Convergence Rate of Transformer with Pruning

VISUALIZATION

Team 4
Taegyu Park, Byeongju Woo

May 30, 2022

POSTECH

Table of Contents

Introduction

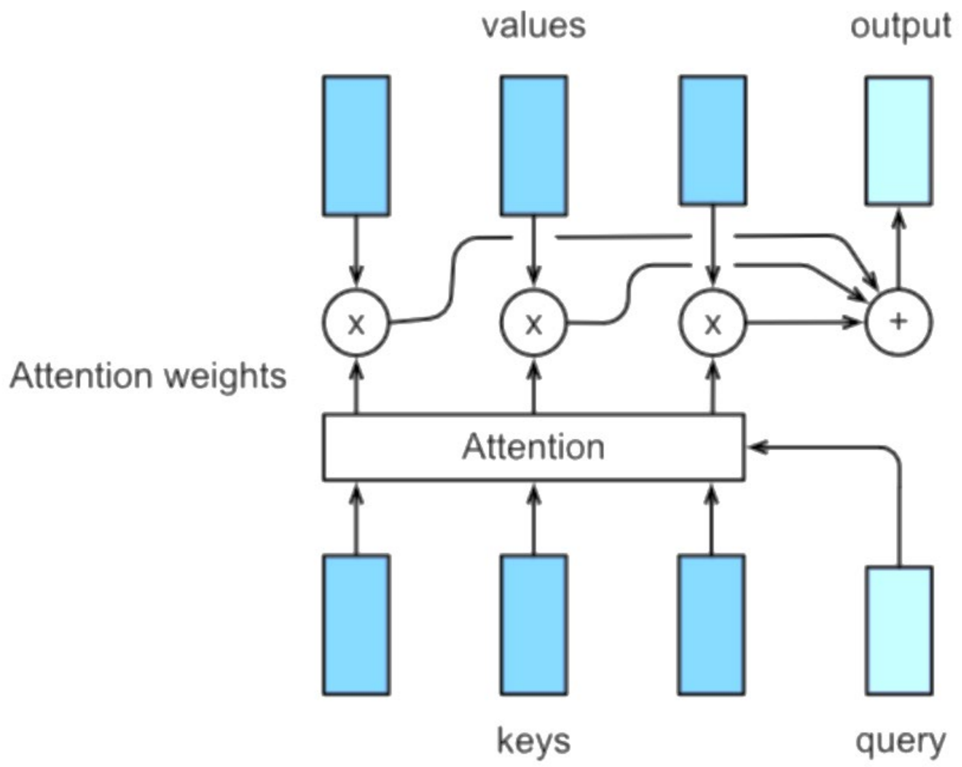
3 Experiments about **Convergency of Transformer**

I. Attention Mechanism
II. Transformer
III. Optimizers
IV. ReduceLROnPlateau
V. Lottery Hypothesis
VI. Loss Landscape

Preliminary

Attention mechanism

Attention mechanism examines bipartite pairwise similarities of a query sequence and a key sequence. And then it uses the result to appropriately mix the value sequence that is correlated to the key sequence.



$$e_{ij} = a(s_{i-1}, h_j) \quad (1)$$

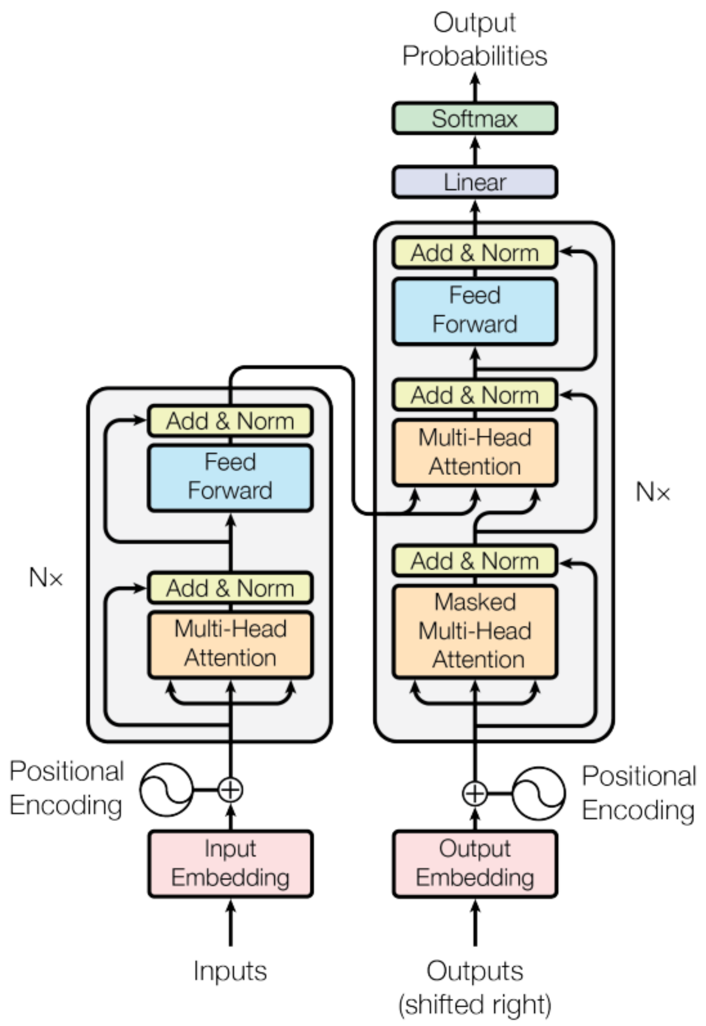
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \quad (2)$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \quad (3)$$

Transformer model

Transformer is an autoencoder model that uses multiple attention blocks.

Since attention blocks have quadratic time complexity, typically Transformer models are heavy and memory-consuming.



Brief Recap

Optimization

In the lectures, we've learned several optimization algorithms such as SGD and Adam.
Also, we've seen that for some configuration, decreasing learning rate is crucial for convergence.

SGD: $w_t = w_{t-1} - \eta \nabla_w L(x; w_{t-1})$

▶ This method requires a decreasing step size $\eta \to 0$ to converge.

(From slide 8)

Adam: $g_t = \nabla_w L(x; w_{t-1}),\ m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t,\ v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2,\ \widehat{m_t} = \frac{m_t}{1-\beta_1^t},\ \widehat{v_t} = \frac{v_t}{1-\beta_2^t}$

$$w_t = w_{t-1} - \gamma \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}} + \epsilon}$$

Proper optimization method for attention mechanism ?

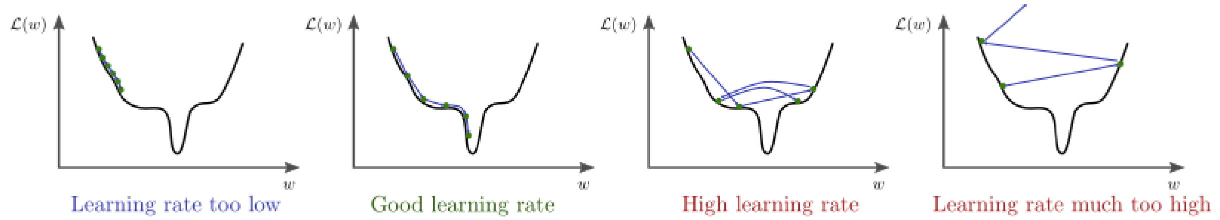
POSTECH

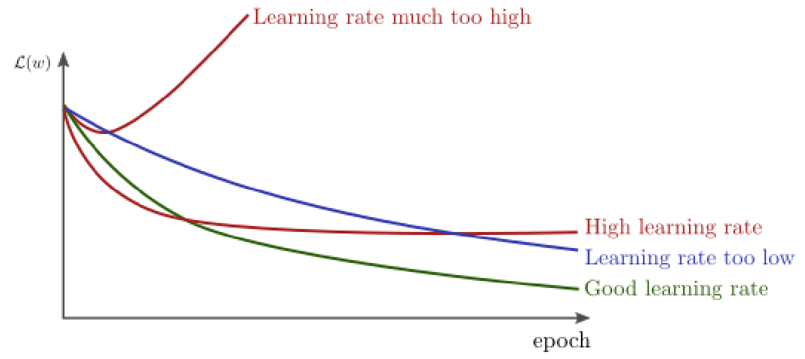
ReduceLROnPlateau

ReduceLROnPlateau learning rate scheduler

ReduceLROnPlateau(RedLR) is a learning rate scheduler
decreases the learning rate, only when a designated metric does not improve.

Why decrease learning rate? Because an optimizer can 'overshoot' the minimal point for some convex functions (and non-convex functions).





https://cs231n.github.io/neural-networks-3/

Lottery Hypothesis

- Lottery Hypothesis

A randomly-initialized, dense neural network contains a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original network after training for at most the same number of iterations.

