Adaptive and Efficient Computation in Deep Neural Networks

Filip Szatkowski

warsaw.ai, 23.10.2025

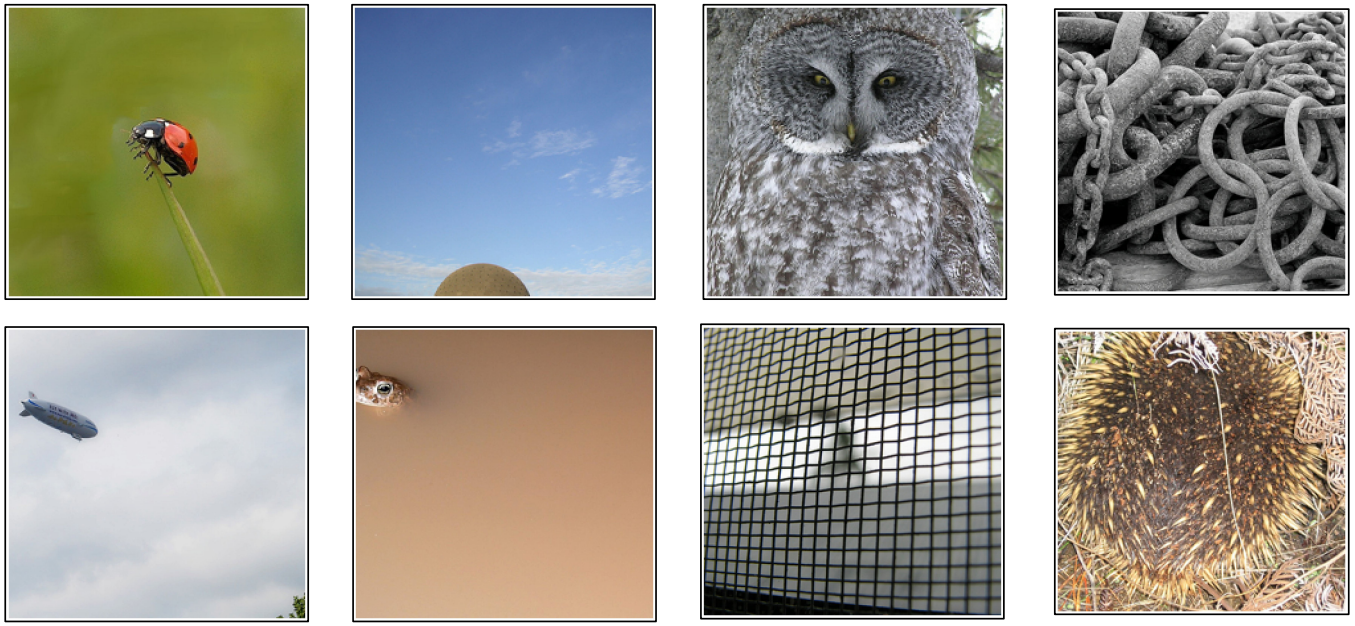
Warsaw University of Technology
Jagiellonian University in Kraków
Sapienza Università di Roma
CVC Centre de Visió per Computador
aws
Universiteit van Amsterdam

Outline

- Short intro to adaptive computation
- Adaptive computation techniques
- Early-exit models
- Activation sparsity
- What’s next?

Why do we want adaptive computation?

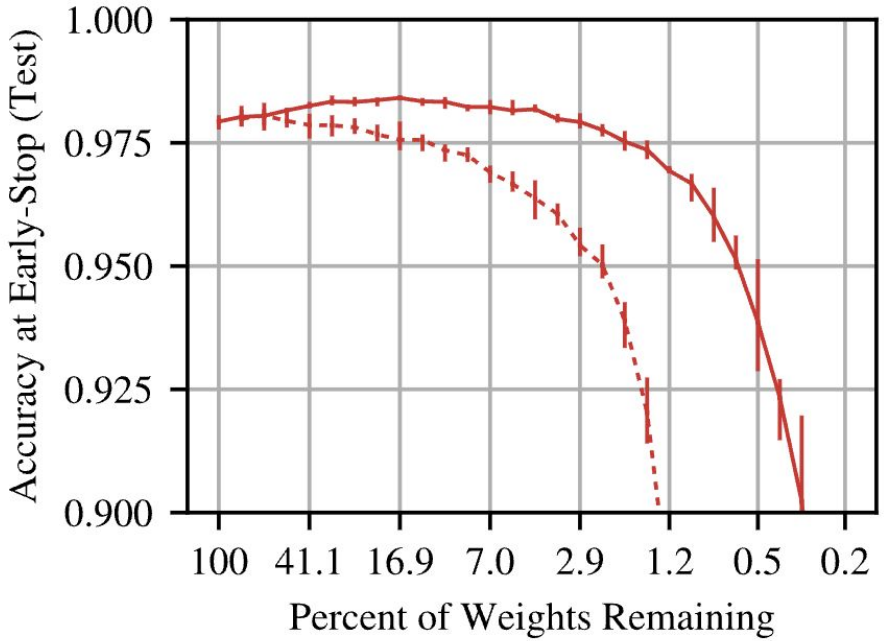
Data points are very different…



…but most neural networks would treat all of the images above the same way!

Current models are not utilizing their parameters well

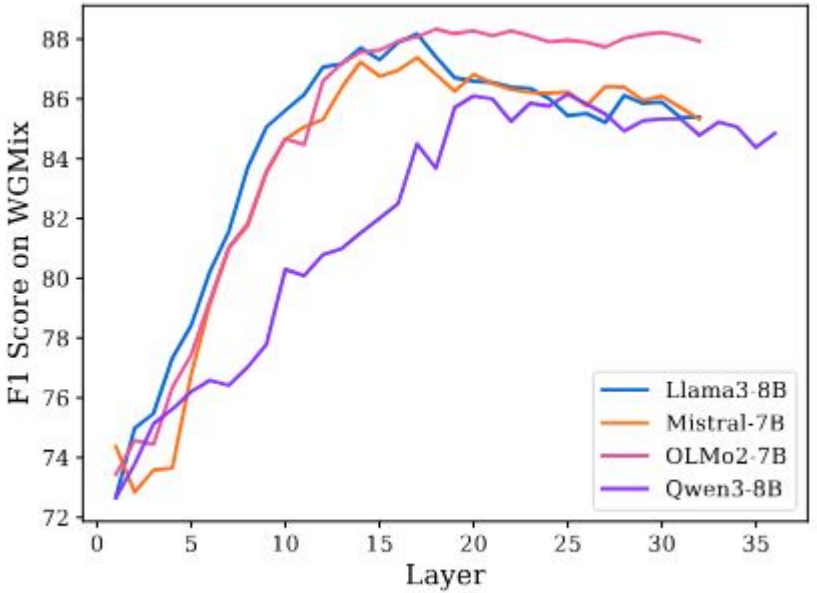
- Large number of parameters translates to better models (see: scaling laws)
- However, models contain small sub-networks that can be trained to the same accuracy as a full network (see: lottery ticket hypothesis)
- We need to use large models mostly because they are easier to optimize



The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks, Frankle et. al 2018

Knowledge forms early in the models

- Existing models can also be probed to a very good accuracy at the early layers already
- Residual design of SOTA models means that deeper layers mostly just refine the information



Efficient Large Language Models Moderation with Multi-Layer Latent Prototypes, Chrabąszcz et. al 2025

Dense computation is often functionally sparse

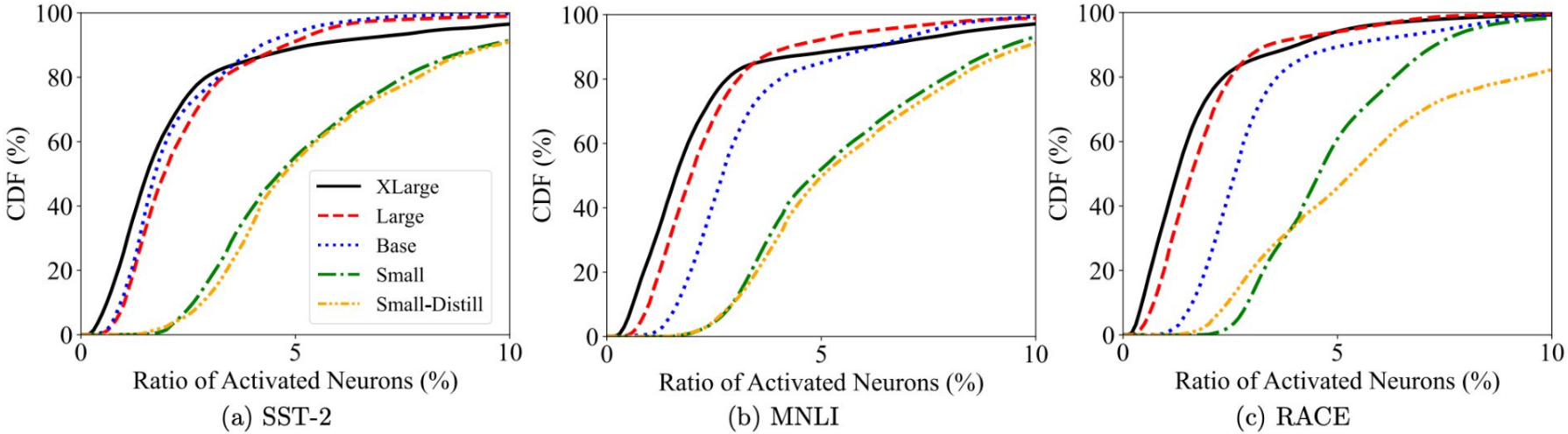
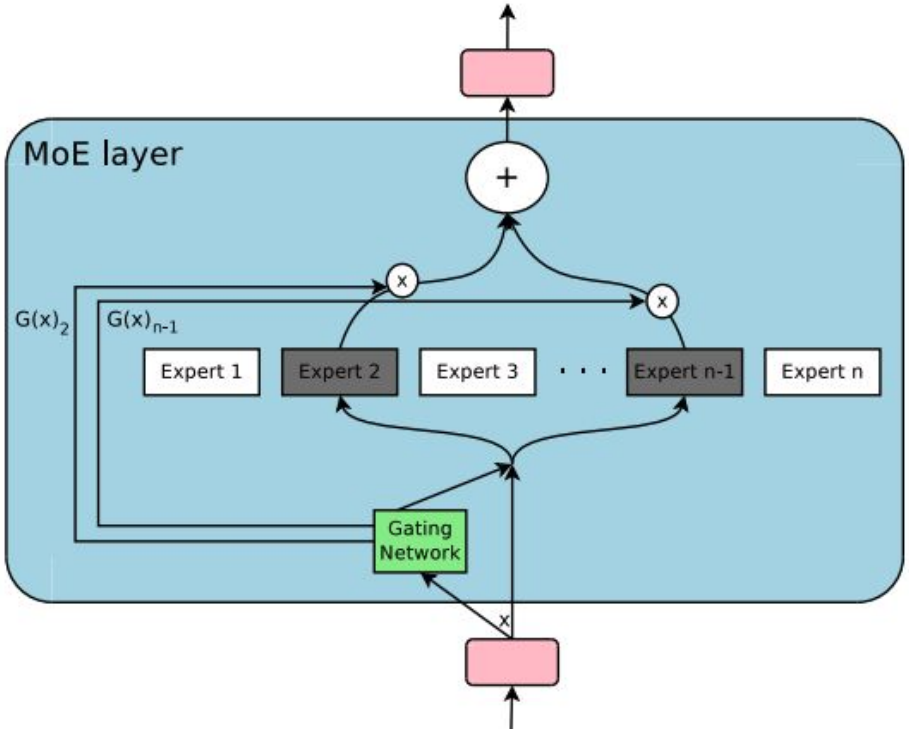


Figure 2: CDF of the ratio of activated neurons for each input with different models on three datasets.

