Stronger Normalization-Free Transformers

Supplementary Material

A. Property Analysis Details

In this section, we provided detailed explanation and visualization on how different function properties affect model training.

A.1. Zero-centeredness

We plot the training curves for λ_{horiz} and λ_{vert} with values $\{0, 0.1, 1\}$ in Fig. 6. The trends are consistent with those observed in top-1 accuracy on ImageNet-1K. For horizontal shifts, the training loss with $\lambda_{\text{horiz}} = 0.1$ nearly overlaps with that of $\lambda_{\text{horiz}} = 0$, and even reaches a slightly lower loss. In contrast, vertical shifts exhibit a monotonic pattern: increasing λ_{vert} consistently raises the training loss, suggesting reduced fitting capacity under larger vertical shift.

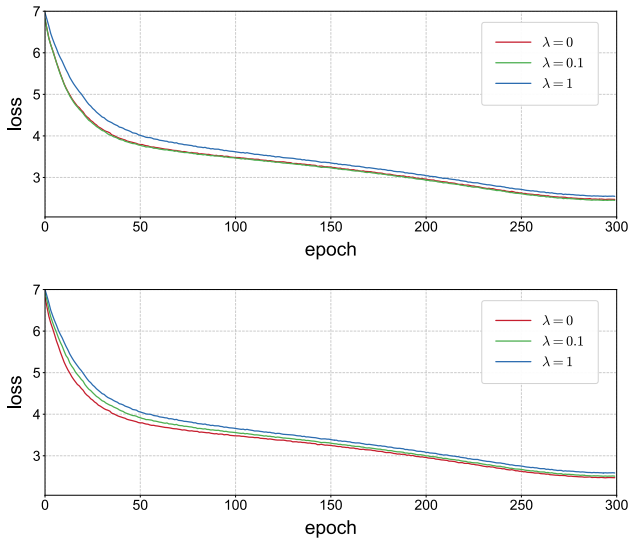

Figure 6. **Training loss curve for horizontal and vertical shifts on the base point-wise function** $\text{erf}(x)$**.** The trends are consistent with the patterns observed in top-1 accuracy on ImageNet-1K.

A.2. Center Sensitivity

We visualize the training losses obtained as λ varies over $\{0, 0.1, 0.5, 1.0, 2.0\}$ on the base point-wise function $\text{erf}(x)$. As shown in Fig. 7, training loss shows a clear monotonic trend: larger λ consistently leads to higher loss, indicating that the width of the flat zone directly limits the model's fitting capacity.

A.3. Monotonicity

We plot the training losses of four functions with distinct monotonicity patterns: the monotonically increasing $\text{erf}(x)$, the monotonically decreasing $\text{negerf}(x)$, the hump-shaped $\text{dampx}(x)$, and the oscillatory $\sin(x)$. As shown in Fig. 8, both increasing and decreasing monotonic functions achieve clearly lower training loss, indicating stronger fitting capacity. In contrast, the non-monotonic functions exhibit higher training loss. This behavior aligns closely with the top-1 accuracy trends observed on ImageNet-1K.

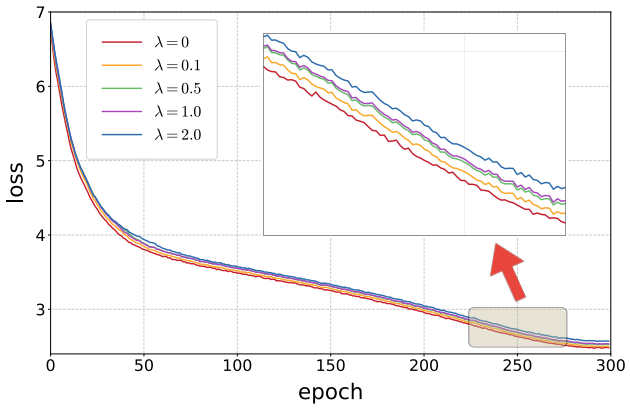

Figure 7. **Training loss curve for different center sensitivity (controlled by** λ**).** A larger λ leads to higher training loss and poorer fitting ability.

