

[W&B Fully Connected](#) > [Articles](#)

An Evaluation of Transformer Variants

Training of different transformer variants for text-to-image generation with DALL-E-mini

[Boris Dayma](#)

Last Updated: Nov 15, 2023

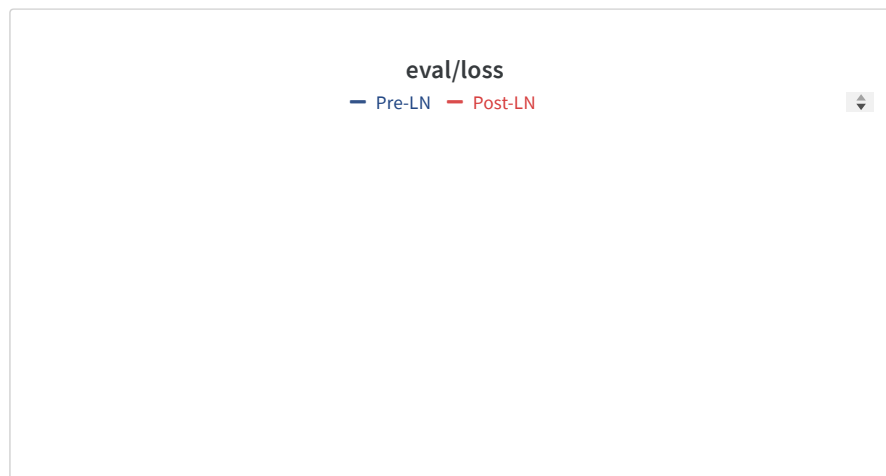
► [Table of contents \(click to expand\)](#)

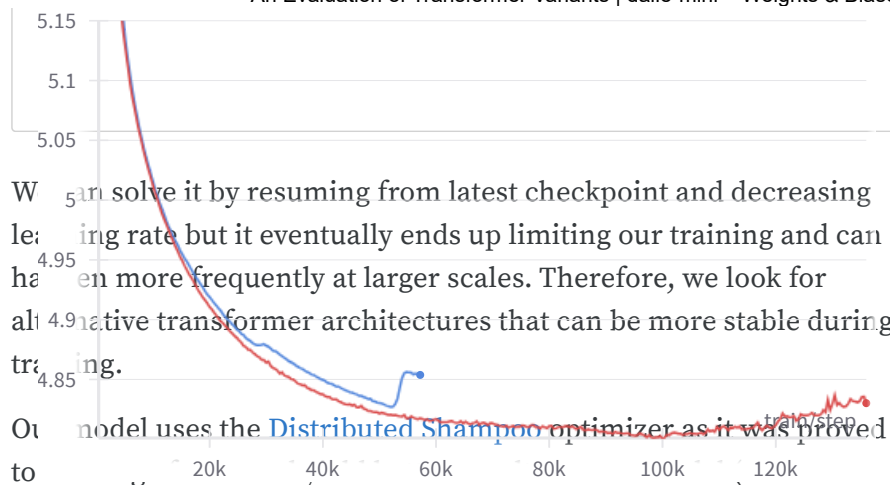
▼ Introduction

Our goal today is simple: we're going to train and compare a few transformer variants for text to image generation with [DALL-E-mini](#).

If you'd like to skip ahead and see what we learned, [jump to the summary before digging in](#).

When training a standard BART model, we observe instability where the evaluation loss can suddenly spike while on the first epoch, whether we use "Pre-LayerNorm" or "Post-LayerNorm" architecture.





We can solve it by resuming from latest checkpoint and decreasing learning rate but it eventually ends up limiting our training and can happen more frequently at larger scales. Therefore, we look for alternative transformer architectures that can be more stable during training.

Our model uses the [Distributed Shampoo](#) optimizer as it was proved to [Evaluation of Distributed Shampoo](#)).

▼ Experimental Setup

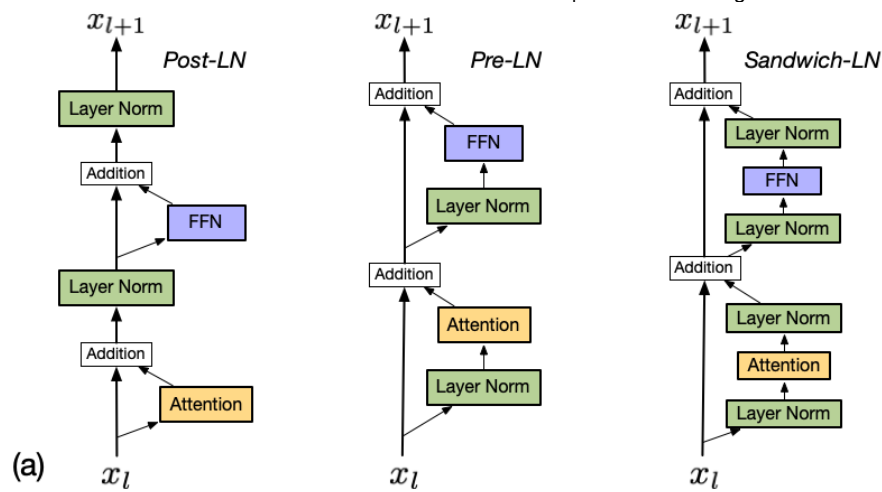
All the transformer architectures are implemented as variants of [BART](#). The details:

- **1 model per device:** each model is about 400M parameters
- **batch size:** max until OOM
- **gradient accumulation steps:** 2 or 3 depending on model so we have at least 800 samples per update
- **learning rate:** constant with 2000 warmup steps
- **loss:** is the cross-entropy on image tokens encoded by the [VQGAN](#)
- **hardware:** TPU v3-8

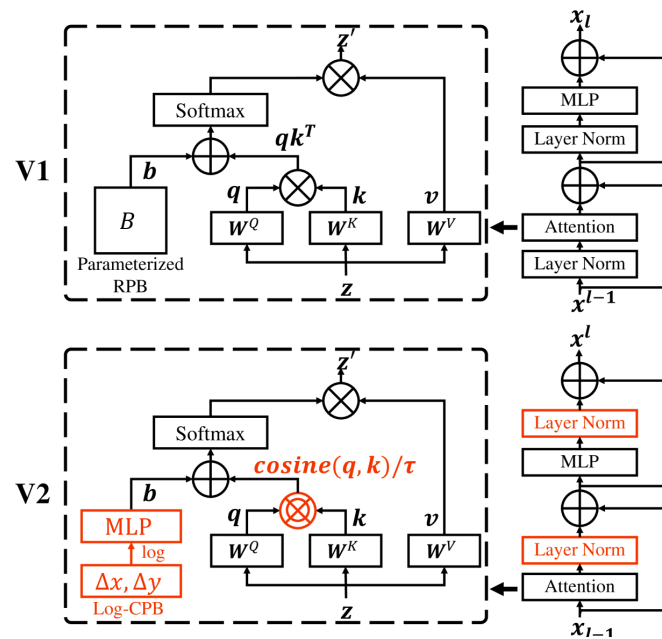
▼ Transformer Variants

The different variants we're looking at today implement a combination of:

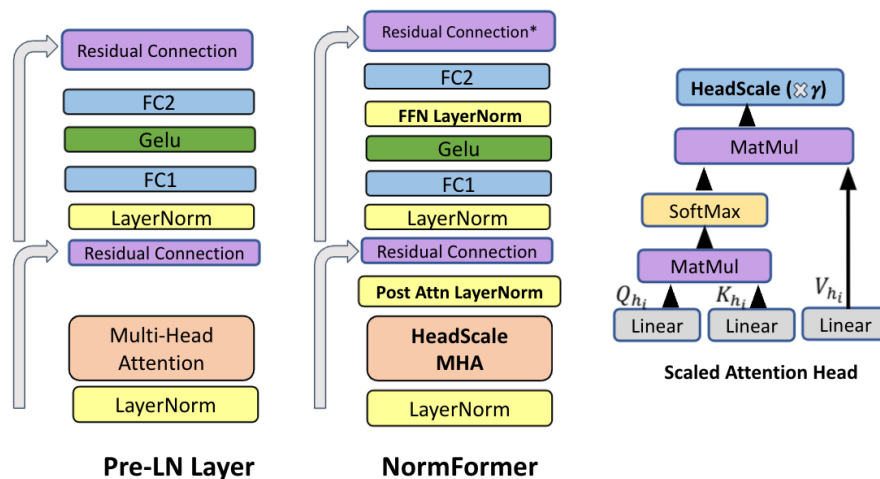
- **Post-LN:** standard BART model with post-LN (like original transformer)
- **Pre-LN:** the LayerNorms are placed as pre-LN
- **Sandwich-LN per [CogView](#):** pre-LN + LN at the end of each block (FFN + Attention)



- **Swin Transformer v2:** post-LN in the non-residual branch + cosine attention with tau + relative position embeddings (the ones from Swin v1 as we don't need continuous ones)