MoEfication: Transformer Feed-forward Layers are Mixtures of Experts, Zhang et al., 2022

Adaptivity already allows for more efficient scaling

- Largest and strongest models right now are Mixtures-of-Experts (e.g., DeepSeek).
- MoE allows for scaling the model capacity without significantly affecting the computational cost.
- Reasoning models are also an adaptive computation technique
- However, from adaptivity and efficiency point of view, these are still missing real specialization, and focus on efficient scaling in large compute regimes rather than efficiency



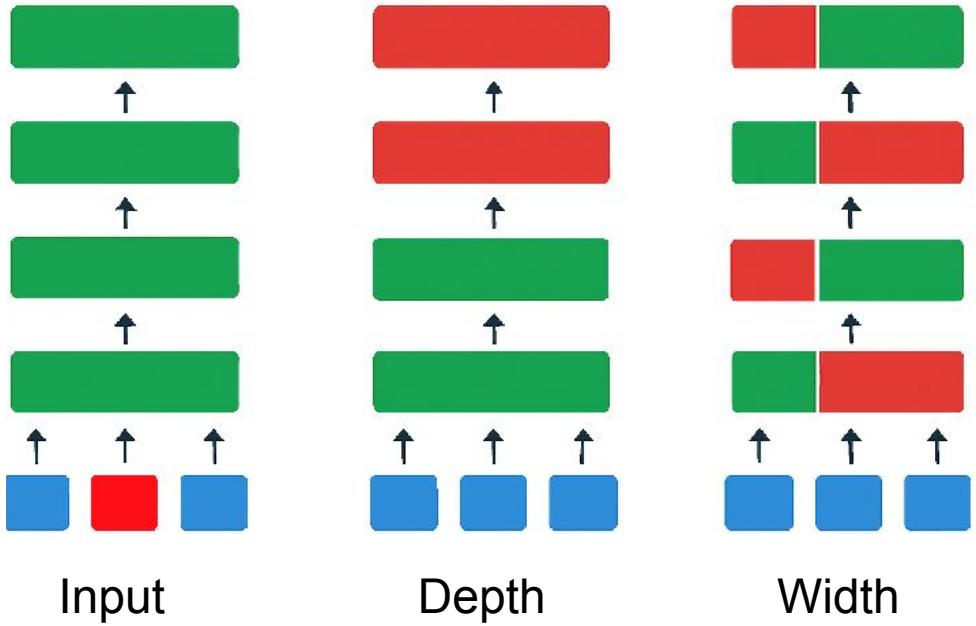
Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer, Shazeer et. al, 2016

We need better (adaptive) models

Adaptive computation techniques

Adaptive computation axes

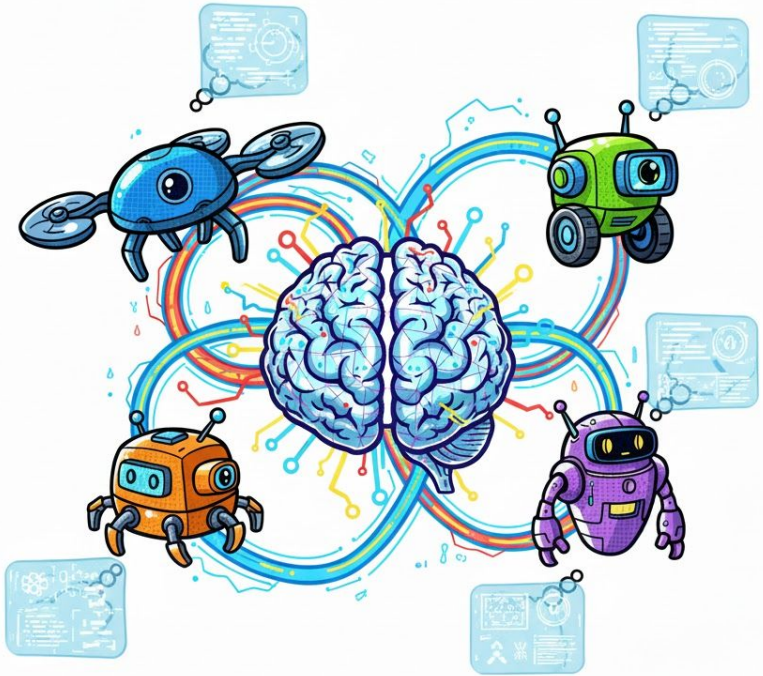
- Input or tokens
- Depth
- Width



Conditional computation in neural networks: principles and research trends, Scardapane et. al, 2024

Adaptive computation axes

- Input or tokens
- Depth
- Width
- *Models or agents*

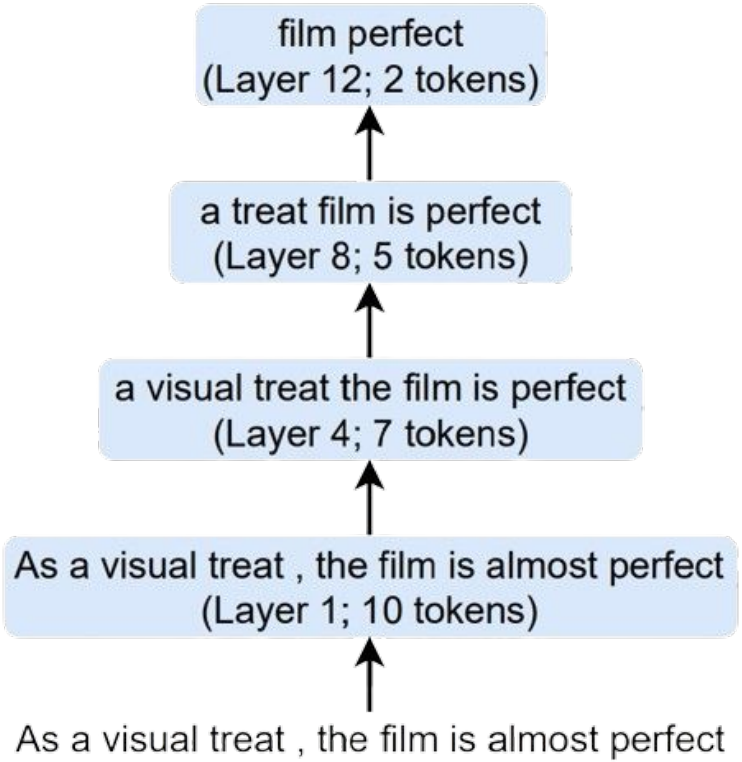


Adaptive computation methods

- Input or tokens
  - Token dropping / pooling / merging
  - KV cache compression
- Depth
  - Recurrent models
  - Early-exits
- Width
  - Matryoshka and incremental models
  - Mixture-of-Experts
  - Activation sparsity
- Models or agents
  - Multi-agent systems
  - Speculative decoding

Token <dropping / merging / etc>

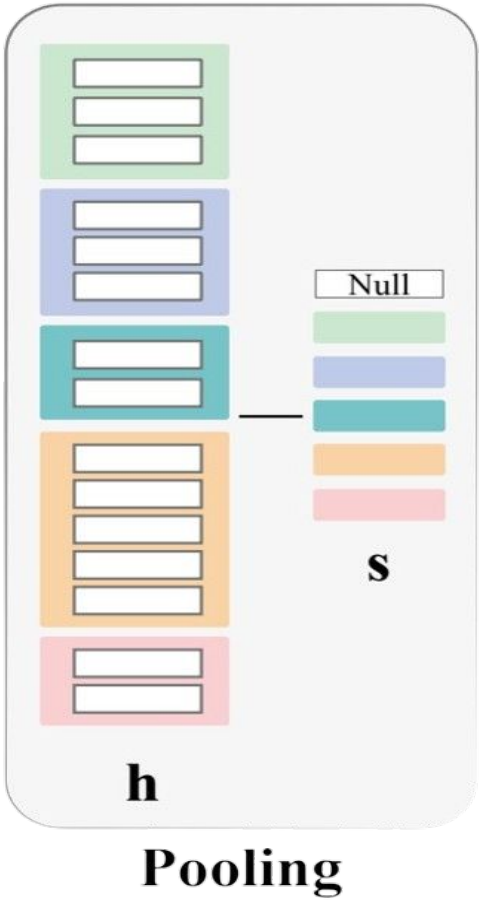
- Transformers operate on tokens
- We can reduce the compute in subsequent layers by reducing number of processed tokens
- This can be achieved through dropping tokens between layers.



Magic Pyramid: Accelerating Inference with Early Exiting and Token Pruning, He et al., 2021

Token <dropping / merging / etc>

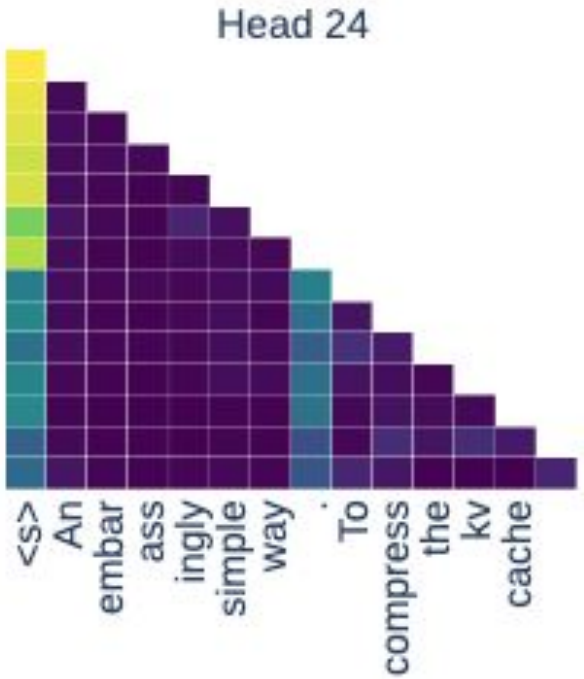
- Transformers operate on tokens
- We can reduce the compute in subsequent layers by reducing number of processed tokens
- This can be achieved through dropping tokens between layers.
- We can also combine the tokens together or pool them, etc.