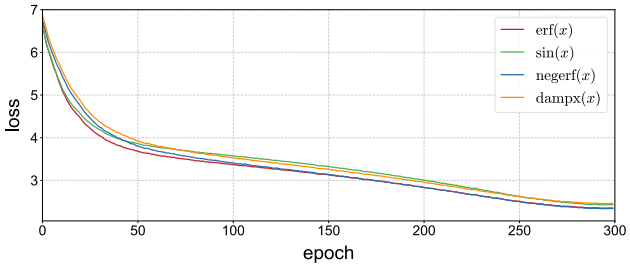

Figure 8. **Training loss curve for different monotonicity.** Monotonic functions consistently achieve lower training loss than non-monotonic functions.

B. Function Search Details

In function search, a wide variety of common functional forms are systematically explored under the constraint of our function properties. The candidates range from polynomial and rational functions to the trigonometric and hyperbolic families, as well as various cumulative distribution functions. Beyond these common functional forms, we also experiment with their variants through translation, scaling, concatenation, and clipping. All candidate functions and their analytical forms are summarized in Tab. 17.

We categorize all candidate functions (see Tab. 7) into four groups: *natural functions*, *transformed basic functions*, *clipped unbounded functions*, and *canonical ratio functions*, and present detailed descriptions and visualizations of how each group is constructed.

Natural functions. This category consists of three functions: $\mathrm{erf}(x)$, $\tanh(x)$, and $\arctan(x)$. As shown in Fig. 9, these functions naturally satisfy all the function properties, including *zero-centeredness*, *boundedness*, *center sensitivity*, and *monotonicity*. Among them, only $\arctan(x)$ is rescaled so that all three functions have their ranges unified to $[-1, 1]$.

Transformed basic functions. This category consists of six functions: $\mathrm{satursin}(x)$, $\mathrm{expsign}(x)$, $\mathrm{exproot}(x)$, $\mathrm{relsign}(x)$, $\mathrm{isru}(x)$, and $\mathrm{cubsign}(x)$. These functions are constructed by starting from simple and commonly used primitives, such as power functions and polynomial forms. Through transformations including translation, scaling, and rotation, we reshape their original structures so that they satisfy all four function properties while preserving the qualitative behavior of the underlying base functions, as shown in Fig. 10.

function	expression
$\mathrm{satursin}(x)$	$\sin\big(\mathrm{clip}(x, -\frac{\pi}{2}, \frac{\pi}{2})\big)$
$\mathrm{isru}(x)$	$x\left(x^2+1\right)^{-1/2}$
$\mathrm{expsign}(x)$	$-\,\mathrm{sign}(x)\left(e^{-\lvert x\rvert}-1\right)$
$\mathrm{smoothsign}(x)$	$x\left(1+\lvert x\rvert\right)^{-1}$
$\mathrm{relsign}(x)$	$x\left(\sqrt{x^2+1}+1\right)^{-1}$
$\mathrm{cubsign}(x)$	$x^3\left(\lvert x\rvert^3+1\right)^{-1}$
$\mathrm{exproot}(x)$	$\mathrm{sign}(x)\left(1-e^{-\sqrt{\lvert x\rvert}}\right)$
$\mathrm{saturlog}(x)$	$\mathrm{sign}(x)\,\ln(\lvert x\rvert+1)\left(\ln(\lvert x\rvert+1)+1\right)^{-1}$
$\tanh(x)$	$\left(e^x-e^{-x}\right)\left(e^x+e^{-x}\right)^{-1}$
$\mathrm{erf}(x)$	$2\pi^{-1/2}\int_0^x e^{-t^2}\,dt$
$\arctan(x)$	$2\pi^{-1}\arctan(x)$
$\mathrm{linear}_{\mathrm{clip}}(x)$	$\mathrm{clip}(x, -1, 1)$
$\mathrm{power23}_{\mathrm{clip}}(x)$	$\mathrm{clip}\big(\mathrm{sign}(x)\,x^{2/3}, -1, 1\big)$
$\mathrm{logsign}_{\mathrm{clip}}(x)$	$\mathrm{clip}(\mathrm{sign}(x)\,\ln(\lvert x\rvert+1), -1, 1)$
$\mathrm{smoothsign}_{\mathrm{clip}}(x)$	$\mathrm{clip}\big(x(1+\lvert x\rvert)^{-1}, -1, 1\big)$
$\mathrm{logquad}_{\mathrm{clip}}(x)$	$\mathrm{clip}\big(\mathrm{sign}(x)\,\ln(x^2+1), -1, 1\big)$

Table 17. Summary of all candidate point-wise functions and their explicit expressions.

