JÜLICH Forschungszentrum

HELMHOLTZAI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT

Polynomial, trigonometric, and tropical activations

Which functions can be used as activations in deep neural networks?

ICLR

Which functions can be used as activations in deep neural networks?

functions

Orthogonal decomposition in a Hilbert space

- Inner product: $< f, g >$.
- Complete.

Orthogonal decomposition in a Hilbert space

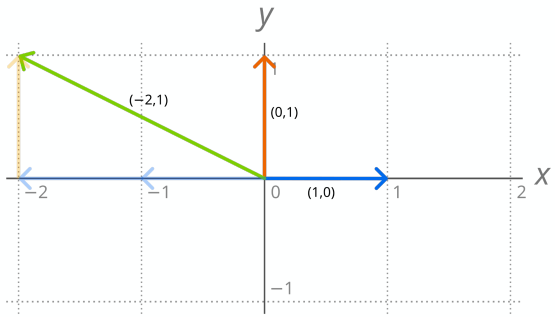
- Inner product: < f, g >.
- Complete.

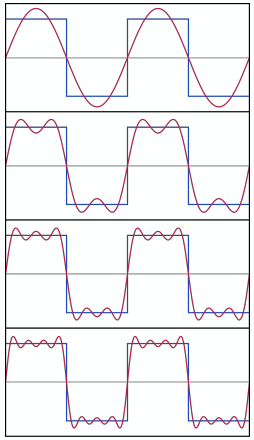
$$x = \sum_{n=0}^{\infty} a_n e_n,$$

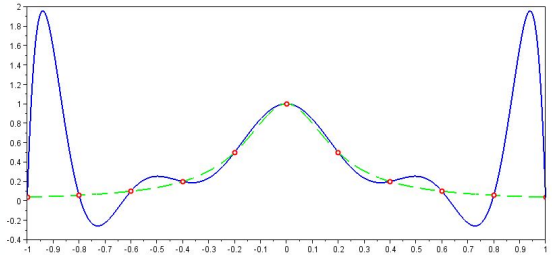
Orthogonal decomposition in a Hilbert space

- Inner product: < f, g >.
- Complete.

$$x = \sum_{n=0}^{\infty} a_n e_n,$$







$$\langle f, g \rangle = \int w(x) f(x) g(x)\, dx.$$

$$w(x) = e^{-x^2/2}$$

$$\langle f, g \rangle = \int w(x) f(x) g(x)\, dx.$$

$$w(x) = e^{-x^2/2}$$

- The **"probabilist's Hermite polynomials"** are given by

$$\mathrm{He}_n(x) = (-1)^n e^{\frac{x^2}{2}} \frac{d^n}{dx^n} e^{-\frac{x^2}{2}},$$

- while the **"physicist's Hermite polynomials"** are given by

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n} e^{-x^2}.$$

$$\langle f, g \rangle = \int w(x) f(x) g(x)\, dx.$$

$$w(x) = e^{-x^2/2}$$

$$\begin{aligned}
\mathrm{He}_0(x) &= 1, \\
\mathrm{He}_1(x) &= x, \\
\mathrm{He}_2(x) &= x^2 - 1, \\
\mathrm{He}_3(x) &= x^3 - 3x, \\
\mathrm{He}_4(x) &= x^4 - 6x^2 + 3, \\
\mathrm{He}_5(x) &= x^5 - 10x^3 + 15x, \\
\mathrm{He}_6(x) &= x^6 - 15x^4 + 45x^2 - 15, \\
\mathrm{He}_7(x) &= x^7 - 21x^5 + 105x^3 - 105x, \\
\mathrm{He}_8(x) &= x^8 - 28x^6 + 210x^4 - 420x^2 + 105,
\end{aligned}$$

$$\langle f, g \rangle = \int w(x) f(x) g(x)\, dx.$$

$$w(x) = e^{-x^2/2}$$

$$\int_{-\infty}^{\infty} \mathrm{He}_m(x)\, \mathrm{He}_n(x) e^{-\frac{x^2}{2}}\, dx = \sqrt{2\pi} n! \delta_{nm}$$