Efficient Transformers with Dynamic Token Pooling, Nawrot et al., 2022

Sparse attention / KV cache compression

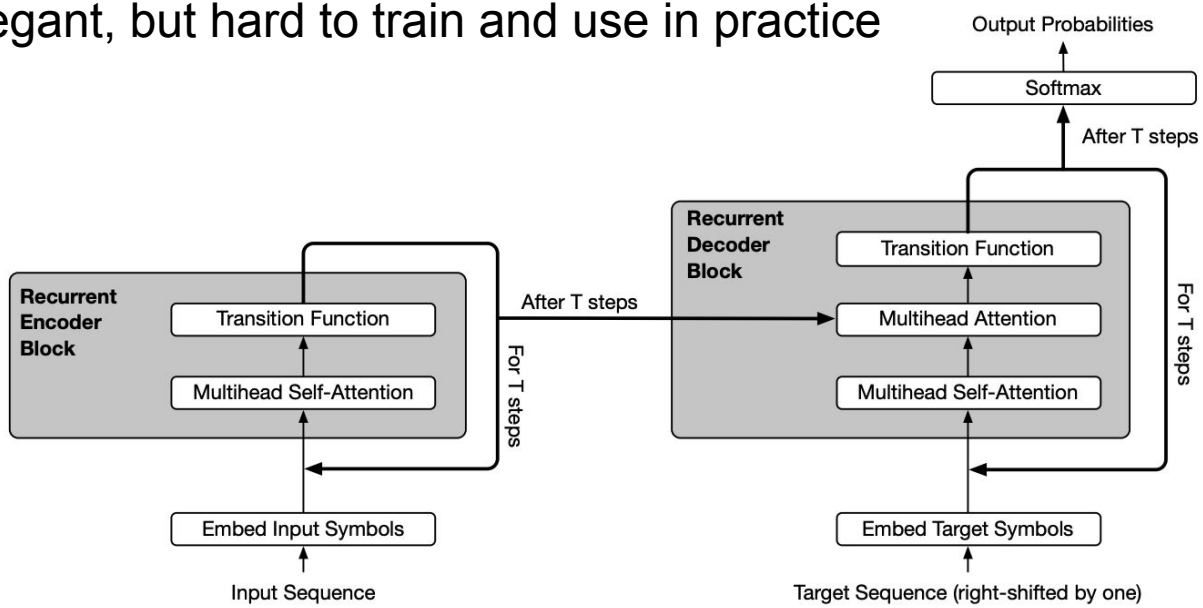
- Sparse attention can be seen as a form of adaptivity across the sequence dimension
- In transformers, during inference we use KV cache to speed up autoregressive generation
- We can drop low-impact elements in KV cache to reduce the memory footprint and accelerate the inference.



A Simple and Effective L2 Norm-Based Strategy for KV Cache Compression, Devoto et al., 2024

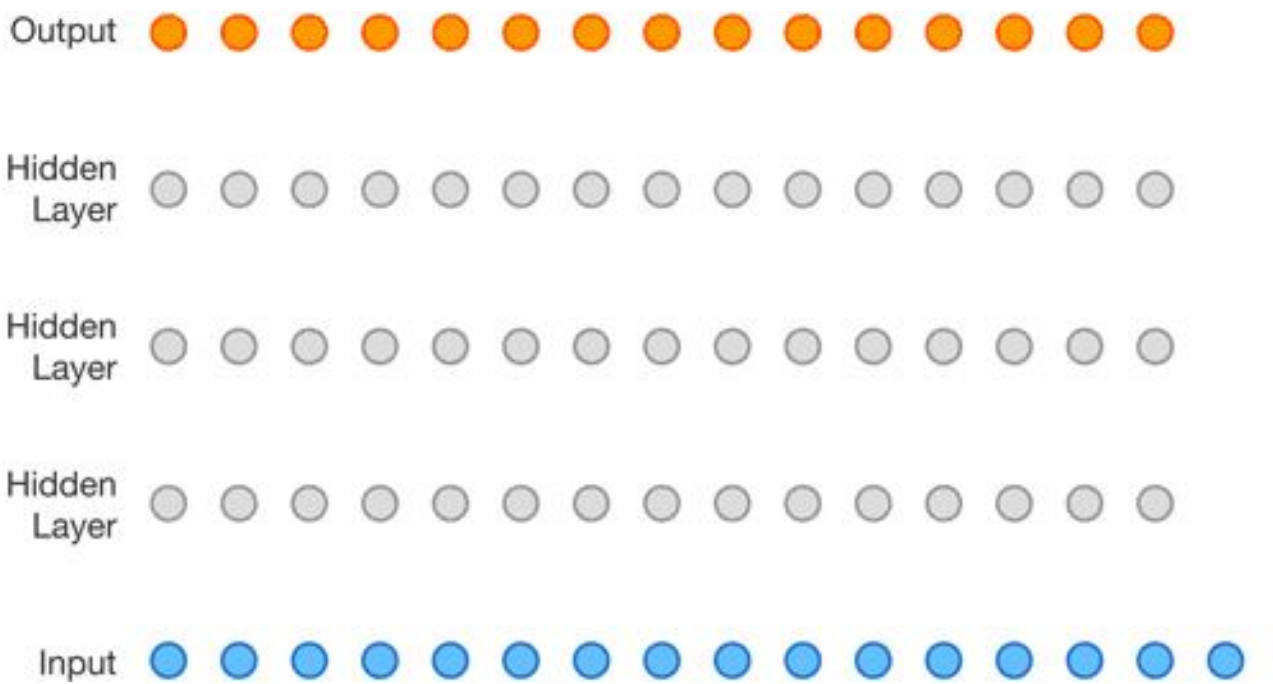
Recurrent transformers

- Universal transformers recurrently use a single layer with a halting criterion
- Different tokens will invoke different number of recurrence steps
- Very elegant, but hard to train and use in practice



Universal Transformers, Deghani et. al, 2018

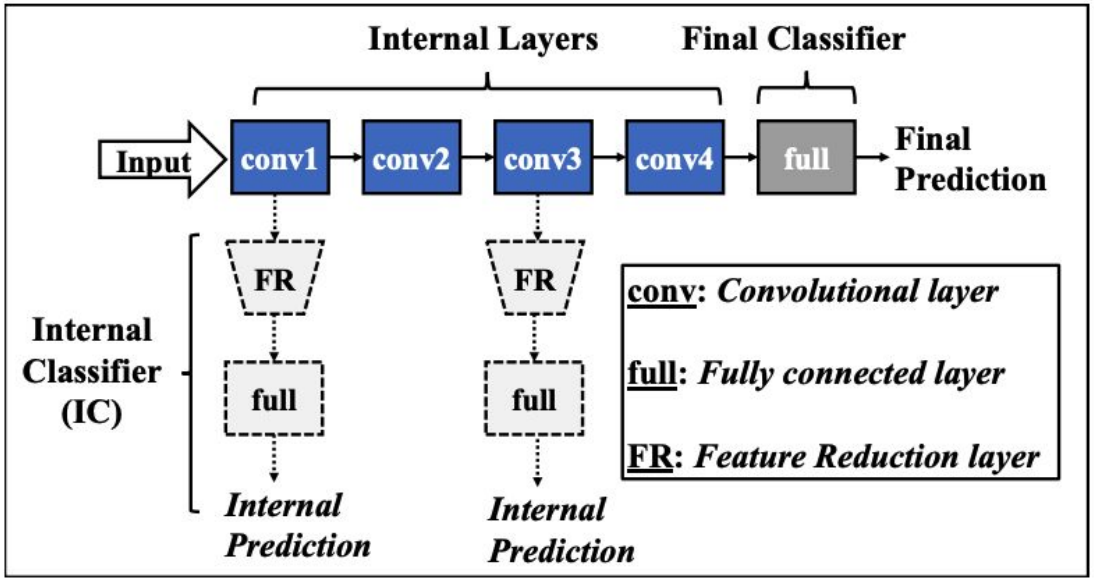
Autoregressive models are adaptive-depth models



WaveNet: A generative model for raw audio - Google DeepMind

Early-exits

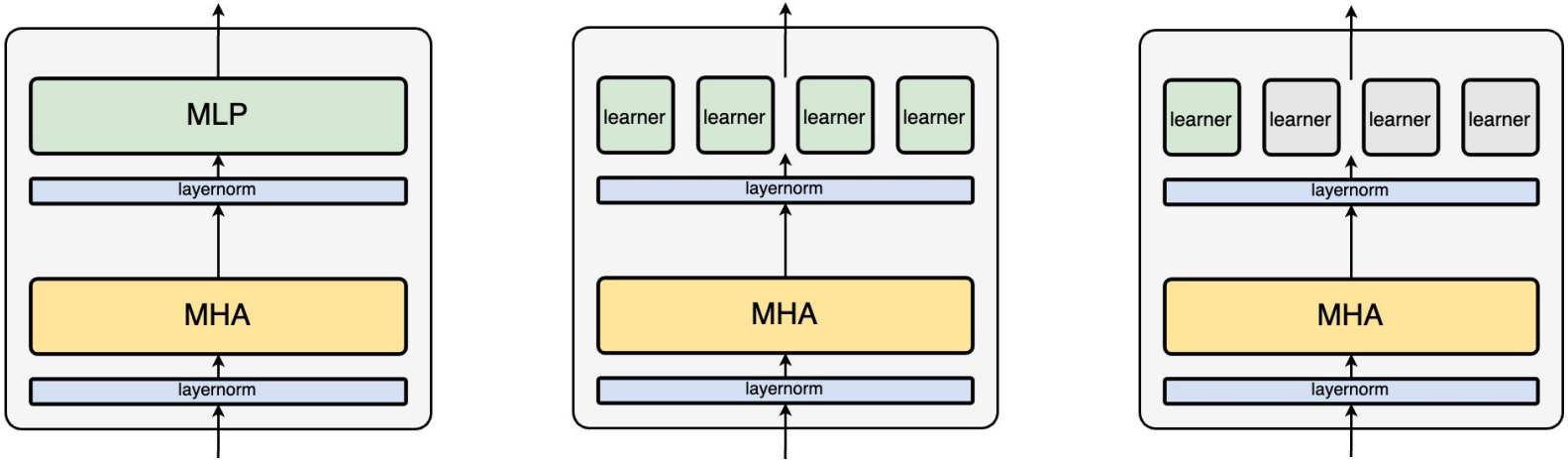
- Model is extended with auxiliary classifiers added to intermediate model layers (early-exits)
- Each classifier predicts sequentially
- If the prediction satisfies the exit rule, it can be returned and the rest of the network gets skipped



Shallow-Deep Networks: Understanding and Mitigating Network Overthinking, Kaya et. al, 2019

Matryoshka models and incremental learners

- Compose FFN of small modules that improve upon each other
- Use a router to determine how many learners are needed for given token
- Ideally, simple tokens will use few learners or modules and save compute



Adaptive Computation Modules: Granular Conditional Computation For Efficient Inference, Wójcik et. al, 2024

Mixture-of-Experts

- An example of adaptive width computation is Mixture-of-Experts (MoE)
- Modern MoEs increasingly often use fine-grained experts e.g., DeepSeek-v3 uses 1 shared expert and routes to 8 out of 256 experts

