Mechanistic Interpretability for Vision Models Optimization

23th July 2025

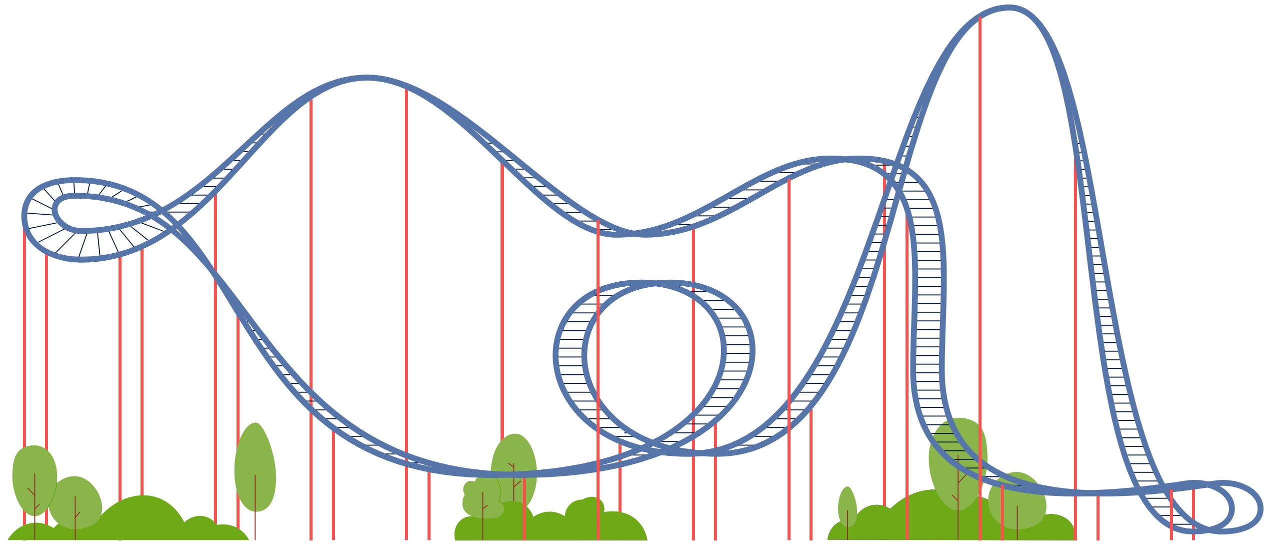
Computer Vision's course project

Authors:
Camilla Giuliani [1883207] & **Pietro D'Annibale** [1917211]

Project resources

🔗 https://colab.research.google.com/drive/1NNMyHI6ySeZPHcacPNtQd6y8-yUvGMZX#scrollTo=6jzzOI7xEby3

🔗 https://github.com/Sassotek/Mechanistic-Interpretability-for-Vision-Models-Optimization



A Note on the Journey

The development of this project has been a complex experience, much like a roller-coaster ride of ups and downs but each difficulty pushed us to grow and find alternative solutions.

Overview

- Goal and ideas
- Hardware and settings
- Dataset
- Model Architecture
- Training
- ACDC → pruning
- Final results and evaluation
- Future improvements
- References

Context and challenges

- ✓ ViTs show very high performance on many vision task.
- ✗ High computational cost makes ViTs not suitable for edge devices with limited hardware capabilities.

Mechanistic Interpretability

Growing research area that aims to reverse-engineer neural networks by understanding their internal components and computations. While it has been mainly applied to small language models, recent studies have started exploring its use in Vision Transformers as well.

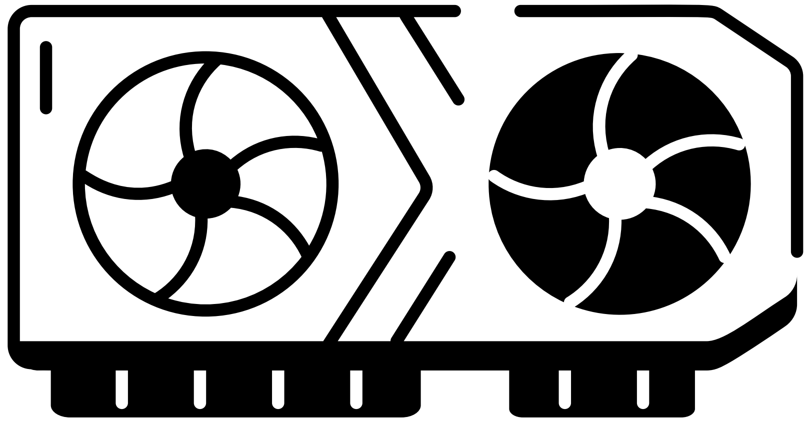
The project goal

Reduce the inference time of a ViT model by adopting the ACDC mechanistic interpretability technique to remove those edges that are more irrelevant for the output computation and analyze how inference time and accuracy are affected.