With δ_{nm} the Kronecker delta.

$$\langle f, g \rangle = \int w(x) f(x) g(x)\, dx.$$

$$w(x) = e^{-x^2/2}$$

$$\int_{-\infty}^{\infty} \mathrm{He}_m(x)\, \mathrm{He}_n(x) e^{-\frac{x^2}{2}}\, dx = \sqrt{2\pi} n! \delta_{nm}$$

With δ_{nm} the Kronecker delta.

$$\langle f, g \rangle = \int w(x) f(x) g(x)\, dx.$$

$$w(x) = e^{-x^2/2}$$

$$\int_{-\infty}^{\infty} \mathrm{He}_m(x)\, \mathrm{He}_n(x) e^{-\frac{x^2}{2}}\, dx = \sqrt{2\pi} n! \delta_{nm}$$

With δ_{nm} the Kronecker delta.

$$x \sim \mathcal{N}(0, 1)$$

Hermite Activation

$$x \sim \mathcal{N}(0,1)$$

$$\langle f, g \rangle = \int w(x) f(x) g(x)\, dx.$$

$$w(x) = e^{-x^2/2}$$

$$\begin{aligned}
\mathrm{He}_0(x) &= 1,\\
\mathrm{He}_1(x) &= x,\\
\mathrm{He}_2(x) &= x^2 - 1,\\
\mathrm{He}_3(x) &= x^3 - 3x,\\
\mathrm{He}_4(x) &= x^4 - 6x^2 + 3,\\
\mathrm{He}_5(x) &= x^5 - 10x^3 + 15x,\\
\mathrm{He}_6(x) &= x^6 - 15x^4 + 45x^2 - 15,\\
\mathrm{He}_7(x) &= x^7 - 21x^5 + 105x^3 - 105x,\\
\mathrm{He}_8(x) &= x^8 - 28x^6 + 210x^4 - 420x^2 + 105,
\end{aligned}$$

$$x \mapsto F(x) = \sum_{k=0}^{n} \frac{a_k}{k!} \mathrm{He}_k(x)$$

$$x \sim \mathcal{U}(-\pi, \pi)$$

$$x \sim \mathcal{U}(-\pi, \pi)$$

$$\begin{cases} \int_{-\pi}^{\pi} \cos(mx)\cos(nx)dx = \pi\delta_{nm} \\ \int_{-\pi}^{\pi} \sin(mx)\sin(nx)dx = \pi\delta_{nm} \\ \int_{-\pi}^{\pi} \cos(mx)\sin(nx)dx = 0 \end{cases}$$

With δ_{nm} the Kronecker delta function.

Fourier Activation

$$x \sim \mathcal{U}(-\pi, \pi)$$

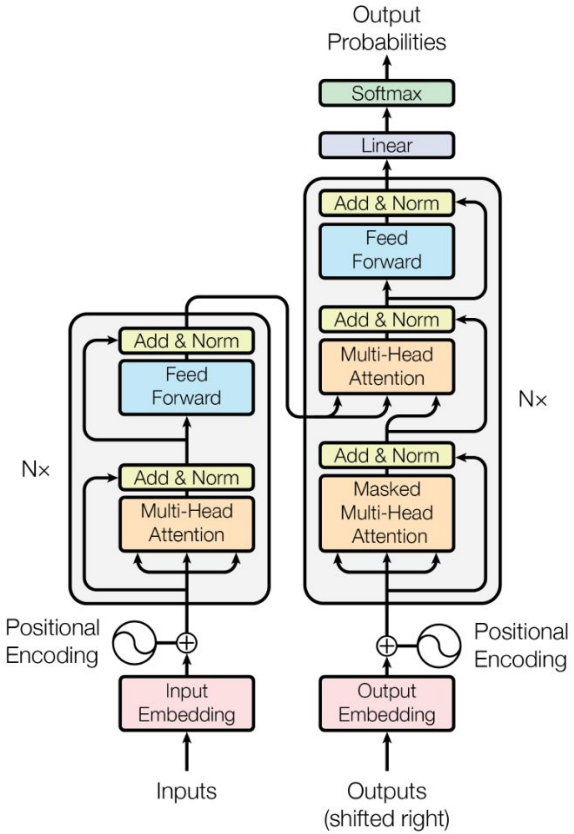
$$\begin{cases} \int_{-\pi}^{\pi} \cos(mx)\cos(nx)dx = \pi\delta_{nm} \\ \int_{-\pi}^{\pi} \sin(mx)\sin(nx)dx = \pi\delta_{nm} \\ \int_{-\pi}^{\pi} \cos(mx)\sin(nx)dx = 0 \end{cases}$$