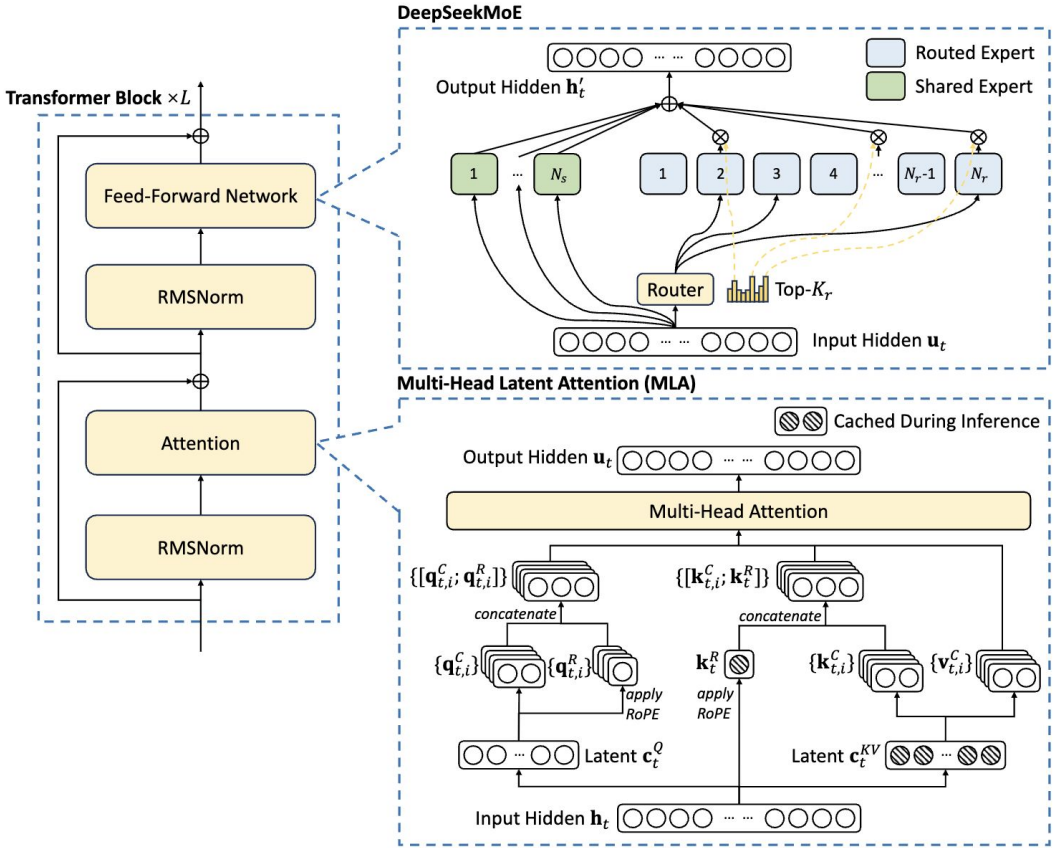
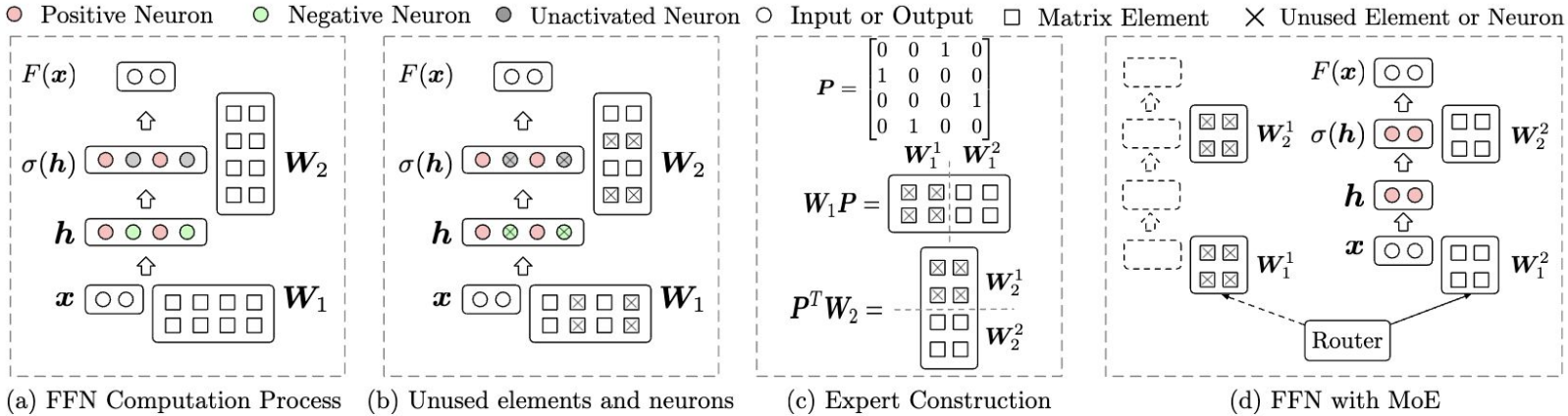


Figure 2 | Illustration of the basic architecture of DeepSeek-V3. Following DeepSeek-V2, we adopt MLA and DeepSeekMoE for efficient inference and economical training.

DeepSeek-V3 Technical Report, DeepSeek AI, 2024

Dense Transformer to MoE conversion (MoEfication)



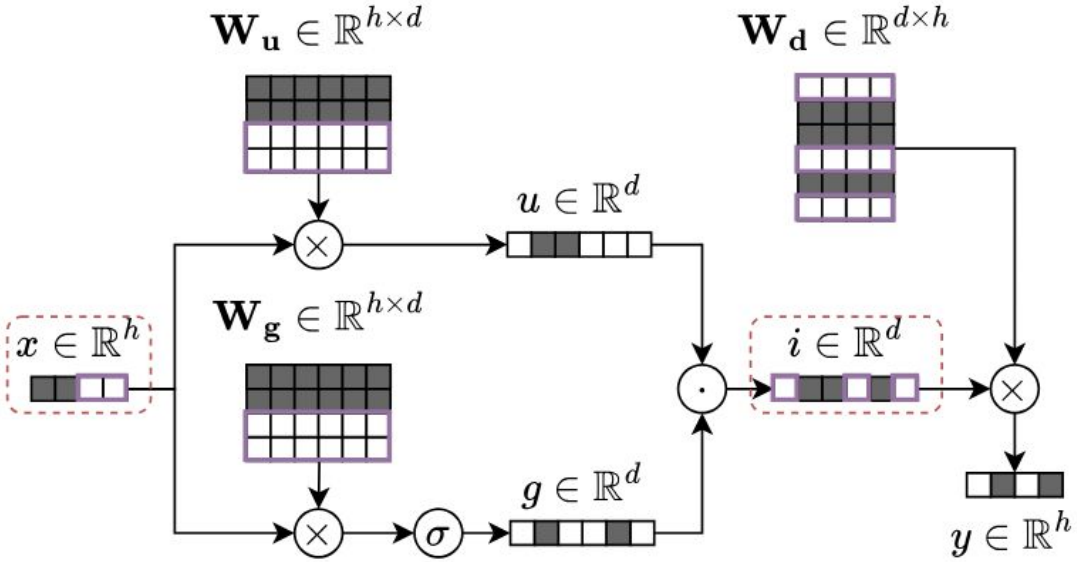
(a) FFN Computation Process (b) Unused elements and neurons (c) Expert Construction (d) FFN with MoE

- MoE paradigm can be also used to accelerate inference of already trained models by exploiting activation sparsity (contextual / dynamic sparsity)
- Model activations can be grouped together like experts
- If we can determine which experts will be used for given token, a lot of computation can be skipped with minimal performance degradation

MoEfication: Transformer Feed-forward Layers are Mixtures of Experts, Zhang et al., 2022

Activation sparsity

- We can further simplify MoEfication paradigm if we treat single neurons as experts
- This is referred to as activation sparsity
- Irrelevant parts of the activations can be skipped alongside corresponding weights during the matmul in the forward pass



Universal Properties of Activation Sparsity in Modern Large Language Models, Szatkowski et al., 2025

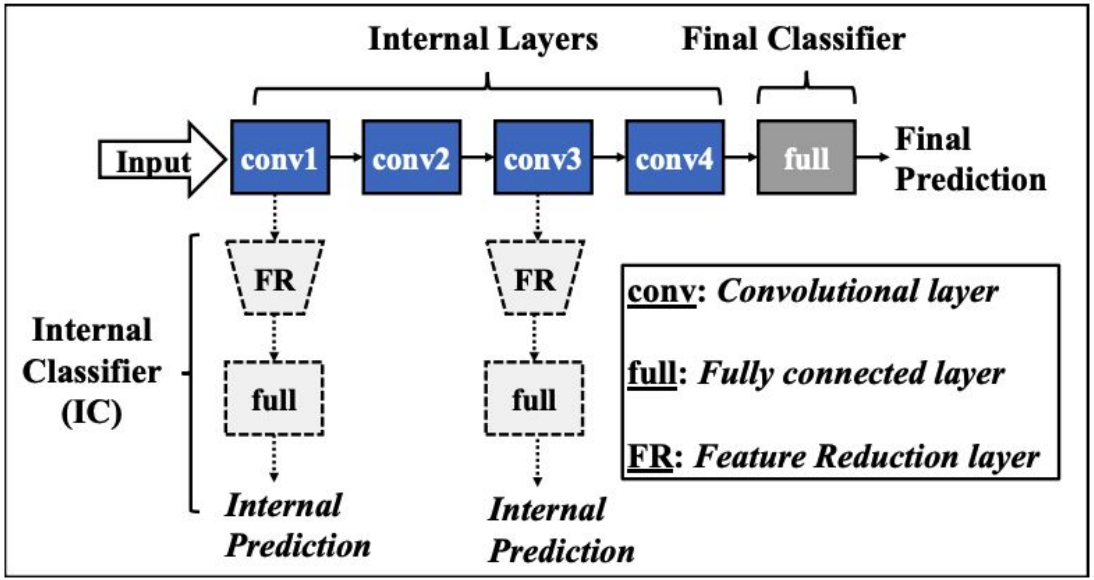
Speculative decoding

https://research.google/blog/looking-back-at-speculative-decoding/

Early-exit models

Early-exits

- Model is extended with auxiliary classifiers added to intermediate model layers (early-exits)
- Each classifier predicts sequentially
- If the prediction satisfies the exit rule, it can be returned and the rest of the network gets skipped



Shallow-Deep Networks: Understanding and Mitigating Network Overthinking, Kaya et. al, 2019

Example of cost-accuracy trade-off with early exits

- Early-exit requires an exit rule, e.g. confidence
- More permissive rule means less computation, but the network performance (loss, accuracy, etc) can degrade
- We can create performance curves for adaptive computation methods by evaluating with different compute budgets

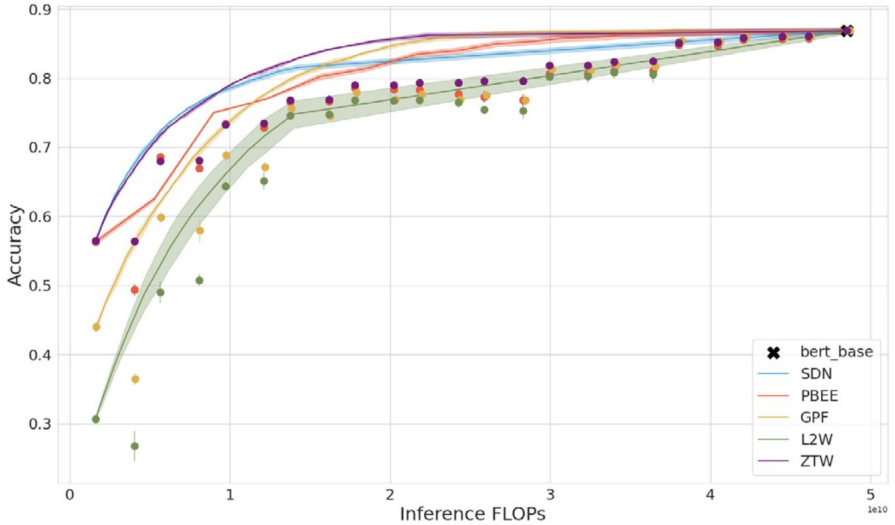


Fig. 3. Accuracy vs. computational cost trade-off obtained by five early-exit methods on the 20 Newsgroups dataset, with BERT-Base as backbone model. Each IC is marked with a point, and the score of threshold-based early-exiting is plotted for a range of τ as a line. Observe that a clear advantage of ZTW over SDN appears after passing a certain number of ICs.