Hardware & settings

2 different GPUs were used while working on colab

Nvidia GeForce RTX 3070



Nvidia Tesla T4

Dataset

Tiny ImageNet

- 200 classes
- 64x64 pixels images
- 110k samples

Augmentations applied {

Random horizontal flip

Random resized crop

Random rotation

Gaussian noise

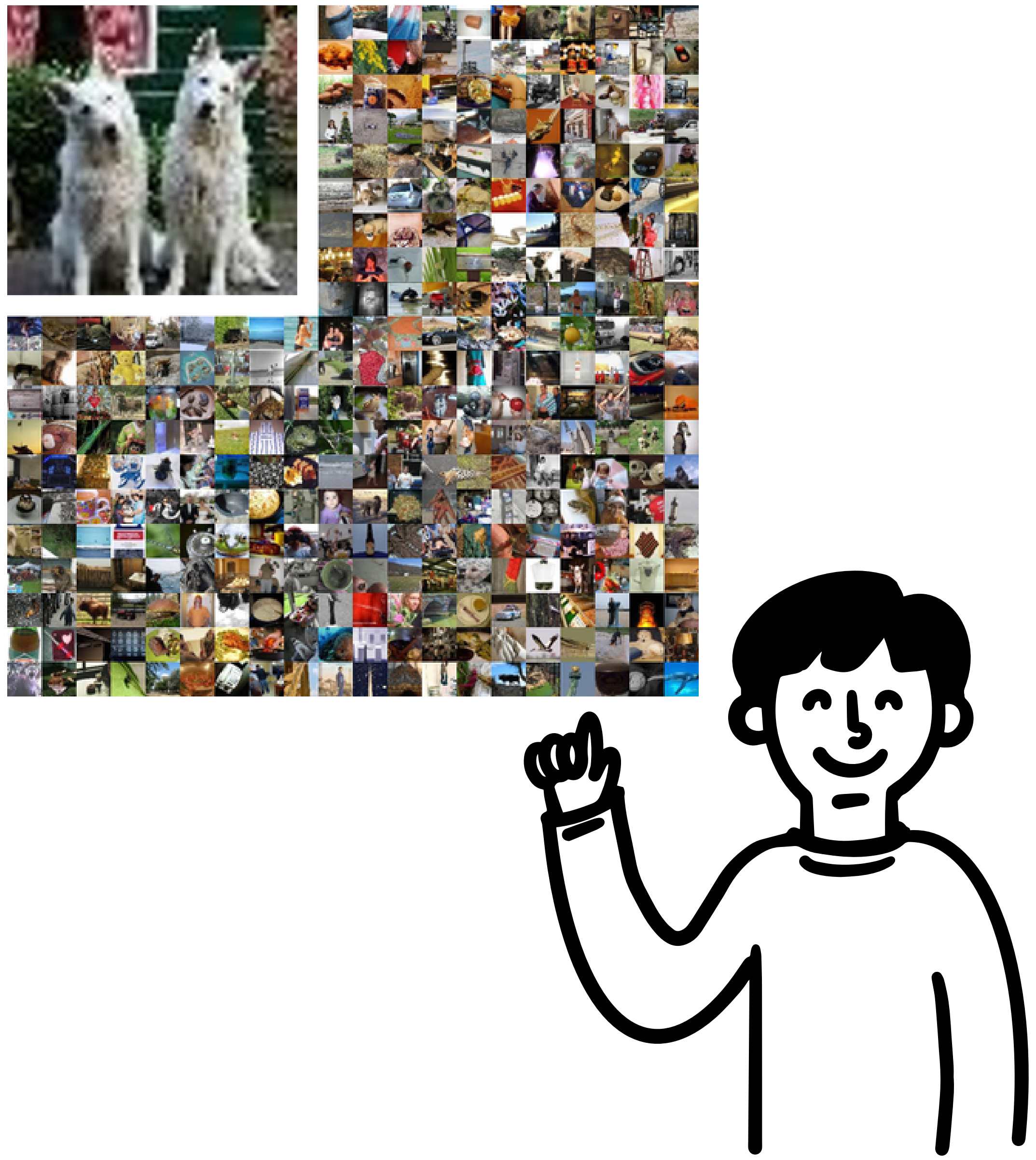
Random erasing

Normalization with ImageNet mean & std

CutMix

MixUp