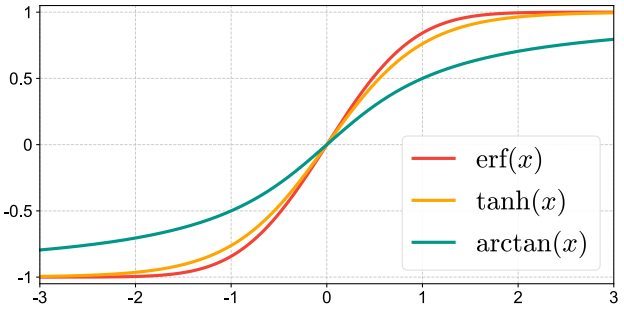

Figure 9. **Visualization of natural functions.**

Clipped unbounded functions. This category consists of five functions: $\mathrm{logsign}(x)$, $\mathrm{logquad}(x)$, $\mathrm{arcsinh}(x)$, $\mathrm{power23}(x)$, and $\mathrm{linear}(x)$. These functions inherently satisfy *zero-centeredness* and *center sensitivity*. For $\mathrm{logsign}_{\mathrm{clip}}(x)$, $\mathrm{logquad}(x)$, and $\mathrm{power23}_{\mathrm{clip}}(x)$, either due to domain asymmetry or because the original form is not monotonic, we construct the negative branch by mirroring the positive side around the origin to ensure *monotonicity*, as shown in Fig. 11. To additionally enforce *boundedness*, we clip their outputs to the interval $[-1, 1]$, which leads to improved performance in practice.

Canonical ratio functions. This category consists of two functions: $\mathrm{saturlog}(x)$ and $\mathrm{smoothsign}(x)$. Both functions are constructed using the canonical ratio form $\frac{f(x)}{|f(x)|+1}$, which naturally enforces *boundedness* and *monotonicity*. By selecting $f(x)$ to be an odd, zero-centered base function, the resulting ratio form automatically satisfies *zero-centeredness* and *center sensitivity* as well. As shown in Fig. 12, this construction yields smooth saturating behaviors that remain stable across a wide input range.

C. Experimental Settings

Vision Transformers. For all supervised classification experiments on ImageNet-1K, we adopt the training configurations summarized in Tab. 18. ViT-B and ViT-L share the same hyperparameters, except that ViT-L employs a modified AdamW momentum setting with (β_1=0.9, β_2=0.95) and a higher stochastic depth rate of 0.5.

Diffusion Transformers. We use the official implementation [41] to train all DiT model sizes as shown in Tab. 19. We observe that the default learning rate is suboptimal for the models in this work. For both the search function experiments and the final evaluation of Derf, we go through three learning rates, 1×10^{-4}, 2×10^{-4}, and 4×10^{-4}, for all models, whether they use LayerNorm or a point-wise function, and report the best result. We also observe that the zero initialization negatively affects the performance of Derf models and other point-wise function models. There-