Zero time waste in pre-trained early exit neural networks, Wójcik et. al, 2023

Improving early-exits

- Not super clear how to train (joint, frozen backbone?)
- Many variants of early exits since SDN paper focusing on selection rules, classifier architecture, training etc
- E.g., early-exit performance can be improved if classifiers are enhanced with cascading connections and ensembling modules

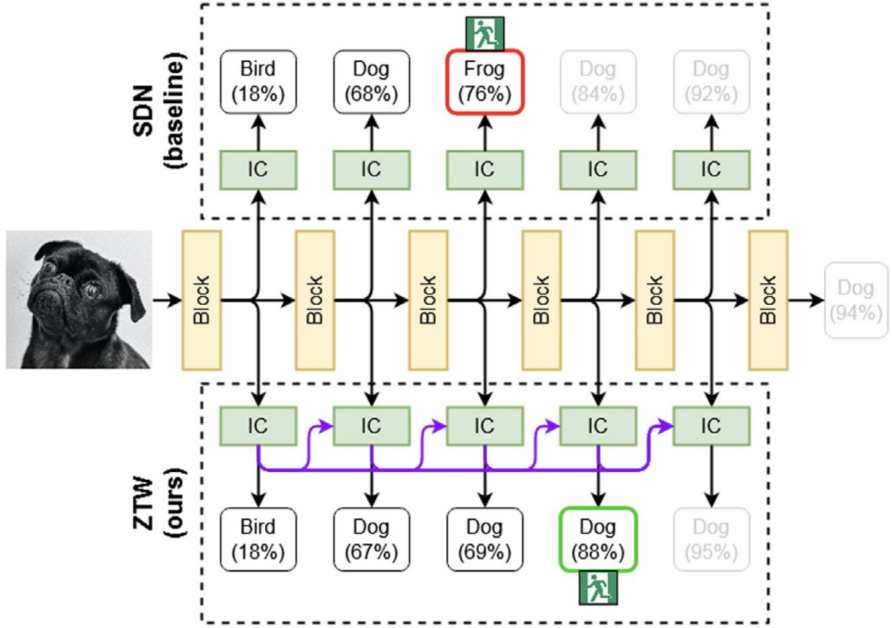


Fig. 1. Comparison of the proposed ZTW (bottom) with a conventional early-exit model, SDN (top). In both approaches, internal classifiers (ICs) attached to the intermediate hidden layers of the base network allow us to return predictions quickly for examples that are easy to process. While SDN discards predictions of uncertain ICs (*e.g.* below a threshold of 75%), ZTW reuses computations from all previous ICs, which prevents information loss and waste of computational resources.

Zero time waste in pre-trained early exit neural networks, Wójcik et. al, 2023

Structured Probabilistic Inference & Generative Modeling workshop

Failure Prediction Is a Better Performance Proxy for Early-Exit Networks Than Calibration

Piotr Kubaty, **Filip Szatkowski**,
Metod Jazbec, Bartosz Wójcik

Calibration in early-exit networks

- Model calibration measures the degree to which a model's predicted confidence aligns with its empirical accuracy.
- Since most early-exit networks use prediction confidence to exit, intuitively the calibration of classifiers is crucial for exit decisions.
- Many works focus on optimizing the classifier calibration.

On Calibration of Modern Neural Networks, Guo et al., 2017

Is calibration all we need?

- We make a simple observation that calibration in context of early-exit networks ignores the cost-accuracy trade-off.
- Some samples are never correctly classified throughout the network.
- For these samples it actually makes sense to exit as early as possible to save compute, but perfectly calibrated classifiers would assign random probabilities to such incorrect samples and never exit.

Failure Prediction Is a Better Performance Proxy for Early-Exit Networks Than Calibration, Kubaty et al., 2025

Early Exit Failure Prediction score

However, this definition of failure prediction was devised for conventional static classifiers, and is not suitable for early-exit networks, as it does not account for the behavior of deeper classifiers. In particular, if a sample is incorrectly classified by the current classifier and all of the deeper classifiers, then it is beneficial to halt computation as early as possible. This crucial observation leads us to adapt the definition of failure prediction to the multi-exit model setup. **We define Early Exit Failure Prediction score (EEFP score)** as:

$$\mathrm{EEFP}_j(\{x_i, y_{j,i}\}) = \mathrm{AUROC}(\{c_j(x_i), \bar{y}_{j,i}\}),$$

where:

$$\bar{y}_{j,i} = \begin{cases} 1, & \text{if } y_{j,i} = 1 \vee (y_{j,i} = 0 \wedge \forall_{l>j} y_{l,i} = 0) \\ 0, & \text{otherwise.} \end{cases}$$

In this formulation, a positive $y_{j,i}$ means that either the current head is correct, or all deeper heads would also be wrong, and exiting is optimal from the computational point of view.

Failure Prediction Is a Better Performance Proxy for Early-Exit Networks Than Calibration, Kubaty et al., 2025

EEFP vs Expected Calibration Error

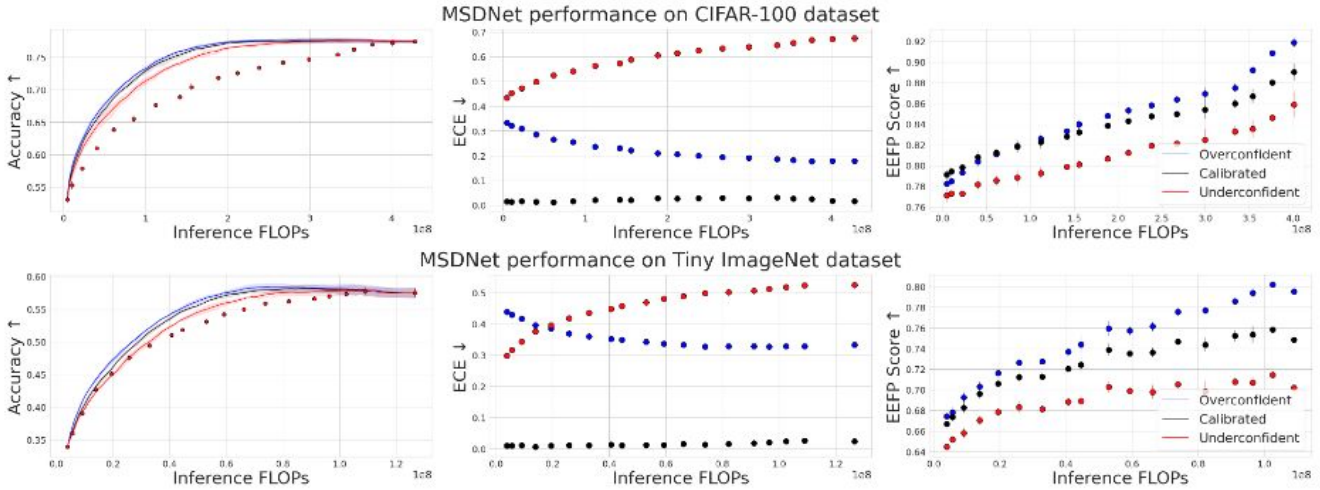


Figure 1: Cost-accuracy curves, head calibration errors, and our proposed EEFP scores for one calibrated model and two decalibrated models with modified temperature values. Calibration fails to capture the quality of the early-exit model, as an overconfident network with higher ECEs performs better than the calibrated one. We propose an alternative metric, *Early-Exit Failure Prediction score* (EEFP score), which more accurately reflects the quality of the multi-exit model.

Failure Prediction Is a Better Performance Proxy for Early-Exit Networks Than Calibration, Kubaty et al., 2025

Rethinking the design of early-exit models

- Cost-accuracy analysis for some samples makes exiting early for wrong predictions beneficial
- Calibration in early-exit models might not be the best base for designing the inference techniques
- Hopefully, we can use our insights and EEFP to design better adaptive models

Failure Prediction Is a Better Performance Proxy for Early-Exit Networks Than Calibration, Kubaty et al., 2025

ICML
International Conference
On Machine Learning

Improving Continual Learning Performance and Efficiency with Auxiliary Classifiers

Filip Szatkowski, Yaoyue Zheng, Fei Yang,
Tomasz Trzciński, Bartłomiej Twardowski, Joost van de Weijer

Warsaw University of Technology

Centre de Visió per Computador

IDEAS NCBR

Continual learning with early-exit models.

- Certain applications need adaptability not only during inference, but also during training
- Continual learning develops training algorithms able to provide adaptability to data shifts

Improving Continual Learning Performance and Efficiency with Auxiliary Classifiers, Szatkowski et al., 2025

Early-exit network for continual learning

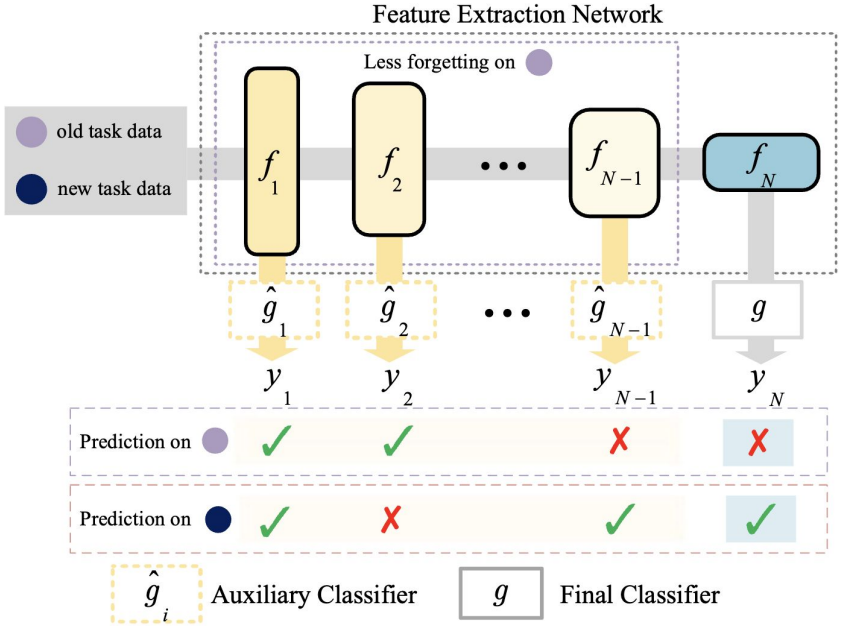


Figure 5. Overview of the network enhanced with ACs in continual learning. The early layers exhibit less forgetting on the old tasks, and can return the correct prediction in cases where the final classifier fails and save computations.