Assume initial parameters are the same.
$f(x;\theta_t)$ has accuracy a with minimum validation loss.
$f(x;m\odot\theta_{t'})$ has accuracy a' with minimum validation loss.
m is a 'mask'.

$$\exists m\ s.t. \sum m \ll |\theta|, a' \geq a, t' \leq t$$

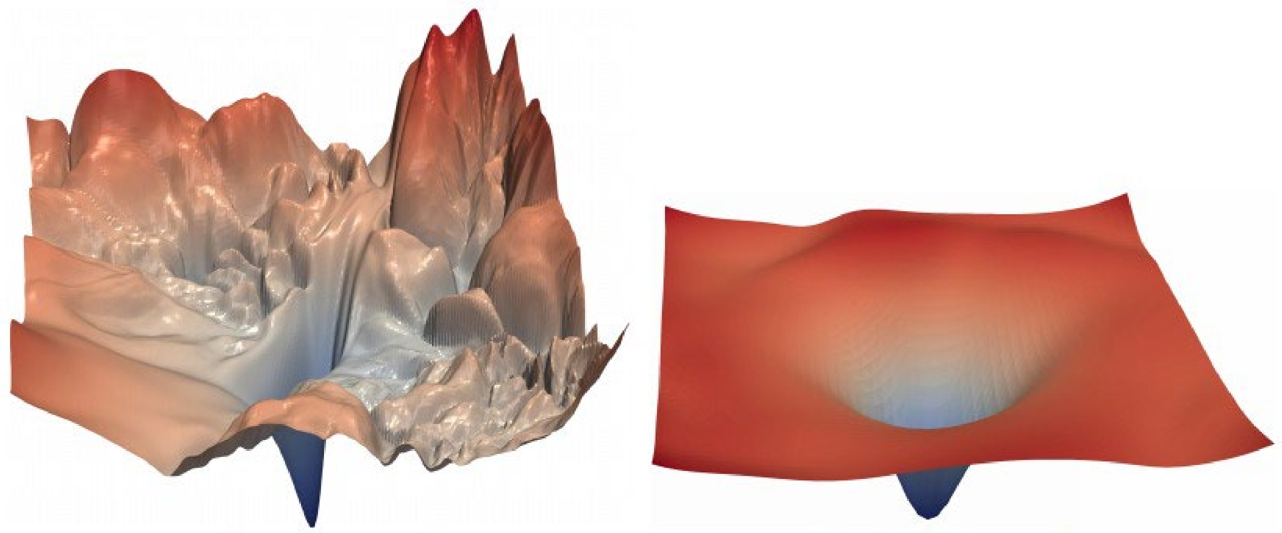
POSTECH

Frankle, J., & Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

Loss Landscape

Drawing loss landscape (1)

For two random tensors x and y, a loss landscape can be drawn by sampling points (α, β) that have the value $L(w + \alpha x + \beta y)$, as introduced in the paper (Li *et al.*, 2018).

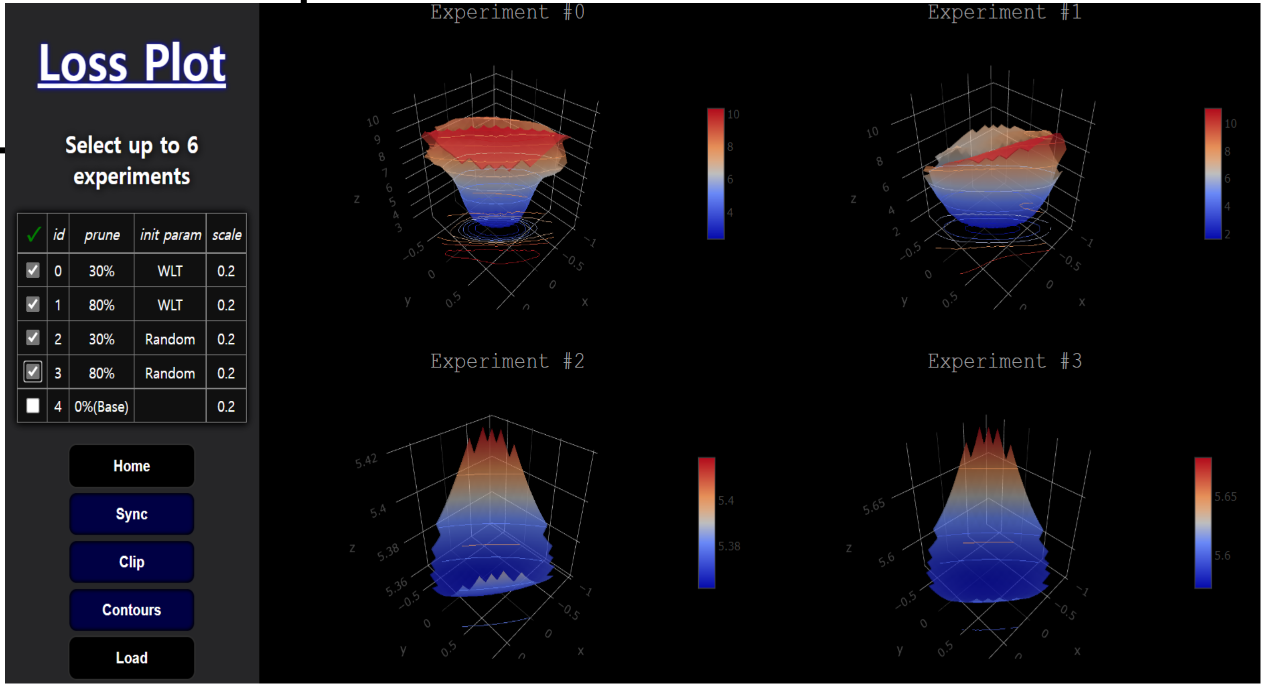


(a) without skip connections (b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

Loss Landscape

```
visualize_loss_landscape(model, grid_num, $x$, $y$):
  $\delta$   ← 2 / (grid_num − 1)
  FOR $\alpha$, $\beta$ EACH VALUE GROWS FROM -1 UP TO 1 BY $\delta$:
    $w$ ← model's parameter
    $w'$ ← $w + \alpha x + \beta y$
    $\mathcal{L}$ ← inference_loss(model($w'$))
    record($\alpha$, $\beta$, $\mathcal{L}$)
```



Loss Plot

Select up to 6 experiments

✓	id	prune	init param	scale
☑	0	30%	WLT	0.2
☑	1	80%	WLT	0.2
☑	2	30%	Random	0.2
☑	3	80%	Random	0.2
☐	4	0%(Base)		0.2

Experiment

I. Optimization for Attention

II. Visualizing Epoch-wise Landscape of Transformer

III. Visualize Loss Landscape of Pruned Transformer

Exp 1 : Optimization for Attention

Experiment settings

Purpose : To find proper optimization method for attention mechanism

First, generate a classification task dataset with a simple generator model. Then we train a single attention mechanism block using various optimization strategies.

Total number of 30,000 data generated in this experiment, and those are split 7:3 to create a training set and a validation set. Cross entropy loss is used.

We use 6 settings:

- SGD
- SGD with RedLR
- NAG
- NAG with RedLR
- Adam
- Adam with RedLR

Exp 2 : Visualizing Epoch-wise Landscape of Transformer

Experiment

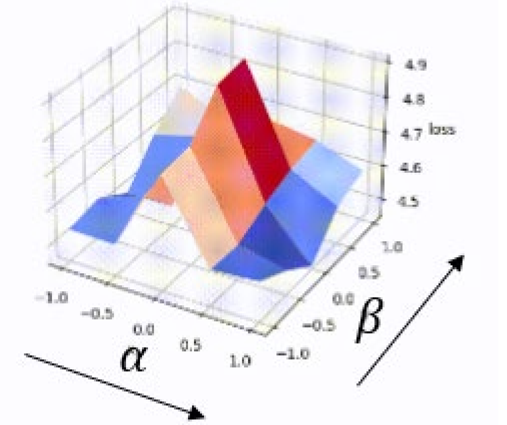
Experiment Steps

Purpose : investigate importance of initial parameter for pruned model training

- **Step1.** Train transformer baseline for WMT'16 Translation task, recording init params, w_0(Adam)
  (Reproduce Pytorch implementation of 'Attention is all you need')

- **Step2.** Prune 10% of the network from **Step1**, and train it from various initial parameter near w_0.