}



Model Architecture

We tested multiple configurations of the ViT Model by varying key hyperparameters such as latent size, patch size, number of encoder layers, and number of MLP heads.

Composition of the **last version** considered :

-Latent Size: 256

-Patch Size: 8

-12 Encoders

-8 MLP Heads

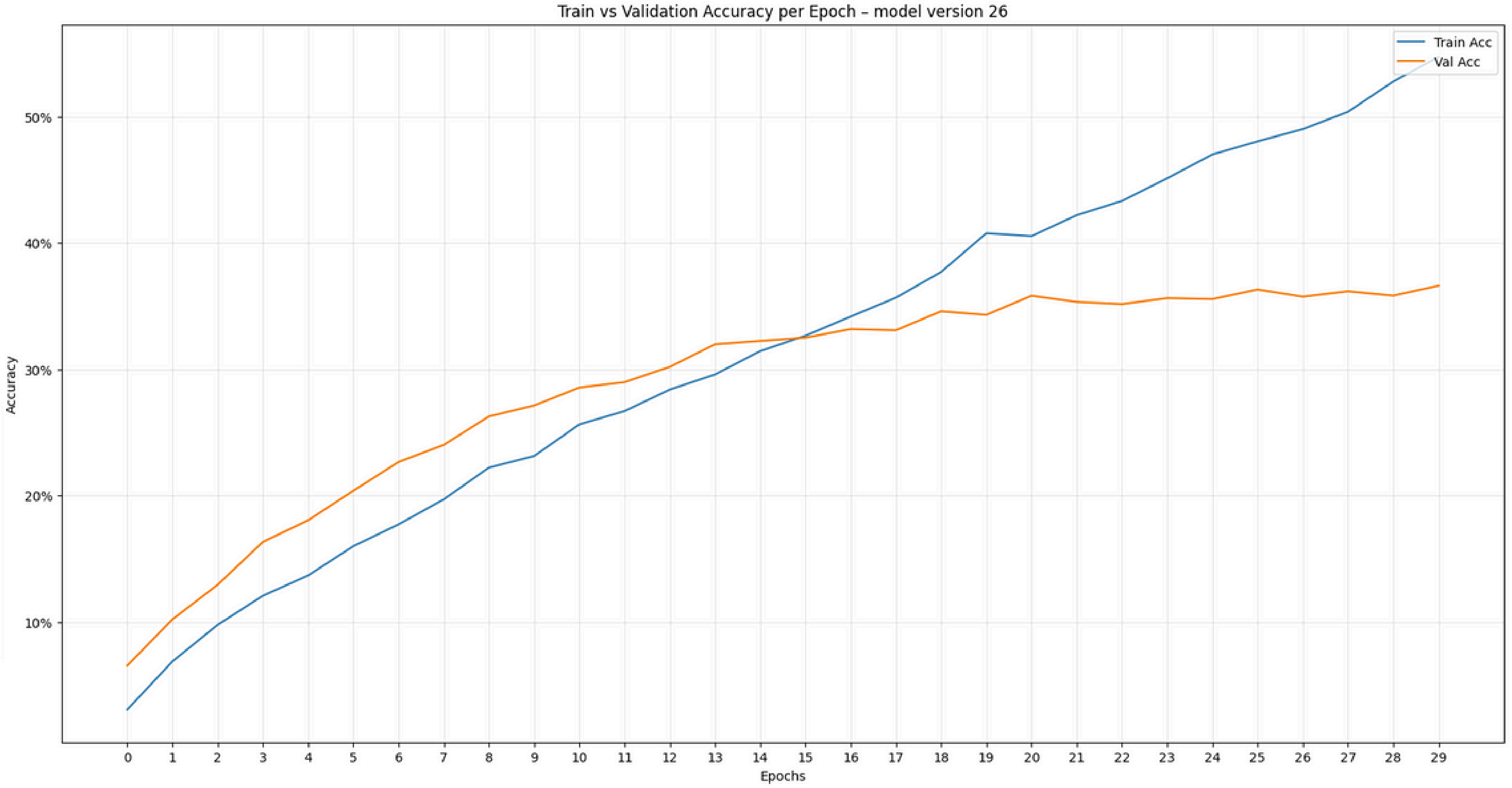
best trade-off between accuracy and computational cost