With δ_{nm} the Kronecker delta function.

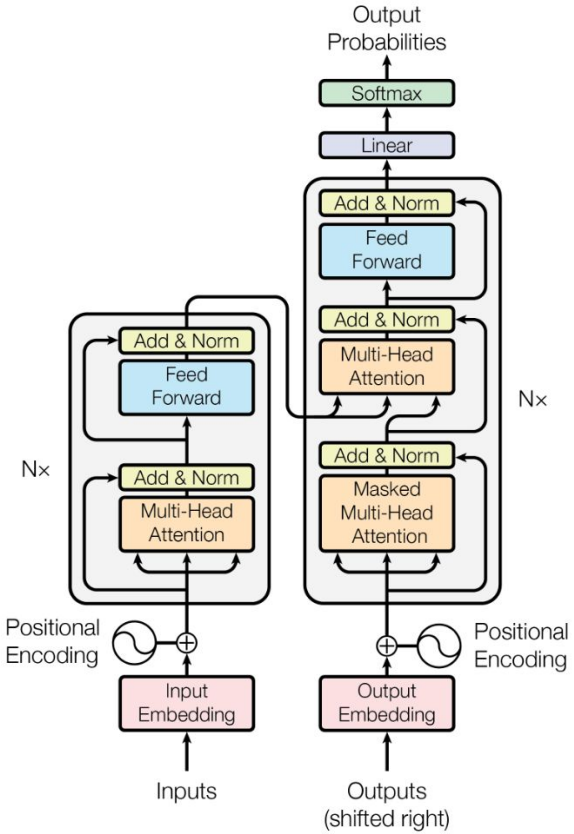
$$x \mapsto F(x) = a_0 + \sum_{k=1}^{n} \frac{(a_k \cos(kx) + b_k \sin(kx))}{k!}$$

Which functions can be used as activations in deep neural networks?

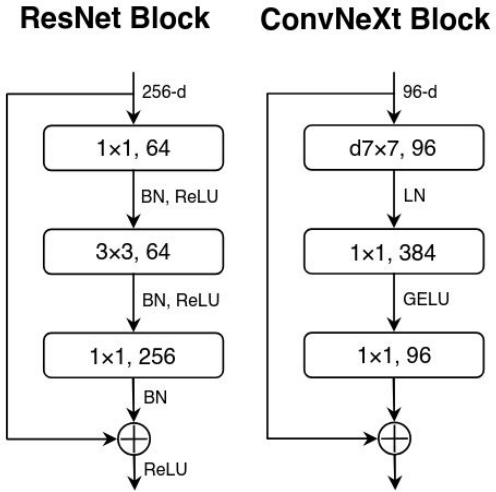
deep neural networks?