  - Training starts from the initial parameter $w_0 + \alpha x + \beta y$.
  - Set x direction as $\widehat{w^* - w_0}$ to observe its effect on landscapes.
  - Set y direction as random tensor



- **Step3.** Plot the 2D surface of a specific criteria : loss, perplexity, or accuracy.
  Also, by recording those 2D surfaces epoch-wise, create videos of changing 2D surfaces.

POSTECH

Exp 3 : Visualize Loss Landscape of Pruned Transformer

Experiment

Experiment Steps

Purpose : To gain insight of convergency of pruned transformer by visualizing loss landscape

- **Step1.** Train transformer baseline for WMT'16 Translation task, recording init params , w_0
  (Reproduce Pytorch implementation of 'Attention is all you need')

- **Step2.** Prune the network from **Step1**, and training from w_0 or random init

Recorded init, 30% pruning	Random init, 30% pruning
Recorded init, 80% pruning	Random init, 80% pruning

Same Implementation details with **Step1**

- **Step3.** Compare 5 models in terms of convergency late, accuracy and **loss landscapes**

Exp 3 : Visualize Loss Landscape of Pruned Transformer

Experiment

■ Implementation Details

Device	Tesla P100-PCIE-16GB (Google Colab+) , 1 GPU
Base Model	Pytorch implementation of *'Attention is all you need'*
Batch size	192
Epoch	100
Lr_mul	0.5
Optimizer	Adam

- Github code will be released as soon as possible

- You can reproduce Experiment with these settings !

■ Visualization Tool Details

- By using method of *'Visualizing the Loss Landscape of Neural Nets'* ,
- Save data as .CSV file, and visualize on website UI tool, named *'LossPlot'*

POSTECH

Result

I. Optimization for Attention

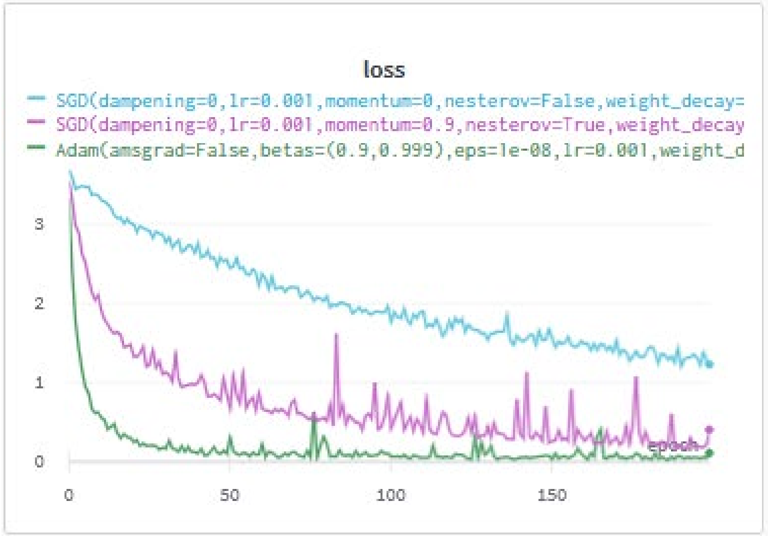
II. Visualizing Epoch-wise Landscape of Transformer

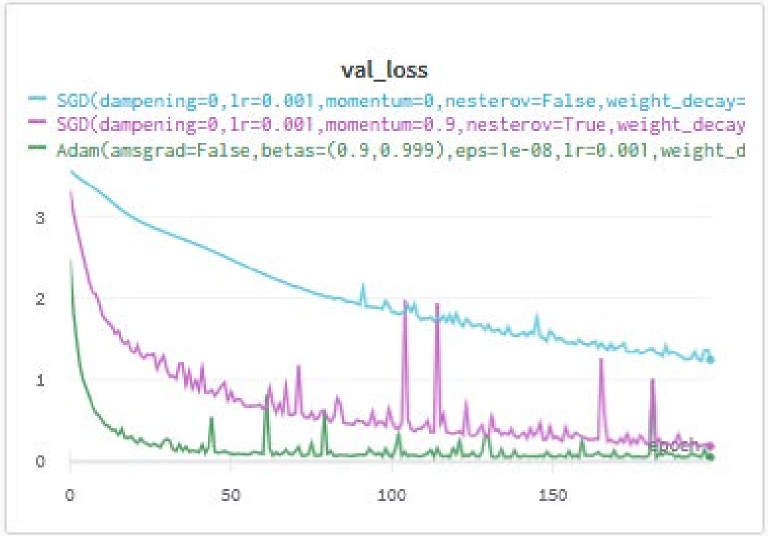
III. Visualize Loss Landscape of Pruned Transformer