- **NormFormer:** pre-LN + LN after attention + LN after activations (middle of FFN block) + head scale (no residual scaling as the paper mentions it does not work on large models)



- **DeepNet**: post-LN (like baseline) + scaling of weights init + scaling of residual connections

Architectures	Encoder		Decoder	
	α	β	α	β
Encoder-only (e.g., BERT)	$(2N)^{\frac{1}{6}}$	$(8N)^{-\frac{1}{6}}$	-	-
Decoder-only (e.g., GPT)	-	-	$(2M)^{\frac{1}{6}}$	$(8M)^{-\frac{1}{6}}$
Encoder-decoder (e.g., NMT, T5)	$0.81(N^4M)^{-\frac{1}{6}}$	$0.87(N^4M)^{-\frac{1}{6}}$	$(3M)^{\frac{1}{6}}$	$(12M)^{-\frac{1}{6}}$

- **GLU** variant instead of standard FFN (about 2/3 of params per dense layer to have same total number of params)

$$\begin{aligned}\text{FFN}_{\text{GLU}}(x, W, V, W_2) &= (\sigma(xW) \otimes xV)W_2 \\ \text{FFN}_{\text{Bilinear}}(x, W, V, W_2) &= (xW \otimes xV)W_2 \\ \text{FFN}_{\text{ReLU}}(x, W, V, W_2) &= (\max(0, xW) \otimes xV)W_2 \\ \text{FFN}_{\text{GELU}}(x, W, V, W_2) &= (\text{GELU}(xW) \otimes xV)W_2 \\ \text{FFN}_{\text{SwiGLU}}(x, W, V, W_2) &= (\text{Swish}_1(xW) \otimes xV)W_2\end{aligned}$$

- RMSNorm instead of standard LayerNorm

$$\bar{a}_i = \frac{a_i}{\text{RMS}(\mathbf{a})} g_i, \quad \text{where } \text{RMS}(\mathbf{a}) = \sqrt{\frac{1}{n} \sum_{i=1}^n a_i^2}.$$

- Activation functions considered are **GeLU**, **Swish** and **SmeLU**

▼ Results

▼ Summary

TL;DR with links to relevant section:

- Final LN in decoder is mandatory. **Final LN in encoder is beneficial.**
- **Do not use bias in dense layers.**
- **DeepNet improves slightly stability vs vanilla Post-LN.**
- **NormFormer is more stable than Sandwich-LN.**
- **The head scale and learnt residual connections in NormFormer are not needed.**
- **We don't need to use a scale in LayerNorm when followed by dense layers.**
- **GLU variants are always beneficial.**
- **The best GLU variants are with GeLU and Swish. Swish is faster but GeLU is more stable.**
- **NormFormer and Swin v2 are the best variants.**
- **RMSNorm is more stable than LayerNorm but on very long runs, it plateau's earlier.**
- **Swin relative positions are not better than absolute position encodings**

- SinkFormers can only be used in the encoder due to causal attention but don't improve performance

These results are specifically applicable to the DalleBart model from DALL-E-mini and use distributed shampoo optimizer which already improves considerably model stability.

Note: I don't think there's a magic recipe that works all the time. I'm mainly reporting what worked in my specific case.



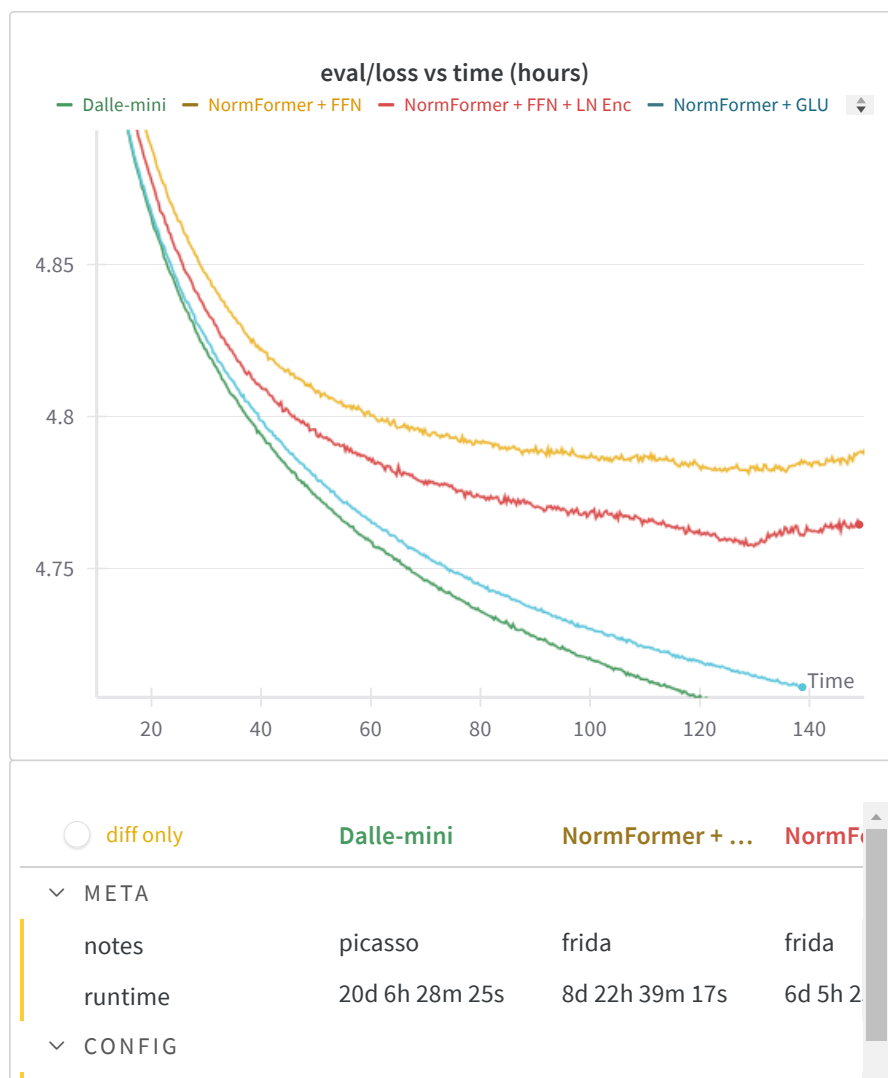
Please comment in this report, raise an issue or submit a PR if you notice any bugs!

▼ Detailed Comparisons

▼ Final LN

In Pre-LN type of architectures (all except Post-LN & DeepNet), the model will not converge unless there is a final LayerNorm in the decoder.

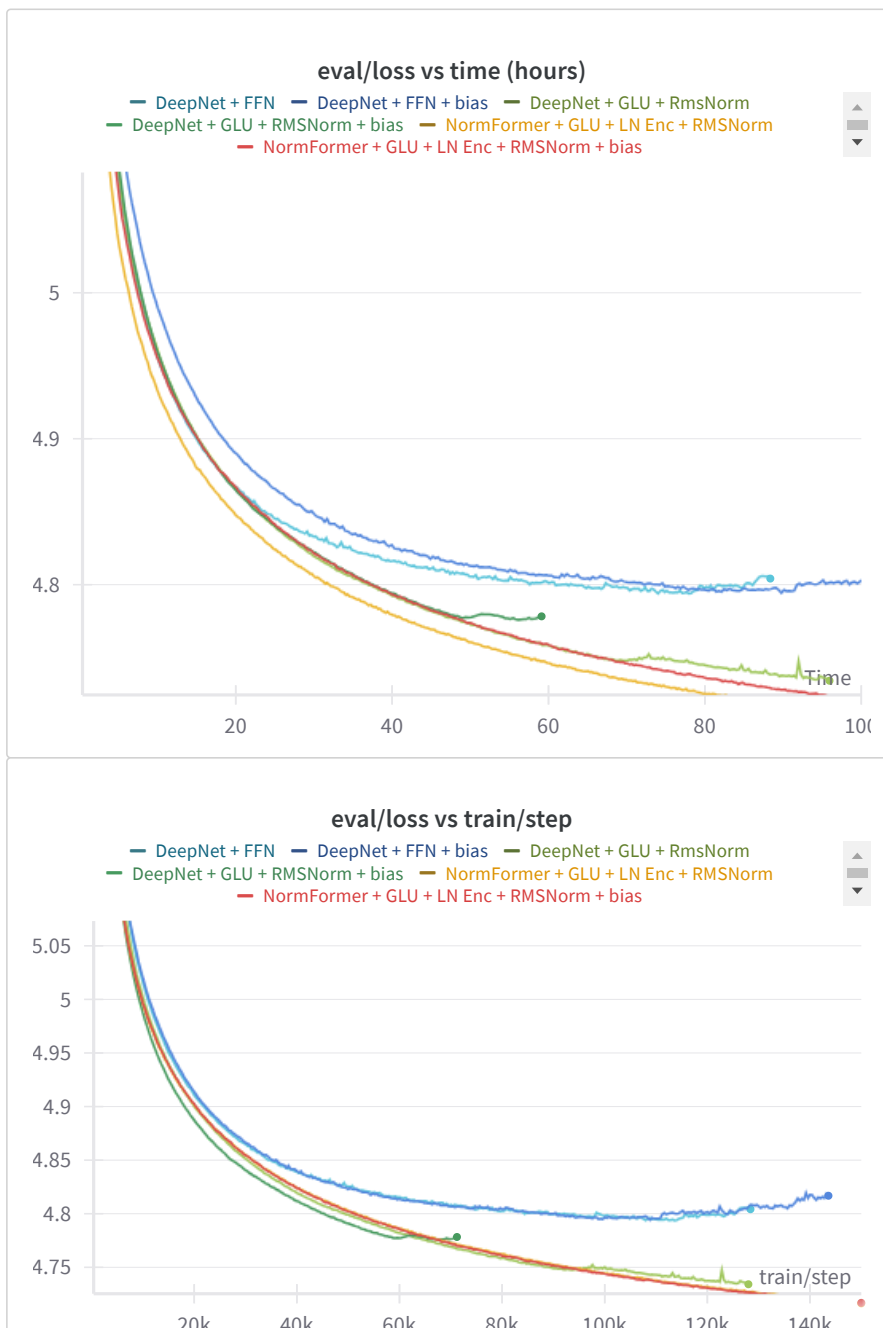
Using a final LayerNorm in the encoder also helps convergence.