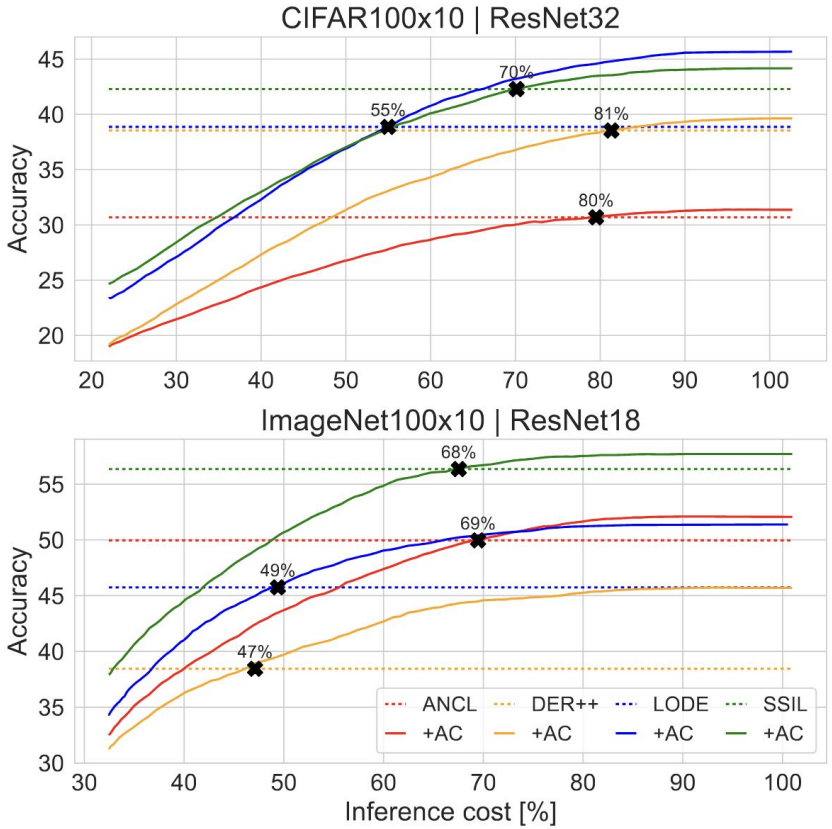
Big table where **ours** is **the best**

Table 1. The final accuracy of diverse continual learning methods enhanced with auxiliary classifiers (ACs) on CIFAR100 and ImageNet100 benchmarks. Adding ACs improves the performance of all methods across both benchmarks, demonstrating the robustness of our idea.

Method	FT	FT+Ex	GDumb	ANCL	BiC	DER++	ER	EWC	LwF	LODE	SSIL	Avg
CIFAR100x5												
Base	18.68±0.31	38.35±0.86	19.09±0.44	37.71±1.14	47.66±0.43	38.96±1.38	34.55±0.21	18.95±0.29	38.26±0.98	42.82±0.84	45.62±0.16	34.60±0.20
+AC	**28.18**±1.07	**38.75**±0.26	**23.29**±0.54	**39.83**±1.22	**50.40**±0.68	**43.49**±0.73	**39.77**±0.32	**28.96**±1.13	**40.55**±0.95	**49.13**±0.35	**48.35**±0.50	**39.15**±0.59
Δ	+9.49±0.96	+0.39±0.90	+4.20±0.16	+2.12±1.03	+2.74±0.83	+4.53±2.05	+5.22±0.38	+10.02±1.39	+2.29±0.25	+6.31±0.81	+2.72±0.42	+4.55±0.38
CIFAR100x10												
Base	10.27±0.05	34.51±0.40	22.22±0.72	30.69±0.62	42.87±1.51	38.54±0.65	32.31±0.82	10.20±0.35	29.56±0.44	38.87±0.45	42.29±0.49	30.21±0.18
+AC	**16.88**±1.08	**36.97**±0.39	**27.74**±0.73	**31.37**±0.94	**46.19**±1.47	**39.64**±1.00	**37.32**±0.28	**19.12**±0.88	**30.31**±1.14	**45.67**±0.52	**44.17**±0.28	**34.13**±0.24
Δ	+6.62±1.06	+2.46±0.31	+5.52±1.13	+0.68±0.79	+3.31±2.62	+1.10±1.08	+5.01±0.98	+8.92±1.06	+0.74±0.91	+6.80±0.93	+1.88±0.77	+3.91±0.41
ImageNet100x5												
Base	23.27±0.39	44.05±0.69	21.29±0.59	60.79±0.06	62.55±0.53	45.33±2.55	38.65±0.43	23.36±0.64	59.60±0.27	49.88±0.56	60.54±0.32	44.48±0.31
+AC	**34.93**±0.65	**46.75**±0.61	**25.30**±1.14	**62.99**±0.30	**65.22**±0.27	**54.14**±0.80	**44.46**±0.47	**35.09**±0.17	**61.07**±0.57	**56.23**±0.66	**63.89**±0.18	**50.01**±0.13
Δ	+11.67±0.77	+2.71±0.85	+4.01±0.61	+2.21±0.35	+2.67±0.79	+8.81±3.34	+5.81±0.66	+11.73±0.71	+1.47±0.45	+6.35±1.22	+3.35±0.48	+5.53±0.41
ImageNet100x10												
Base	14.40±0.30	35.94±0.86	22.55±0.62	49.96±0.46	56.32±0.47	38.45±1.95	32.45±0.35	14.69±0.20	49.15±0.38	45.75±0.50	56.35±0.51	37.82±0.35
+AC	**22.14**±0.16	**39.26**±0.61	**25.93**±0.52	**52.07**±0.50	**57.23**±0.87	**45.70**±0.40	**37.10**±1.20	**23.25**±0.55	**49.51**±0.71	**51.39**±0.91	**57.71**±0.08	**41.93**±0.25
Δ	+7.74±0.37	+3.32±0.90	+3.38±0.37	+2.11±0.32	+0.91±0.42	+7.25±1.55	+4.65±0.88	+8.56±0.39	+0.36±1.05	+5.64±0.90	+1.35±0.59	+4.12±0.13

Dynamic inference in continual learning

- Early-exit networks outperform the base variants across several CL methods
- Now we investigate the cost-performance trade-offs
- Early-exit models match standard ones at 50-80% of computation, offering free cost reduction
- The overhead from additional classifiers is negligible



Activation sparsity

Mixture-of-Experts

- A currently very popular example of width-adaptive computation is Mixture-of-Experts (MoE)
- Modern MoEs increasingly often use fine-grained experts e.g., DeepSeek-v3 uses 1 shared expert and routes to 8 out of 256 experts

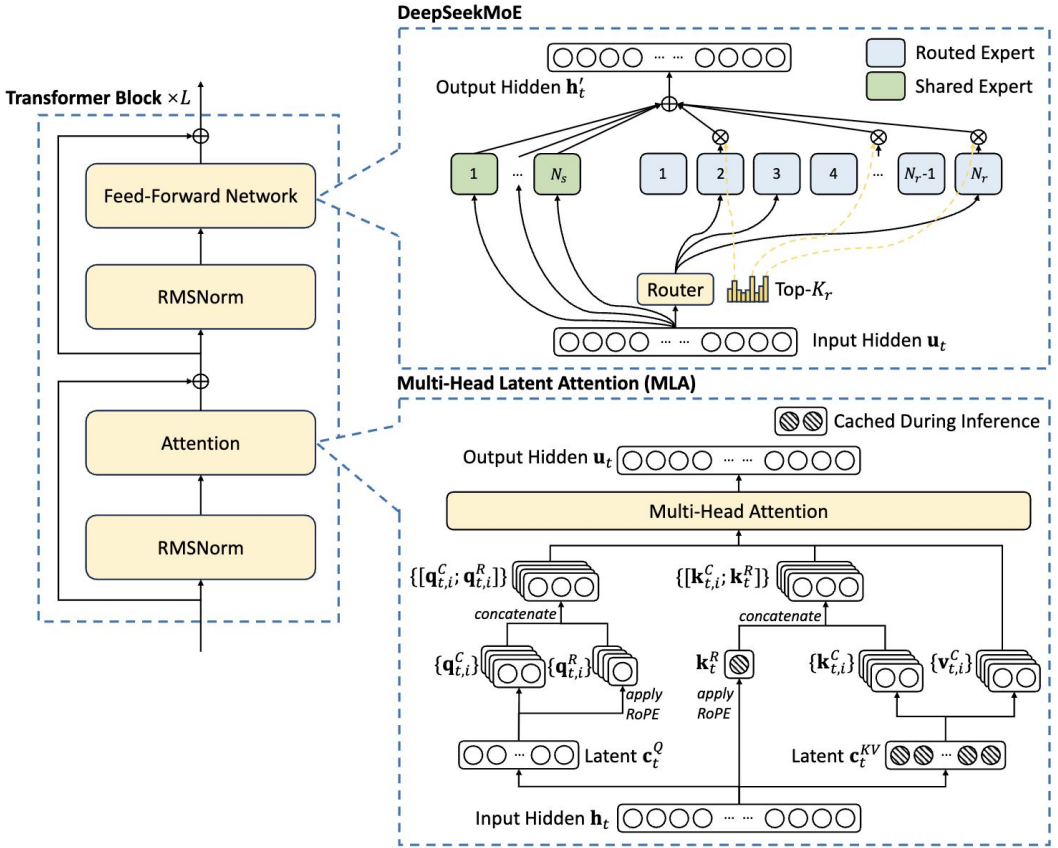
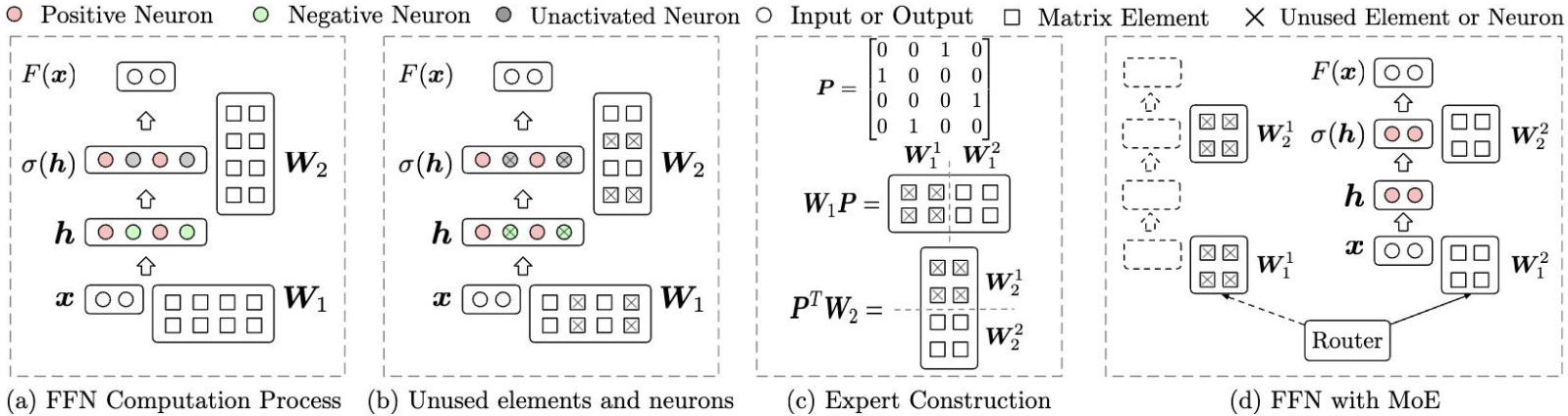


Figure 2 | Illustration of the basic architecture of DeepSeek-V3. Following DeepSeek-V2, we adopt MLA and DeepSeekMoE for efficient inference and economical training.