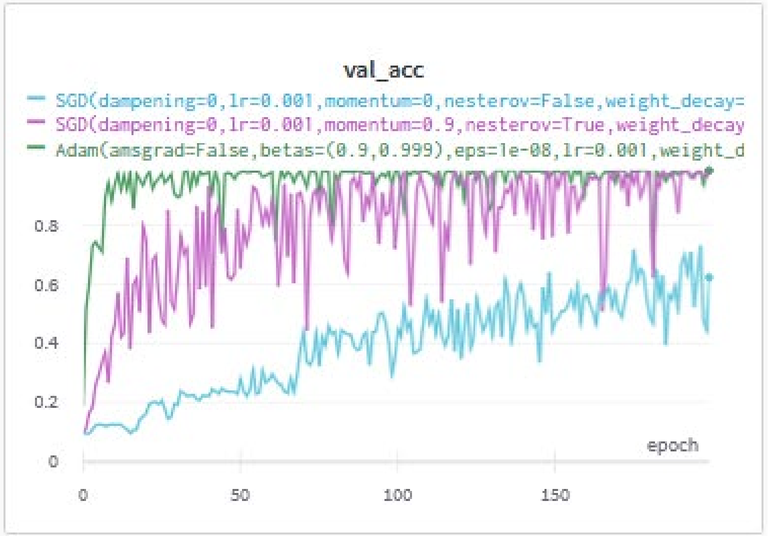
Exp 1 : Optimization for Attention

Result

■ Result graph







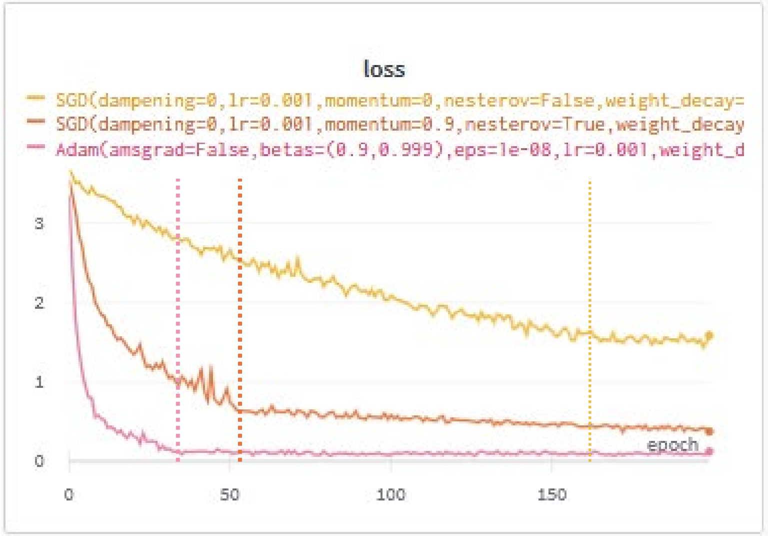
SGD
NAG
Adam

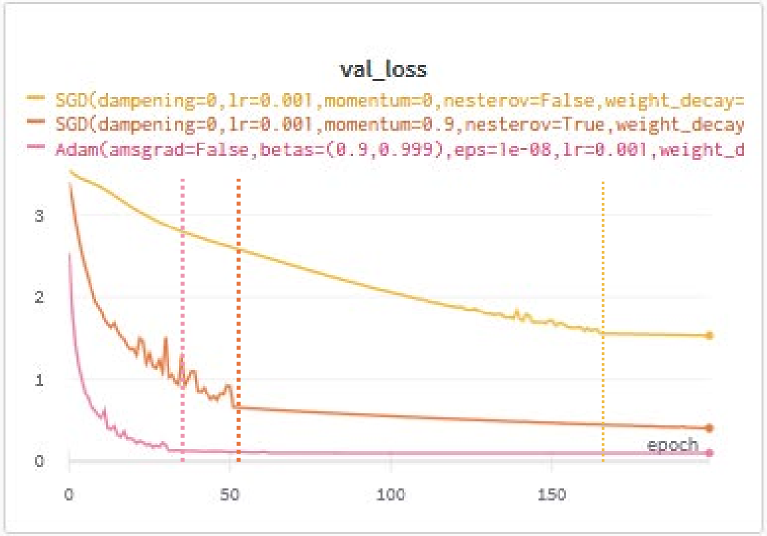
Convergence rate: Adam > NAG > SGD

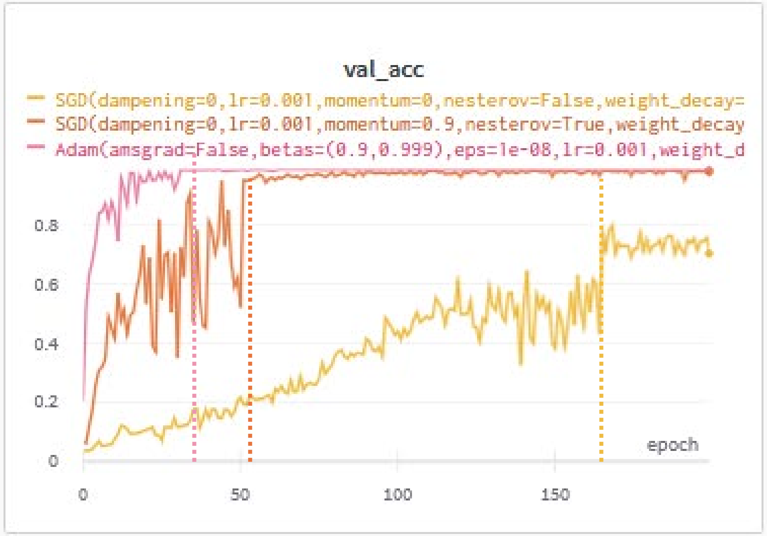
Exp 1 : Optimization for Attention

Result

Result graph







SGD+RedLR
NAG+RedLR
Adam+RedLR

Convergence rate: Adam+RedLR > NAG+RedLR > SGD+RedLR
(dotted lines represent learning rate decrease)

