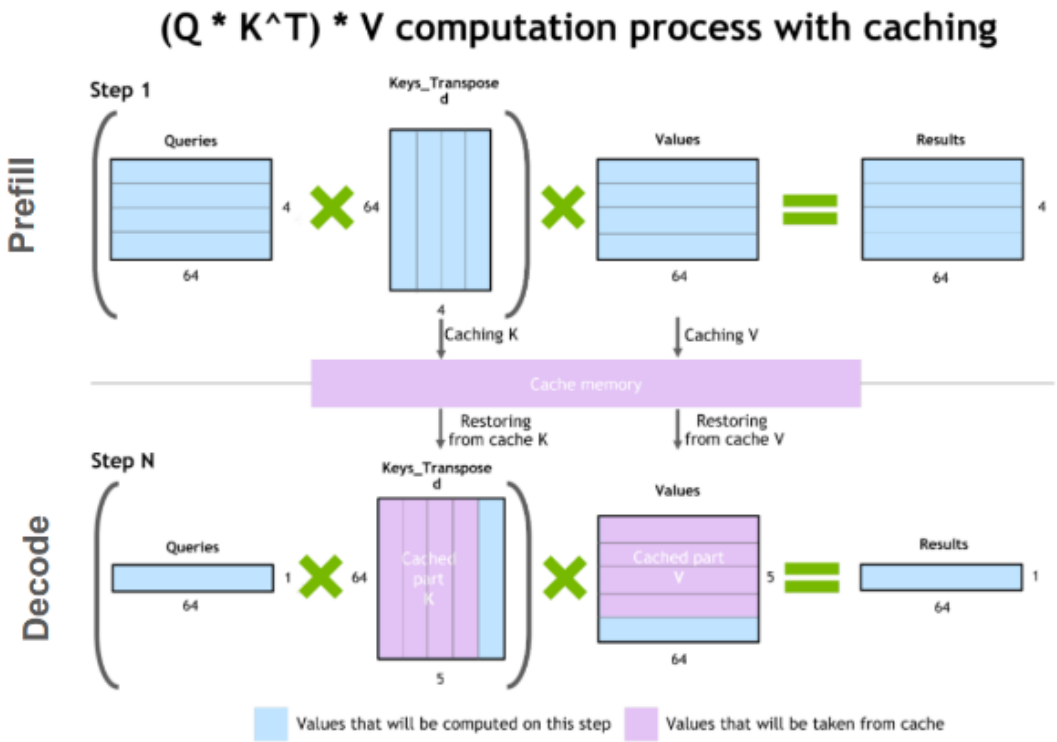
Attention



Outline

Two main phases for LLM deployment

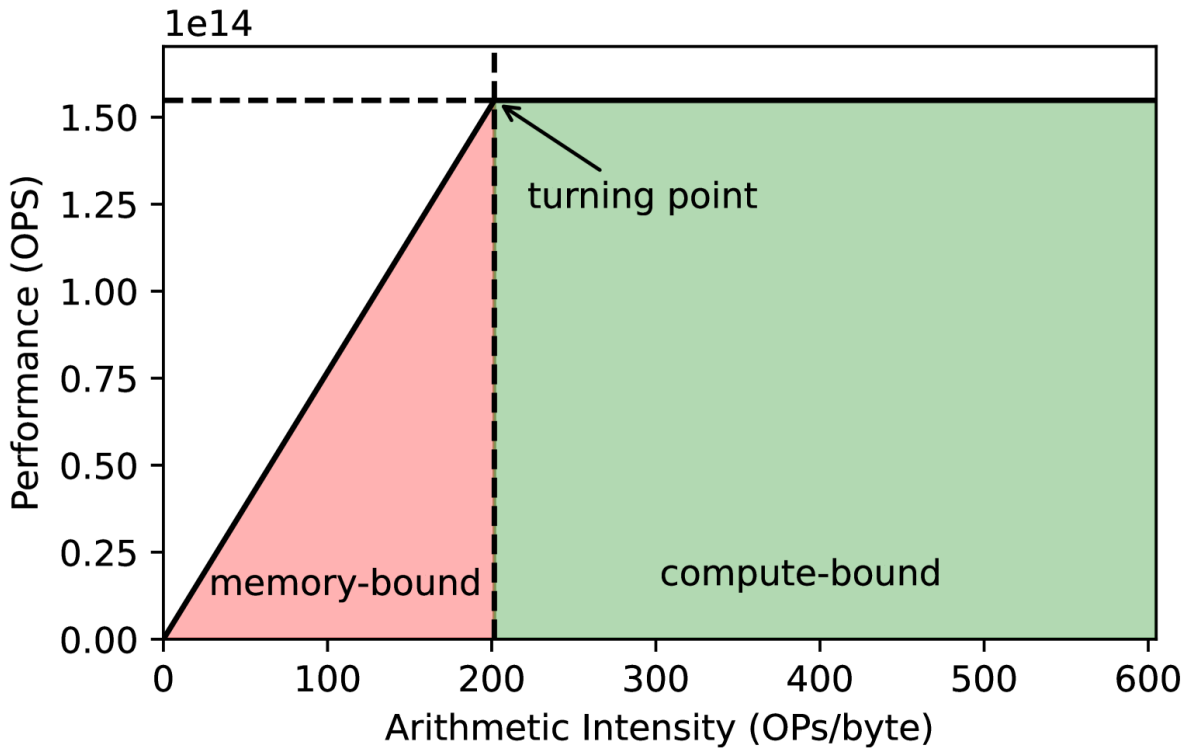
- **Prefilling**: compute keys and values for the input
- **Generation**: autoregressive output generation



From `https://developer.nvidia.com/blog/mastering-llm-techniques-inference-optimization/`

Challenges of Each Phase

- **Prefilling**: computation heavy, dominated by the quadratic complexity of attention wrt the sequence length
- **Generation**: linear wrt sequence length but the key–value cache grows linearly. Attention is memory-bound (retrieving keys and values from memory takes more time than the actual computation)



Sparse Attention

We can sparsify keys–values to alleviate these issues:

- **Prefilling**: identify a subset of important token interactions and approximate the attention output with this subset.
- **Generation**: reduce the memory footprint by discarding less important tokens