DeepSeek-V3 Technical Report, DeepSeek AI, 2024

Dense Transformer to MoE conversion (MoEfication)



(a) FFN Computation Process (b) Unused elements and neurons (c) Expert Construction (d) FFN with MoE

- MoE paradigm can be also used to accelerate inference of already trained models by exploiting activation sparsity (contextual / dynamic sparsity)
- Model activations can be grouped together like experts
- If we can determine which experts will be used for given token, a lot of computation can be skipped with minimal performance degradation

MoEfication: Transformer Feed-forward Layers are Mixtures of Experts, Zhang et al., 2022

Exploiting Activation Sparsity with Dense to Dynamic-k Mixture-of-Experts Conversion

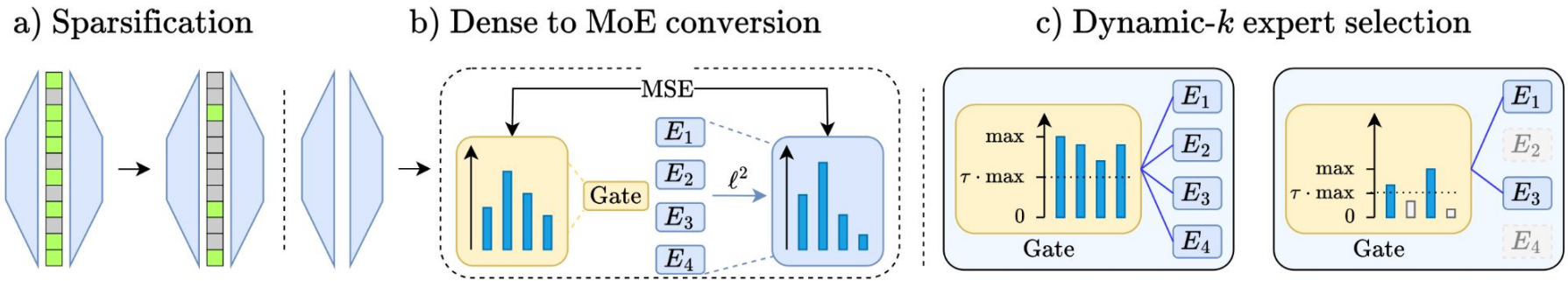
Filip Szatkowski*, Bartosz Wójcik*,
Mikołaj Piórczyński*, Simone Scardapane


Warsaw University of Technology

IDEAS NCBR


JAGIELLONIAN UNIVERSITY IN KRAKÓW

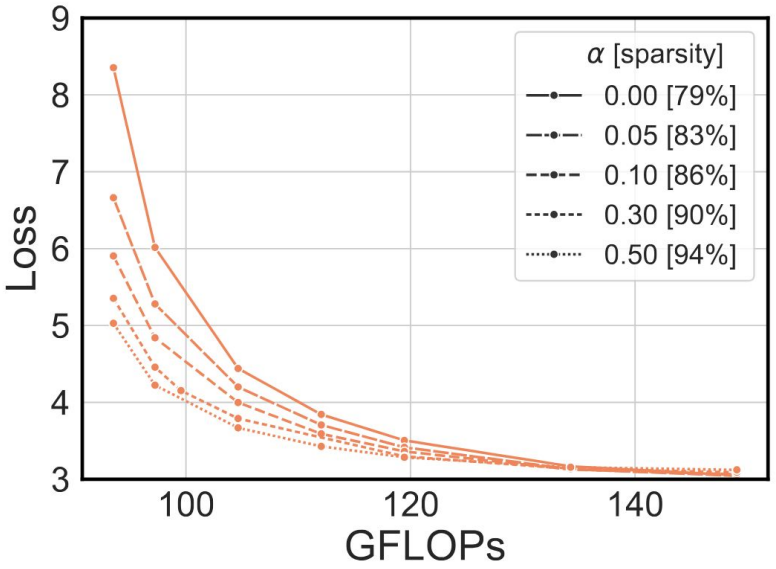
Dense-to-Dynamic-k Mixture-of-Experts (D2DMoE) outline



Exploiting Activation Sparsity with Dense to Dynamic-k Mixture-of-Experts Conversion, Szatkowski et al., 2025

Activation sparsity in transformers

- We analyze the relationship between the activation sparsity of the starting model and the efficiency of the final MoE model.
- We introduce lightweight fine-tuning with auxiliary loss component before conversion to MoE to enforce higher activation sparsity.
- Improved activation sparsity directly leads to a substantially improved cost-to-performance trade-off.

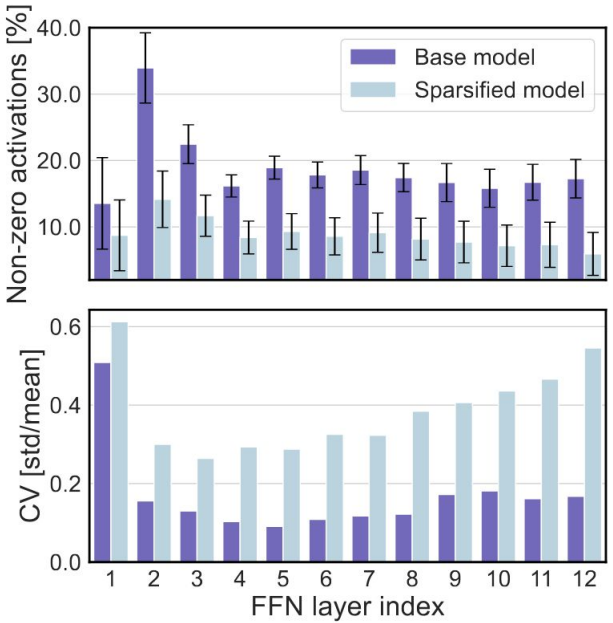


(a) Impact of sparsity on MoE conversion

Exploiting Activation Sparsity with Dense to Dynamic-k Mixture-of-Experts Conversion, Szatkowski et al., 2025

Sparsification leads to higher variance in activations

- We investigate the average number of non-zero activations across different model layers.
- After sparsification, the variance in this number becomes more prominent.
- Resulting MoE model would need to use different number of experts to replicate the original activations.

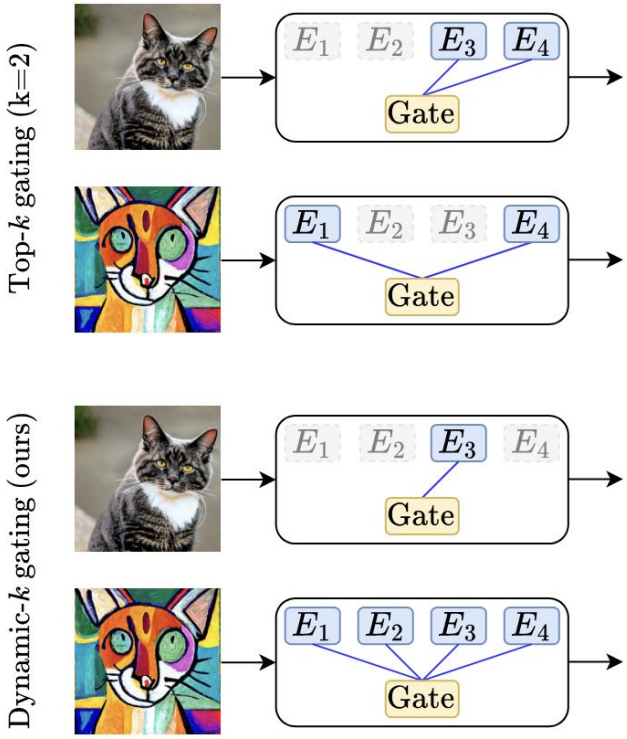


(b) Non-zero activations distribution

[Exploiting Activation Sparsity with Dense to Dynamic-k Mixture-of-Experts Conversion](), Szatkowski et al., 2025

Top-k gating is not enough

- Standard Mixture-of-Experts architecture always activates the same number of experts.
- As demonstrated previously, this is unsuitable for sparsified models where number of non-zero activations vary significantly depending on the input.
- We propose to replace standard top-k gating with dynamic-k routing that can adapt the number of activated experts to the difficulty of the input, leading to more optimal resource usage.

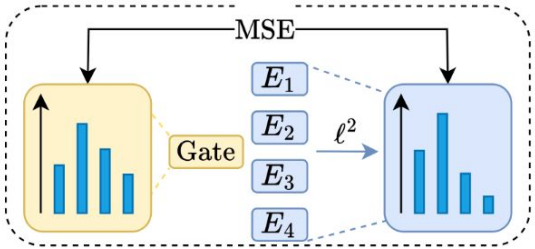


(c) Top-k vs dynamic-k gating

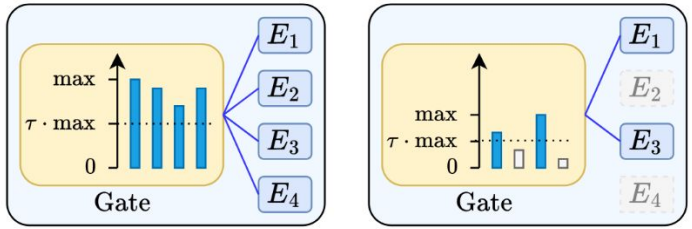
[Exploiting Activation Sparsity with Dense to Dynamic-k Mixture-of-Experts Conversion](), Szatkowski et al., 2025

Expert contribution routing and dynamic-k gating

- We improve the router training scheme and frame it as a regression problem.
- Our routers predict the L2 norm of each expert output, which allows for approximation of each expert's relative contribution.

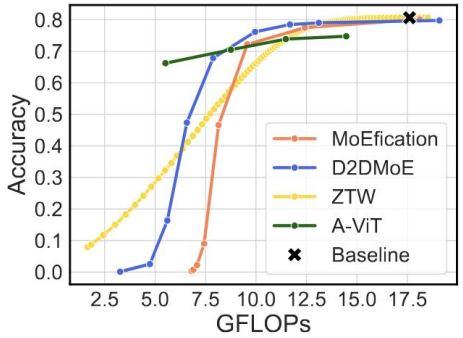


- This allows us to dynamically determine the number of experts to execute on a per-token basis.
- With our gating, we can control the average computation of the model through the hyperparameter without retraining.

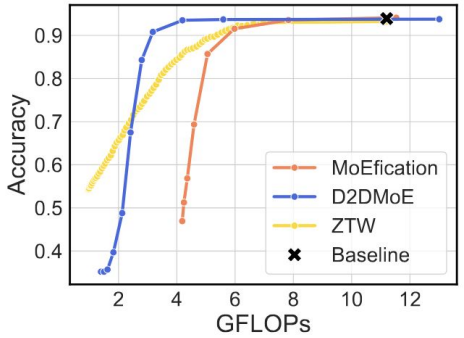


$$G(z)_i = \begin{cases} 1 & \text{if } R(z)_i \geq \tau \cdot \max R(z) \\ 0 & \text{if } R(z)_i < \tau \cdot \max R(z) \end{cases}$$

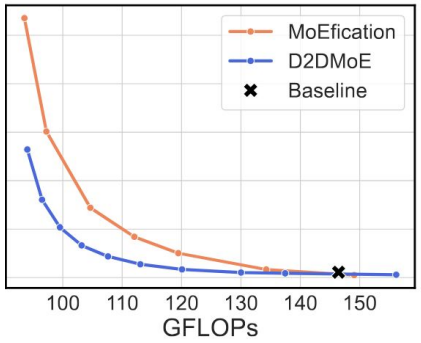
Cost-performance characteristics for D2DMoE



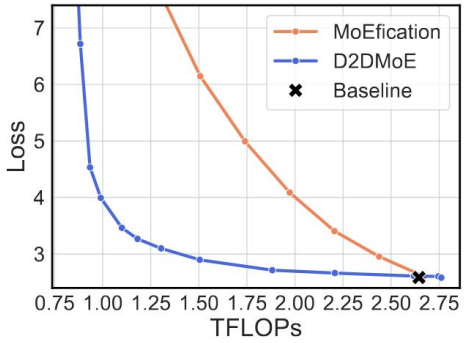
(a) ViT-B on ImageNet-1k



(b) BERT-base on CARER



GPT-2-base on OpenWebText



(d) Gemma-2B on C4

- D2DMoE offers better cost-performance characteristic than other adaptive computation methods

Exploiting Activation Sparsity with Dense to Dynamic-k Mixture-of-Experts Conversion, Szatkowski et al., 2025

End-to-end performance vs latency for D2DMoE

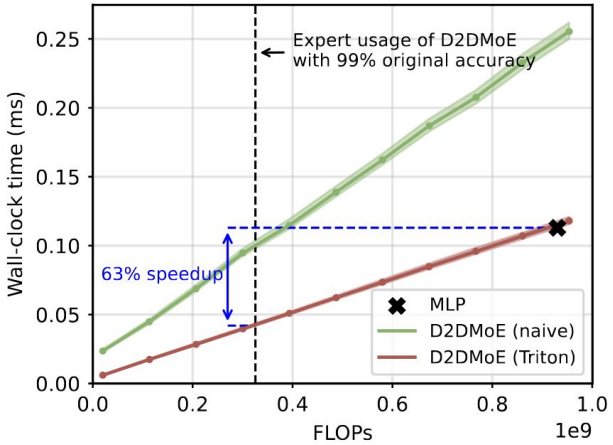


Figure 5: Single D2DMoE layer execution wall-clock time.