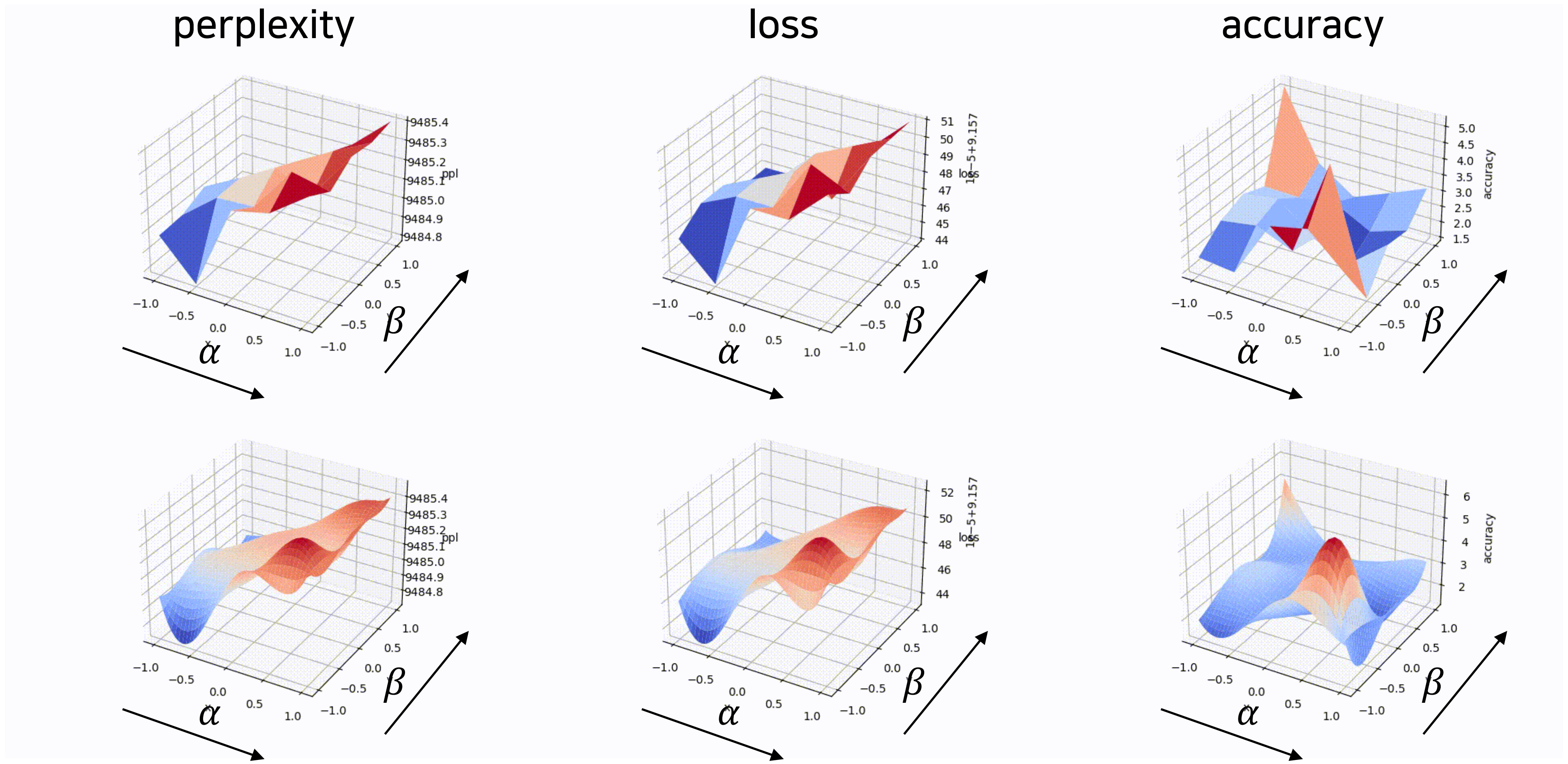
Much smoother as learning goes on

POSTECH

Exp 2 : Visualizing Epoch-wise Landscape of Transformer

Result

- Result graph: train set

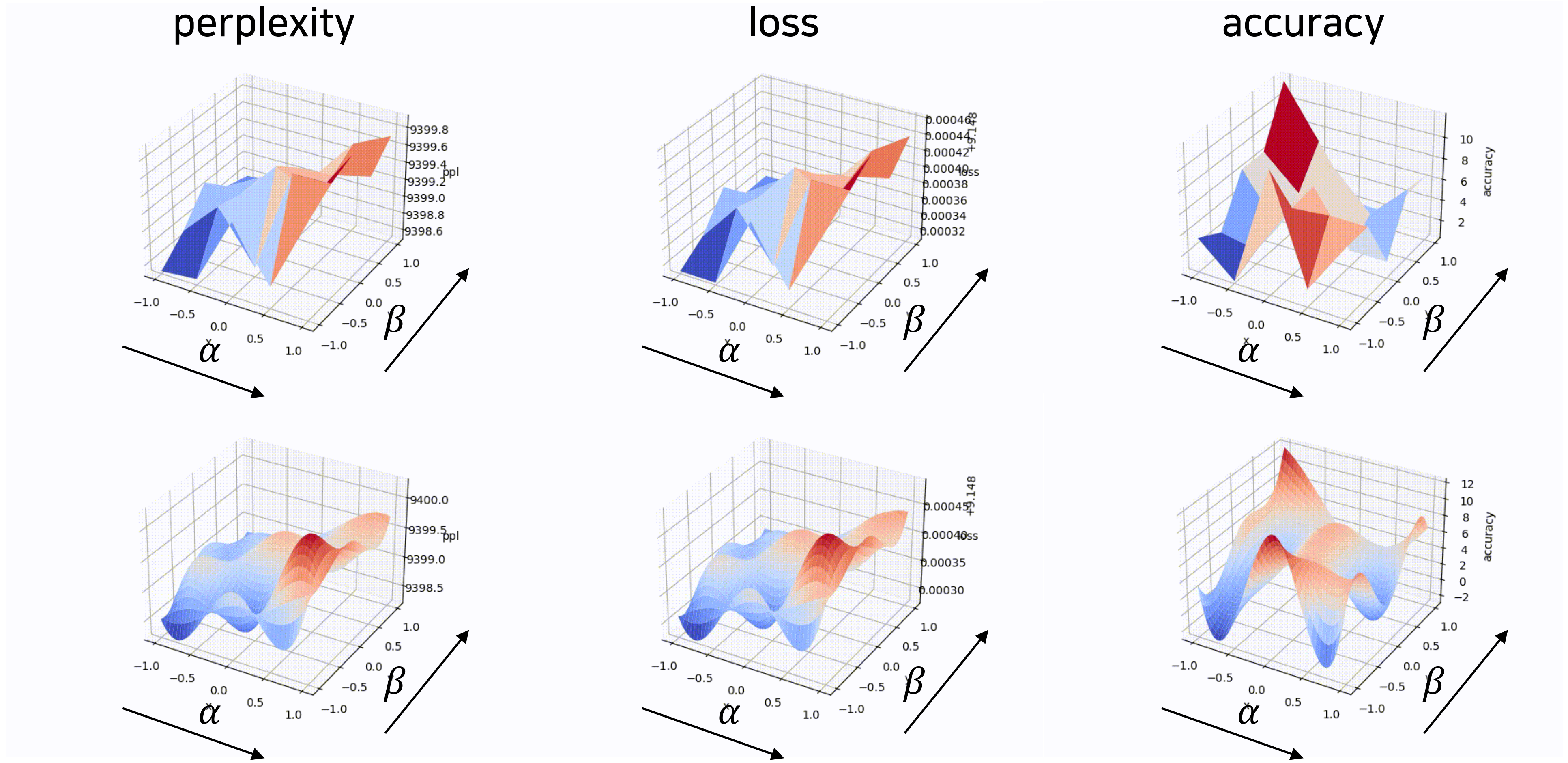


Up: original result. Down: smooth approximate

Exp 2 : Visualizing Epoch-wise Landscape of Transformer

Result

- Result graph: validation set



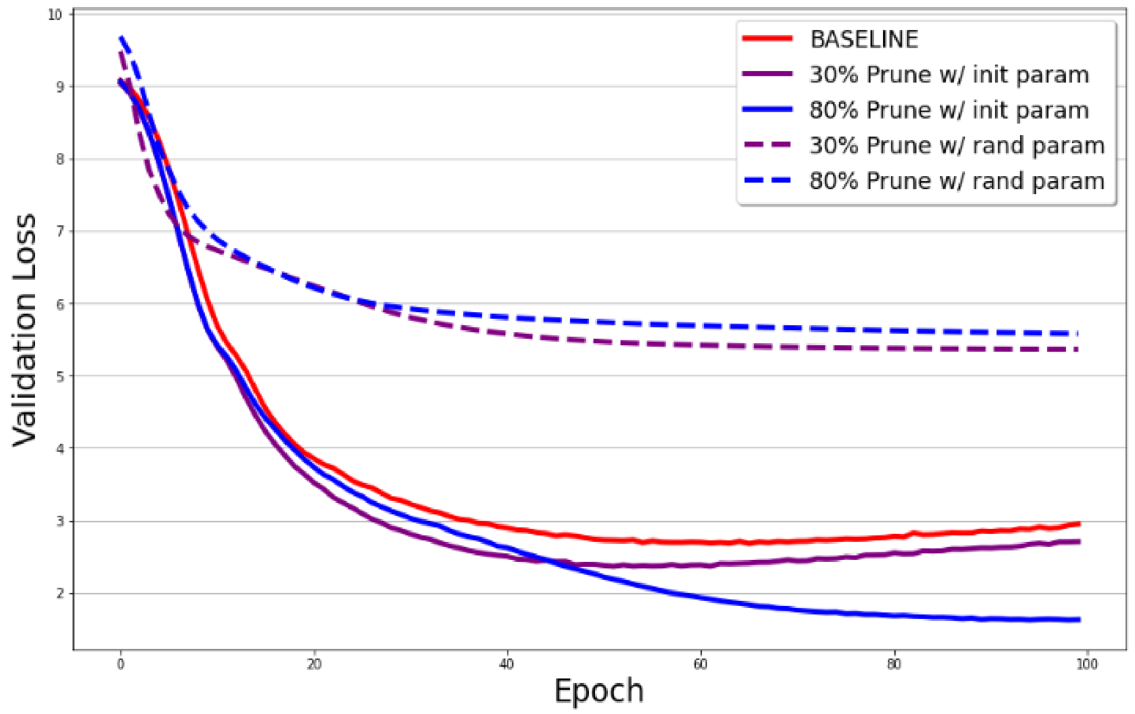
Up: original result. Down: smooth approximate