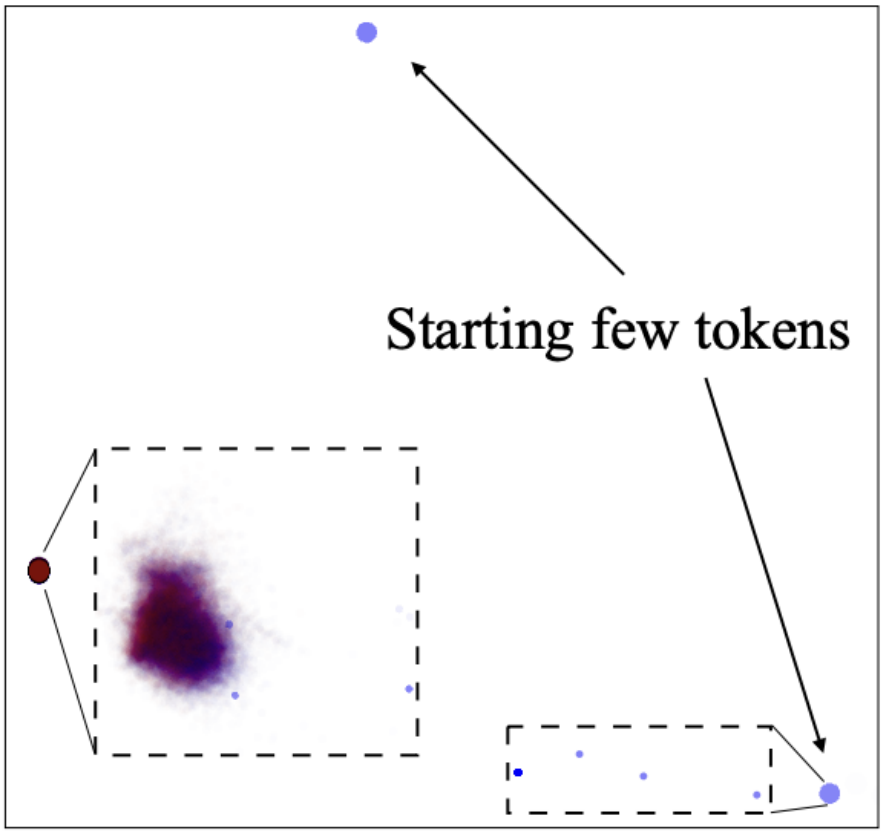
Other benefits include extending LLM "horizon":

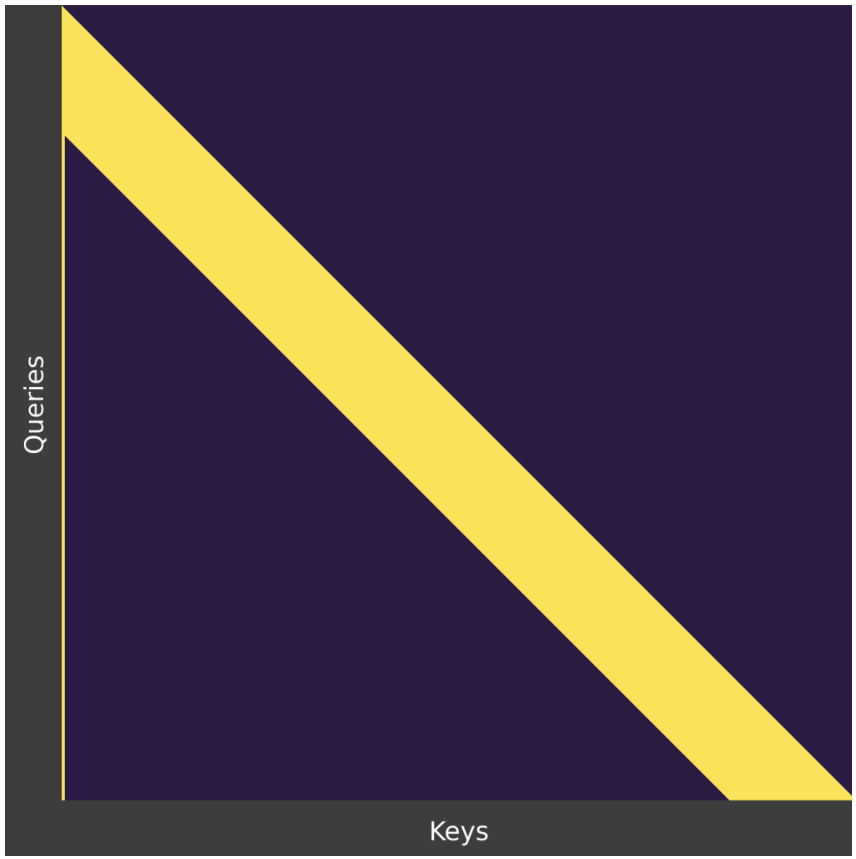
- Prefilling for long context
- Generation for chain-of-thought reasoning

Check out the Jupyter notebook on our tutorial webpage!

Static sparsity in Prefilling

- Local Window: This reflects a Markov assumption where most recent keys–values are always retained
- Sink Tokens (Xiao et al., 2024): initial tokens have strong attention scores (independent of semantics) and are differently distributed (Han et al., 2024) (left plot)





Vertical+Slash Attention (MInference) (Jiang et al., 2024b)

1. calculates partial attention scores
2. Then creates a mask combining two types of connectivity:
   - Attention to top-K most relevant tokens (content-based)
   - Attention along top-K diagonal stripes (structure-based)

Algorithm 2 Vertical-Slash Head

Input: $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{S \times d_h}, k_v, k_s \in \mathbb{N}$

 # Approximate vertical and slash pattern (last_q = 64)

$\widehat{\boldsymbol{A}} \leftarrow \text{softmax}\left(\boldsymbol{Q}_{[-\text{last\_q}:]}\boldsymbol{K}^\top/\sqrt{d} + \boldsymbol{m}_{\text{casual}}\right)$

 # Indices of top k_v vertical line, sum in vertical

$\boldsymbol{i}_v \leftarrow \text{argtopk}\left(\text{sum}_v(\widehat{\boldsymbol{A}}), k_v\right)$

 # Indices of top k_s slash line, sum in slash

$\boldsymbol{i}_s \leftarrow \text{argtopk}\left(\text{sum}_s(\widehat{\boldsymbol{A}}), k_s\right)$

 # Build sparse attention index

$\boldsymbol{i}_{vs} \leftarrow \text{sparseformat}(\boldsymbol{i}_v, \boldsymbol{i}_s)$

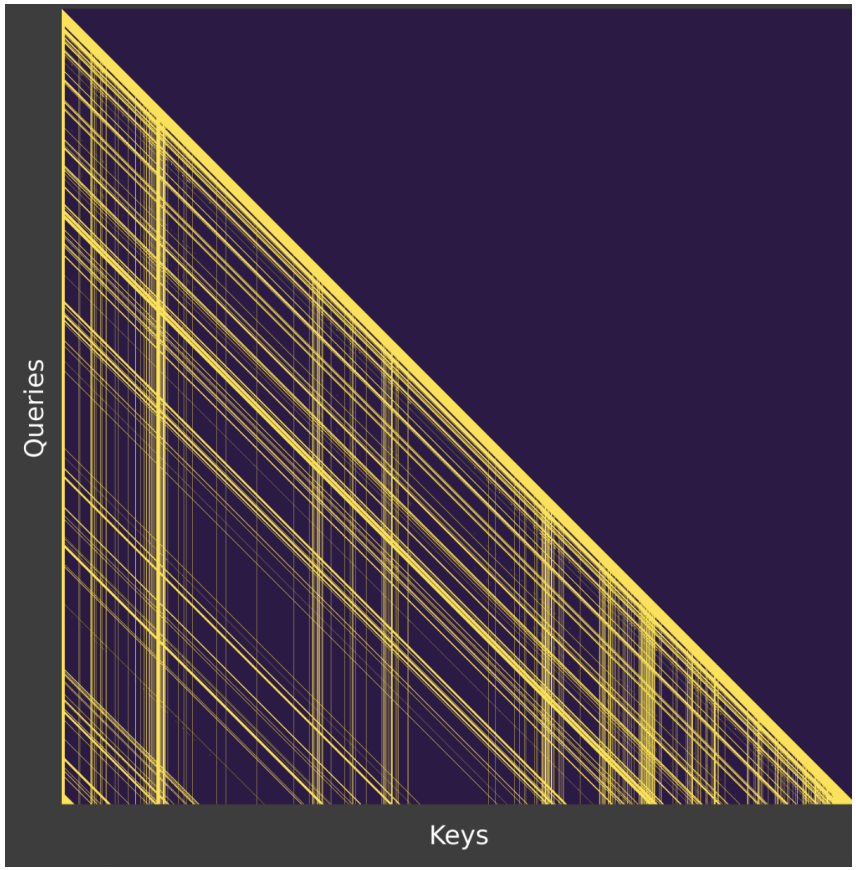
 # Final dynamic sparse attention scores (only index block)