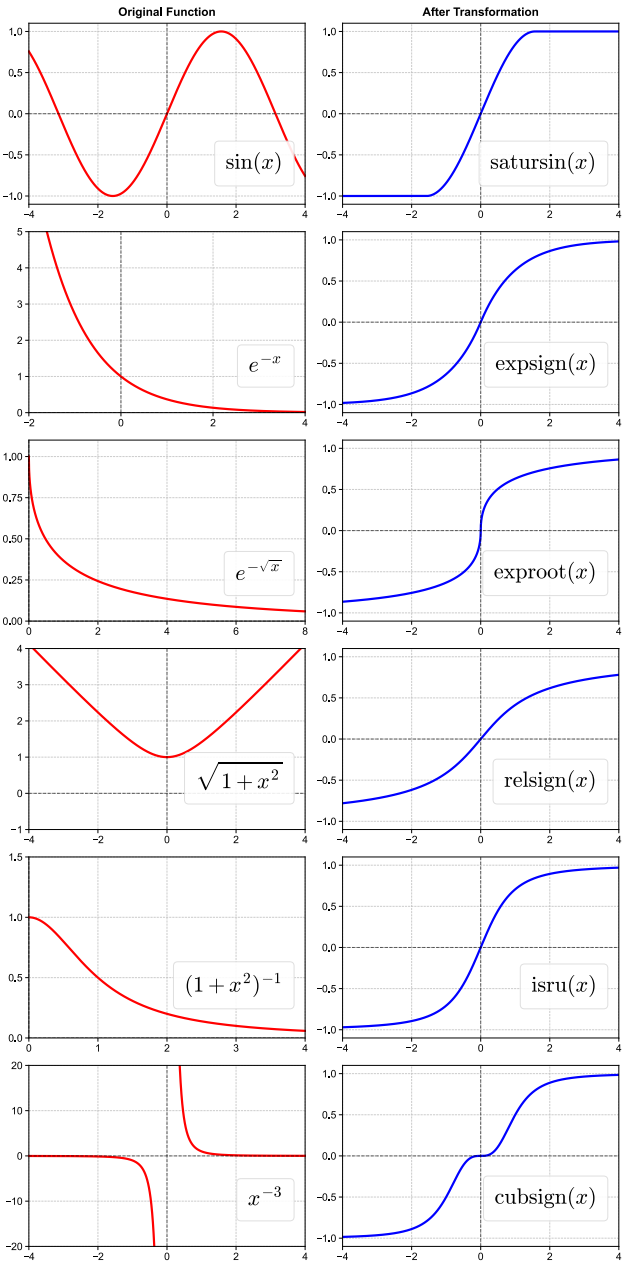

Figure 10. **Visualization of transformed basic functions.**

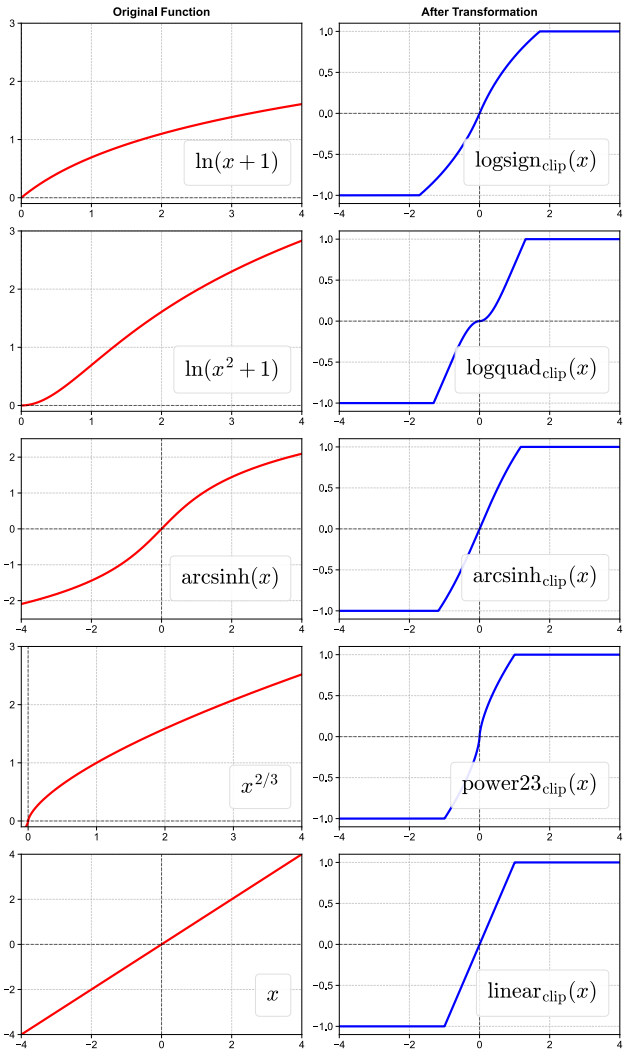

Figure 11. **Visualization of clipped unbounded functions.**

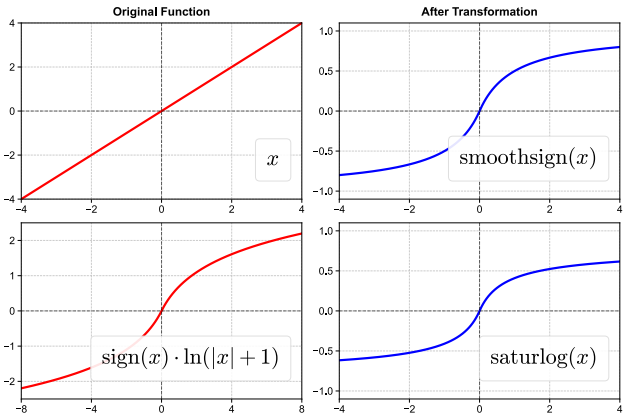

Figure 12. **Visualization of canonical ratio functions.**

fore, we retain the zero initialization for LN models but remove it for the other models.