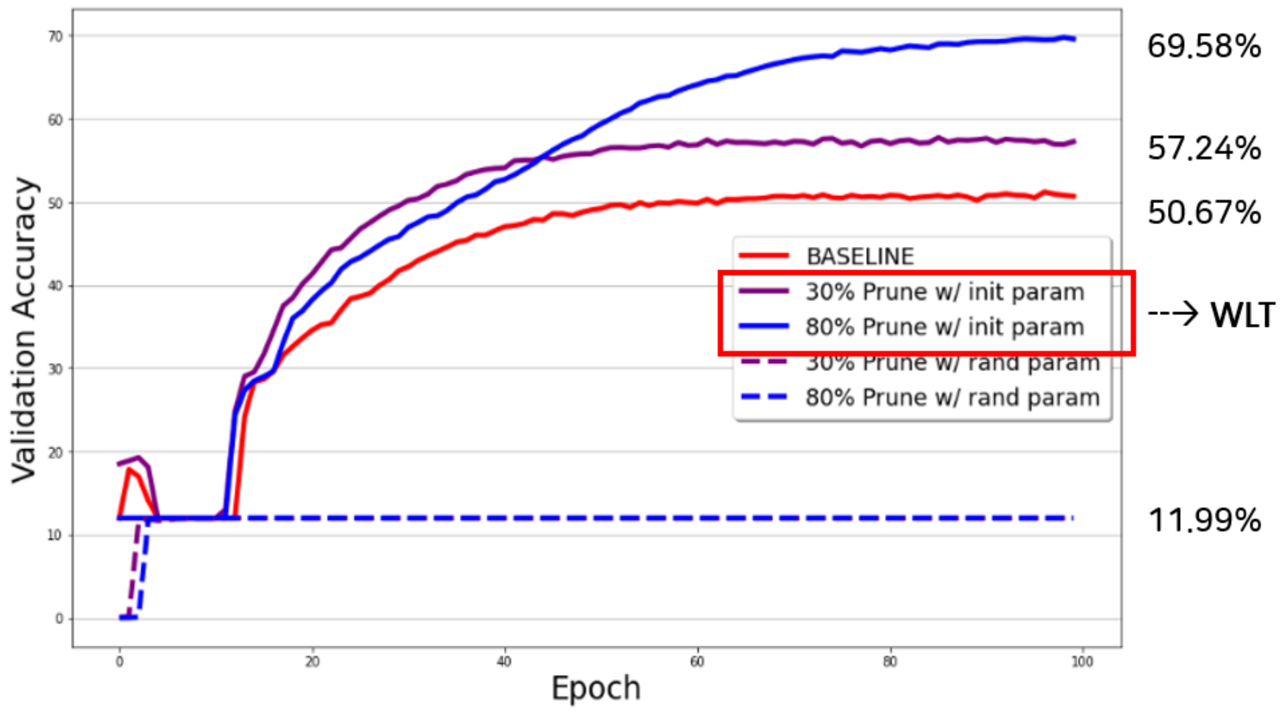
POSTECH

Exp 3 : Visualize Loss Landscape of Pruned Transformer

Result

- Validation loss & accuracy as training proceeds





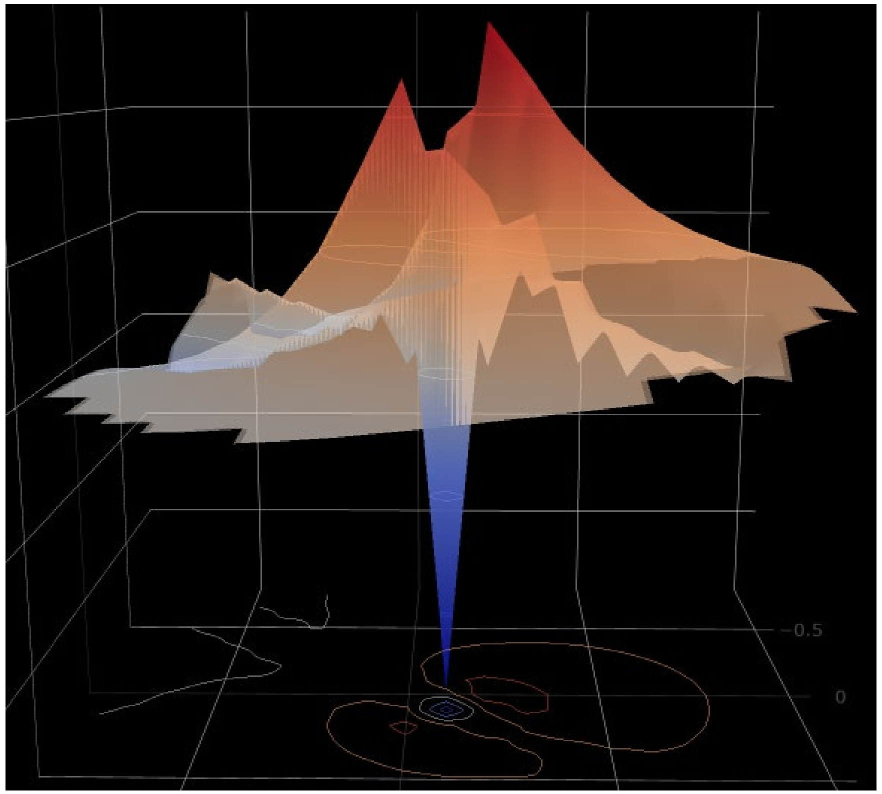
POSTECH

Exp 3 : Visualize Loss Landscape of Pruned Transformer

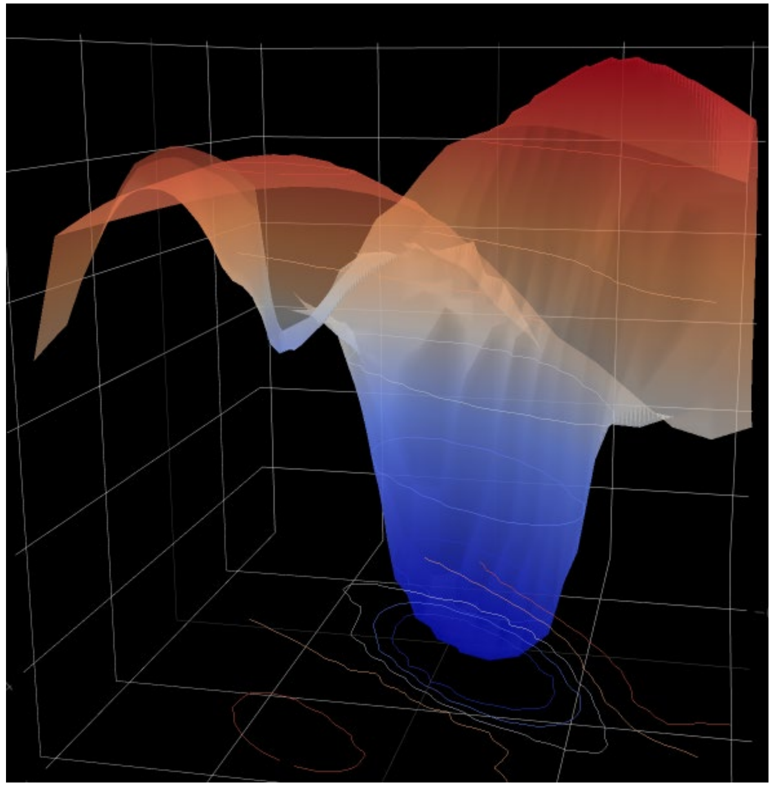
Result

- Baseline Visualization with different scales

Baseline, Global View (scale=1.0)



Baseline, Local View (scale=0.2)



※ Scale means the **range of search space** of landscape

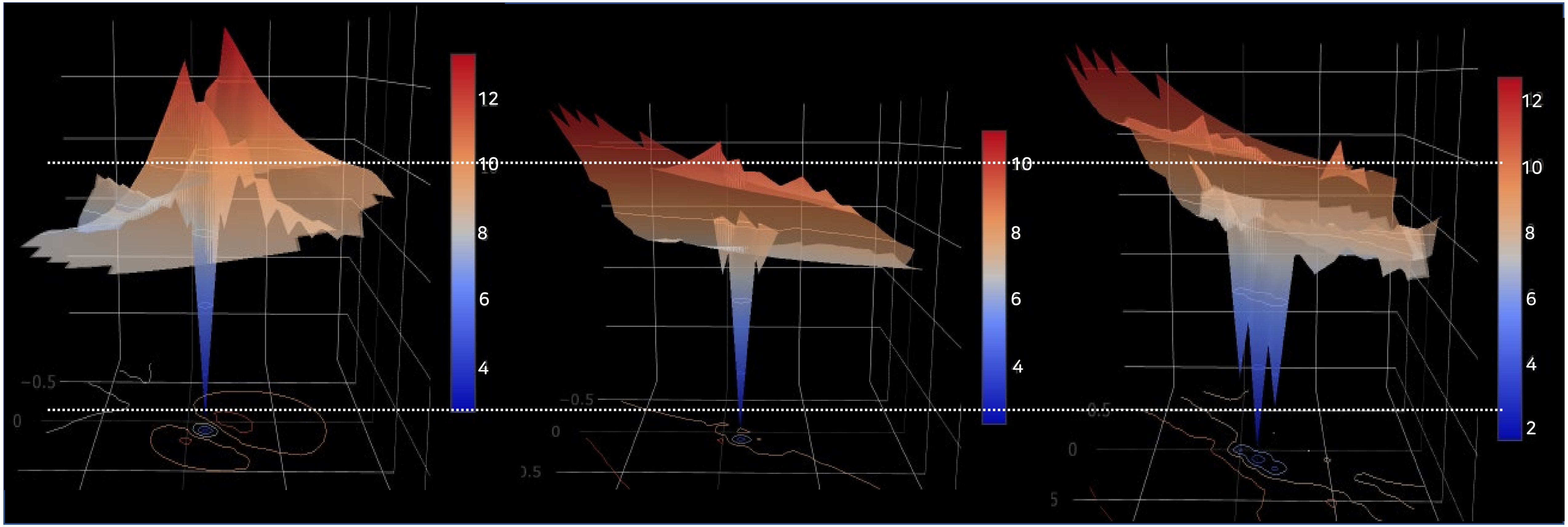
$$L(w + scale * \alpha x + scale * \beta y)$$

POSTECH

Exp 3 : Visualize Loss Landscape of Pruned Transformer

Result

- Compare WLTs with baseline, Global View



Baseline

30% pruned WLT

80% pruned WLT

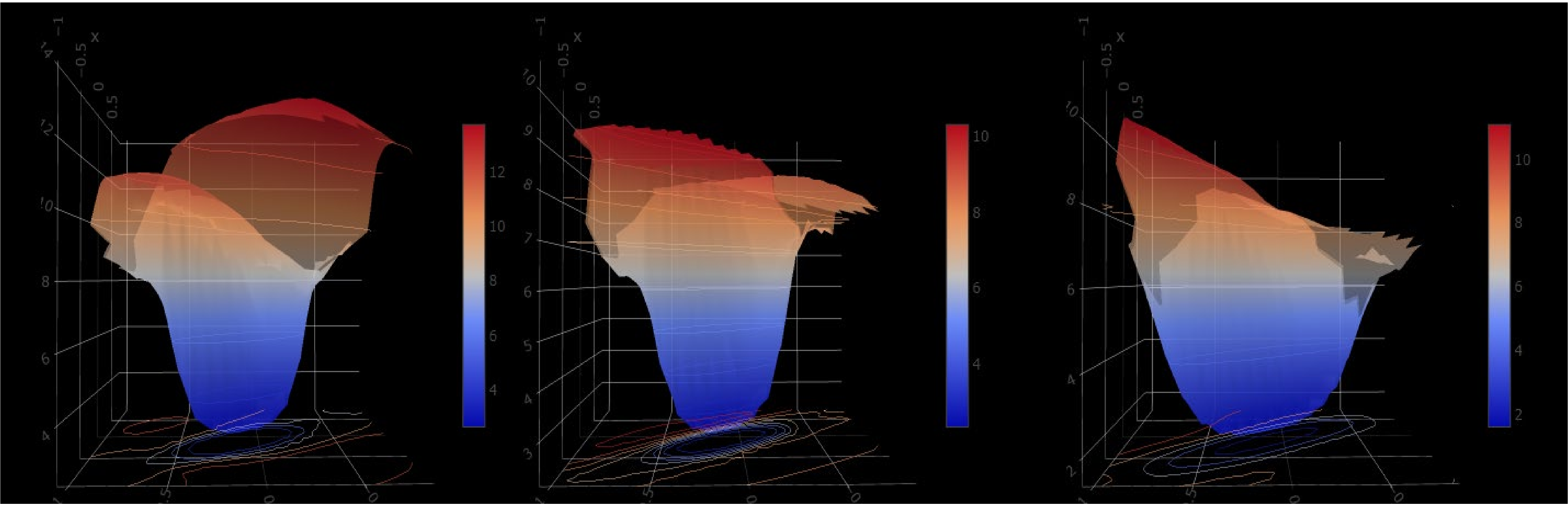
Landscape smoothened. 80% pruned one has deeper & wider minima.