$\boldsymbol{A} \leftarrow \text{softmax}\left(\text{sparse}(\boldsymbol{Q}\boldsymbol{K}^\top, \boldsymbol{i}_{vs})/\sqrt{d}\right)$

 # Sparse mixed scores and values

$\boldsymbol{y} \leftarrow \text{sparse}(\boldsymbol{A}\boldsymbol{V}, \boldsymbol{i}_{vs})$

return $\boldsymbol{y}$



Block Sparse Attention (Jiang et al., 2024b)

1. Divides the sequence into fixed-size chunks.
2. Computes chunk-level attention scores with averaged token representations.
3. Allows each chunk to attend to the top-K most relevant chunks.

Algorithm 3 Block-Sparse Head

Input: $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{S \times d_h}$, $k_b \in \mathbb{N}$

 # Approximate block-sparse pattern (block_size = 64)

$\widehat{\boldsymbol{Q}} \leftarrow \text{MeanPooling}(\boldsymbol{Q}, block\_size)$

$\widehat{\boldsymbol{K}} \leftarrow \text{MeanPooling}(\boldsymbol{K}, block\_size)$

$\widehat{\boldsymbol{A}} \leftarrow \text{softmax}\left(\widehat{\boldsymbol{Q}}\widehat{\boldsymbol{K}}^\top / \sqrt{d} + \boldsymbol{m}_{\text{casual}}\right)$

 # Indices of top k blocks

$\boldsymbol{i}_b \leftarrow \text{argtopk}\left(\widehat{\boldsymbol{A}}, k_b\right)$

 # Build sparse attention index

$\boldsymbol{i}_b \leftarrow \text{sparseformat}(\boldsymbol{i}_b)$

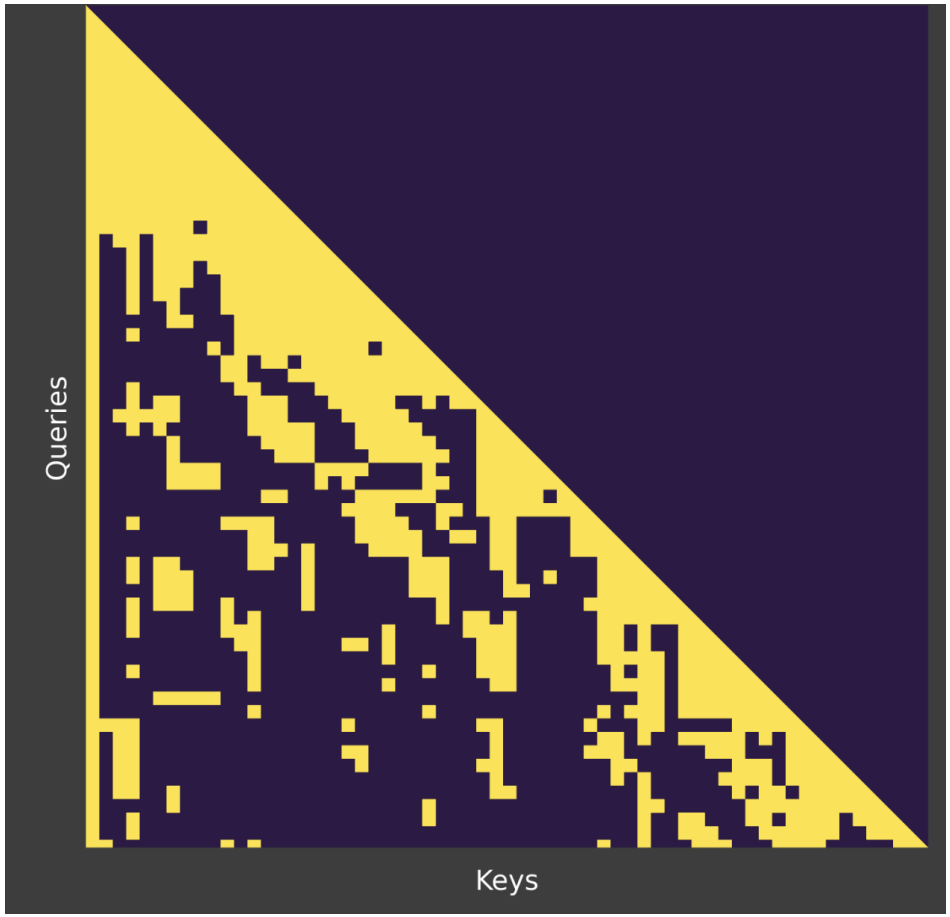
 # Final dynamic sparse attention scores (only index block)

$\boldsymbol{A} \leftarrow \text{softmax}\left(\text{sparse}(\boldsymbol{Q}\boldsymbol{K}^\top, \boldsymbol{i}_b)/\sqrt{d}\right)$

 # Sparse mixed scores and values

$\boldsymbol{y} \leftarrow \text{sparse}(\boldsymbol{A}\boldsymbol{V}, \boldsymbol{i}_b)$