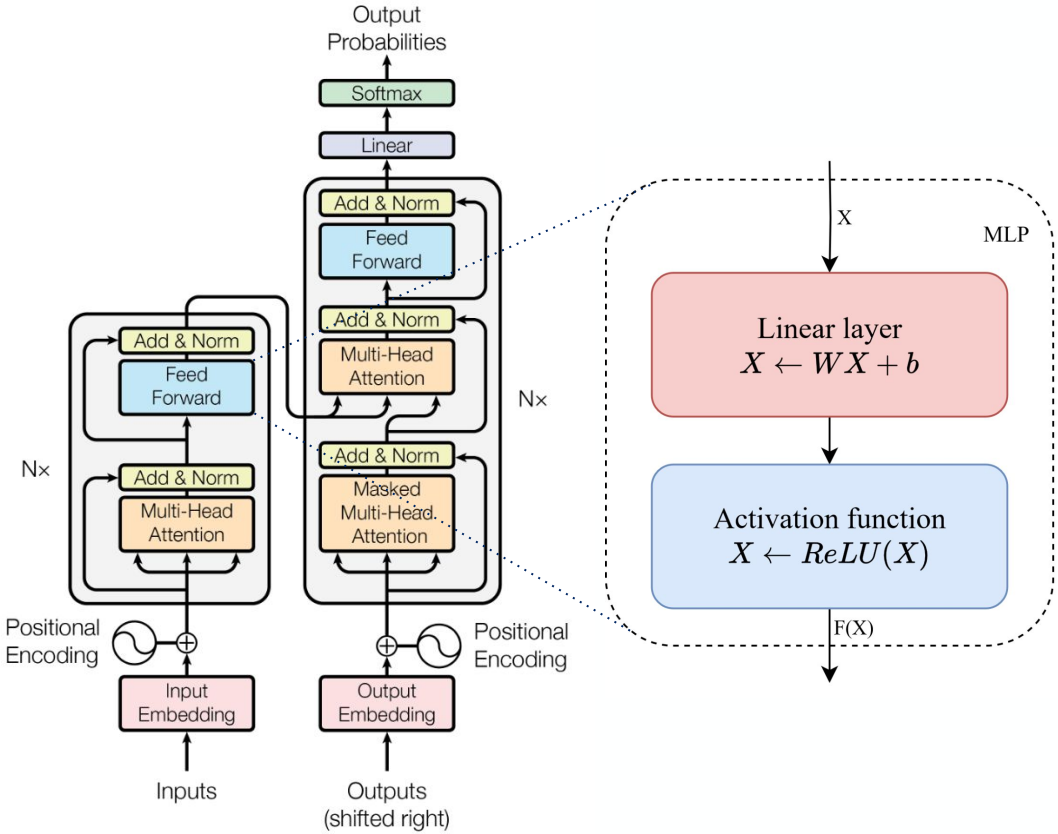
[Vaswani et al. 2017]