Speech models. For both wav2vec 2.0 models, we retain the first GroupNorm layer and the LayerNorm located after the convolutional feature extractor, since both primarily serve as data normalization to handle the unnormalized input data. We use the official implementation [4] for both the Base and Large models, keeping all hyperparameters identical to the original setup, as shown in Tab. 20. The only change we make is running

config	value
optimizer	AdamW
base learning rate	4e-3
weight decay	0.05
optimizer momentum	β_1=0.9, β_2=0.999 (B), 0.95 (L)
effective batch size	4096
learning rate schedule	cosine decay
warmup epochs	20
training epochs	300
augmentation	rand-m9-mstd0.5-inc1
label smoothing [50]	0.1
mixup [64]	0.8
cutmix [62]	1.0
random erase [66]	0.25
drop path [24]	0.15 (B), 0.5 (L)
exp. moving average (EMA)	0.9999

Table 18. **Training Configurations of ViT.**

config	value
optimizer	AdamW
base learning rate	{1e-4, 2e-4, 4e-4}
weight decay	0
optimizer momentum	β_1=0.9, β_2=0.999
effective batch size	256
learning rate schedule	constant
training epochs	80
exp. moving average (EMA)	0.9999

Table 19. **Training Configurations of DiT.**

all models—whether normalization-based or point-wise-function-based—in `fp32` precision instead of the default `bf16`. We report the final validation loss.

config	value
optimizer	Adam
learning rate	5e-4 (B), 3e-4 (L)
weight decay	0.01
optimizer momentum	β_1=0.9, β_2=0.98
max tokens	1400000 (B), 1200000 (L)
learning rate schedule	polynomial decay
warmup updates	32000 (B), 20000 (L)
max updates	400000 (B), 250000 (L)
dropout (input to encoder)	0.1
dropout (target features)	0.1
dropout (transformer)	0.0 (B), 0.1 (L)
layer dropout	0.05 (B), 0.2 (L)
feature grad mult	0.1
latent temp	[2,0.5,0.999995] (B), [2.0,0.1,0.999995] (L)
max sample size	250000 (B), 320000 (L)

Table 20. **Training Configurations of wav2vec 2.0.**

DNA models. For both the HyenaDNA model [38] and the Caduceus model [45], we directly follow their official implementations without modifying hyperparameters, as shown in Tab. 21. In particular, Hyena uses LayerNorm and Caduceus uses RMSNorm. For our evaluation, we replace each model's original normalization layer with Derf and report the average accuracy across all tasks.

config	value
optimizer	AdamW
learning rate	6e-4 (H), 8e-3 (C)
sequence length	1024 (H), 131072 (C)
effective batch size	1024 (H), 8 (C)
training steps	10000 (H), 50000 (C)
RC augmentation	true (H), false (C)
MLM probability	0.0 (H), 0.15 (C)
bidirectional	false (H), true (C)

Table 21. **Training Configurations of HyenaDNA and Caduceus.** H denotes HyenaDNA, C denotes Caduceus.

Language models. For the GPT-2 (124M) model, we follow the hyperparameters as shown in Tab. 22. For Derf and DyT, we configure the α initialization separately for the point-wise function layer following the attention layer and for the other point-wise function layers. We try multiple combinations of these initialization settings and report the best validation loss.

config	value
optimizer	AdamW
base learning rate	6e-4
α initialization attention	{0.5, 1.0, 2.0, 4.0}
α initialization other	{0.1, 0.3, 0.5, 1.0}
weight decay	0.1
optimizer momentum	β_1=0.9, β_2=0.95
gradient clipping	1.0
block size	1024
gradient accumulation steps	40
effective batch size	491,520
learning rate schedule	cosine decay
warmup iterations	2,000
training iterations	300,000
dropout	0.0
mixed precision	`bf16`

Table 22. **Training Configurations of GPT-2 (124M).**

D. Additional Results

Beyond evaluating each model with its default normalization layer (typically LN), we additionally test RMSNorm and GroupNorm (GN) to enable a more complete comparison. RMSNorm is widely used in modern large language models, including T5 [42], LLaMA [17, 52, 53], Qwen [5, 60], and DeepSeek [20, 30], while GN is employed in several vision architectures, including ConvNeXt [34], DETR [10], and Swin Transformer [33].