_wandb	{ "desc": null, "val... { "desc": null, "val... { "desc": null, "val...		
batch_size_per_step	816	832	832
config_name	config/mini_glu	config/mini	config/
gradient_accumula...	3	2	2
model_config			
decoder_ffn_dim	2730	4096	4096
encoder_ffn_dim	2730	4096	4096
use_absolute_po...	-	true	-

▼ Bias in Dense layers

Using bias in dense layers adds 15% of training time per step and hurts convergence.

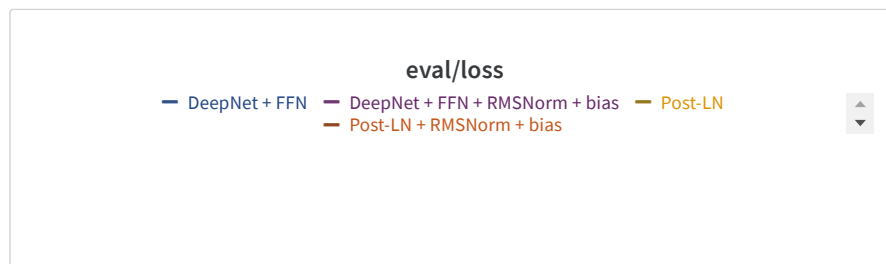


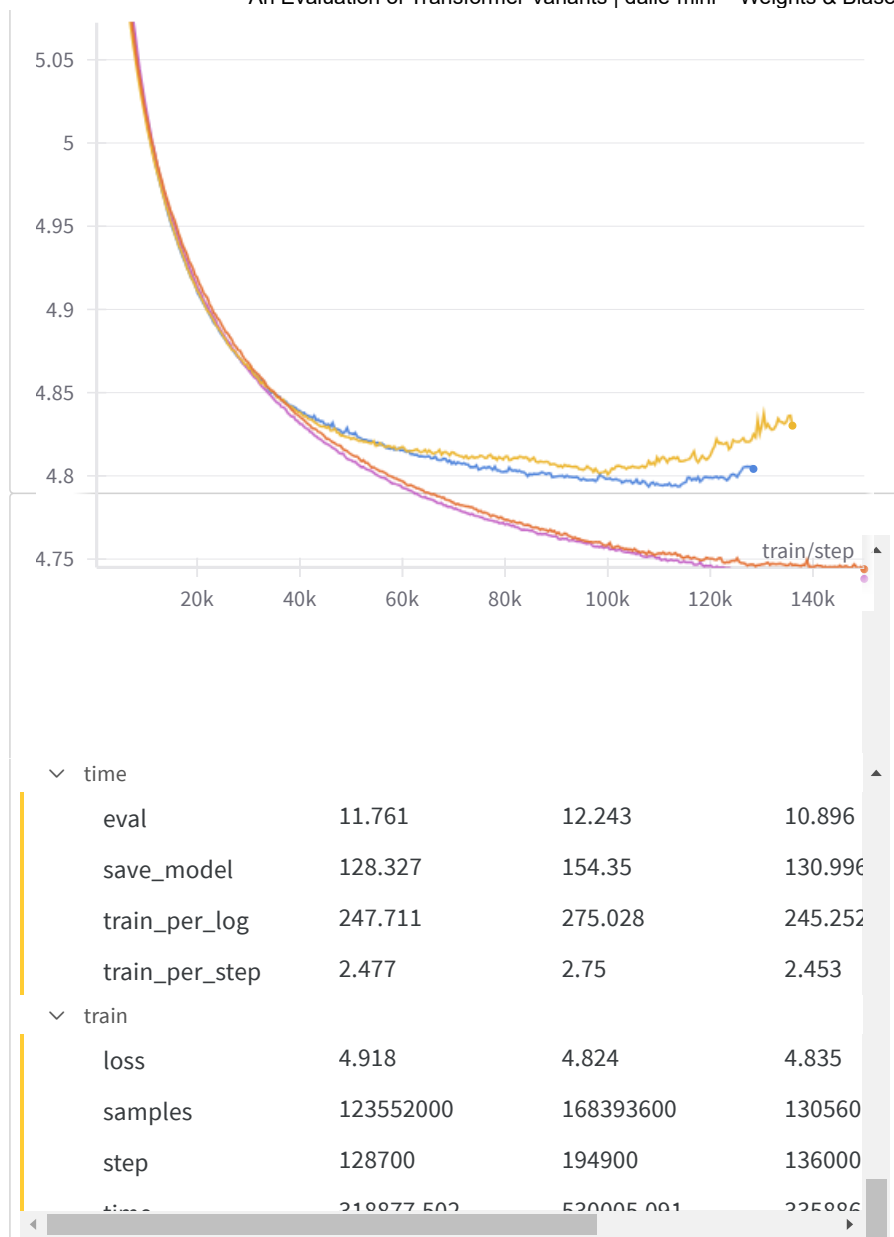


▼ DeepNet vs Post-LN

DeepNet just improves slightly stability vs vanilla Post-LN by adding a scaling factor at weights initialization and in residual branches.

If you use Post-LN, you may as well implement it.