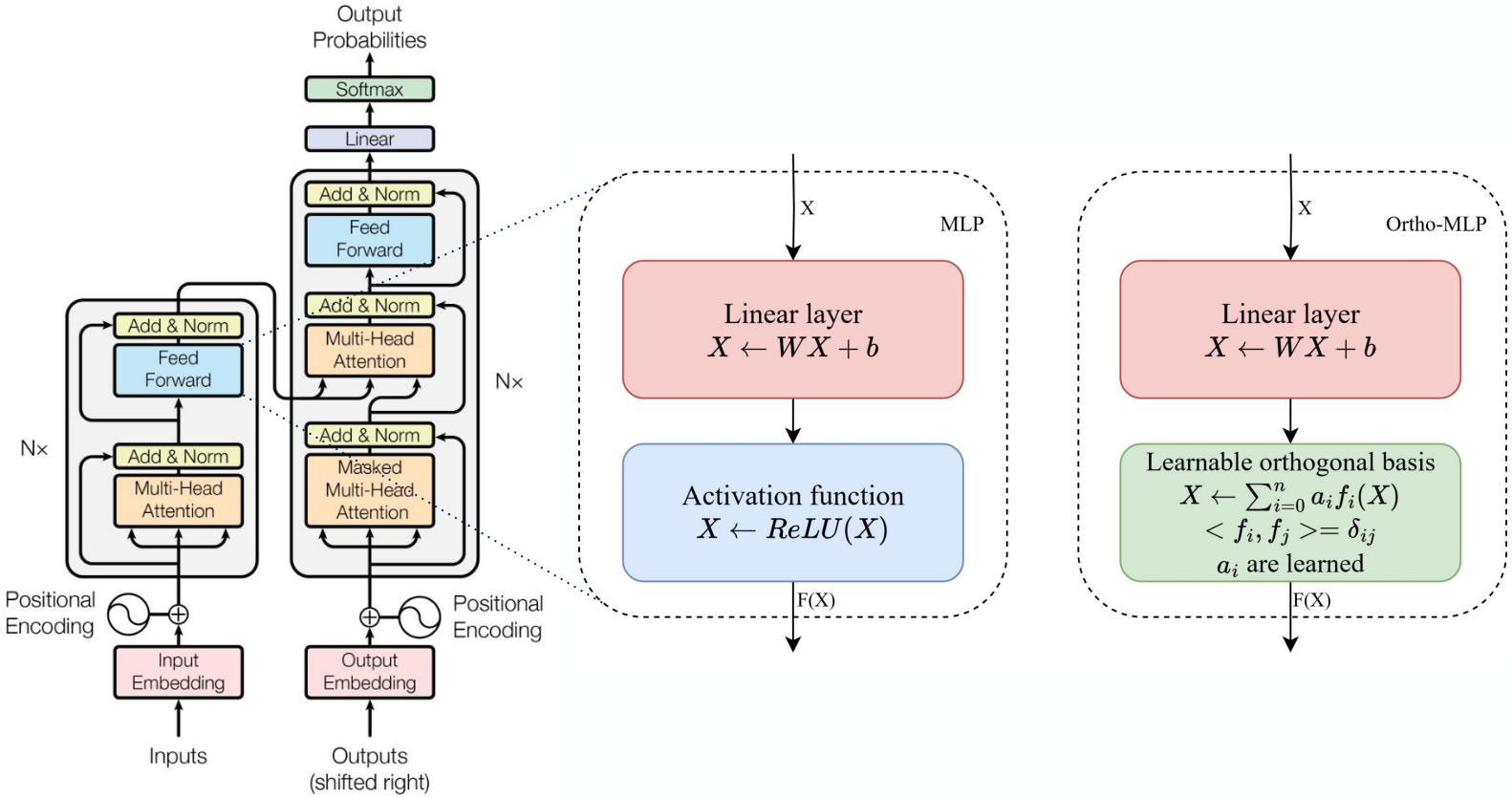
[Vaswani et al. 2017]



[He et al. 2015] [Liu et al. 2020]



[Vaswani et al. 2017]



[Vaswani et al. 2017]

The difficulty of training polynomial activations

- Numerical instability.
- Exploding / vanishing activations / gradients.
- x^n grows / decays rapidly.

Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun

Microsoft Research

{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Abstract

Rectified activation units (rectifiers) are essential for state-of-the-art neural networks. In this work, we study rectifier neural networks for image classification from two

and the use of smaller strides [33, 24, 2, 25]), new non-linear activations [21, 20, 34, 19, 27, 9], and sophisticated layer designs [29, 11]. On the other hand, better generalization is achieved by effective regularization techniques [12, 26, 9, 31], aggressive data augmentation

The analysis shown in He et al. (2015) could be stated as *the output signal of each MLP block should have the same variance as the input signal*. And since learning is performed with backpropagation, this same rule should apply to the gradients as well, meaning that *the variance of the gradient of the input should also be equal to the variance of the gradient of the output of the MLP*.