All evaluations follow the same experimental settings described in the previous section. These additional results show that Derf not only surpasses the default choices used in each model, but also outperforms the other normalization alternatives we evaluate.

Vision Transformers. For both ViT-Base and ViT-Large [16], the default normalization layer is LayerNorm. To complement the results, we also evaluate RMSNorm [63]

and GN [56] as additional replacements in Tab. 23. Compared to all other methods, Derf achieves clearly higher top-1 accuracy, demonstrating its effectiveness in vision transformer architectures.

model	LN	DyT	Derf	RMSNorm	GN
ViT-B	82.3%	82.5%	**82.8%**	82.4%	82.5%
ViT-L	83.1%	83.6%	**83.8%**	83.0%	83.1%

Table 23. **Supervised classification accuracy on ImageNet-1K.** Derf achieves higher top-1 accuracy than all other methods on different model sizes.

Diffusion Transformers. For DiT models [41], we additionally evaluate RMSNorm [63] as an alternative normalization layer and compare its performance with LN, DyT, and Derf. As shown in Tab. 24, Derf achieves a clear improvement in FID compared to all other methods.

model	LN	DyT	Derf	RMSNorm
DiT-B/4	64.93	63.94	**63.23**	65.08
DiT-L/4	45.91	45.66	**43.94**	45.02
DiT-XL/2	19.94	20.83	**18.92**	20.76

Table 24. **Image generation quality (FID) on ImageNet.** Lower FID indicates better image generation quality. Derf achieves lower FID scores than all other methods across different DiT models.

Speech models. For two wav2vec 2.0 Transformer models [4], we additionally evaluate RMSNorm [63] as an alternative normalization layer and compare its performance with LN, DyT, and Derf in Tab. 25. Compared to other methods, Derf yields lower validation loss on different model sizes

model	LN	DyT	Derf	RMSNorm
wav2vec 2.0 Base	1.95	1.95	**1.93**	1.95
wav2vec 2.0 Large	1.92	1.91	**1.90**	1.93

Table 25. **Speech pretraining validation loss on the LibriSpeech dataset.** Derf achieves lower validation loss than all other methods across two wav2vec 2.0 models.

DNA models. For the HyenaDNA model [38] and the Caduceus model [45], we additionally evaluate both LayerNorm and RMSNorm for each architecture, regardless of their default choices, and compare their performance with DyT and Derf in Tab. 26.

model	LN	DyT	Derf	RMSNorm
Hyena	85.2%	85.2%	**85.7%**	85.2%
Caduceus	87.0%	86.9%	**87.3%**	86.9%

Table 26. **DNA classification accuracy on the GenomicBenchmarks dataset**, averaged over each subtask. Derf consistently outperforms other methods across two different DNA models.

Language models. For the GPT-2 (124M) model, we additionally evaluate RMSNorm [63] for a more complete comparison of normalization choices. As shown in Tab. 27, Derf achieves comparable performance to both LN and RMSNorm, while clearly outperforming DyT.

model	LN	DyT	Derf	RMSNorm
GPT-2	**2.94**	2.97	**2.94**	2.95

Table 27. **GPT-2 validation loss on the OpenWebText dataset.** Derf matches the performances of both LayerNorm and RMSNorm while achieving lower validation loss than DyT.

E. Loss Calculation Details

Vision Transformers. For ViT models, we measure fitting capacity under a deterministic evaluation setup. We switch the model to evaluation mode, disable drop-path, mixup, cutmix, label smoothing, and all data augmentations, and apply only the standard test-time preprocessing (center crop and normalize). The cross-entropy loss is then computed on the training set and averaged over all samples.

Diffusion Transformers. For DiT models, we evaluate fitting capacity by switching the model to evaluation mode. We apply the standard test-time preprocessing (center crop, random horizontal flip, and normalize). Since DiT does not employ drop-path, no stochastic regularization needs to be disabled. We then compute the diffusion MSE loss over the first 100 training batches and report the average.

Other models. For all other models, wav2vec 2.0, HyenaDNA, Caduceus, and GPT2, we simply apply the same procedure: use the standard test-time preprocessing, disable drop-path or dropout when present, and compute the training loss over the full training set, reporting the average.