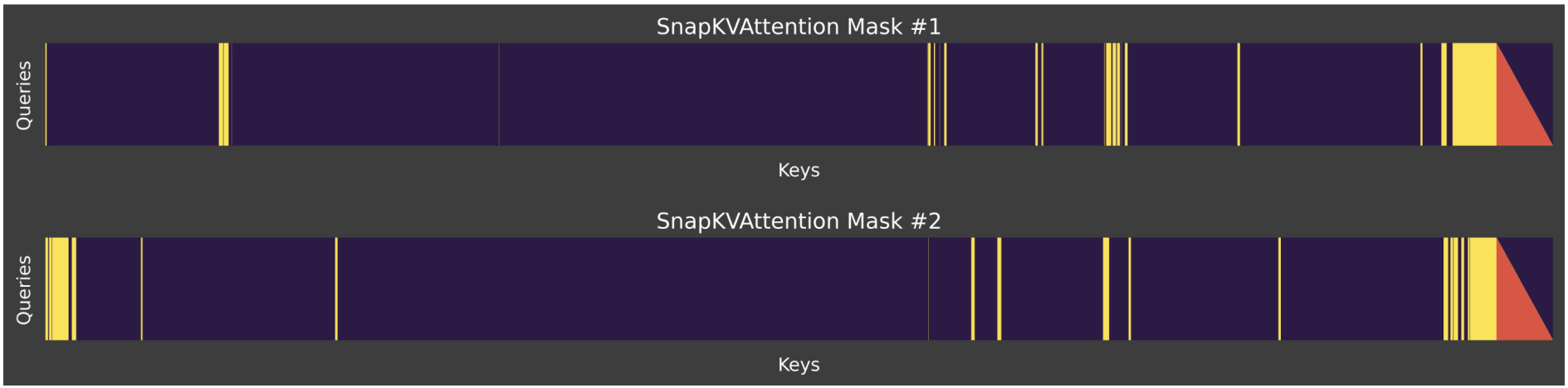
return $\boldsymbol{y}$



SnapKV Attention (Li et al., 2024)

Now turning to sparse attention during generation!

1. Approximates attention scores using a suffix window of queries
2. Uses attention score pooling to identify important token clusters
3. Keeps only the most relevant tokens plus a recent window

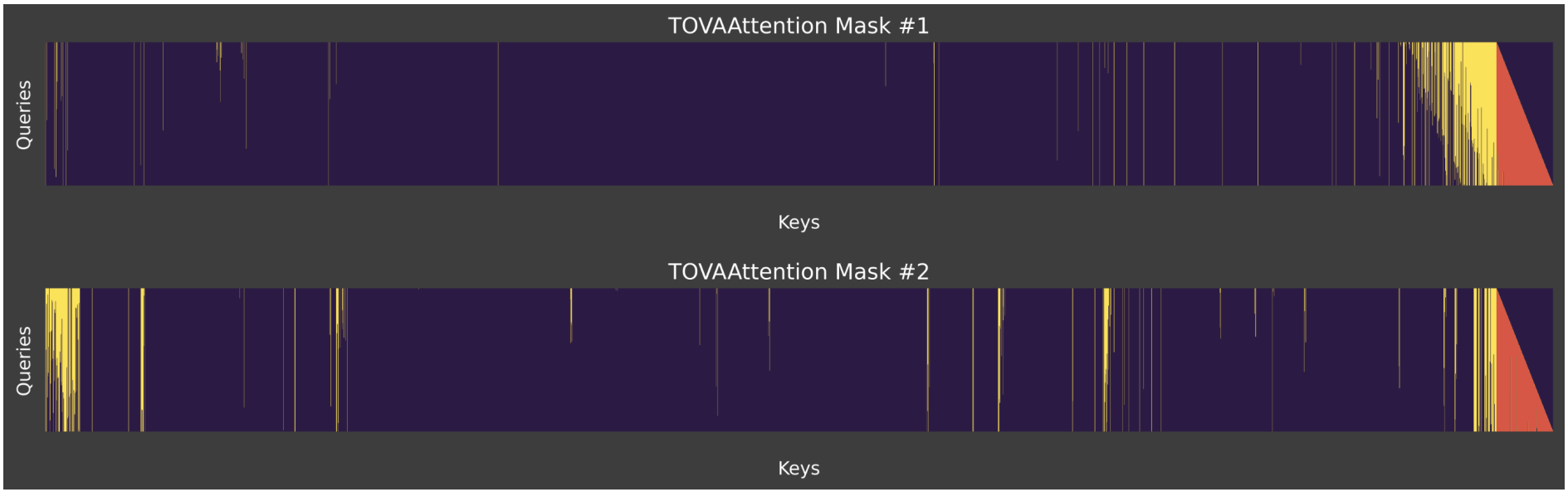


Interaction colours: Red (allowed generation query – generation key), Yellow (allowed generation query – prefilling key), Blue (forbidden)

TOVA Attention (Oren et al., 2024)

Dynamically prunes the KV cache during generation by:

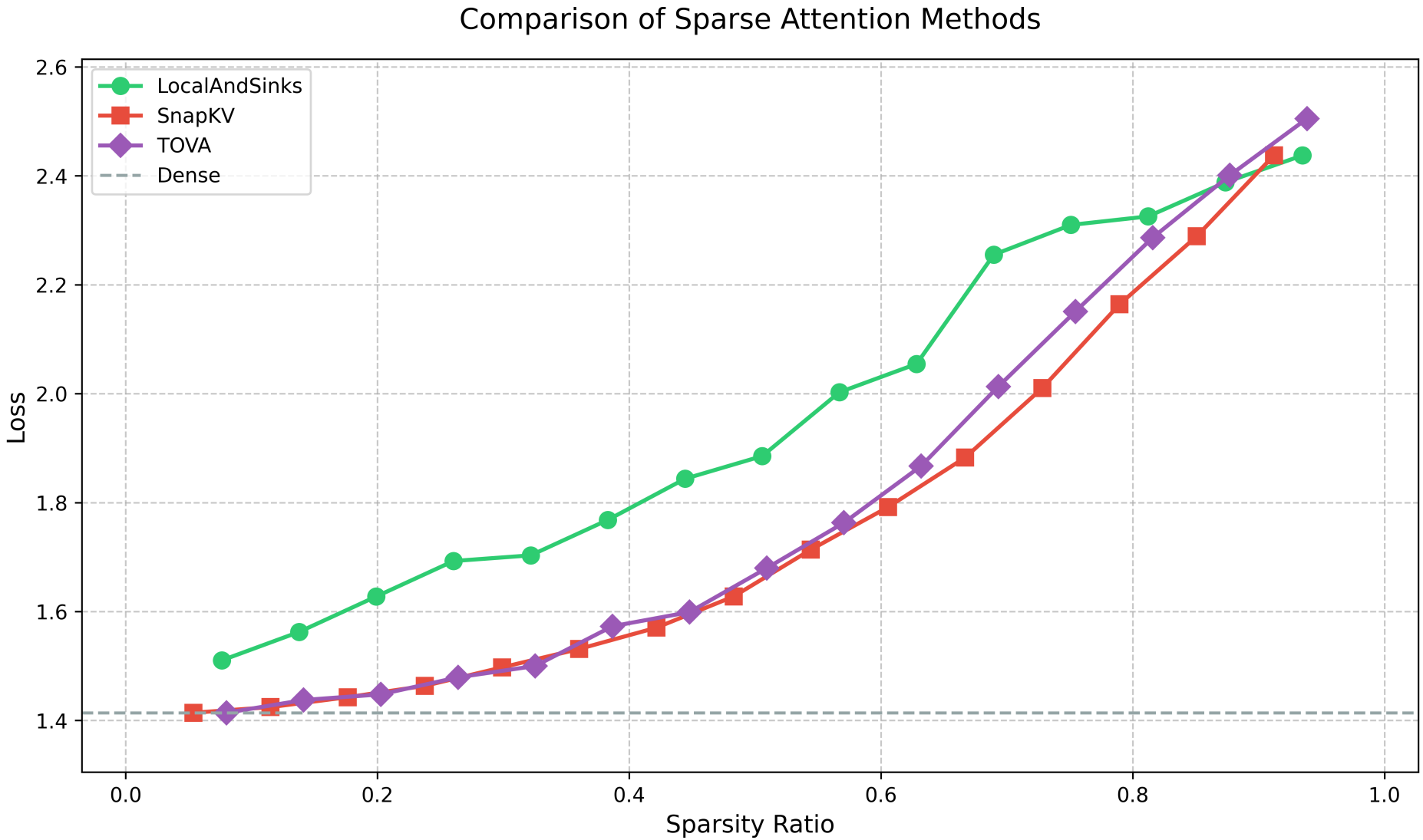
1. Uses the last prefilling token to identify initially important tokens
2. During generation, maintains a fixed-size cache by removing the least attended token after processing each new token



Note some vertical lines are cut off!