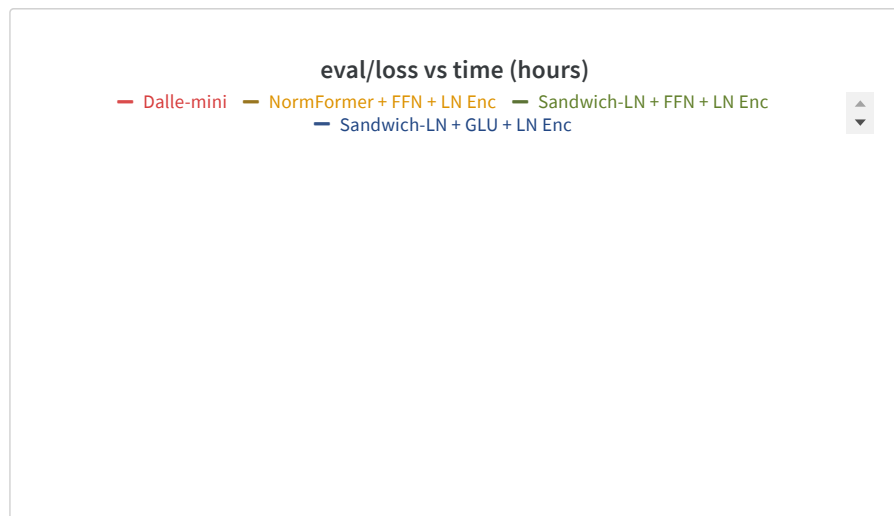


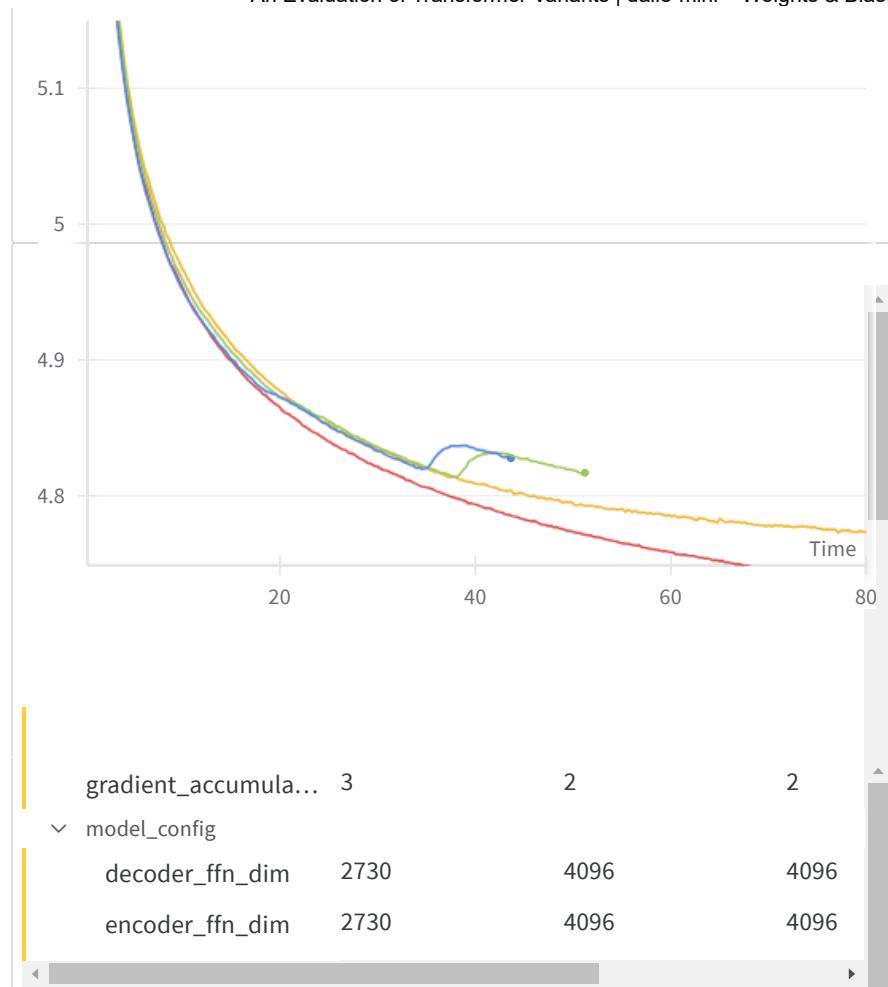


▼ NormFormer vs Sandwich-LN

NormFormer is more stable than Sandwich-LN, which has same

LayerNorm positions except in the MLP block





▼ NormFormer variants

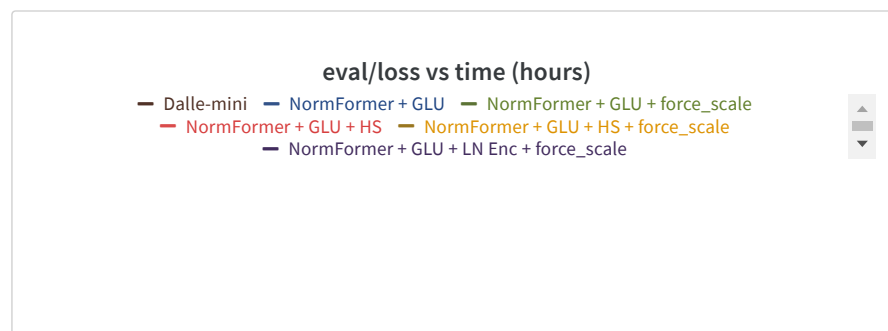
NormFormer head scale hurts convergence, probably because it is followed with a LayerNorm. It also slows down training.

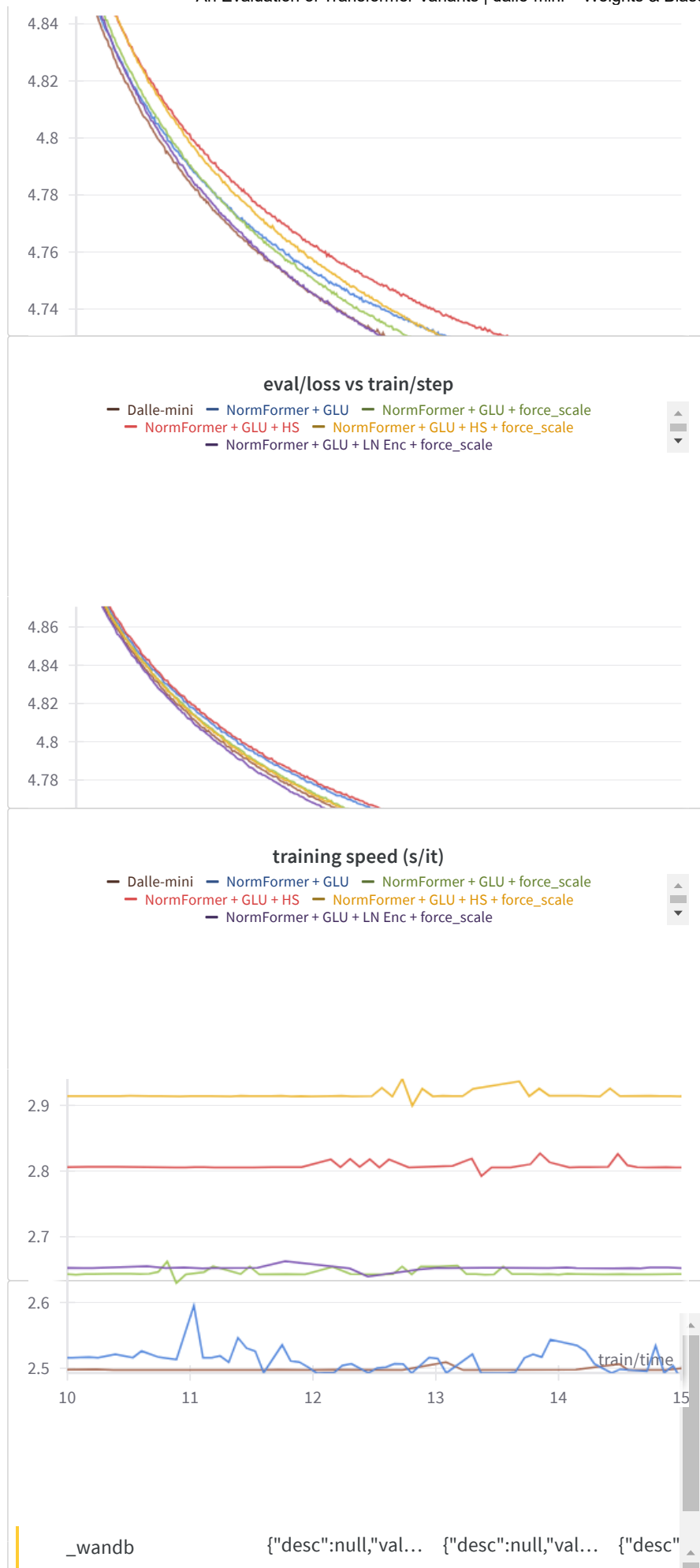
The learnt residual connections have not been used as the paper mentions it is not beneficial at larger scale.

We also present a `force_scale` variant which uses the LayerNorm learnt scale even when directly followed by dense layers. It can

lower learning rate on parameter updates: $(\text{learning_rate} * \text{scale}_1) * (\text{learning_rate} * \text{scale}_2)$. Overall it is not recommended to use it

as we should get the same improvements through the learning rate schedule and it adds about 5% of training time per step.