POSTECH

Exp 3 : Visualize Loss Landscape of Pruned Transformer

Result

- Compare WLTs with baseline, Local View



Baseline

30% pruned WLT

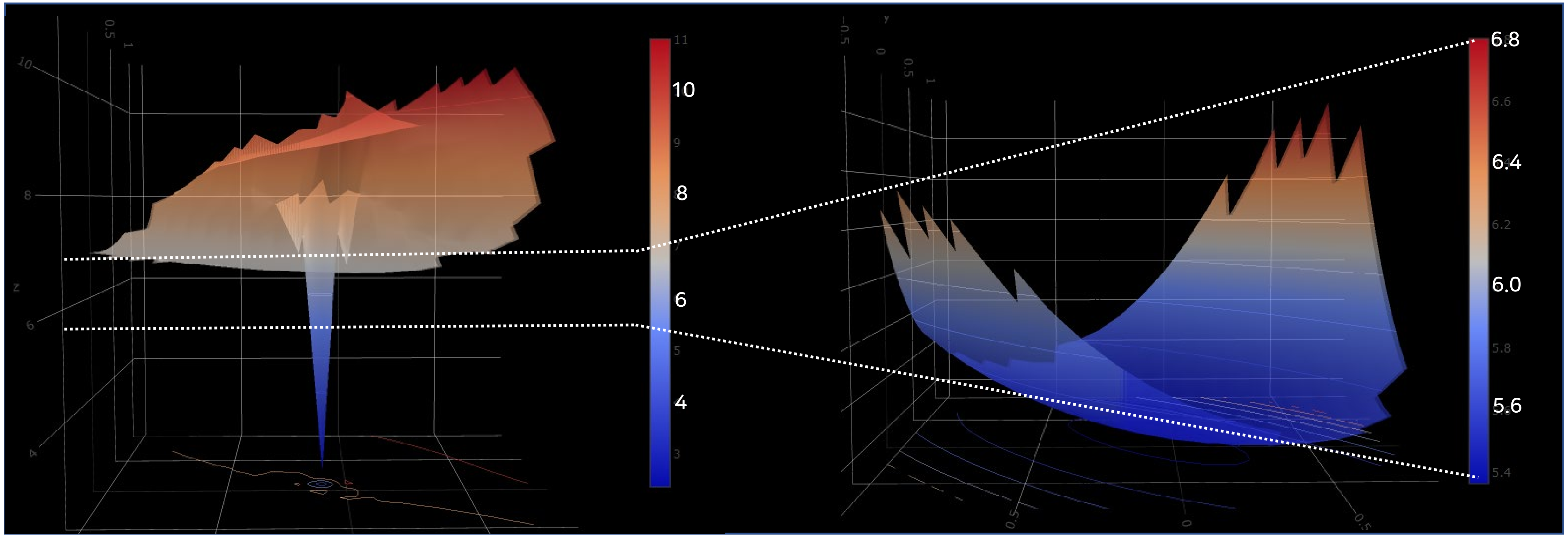
80% pruned WLT

The more the model is pruned, the wider local minimum becomes.

Exp 3 : Visualize Loss Landscape of Pruned Transformer

Result

- Compare WLTs with init param, Global View



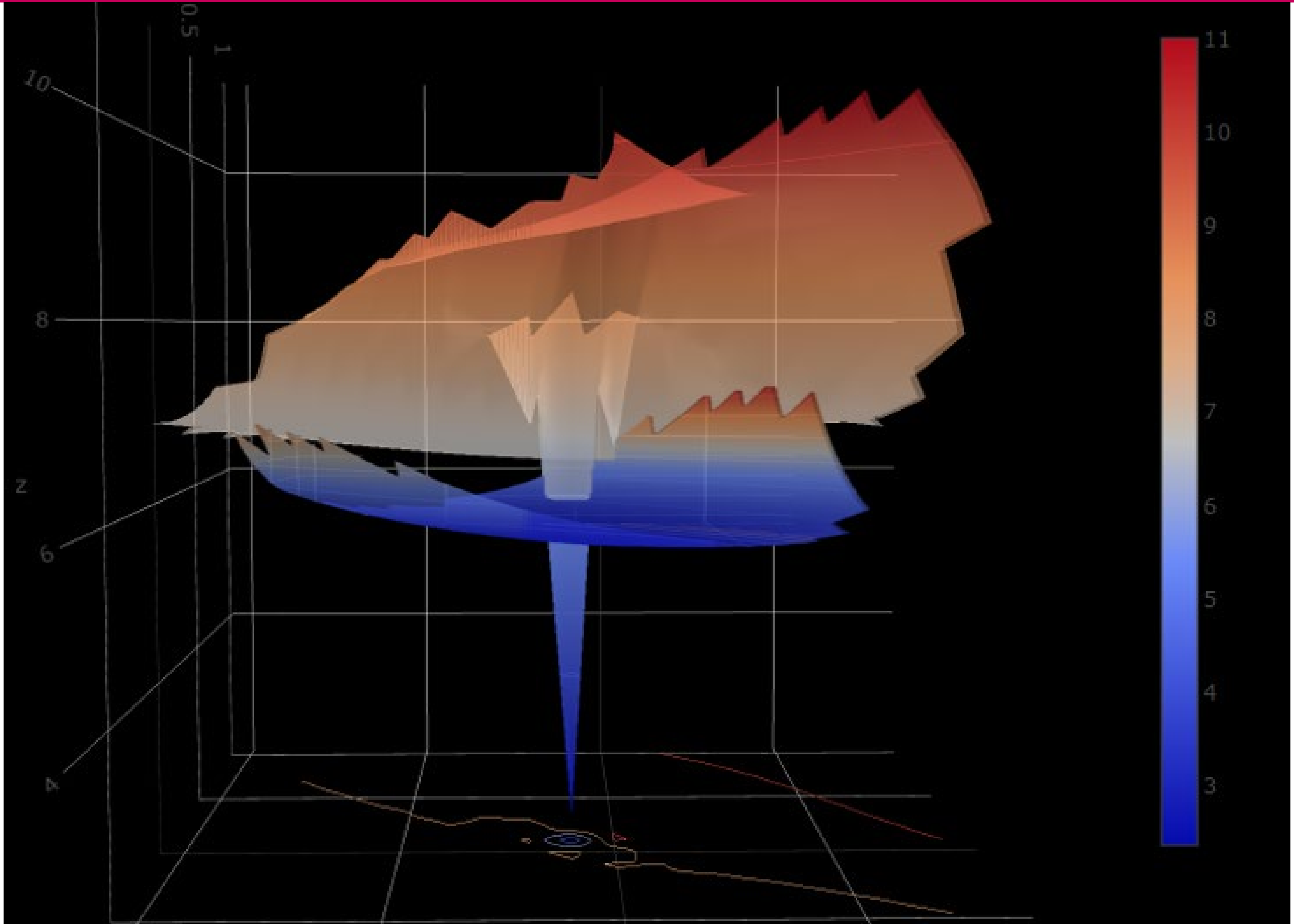
30% pruned WLT

30% pruned random init

For random ticket, loss range become much lesser, and become very flat

Exp 3 : Visualize Loss Landscape of Pruned Transformer

Result



Conclusion

I. Optimization for Attention

II. Visualizing Epoch-wise Landscape of Transformer

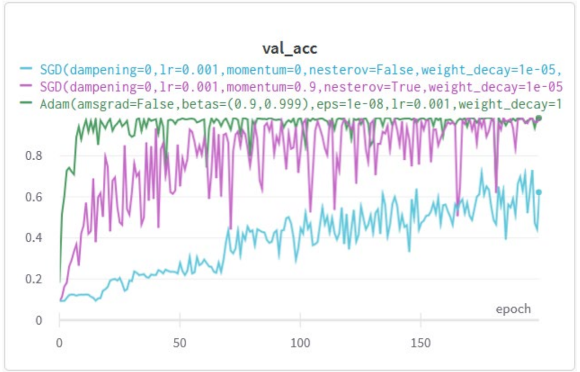
III. Visualize Loss Landscape of Pruned Transformer

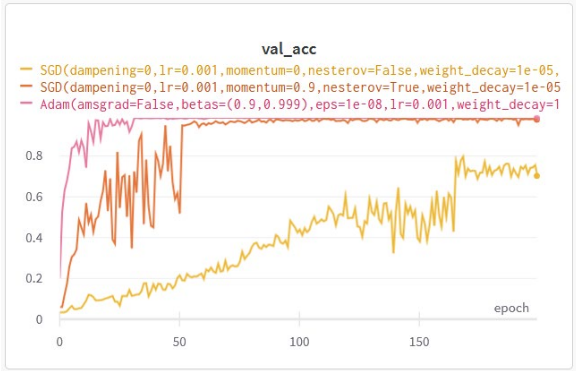
IV. Discussion

Exp 1 : Optimization for Attention

Experiment summary

Convergence rate: Adam = Adam+RedLR > NAG+RedLR > NAG ≫ SGD+RedLR > SGD
SGD showed a linear loss decrease, while Adam and NAG showed a much faster loss decrease.
RedLR scheduler helped smooth graphs and slightly improved SGD's validation accuracy.