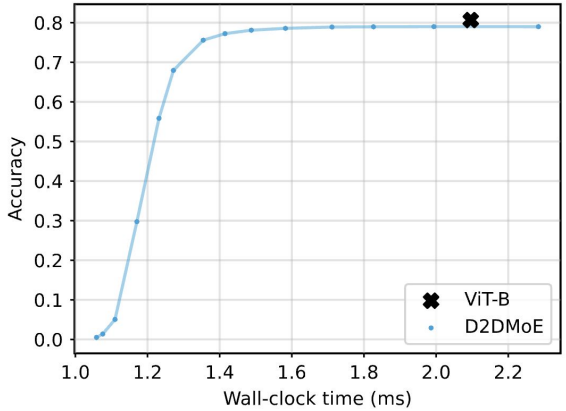
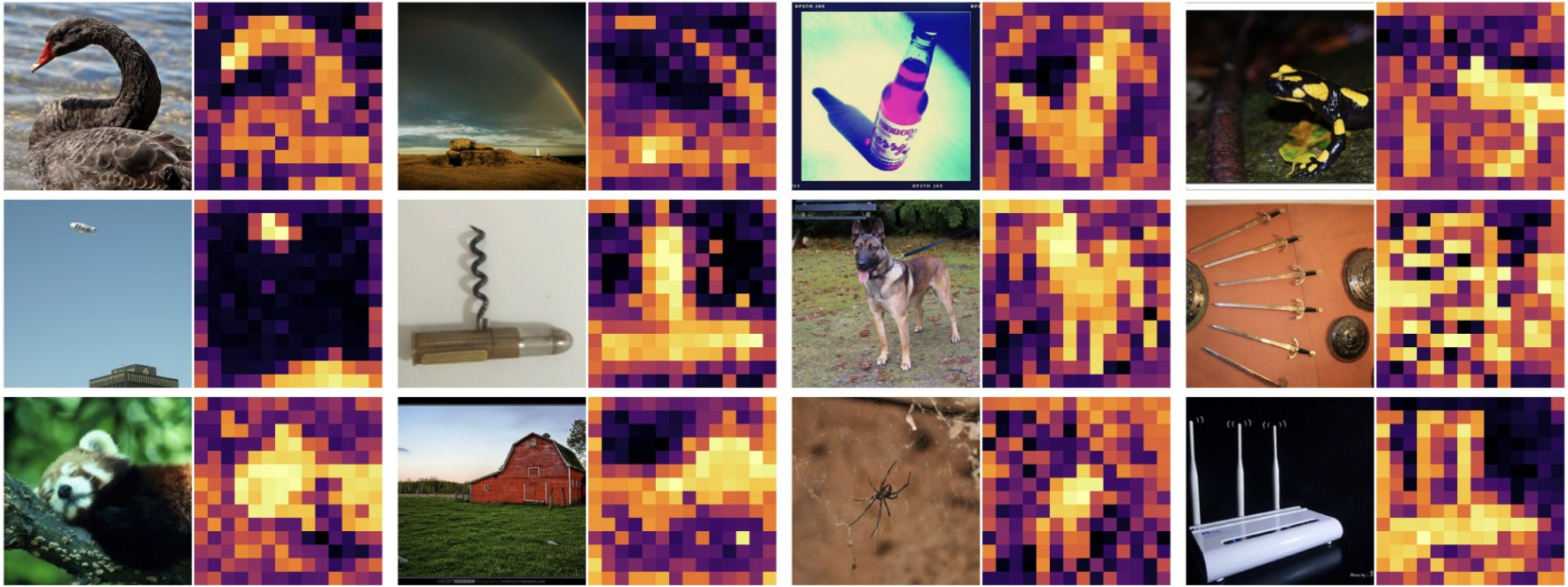


Figure 11: Wall-clock time measurements of the ViT-B model and its corresponding D2DMoE model.

- We implement D2DMoE kernel in Triton to transfer FLOPs savings to latency
- End-to-end, we achieve around 30% inference time acceleration before the accuracy degrades

Exploiting Activation Sparsity with Dense to Dynamic-k Mixture-of-Experts Conversion, Szatkowski et al., 2025

Expert utilization maps for D2DMoE-ViT



NEURAL INFORMATION PROCESSING SYSTEMS **UniReps workshop**

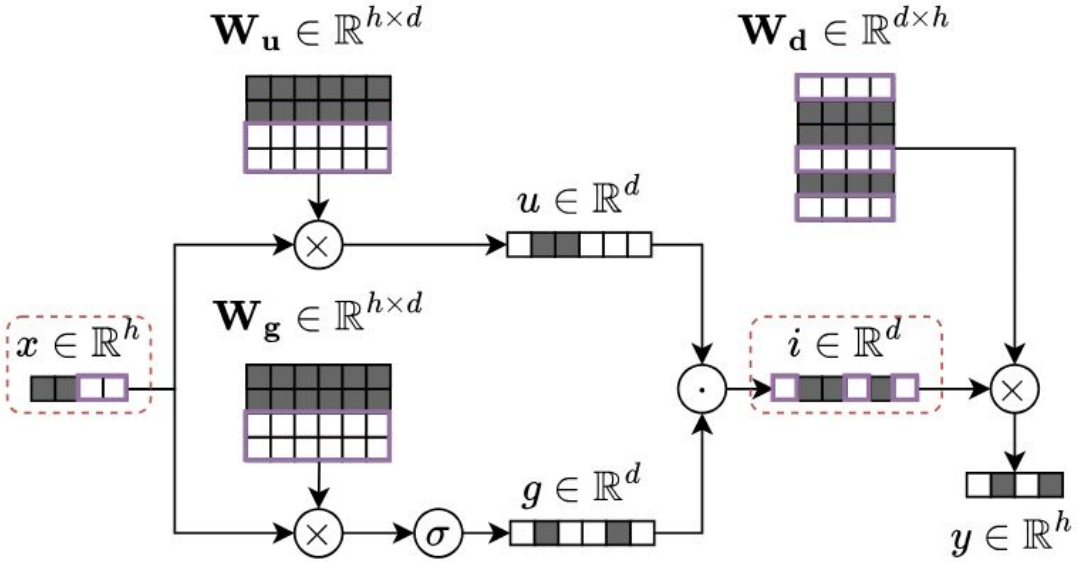
Universal Properties of Activation Sparsity in Modern Large Language Models

Filip Szatkowski, Patryk Będkowski, Alessio Devoto, Jan Dubiński, Pasquale Minervini, Mikołaj Piórczyński, Simone Scardapane, Bartosz Wójcik

Warsaw University of Technology IDEAS NCBR JAGIELLONIAN UNIVERSITY IN KRAKÓW SAPIENZA UNIVERSITÀ DI ROMA NASK THE UNIVERSITY OF EDINBURGH

Router-free activation sparsity

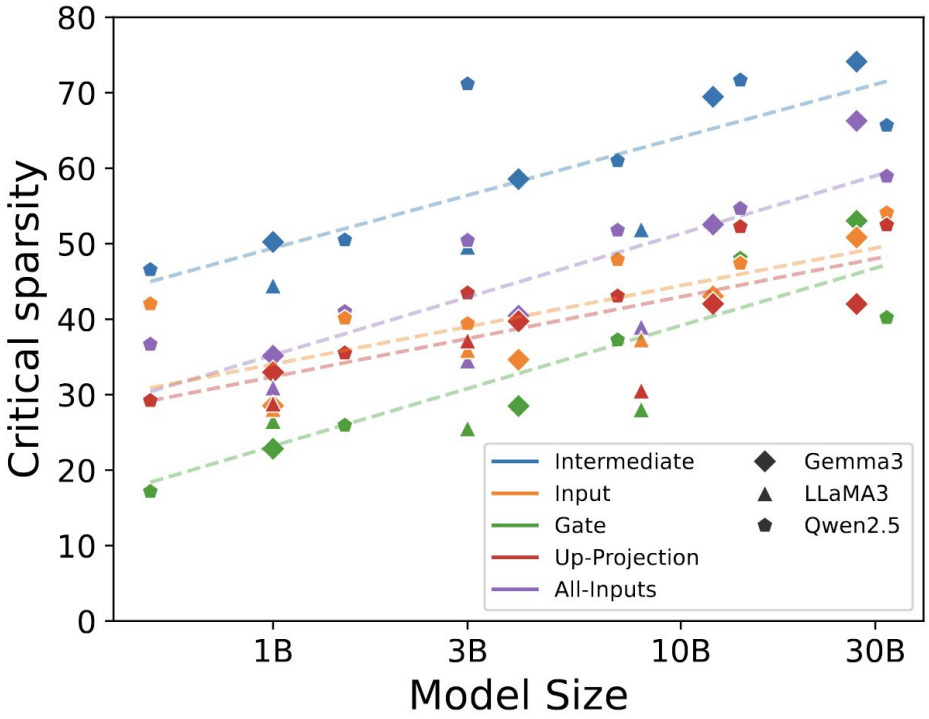
- We can further simplify MoEfication paradigm if we treat single neurons as experts
- This is referred to as activation sparsity
- Irrelevant parts of the activations can be skipped alongside corresponding weights during the matmul in the forward pass



Universal Properties of Activation Sparsity in Modern Large Language Models, Szatkowski et al., 2025

Activation sparsity is prevalent everywhere in LLM FFNs

- Modern LLMs are harder to directly utilize with activation sparsity, as they do not use ReLU and their FFNs structure is more complex.
- To examine activation sparsity in these models, we sparsify the activations in the model during the evaluation and measure critical sparsity, the highest activation sparsity where the model retains at least 99% performance.
- Interestingly, we find relatively high functional sparsity across all components of FFN layers, even without any architectural biases towards sparsity, e.g. at FFN inputs.



Universal Properties of Activation Sparsity in Modern Large Language Models, Szatkowski et al., 2025

Conclusions

Conclusions

- Adaptive computation methods allows us to trade-off the model performance and computational efficiency.
- Adaptive models can also be more expressive in some use cases.
- Aspects of these models are already used in the frontier architectures.
- Unfortunately, these methods provide lossy acceleration.
- Current frameworks and hardware are also not very friendly for flexible information processing algorithms.

Thanks for listening