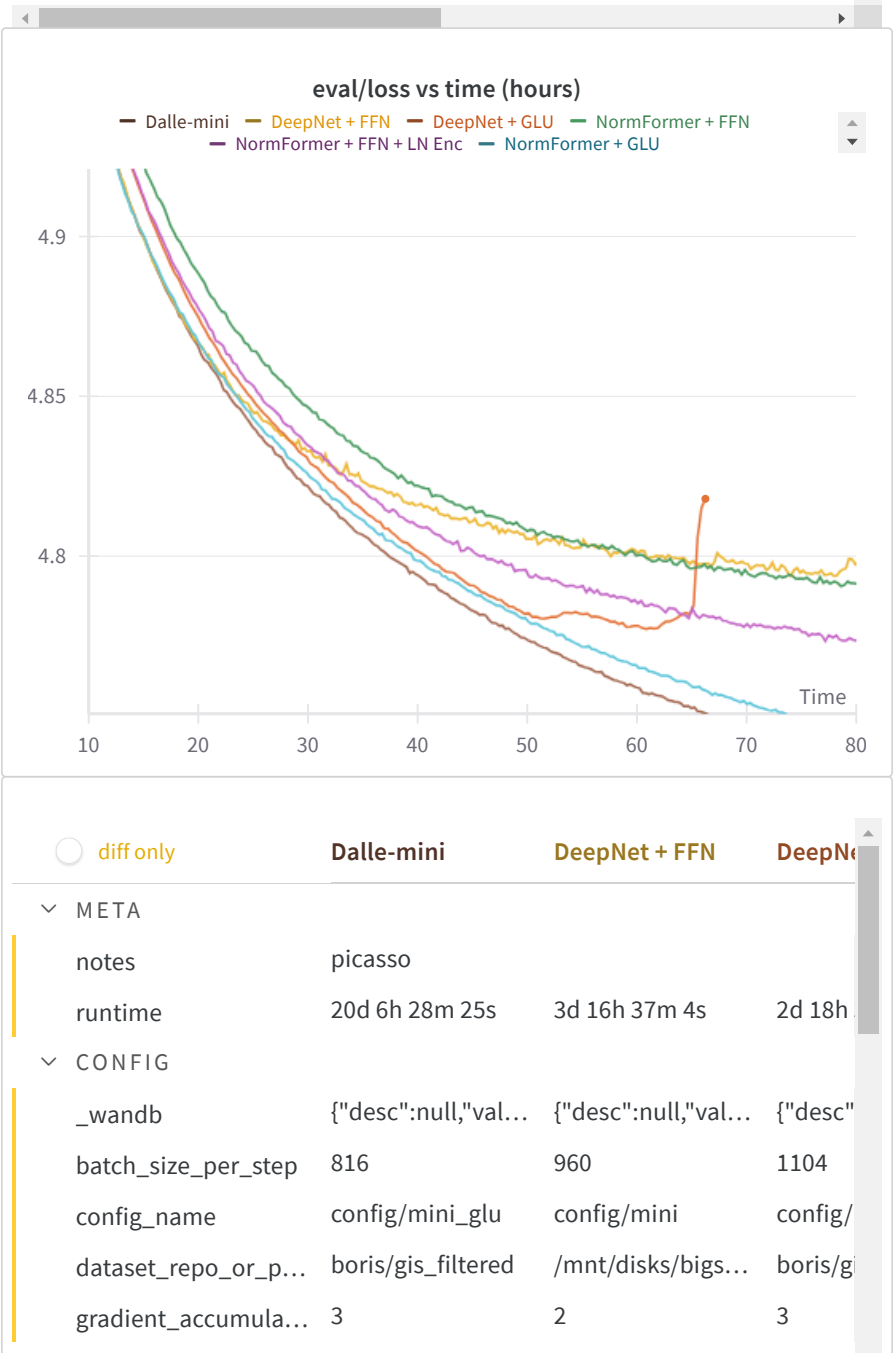




dataset_repo_or_p...	boris/gis_filtered	boris/gis_filtered	/mnt/di
model_config			
force_ln_scale	false	false	true
use_absolute_po...	-	-	-
use_bias	-	-	-
use_final_ln_enc...	true	false	false
use_head_scale	false	false	false

GLU variants

GLU variants are always beneficial even if they require extra memory for the same amount of parameters (using 2/3 per dense layer vs FFN) and therefore a lower batch size.

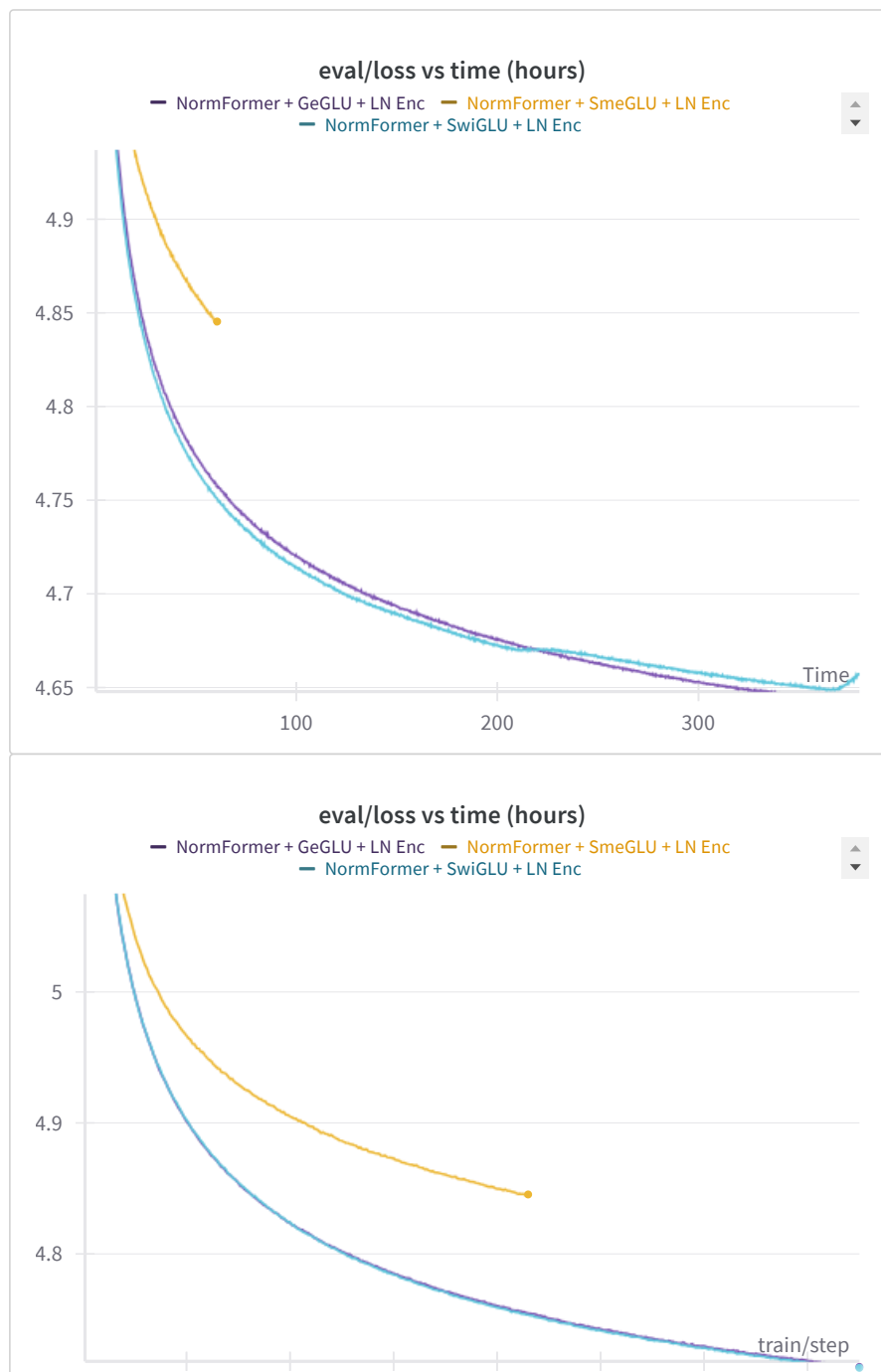


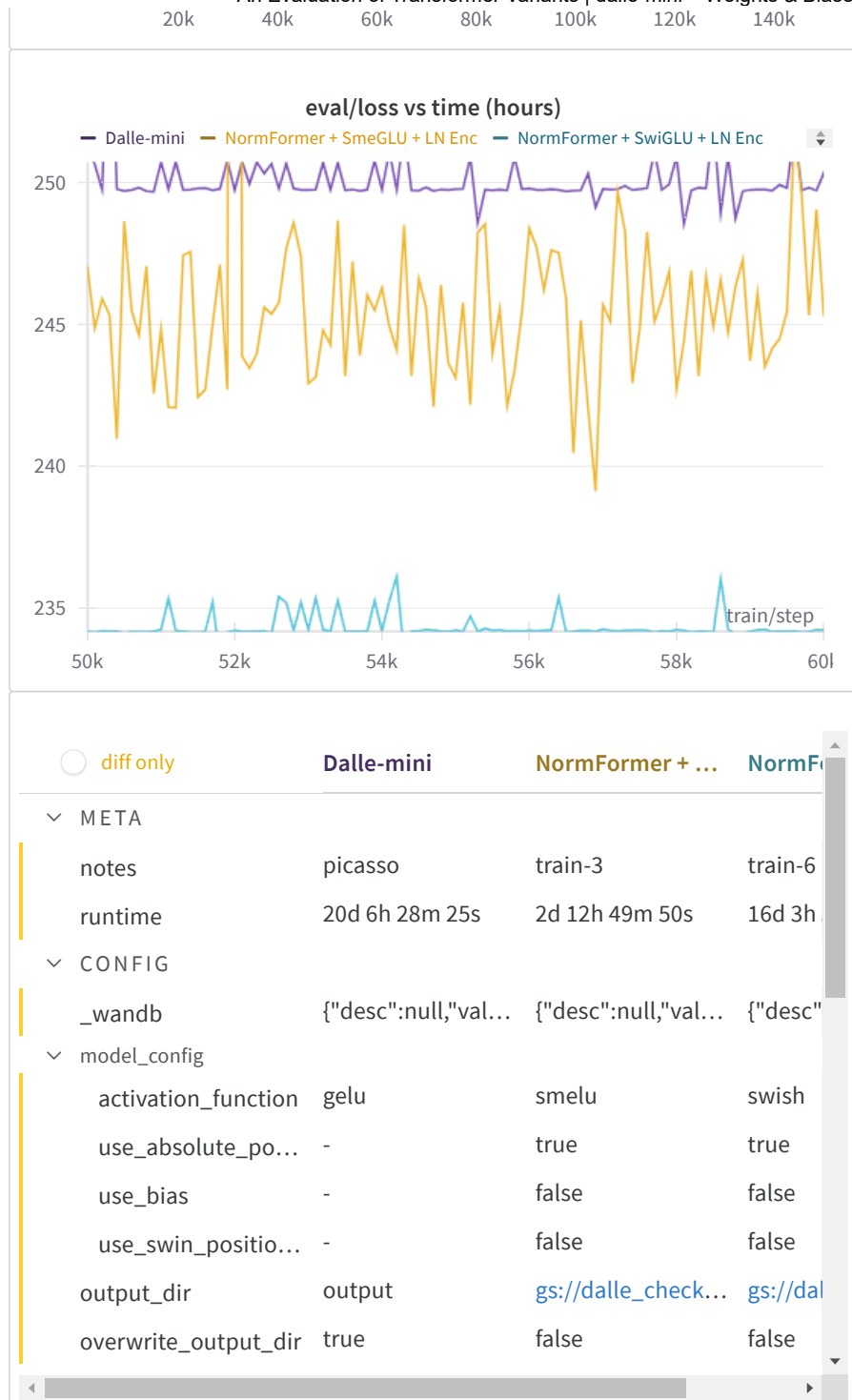
model_config			
decoder_ffn_dim	2730	4096	2730
encoder_ffn_dim	2730	4096	2730