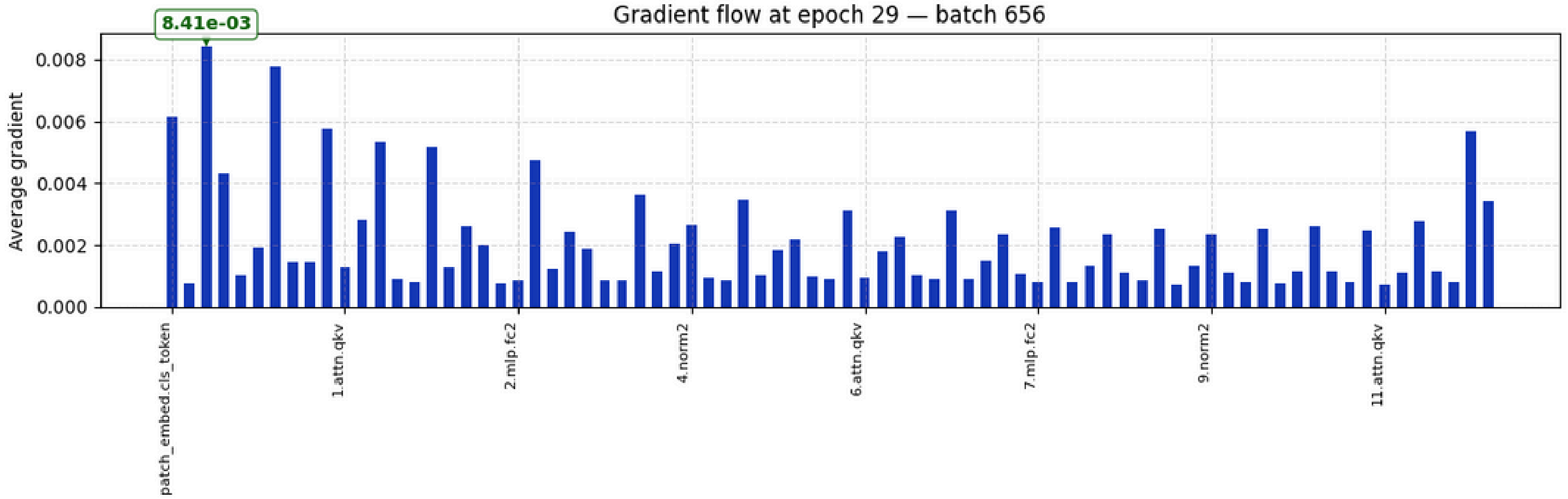
Training

last version of our ViT trained with

- 30 epochs
- Batch size: 128
- Gradient flow visualization
- AMP: Automatic Mixed Precision
- CosineAnnealingLR