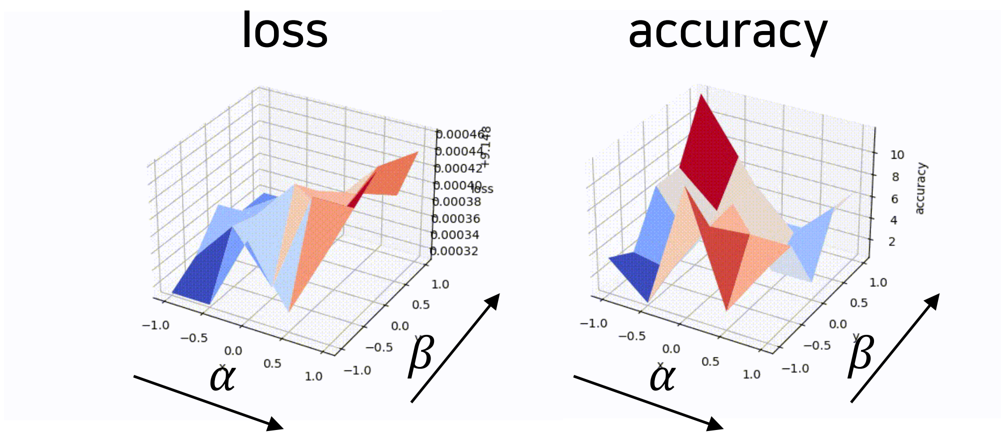
Conclusion

We've confirmed that attention mechanism well follows the theoretical convergence curve seen in lectures, such as $\epsilon \sim \mathcal{O}\left(\frac{1}{T}\right)$ and $\epsilon \sim \mathcal{O}\left(\frac{1}{T^2}\right)$. Also, RedLR can lead to much stable convergence for attention mechanism-based models.

Exp 2 : Visualizing Epoch-wise Landscape of Transformer

Experiment summary

Points with higher α, the direction that points the optimal point, showed much better accuracy and lower loss compared to other points, even though their initial loss were larger than others. Likewise, points with lower α showed relatively low accuracy and high loss. However, some points like (-1, 0), (-0.5, 1) or (0, 1) showed higher performance.



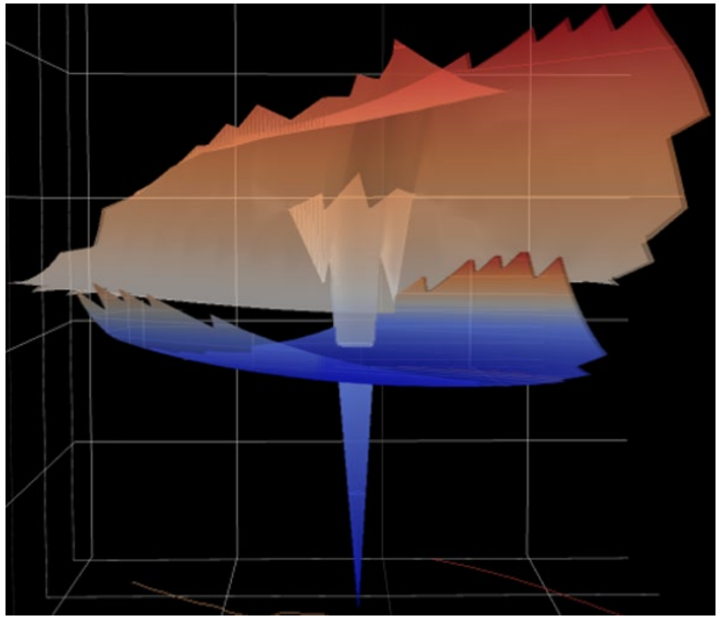
Conclusion

We've seen that appropriate initialization point is crucial to final performance. Although initial points closer to optimal point scored better accuracy, whether an initial point is closer to the optimal point is not necessary for scoring high accuracy. And unfortunately, this initialization point with 10% pruning wasn't a winning ticket.
Also, there is no significant difference in the rate of change of metrics for each point.

Exp 3 : Visualize Loss Landscape of Pruned Transformer

Conclusion

- We found that finding WLTs of transformer is quite easy, with large performance gap (51%→70%)
  → Implies **Transformer is over-parameterized**, many connections are unnecessary
- Empirical results show that **WLT method help smooth** Loss Landscape a little bit, with deeper minimum point



- Below argument of *Li et al[1]* confirmed empirically

  " Training a pruned model from scratch performs worse than retraining a pruned model, which may indicate the difficulty of training a network with a small capacity."

→ Very flat loss surface of "30% pruned w/ random" prevent training, by providing near 0 curvature

→ If the point move far away from the optima, loss value does not increase significantly *(Even less than WLT)*

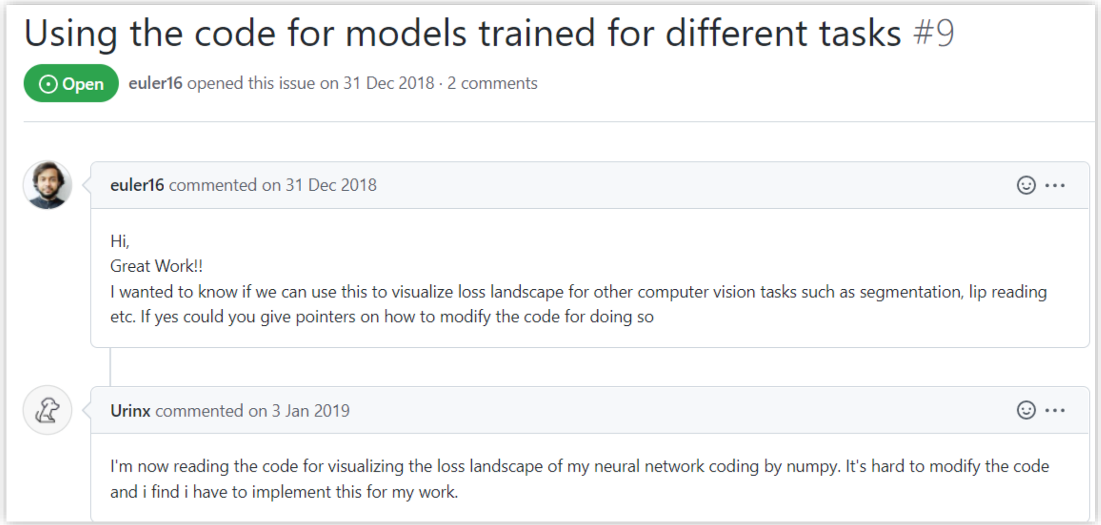
→ Implies **the expressiveness of the model is insensitive to parameter change : can be explained by small capacity**

Discussion

- For attention block, optimizers seen in lectures work as intended even if applied to a complex function with a randomly generated dataset.
- Choosing nice initial parameter is important for final performance. However, even if started on another initial point near the original one, the model may get near optimal performance like WLTs. This means there are some group of initial parameters that acts like a winning ticket.
- 80% pruned winning ticket of Transformer performed better than the original one and had better loss landscape, while randomly initialized one didn't. It is because randomly initialized model has flat loss landscape compared to winning tickets.
- If we had much time and computational resource, we could have additional experiments like finding optimal prune rate, broader initial parameter landscape, loss landscape of a single attention head w/o residual connection, etc.

Additionally…

■ Official code for "visualizing loss landscape of neural nets" is very difficult to use!



Using the code for models trained for different tasks #9

Open euler16 opened this issue on 31 Dec 2018 · 2 comments

euler16 commented on 31 Dec 2018

Hi,
Great Work!!
I wanted to know if we can use this to visualize loss landscape for other computer vision tasks such as segmentation, lip reading etc. If yes could you give pointers on how to modify the code for doing so

Urinx commented on 3 Jan 2019

I'm now reading the code for visualizing the loss landscape of my neural network coding by numpy. It's hard to modify the code and i find i have to implement this for my work.

1. It requires additional viewer program
2. Its dataset loader, model loader is hard corded.
3. Code are distributed messy. You should revise a lot part of code for your own dataset and model

→ But for our visualization code is simply applicable for any pytorch model which provide

model.state_dict() **and** **model.load()**

→ We hope you use our code if you want to visualize your own model

Thank you for listening ☺

Our code is available at https://github.com/DelVel/CSED490Y

Taegyu Park, Byeongju Woo

POSTECH