▼ GLU activation functions

We use different activation functions in combination with GLU variants: GeLU, Swish and SmeLU.

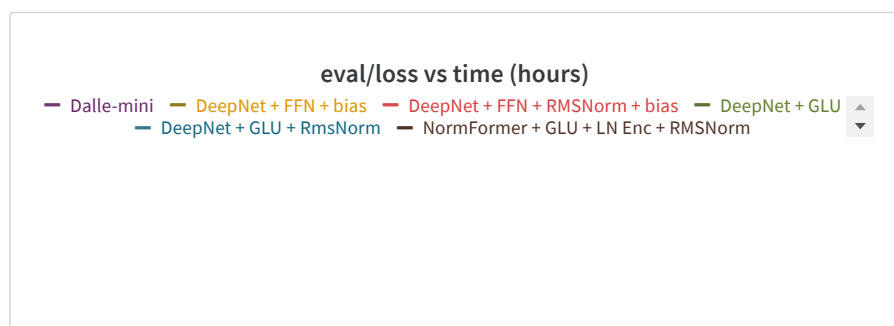
GeGLU (with GeLU) and SwiGLU (with Swish) perform the best. SwiGLU is slightly better up to a certain point because it is faster to compute but eventually GeGLU seems more stable.

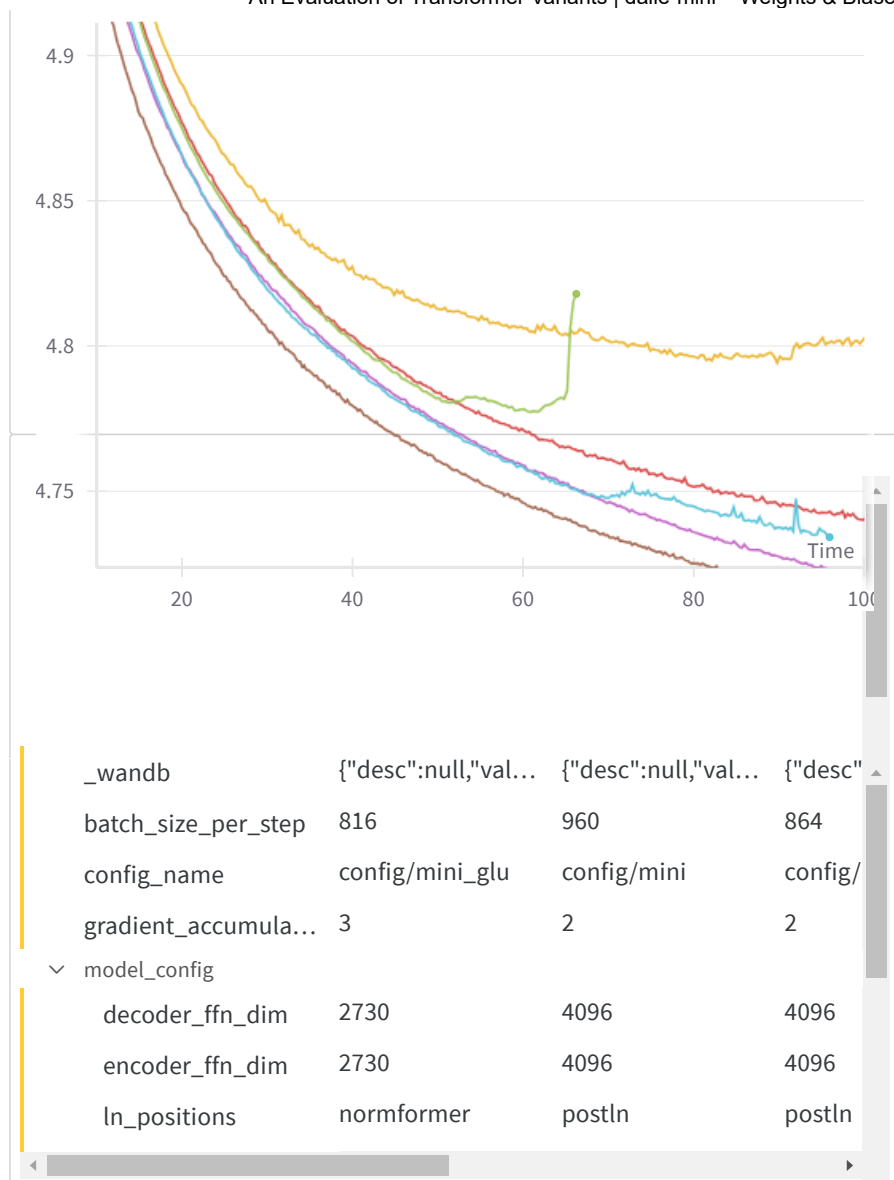




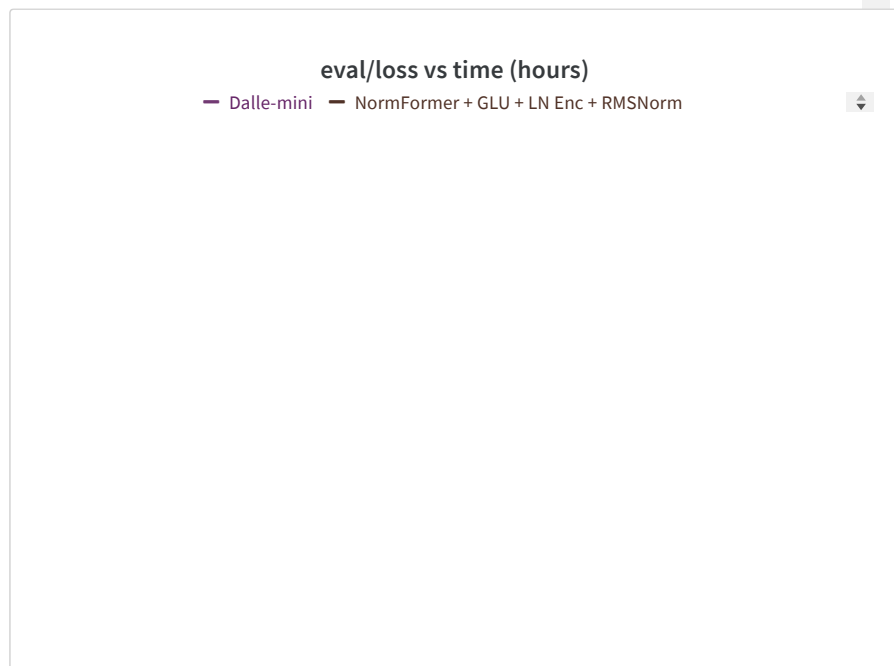
▼ RMSNorm

RMSNorm is better than LayerNorm for a very long time.