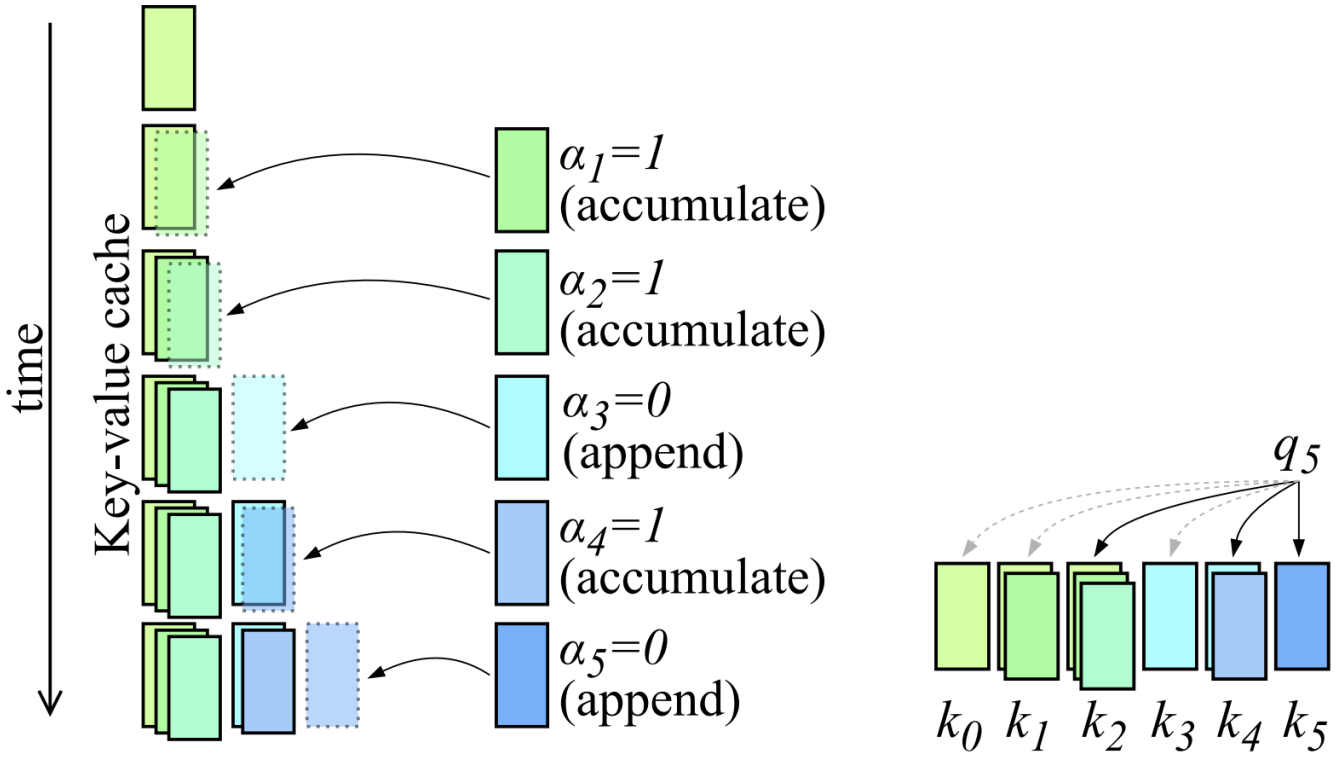
Comparison on Summarisation Task

- Dynamic methods define a better Pareto frontier than static ones
- Yet, they still degrade performance beyond low sparsity ratios



Dynamic Memory Compression (Nawrot et al., 2024)

- Compress representations instead of dropping them.
- Decide if to append or accumulate new keys–values to the cache



Generation (left) vs Training (right)

During training, the binary decision variable α is continuously relaxed

To parallelise training, partly accumulated tokens are kept, too

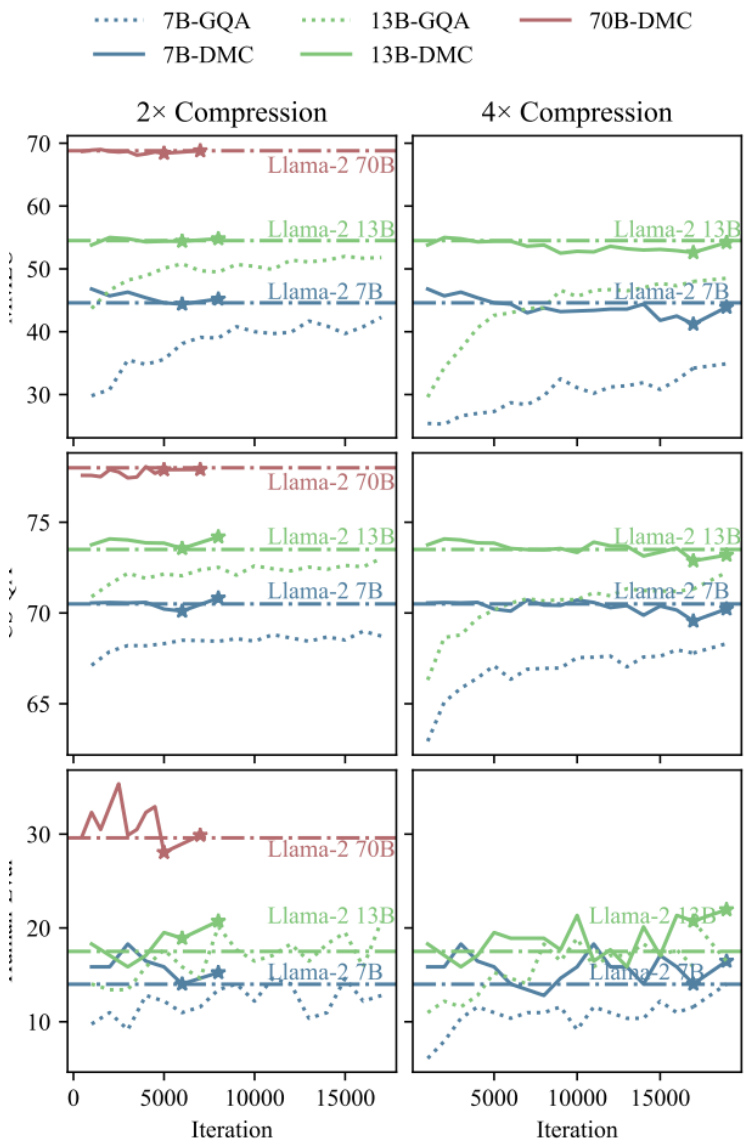
α modulates query–key interactions through an extra mask to progressively remove items inaccessible at inference time