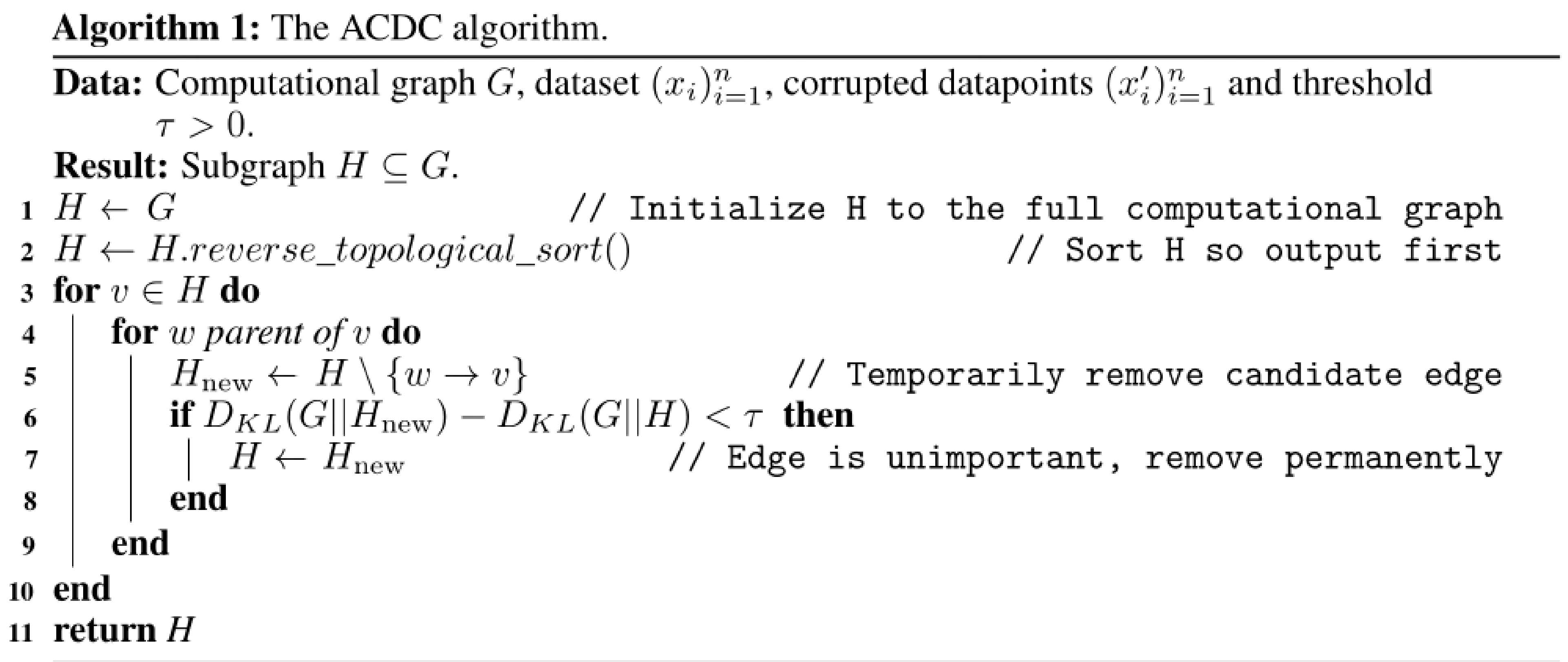
Training was guided by (Soft Target) Cross Entropy Loss, and performance was evaluated using Accuracy.


Train vs Validation Accuracy per Epoch – model version 26


Gradient flow at epoch 29 — batch 656

acdc

automatic circuit discovery



Algorithm 1: The ACDC algorithm.

```
Data: Computational graph $G$, dataset $(x_i)_{i=1}^n$, corrupted datapoints $(x'_i)_{i=1}^n$ and threshold
      $\tau > 0$.
Result: Subgraph $H \subseteq G$.
$H \leftarrow G$                                      // Initialize H to the full computational graph
$H \leftarrow H.reverse\_topological\_sort()$                          // Sort H so output first
for $v \in H$ do
    for $w$ parent of $v$ do
        $H_{\text{new}} \leftarrow H \setminus \{w \to v\}$                // Temporarily remove candidate edge
        if $D_{KL}(G||H_{\text{new}}) - D_{KL}(G||H) < \tau$ then
            $H \leftarrow H_{\text{new}}$                      // Edge is unimportant, remove permanently
        end
    end
end
return $H$
```

We implemented the ACDC algorithm based on the method proposed in [1](A. Conmy et al. ,2023).

The algorithm identifies and removes edges from the **computational graph** that have minimal impact on the final output.