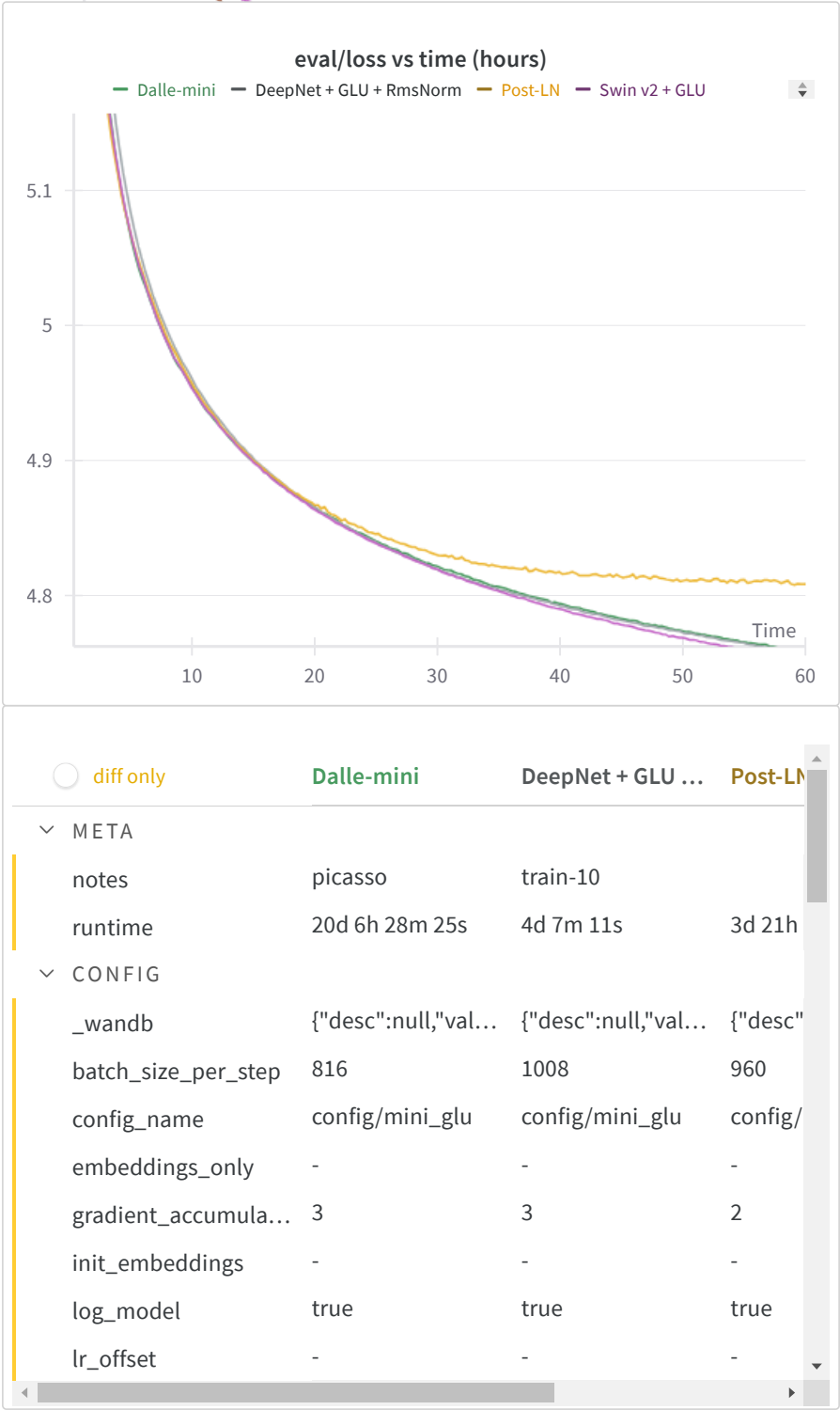


However after a long time on the best runs, RMSNorm plateau's before LayerNorm.



▼ Swin v2

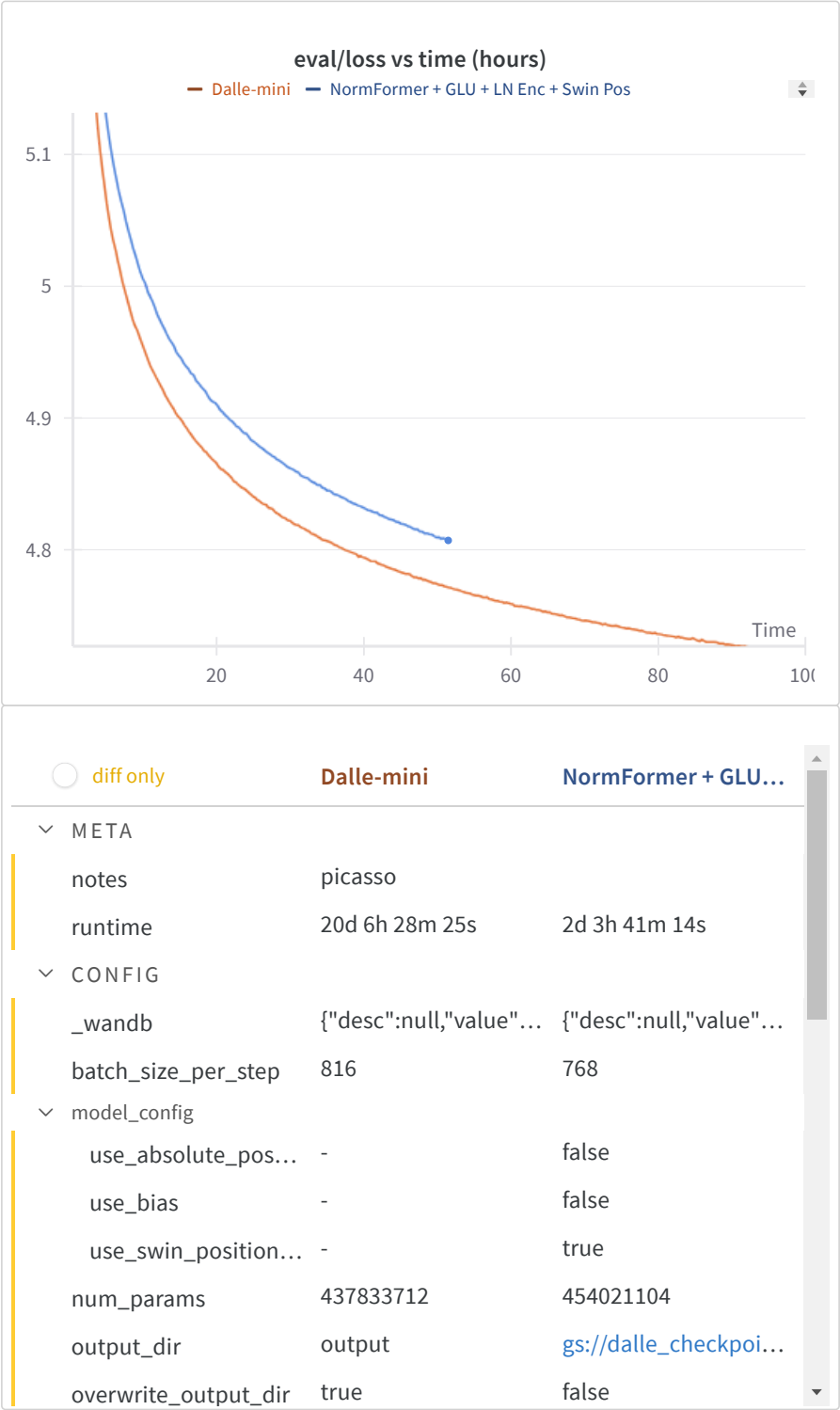
Swin v2 trains as well as NormFormer. We used a fixed scale tau in the test.