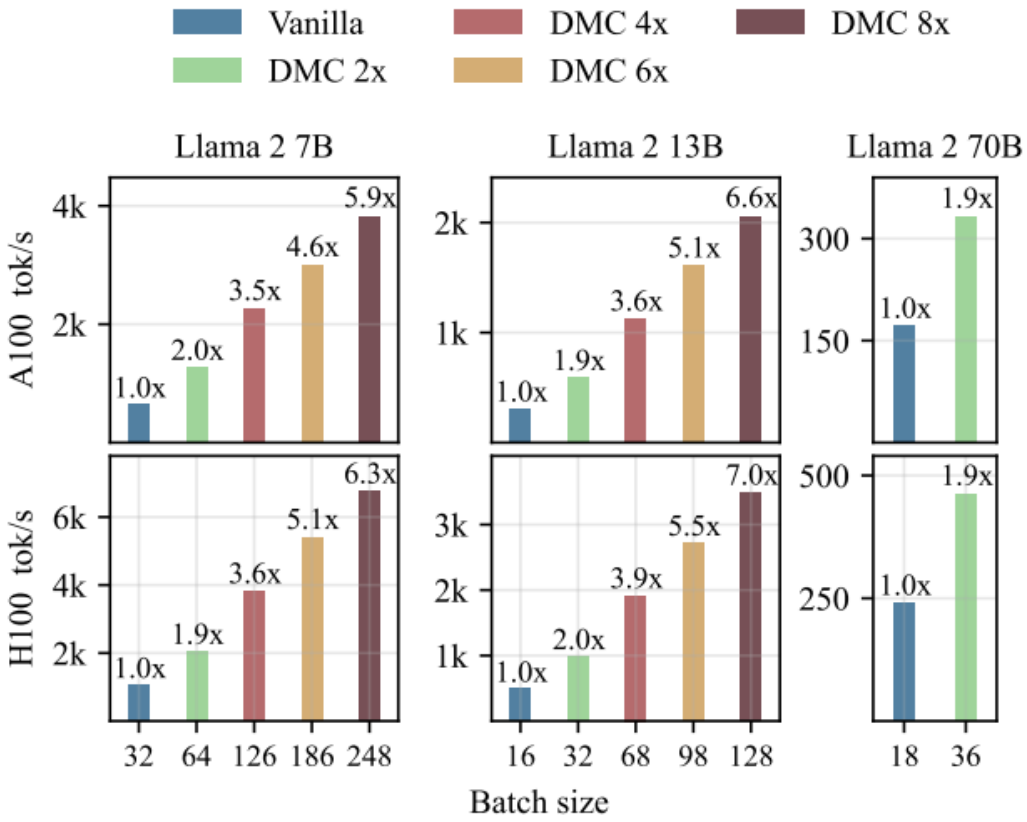
$$
\begin{aligned}
z_0 &\leftarrow \omega_0, & z_i &\leftarrow z_{i-1}\alpha_i + \omega_i, \\
\mathbf{k}_0 &\leftarrow \mathbf{k}_0, & \mathbf{k}_i &\leftarrow \frac{\alpha_i \mathbf{k}_{i-1} z_{i-1} + \mathbf{k}_i \omega_i}{z_i}, \\
\mathbf{v}_0 &\leftarrow \mathbf{v}_0, & \mathbf{v}_i &\leftarrow \frac{\alpha_i \mathbf{v}_{i-1} z_{i-1} + \mathbf{v}_i \omega_i}{z_i}.
\end{aligned}
$$

$$
\begin{array}{c} \\ \mathbf{q_0} \\ \mathbf{q_1} \\ \mathbf{q_2} \\ \vdots \\ \mathbf{q_n} \end{array}
\begin{array}{c}
\begin{array}{ccccc} \mathbf{k_0} & \mathbf{k_1} & \mathbf{k_2} & \dots & \mathbf{k_n} \end{array} \\
\begin{bmatrix}
0 & -\infty & -\infty & \dots & -\infty \\
\log(1 - \alpha_1) & 0 & -\infty & & -\infty \\
\log(1 - \alpha_1) & \log(1 - \alpha_2) & 0 & & -\infty \\
\vdots & \vdots & & \ddots & \vdots \\
\log(1 - \alpha_1) & \log(1 - \alpha_2) & \log(1 - \alpha_3) & \dots & 0
\end{bmatrix}
\end{array}
$$

Foundation models can be retrofitted with DMC behaviour with negligible extra data

Result on LLM Benchmarks





Resources

- Training-free methods for memory sparsification:
  `https://github.com/NVIDIA/kvpress`
- Retrofitting for memory compression:
  `https://github.com/NVIDIA/Megatron-LM/tree/dmc`

Outline

Summary: Part II

- In foundation model architectures, sparsity occurs among experts/adapters, in a model's depth, or among tokens in attention;
- Will Sparse MoEs become the default? Scaling laws and post-training are reasons for optimism, but CAP trade-offs, lack of specialisation, and factual vs reasoning tasks should make us cautious.
- Mixtures-of-adapters could unlock the potential of asynchronous and independently developed, massively multitask LLMs
- Sparse memory has different challenges and solutions in prefilling and generation stages $\rightarrow$ may be the key to long context and long reasoning.
- Dynamic sparsification is a special case of dynamic compression, which requires extra training
- The hardware lottery still plays a significant role in the design of MoEs and attention

Thank you!

We have open PhD and post-doc positions in the SARDINE Lab!

I am looking for PhDs at Edinburgh NLP, too!

Acknowledgments






- Sara Hooker, Alessandro Sordoni, Zita Marinho (panelists)
- Duarte Alves, Piotr Nawrot, Alessandro Sordoni, Saul Santos (for Jupyter notebooks)
- DECOLLAGE ERC-CoG project (2023–28) ▸ Link

Discussion Panel



- Sara Hooker, Alessandro Sordoni, Zita Marinho.

Some Questions

- To what extent is the design of sparse architectures constrained by the hardware lottery? How can it escape this constraint?
- To what extent is functional specialisation necessary for sparse and modular architectures?
- Can we use dynamic sparsity to make search/planning more efficient (e.g. in RL)?
- Modular architectures in foundation models have become popular (Grok, Mixtral, etc). Will FM architectures be all sparse in 5 years?

References I

Amari, S.-I. (1972). Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Transactions on computers*, 100(11):1197–1206.

Amos, B. and Kolter, J. Z. (2017). Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, pages 136–145. PMLR.