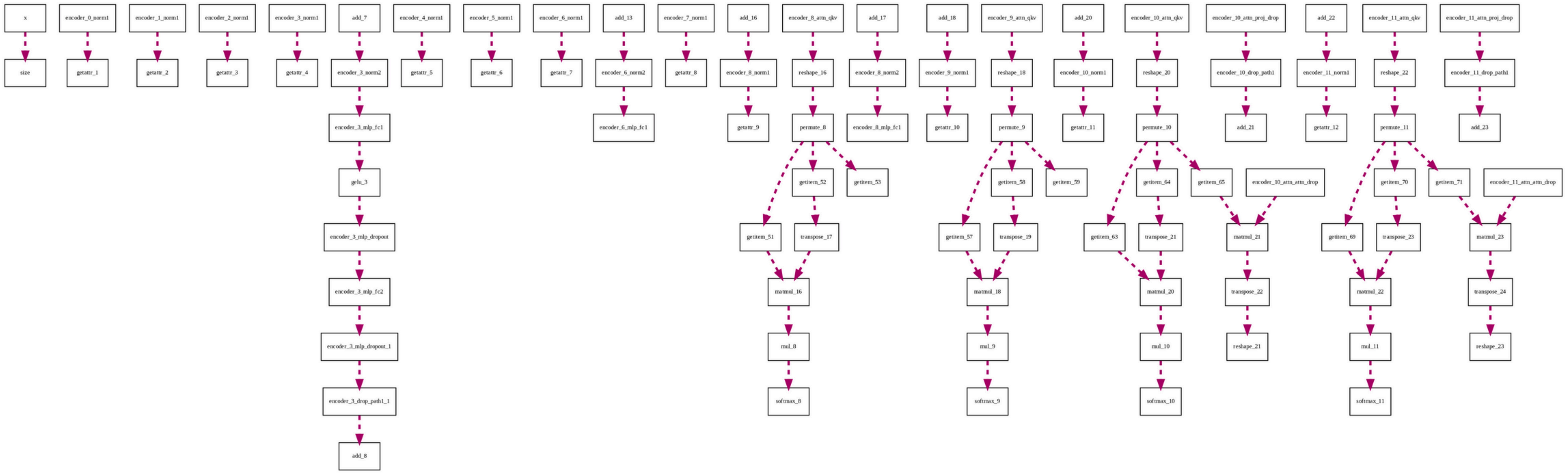
metric employed for pruning

KL-divergence

Several values of the τ parameter were tested

Pruning Phase

last version of our ViT after ACDC with τ = 5e-2 had **81 cuts / 518 edges**



⊘ Make zero the contribution of the nodes that contribute less to the output computation.

✂ Remove from the computational graph the edges with nodes whose contribution is zero

FINAL RESULTS

Model	Test loss	Test Accuracy	InferenceTime
Baseline	361.09864	36.25%	61.31 s
Pruned	383.527899	33.00%	59.87 s
Baseline Re-Trained	368.932719	36.34%	60.51 s
Pruned Trained	365.003407	36.07	58.11

Inference time remains nearly constant after pruning

Another 5 epochs training phase done on both pruned model and baseline model, the training gap between non-pruned and pruned model is recovered during training.

Future improvements

Optimizing the balance between inference efficiency and accuracy remains an open challenge. Future directions could include experimenting with higher τ values while compensating for accuracy loss using techniques like Knowledge Distillation or selective re-training of key components.

Edge Attribution Patching (EAP), proposed in [2] (Attribution Patching Outperforms Automated Circuit Discovery, A. Syed et al., 2024), is a faster and more efficient alternative to ACDC and could be considered in future work — potentially in combination with ACDC itself, as suggested in the original paper.

The success of this procedure is very important because it can lead to model response time useful to be employed in real time applications.

References

[1] A. Conmy et al. (2023). Towards Automated Circuit Discovery for Mechanistic Interpretability.
In: Advances in Neural Information Processing Systems 36 (NeurIPS 2023)
[https://arxiv.org/abs/2304.14997]

[2] A. Syed, C. Rager and A.Conmy, (2024). Attribution Patching Outperforms Automated Circuit Discovery,
BlackboxNLP 2024.
[https://arxiv.org/abs/2310.10348]

[3] A. Vaswani et al. (2017). Attention is all you need.
In: Advances in Neural Information Processing Systems 36 (NeurIPS 2017)
[https://arxiv.org/abs/1706.03762]

[4] TinyImageNet dataset [https://www.kaggle.com/datasets/wissamsalam/tiny-imagenet-cleaned-for-classification]

[5] A.Dosovitskiy et al.(2021).An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale
[https://arxiv.org/abs/2010.11929]

[6] Einops Guide [https://nbviewer.org/github/arogozhnikov/einops/blob/main/docs/1-einops-basics.ipynb]

[7] VISO.ai [https://viso.ai/deep-learning/vision-transformer-vit/]