▼ Swin Relative Positions

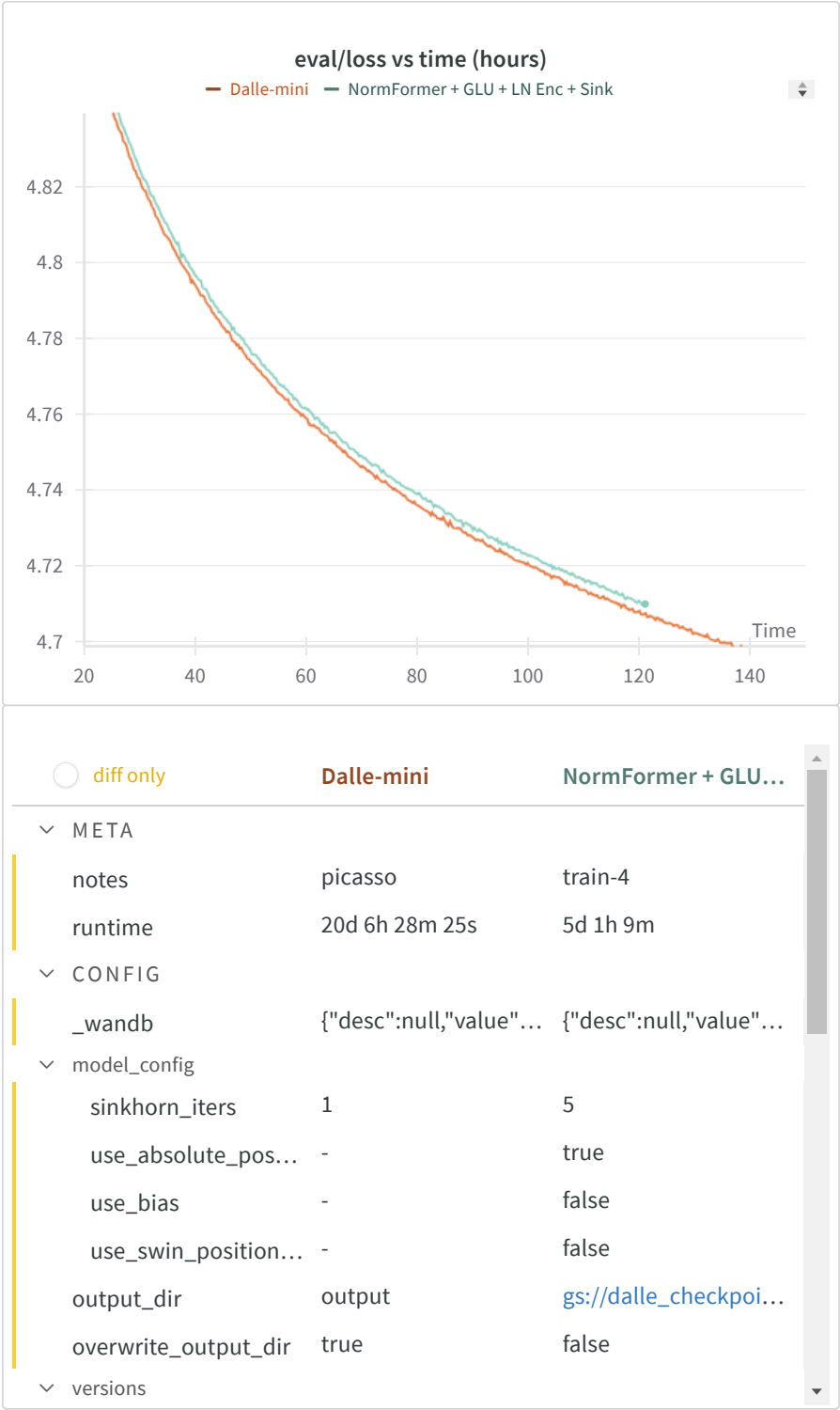
We encode relative positions as learnt parameters in attention layers per Swin Transformers and compare it with regular absolute position embeddings.

Swin relative positions are slower to train.



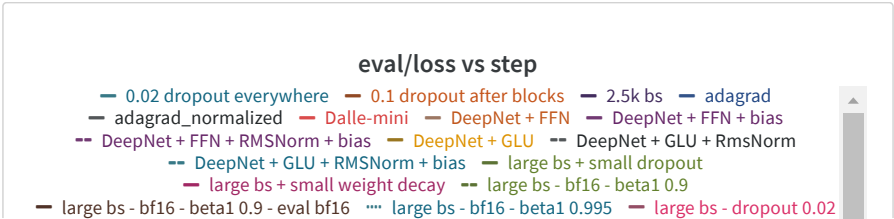
▼ SinkFormers

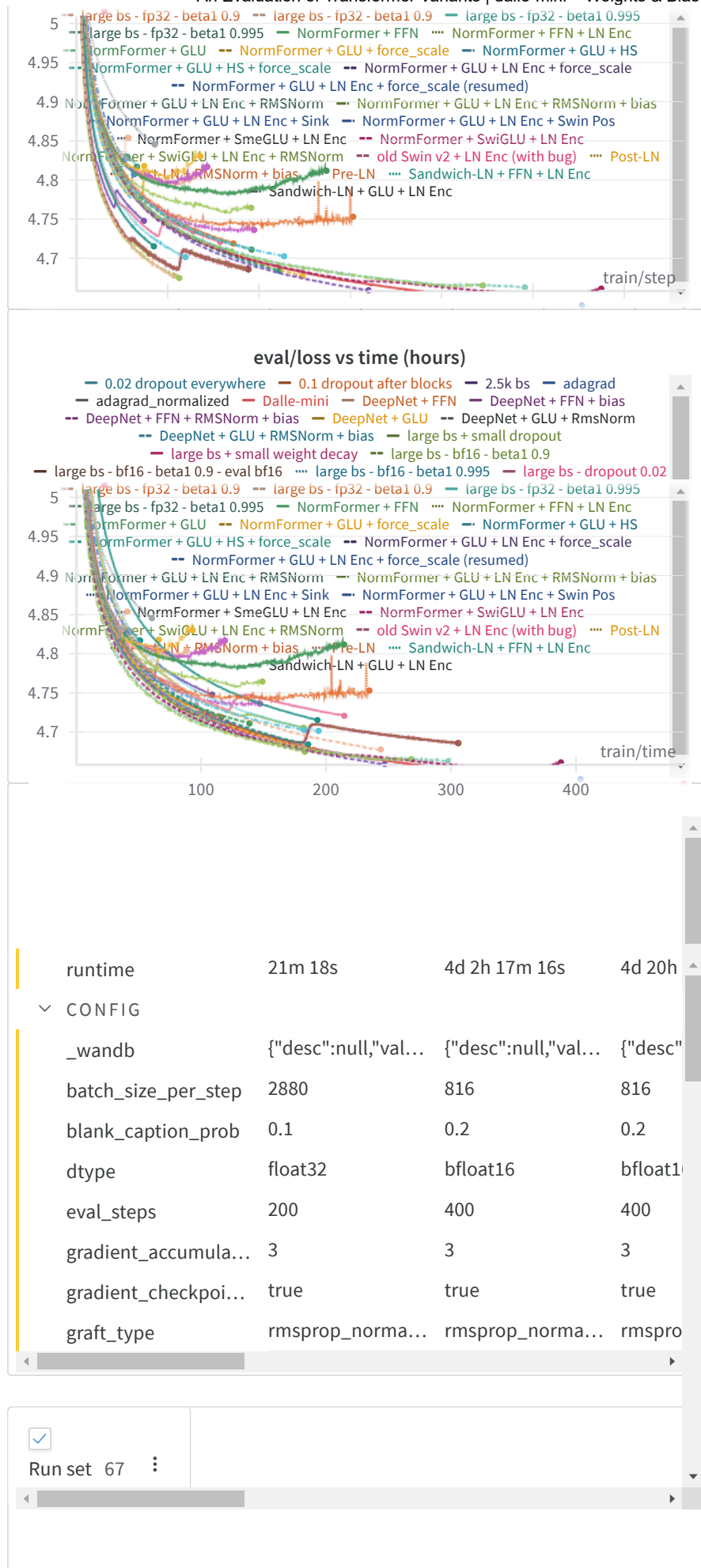
We can only use the SinkFormer in the encoder but not in the decoder due to causal attention however it does not help the model.



▼ All experiments

💡 Select the runs you want to display from bottom table



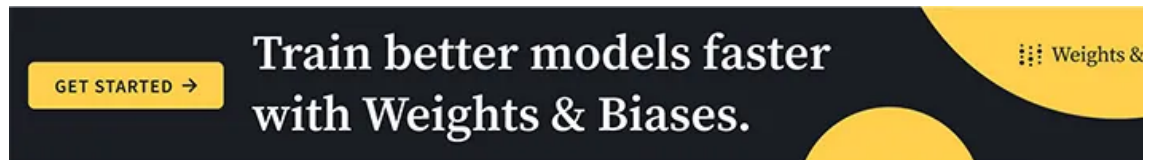


▼ Acknowledgements

- [Rohan Anil](#) for setting up Distributed Shampoo optimizer and continuous feedback

- [Phil Wang](#) who has also implemented many variants in PyTorch and shares his insights through [x-transformers](#)
- Google [TPU Research Cloud \(TRC\) program](#) for providing computing resources and [Pedro Cuenca](#) for running some of these experiments
- [Weights & Biases](#) for providing the infrastructure for experiment tracking and model management
- 🤗 [Hugging Face](#) for the JAX implementation of Bart

Tags: Artifacts, Domain Agnostic, Experiment



Created with ❤️ on Weights & Biases.


<https://wandb.ai/dalle-mini/dalle-mini/reports/An-Evaluation-of-Transformer-Variants--VmldzoxNjk4MTlw>

Made with Weights & Biases. [Sign up](#) or [log in](#) to create reports like this one.

Never lose track of another ML
project. **Try W&B today.**

[SIGN UP](#)

[TRY W&B NOW](#)

 **Weights & Biases**
Get weekly updates with the latest ML news.

[Subscribe](#)

PRODUCTS

[Dashboard](#) [Sweeps](#) [Artifacts](#) [Reports](#) [Tables](#)

QUICKSTART

[Documentation](#)

RESOURCES

[Courses](#) [Forum](#) [Tutorials](#) [Benchmarks](#)

W&B

[About Us](#) [Authors](#) [Contact](#) [Terms of Service](#) [Privacy Policy](#)

Copyright ©2024 Weights & Biases. All rights reserved.