Ansell, A., Ponti, E., Korhonen, A., and Vulić, I. (2022). Composable sparse fine-tuning for cross-lingual transfer. In Muresan, S., Nakov, P., and Villavicencio, A., editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1778–1796, Dublin, Ireland. Association for Computational Linguistics.

Ansell, A., Vulić, I., Sterz, H., Korhonen, A., and Ponti, E. M. (2024). Scaling sparse fine-tuning to large language models. *arXiv preprint arXiv:2401.16405*.

Artetxe, M., Bhosale, S., Goyal, N., Mihaylov, T., Ott, M., Shleifer, S., Lin, X. V., Du, J., Iyer, S., Pasunuru, R., Anantharaman, G., Li, X., Chen, S., Akin, H., Baines, M., Martin, L., Zhou, X., Koura, P. S., O'Horo, B., Wang, J., Zettlemoyer, L., Diab, M., Kozareva, Z., and Stoyanov, V. (2022). Efficient large scale language modeling with mixtures of experts. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11699–11732, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Bastings, J., Aziz, W., and Titov, I. (2019). Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2963–2977, Florence, Italy. Association for Computational Linguistics.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

References II

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Bengio, Y., Léonard, N., and Courville, A. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. *preprint arXiv:1308.3432*.

Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J.-P., and Bach, F. (2020). Learning with differentiable pertubed optimizers. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9508–9519. Curran Associates, Inc.

Blondel, M., Martins, A. F. T., and Niculae, V. (2020). Learning with fenchel-young losses. *Journal of Machine Learning Research*, 21(35):1–69.

Cai, W., Jiang, J., Wang, F., Tang, J., Kim, S., and Huang, J. (2024). A survey on mixture of experts. *arXiv preprint arXiv:2407.06204*.

Child, R., Gray, S., Radford, A., and Sutskever, I. (2019). Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Chow, Y., Nachum, O., and Ghavamzadeh, M. (2018). Path consistency learning in tsallis entropy regularized mdps. In *International conference on machine learning*, pages 979–988. PMLR.

Correia, G., Niculae, V., and Martins, A. F. T. (2019). Adaptively sparse transformers. In *Proceedings of the Empirical Methods for Natural Language Processing*.

Corro, C. and Titov, I. (2019). Differentiable perturb-and-parse: Semi-supervised parsing with a structured variational autoencoder. In *International Conference on Learning Representations*.

References III

Eigen, D., Ranzato, M., and Sutskever, I. (2013). Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*.

Elbayad, M., Gu, J., Grave, E., and Auli, M. (2020). Depth-adaptive transformer. In *International Conference on Learning Representations*.

Farinhas, A., Aziz, W., Niculae, V., and Martins, A. F. (2022). Sparse communication via mixed distributions. In *Proc. of ICLR*.

Fedus, W., Zoph, B., and Shazeer, N. (2022). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Fernandes, P., Treviso, M., Pruthi, D., Martins, A. F., and Neubig, G. (2022). Learning to scaffold: Optimizing model explanations for teaching. *arXiv preprint arXiv:2204.10810*.

Figurnov, M., Collins, M. D., Zhu, Y., Zhang, L., Huang, J., Vetrov, D., and Salakhutdinov, R. (2017). Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Fu, Y., Jiang, Y., Huang, Y., Nie, P., Lu, Z., Xue, L., He, C., Sit, M.-K., Xue, J., Dong, L., Miao, Z., Zou, K., Ponti, E., and Mai, L. (2024). MoE-CAP: Cost-accuracy-performance benchmarking for mixture-of-experts systems.

Gale, T., Narayanan, D., Young, C., and Zaharia, M. (2023). Megablocks: Efficient sparse training with mixture-of-experts. In Song, D., Carbin, M., and Chen, T., editors, *Proceedings of Machine Learning and Systems*, volume 5, pages 288–304. Curan.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep Sparse Rectifier Neural Networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323.

References IV

Graves, A. (2016). Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*.

Guerreiro, N. M. and Martins, A. F. (2021). Spectra: Sparse structured text rationalization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6534–6550.

Han, C., Wang, Q., Peng, H., Xiong, W., Chen, Y., Ji, H., and Wang, S. (2024). LM-infinite: Zero-shot extreme length generalization for large language models. In Duh, K., Gomez, H., and Bethard, S., editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3991–4008, Mexico City, Mexico. Association for Computational Linguistics.

Hinton, G. E. and Ghahramani, Z. (1997). Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 352(1358):1177–1190.

Hooker, S. (2021). The hardware lottery. *Commun. ACM*, 64(12):58–65.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

Hu, J. Y.-C., Yang, D., Wu, D., Xu, C., Chen, B.-Y., and Liu, H. (2023). On sparse modern hopfield model. In *Advances in Neural Information Processing Systems*.

Jacobs, R. A., Jordan, M. I., and Barto, A. G. (1991a). Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive science*, 15(2):219–250.

References V

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991b). Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Jelassi, S., Mohri, C., Brandfonbrener, D., Gu, A., Vyas, N., Anand, N., Alvarez-Melis, D., Li, Y., Kakade, S. M., and Malach, E. (2024). Mixture of parrots: Experts improve memorization more than reasoning. *arXiv preprint arXiv:2410.19034*.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., Casas, D. d. l., Hanna, E. B., Bressand, F., et al. (2024a). Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Jiang, H., LI, Y., Zhang, C., Wu, Q., Luo, X., Ahn, S., Han, Z., Abdi, A. H., Li, D., Lin, C.-Y., Yang, Y., and Qiu, L. (2024b). MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models.

Lee, K., Choi, S., and Oh, S. (2018). Sparse Markov decision processes with causal sparse Tsallis entropy regularization for reinforcement learning. *IEEE Robotics and Automation Letters*, 3(3):1466–1473.

Lee, K., Kim, S., Lim, S., Choi, S., and Oh, S. (2019). Tsallis reinforcement learning: A unified framework for maximum entropy reinforcement learning. *arXiv preprint arXiv:1902.00137*.

References VI

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. (2021). GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*.

Li, Y., Huang, Y., Yang, B., Venkitesh, B., Locatelli, A., Ye, H., Cai, T., Lewis, P., and Chen, D. (2024). SnapKV: Llm knows what you are looking for before generation. *arXiv preprint arXiv:2404.14469*.

Louizos, C., Welling, M., and Kingma, D. P. (2018). Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*.

Ludziejewski, J., Krajewski, J., Adamczewski, K., Pióro, M., Krutul, M., Antoniak, S., Ciebiera, K., Król, K., Odrzygóźdź, T., Sankowski, P., Cygan, M., and Jaszczur, S. (2024). Scaling laws for fine-grained mixture of experts. In *Forty-first International Conference on Machine Learning*.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2017). The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Malaviya, C., Ferreira, P., and Martins, A. F. T. (2018). Sparse and Constrained Attention for Neural Machine Translation. In *Proc. of the Annual Meeting of the Association for Computation Linguistics*.

Martins, A. and Astudillo, R. (2016). From softmax to sparsemax: A sparse model of attention and multi-label classification. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1614–1623, New York, New York, USA. PMLR.

Martins, A., Farinhas, A., Treviso, M., Niculae, V., Aguiar, P., and Figueiredo, M. (2020a). Sparse and continuous attention mechanisms. *Advances in Neural Information Processing Systems*, 33.

References VII

Martins, A. F. T. and Kreutzer, J. (2017). Fully differentiable neural easy-first taggers. In *Proc. of Empirical Methods for Natural Language Processing*.

Martins, A. F. T., Treviso, M., Farinhas, A., Aguiar, P. M., Figueiredo, M. A., Blondel, M., and Niculae, V. (2022a). Sparse continuous distributions and Fenchel-Young losses. *Journal of Machine Learning Research (to appear)*, 23(257):1–74.

Martins, P. H., Marinho, Z., and Martins, A. F. (2020b). Sparse text generation. In *Empirical Methods for Natural Language Processing*.

Martins, P. H., Marinho, Z., and Martins, A. F. T. (2022b). ∞-former: Infinite memory transformer. In *Proc. of ACL*.

Martins, P. H., Niculae, V., Marinho, Z., and Martins, A. F. (2021). Sparse and structured visual attention. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 379–383. IEEE.

Muennighoff, N., Soldaini, L., Groeneveld, D., Lo, K., Morrison, J., Min, S., Shi, W., Walsh, P., Tafjord, O., Lambert, N., et al. (2024). OLMoE: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*.

Nakano, K. (1972). Associatron – a model of associative memory. *IEEE Transactions on Systems, Man, and Cybernetics*, (3):380–388.

Nawrot, P., Chorowski, J., Lancucki, A., and Ponti, E. M. (2023). Efficient transformers with dynamic token pooling. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6403–6417, Toronto, Canada. Association for Computational Linguistics.

References VIII

Nawrot, P., Łańcucki, A., Chochowski, M., Tarjan, D., and Ponti, E. (2024). Dynamic memory compression: Retrofitting LLMs for accelerated inference. In *Forty-first International Conference on Machine Learning*.

Niculae, V. and Blondel, M. (2017). A regularized framework for sparse and structured neural attention. *arXiv preprint arXiv:1705.07704*.

Niculae, V. and Martins, A. F. (2020). LP-SparseMAP: Differentiable relaxed optimization for sparse structured prediction. In *International Conference on Machine Learning*.

Niculae, V., Martins, A. F. T., Blondel, M., and Cardie, C. (2018). SparseMAP: Differentiable Sparse Structured Inference. In *Proc. of the International Conference on Machine Learning*.

Niepert, M., Minervini, P., and Franceschi, L. (2021). Implicit mle: backpropagating through discrete exponential family distributions. *Advances in Neural Information Processing Systems*, 34:14567–14579.

Oren, M., Hassid, M., Yarden, N., Adi, Y., and Schwartz, R. (2024). Transformers are multi-state RNNs. *arXiv preprint arXiv:2401.06104*.

Ostapenko, O., Su, Z., Ponti, E., Charlin, L., Le Roux, N., Caccia, L., and Sordoni, A. (2024). Towards modular LLMs by building and reusing a library of loras. In *Forty-first International Conference on Machine Learning*.

Palmer, A. W., Hill, A. J., and Scheding, S. J. (2017). Methods for stochastic collection and replenishment (scar) optimisation for persistent autonomy. *Robotics and Autonomous Systems*, 87:51–65.

References IX

Papandreou, G. and Yuille, A. L. (2011). Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models. In *2011 International Conference on Computer Vision*, pages 193–200.

Paulus, M., Choi, D., Tarlow, D., Krause, A., and Maddison, C. J. (2020). Gradient estimation with stochastic softmax tricks. *Advances in neural information processing systems*, 33:5691–5704.

Peters, B. and Martins, A. F. (2021). Smoothing and shrinking the sparse seq2seq search space. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2642–2654.

Peters, B., Niculae, V., and Martins, A. F. T. (2019). Sparse sequence-to-sequence models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., and Gurevych, I. (2021). AdapterFusion: Non-destructive task composition for transfer learning. In Merlo, P., Tiedemann, J., and Tsarfaty, R., editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Pfeiffer, J., Ruder, S., Vulić, I., and Ponti, E. (2023). Modular deep learning. *Transactions on Machine Learning Research*. Survey Certification.

Ponti, E. M., Sordoni, A., Bengio, Y., and Reddy, S. (2023). Combining parameter-efficient modules for task-level generalisation. In Vlachos, A. and Augenstein, I., editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 687–702, Dubrovnik, Croatia. Association for Computational Linguistics.

References X

Pruthi, D., Bansal, R., Dhingra, B., Soares, L. B., Collins, M., Lipton, Z. C., Neubig, G., and Cohen, W. W. (2022). Evaluating explanations: How much do explanations from the teacher aid students? *Transactions of the Association for Computational Linguistics*, 10:359–375.

Raposo, D., Ritter, S., Richards, B., Lillicrap, T., Humphreys, P. C., and Santoro, A. (2024). Mixture-of-Depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*.

Santos, S., Niculae, V., McNamee, D., and Martins, A. F. (2024). Sparse and structured hopfield networks. In *International Conference on Machine Learning*.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Shen, S., Hou, L., Zhou, Y., Du, N., Longpre, S., Wei, J., Chung, H. W., Zoph, B., Fedus, W., Chen, X., Vu, T., Wu, Y., Chen, W., Webson, A., Li, Y., Zhao, V. Y., Yu, H., Keutzer, K., Darrell, T., and Zhou, D. (2024). Mixture-of-experts meets instruction tuning: A winning combination for large language models. In *The Twelfth International Conference on Learning Representations*.

Teerapittayanon, S., McDanel, B., and Kung, H. (2016). Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469.

Treviso, M. and Martins, A. F. (2020). The explanation game: Towards prediction explainability through sparse communication. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 107–118.

Treviso, M., Ross, A., Guerreiro, N. M., and Martins, A. F. T. (2023). CREST: A joint framework for rationalization and counterfactual text generation. In *Proc. of ACL*.

References XI

Tsallis, C. (1988). Possible generalization of Boltzmann-Gibbs statistics. *Journal of Statistical Physics*, 52:479–487.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. (2019). Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808.

Wu, D., Hu, J. Y.-C., Li, W., Chen, B.-Y., and Liu, H. (2024). Stanhop: Sparse tandem hopfield model for memory-enhanced time series prediction. In *International Conference on Learning Representations*. ICLR.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. (2024). Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

Yang, W., Li, X., and Zhang, Z. (2019). A regularized approach to sparse optimal policy in reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32.

Zhu, L., Chen, Z., Schlegel, M., and White, M. (2023). Generalized munchausen reinforcement learning using tsallis kl divergence. In *Advances in Neural Information Processing Systems*.

Zhu, L., Shah, H., Wang, H., and White, M. (2024). q-exponential family for policy optimization. *arXiv preprint arXiv:2408.07245*.

Zoph, B., Bello, I., Kumar, S., Du, N., Huang, Y., Dean, J., Shazeer, N., and Fedus, W. (2022). ST-MoE: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*.