The analysis shown in He et al. (2015) could be stated as *the output signal of each MLP block should have the same variance as the input signal*. And since learning is performed with backpropagation, this same rule should apply to the gradients as well, meaning that *the variance of the gradient of the input should also be equal to the variance of the gradient of the output of the MLP*.

$$\alpha = \mathrm{Var}[x] \cdot \mathbb{E}\left[F(x)^2\right]^{-1}$$

The analysis shown in He et al. (2015) could be stated as *the output signal of each MLP block should have the same variance as the input signal*. And since learning is performed with backpropagation, this same rule should apply to the gradients as well, meaning that *the variance of the gradient of the input should also be equal to the variance of the gradient of the output of the MLP*.

$$\alpha = \mathrm{Var}[x] \cdot \mathbb{E}\left[F(x)^2\right]^{-1}$$

$$\alpha' = \mathrm{Var}[x] \cdot \mathbb{E}\left[F'(x)^2\right]^{-1}$$

Variance preserving initialization: ReLU Activation

$$\mathbb{E}[\mathrm{ReLU}(x)^2] = \int_0^\infty x^2 \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx = \frac{1}{2}$$

$$\mathbb{E}\left[\left(\frac{d}{dx}\mathrm{ReLU}(x)\right)^2\right] = \int_0^\infty \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx = \frac{1}{2}$$

Variance preserving initialization: ReLU Activation

torch.nn.init.calculate_gain(*nonlinearity*, *param=None*) [SOURCE]

Return the recommended gain value for the given nonlinearity function.

The values are as follows:

nonlinearity	gain
Linear / Identity	1
Conv{1,2,3}D	1
Sigmoid	1
Tanh	$\frac{5}{3}$
ReLU	$\sqrt{2}$
Leaky Relu	$\sqrt{\frac{2}{1+\text{negative\_slope}^2}}$
SELU	$\frac{3}{4}$

Variance preserving initialization: Hermite Activation

$$x \mapsto F(x) = \sum_{k=0}^{n} \frac{a_k}{k!} \mathrm{He}_k(x)$$

Variance preserving initialization: Hermite Activation

$$x \mapsto F(x) = \sum_{k=0}^{n} \frac{a_k}{k!} \mathrm{He}_k(x)$$

$$\forall k \in [\![1, n]\!] \; a_k = 1 \; and \; a_0 = \sqrt{1 - \frac{1}{n!}}$$

$$\alpha = \alpha' = \left(\sum_{k=0}^{n-1} \frac{1}{k!} \right)^{-1}$$

Variance preserving initialization: Hermite Activation

$$\mathbb{E}\left[F(x)^2\right] = \int_{-\infty}^{+\infty} F^2(x) \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}} dx$$

$$= \int_{-\infty}^{+\infty} \left(\sum_{k=0}^{n} \frac{a_k}{k!} \mathrm{He}_k(x)\right)^2 \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}} dx$$

$$\mathbb{E}\left[F(x)^2\right] = \int_{-\infty}^{+\infty} \sum_{k=0}^{n} \frac{a_k^2}{(k!)^2} \mathrm{He}_k(x)^2 \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}} dx$$

$$x \mapsto F(x) = \sum_{k=0}^{n} \frac{a_k}{k!} \mathrm{He}_k(x)$$

$$\forall k \in [\![1, n]\!]\ a_k = 1 \text{ and } a_0 = \sqrt{1 - \frac{1}{n!}}$$

$$\alpha = \alpha' = \left(\sum_{k=0}^{n-1} \frac{1}{k!}\right)^{-1}$$

Variance preserving initialization: Hermite Activation

$$x \mapsto F(x) = \sum_{k=0}^{n} \frac{a_k}{k!} \mathrm{He}_k(x)$$

$$a_k = \frac{1}{\sqrt{e}} \text{ and } a_0 = \frac{1}{\sqrt{e}} \sqrt{1 - \frac{1}{n!}}$$

Variance preserving initialization: Fourier Activation

$$x \mapsto F(x) = a_0 + \sum_{k=1}^{n} \frac{(a_k \cos(kx) + b_k \sin(kx))}{k!}$$

$$a_k = \frac{1}{\sqrt{I_0(2)}} \text{ and } a_0 = \frac{1}{\sqrt{I_0(2)}} \sqrt{1 - \frac{1}{(n!)^2}}$$

$$I_0(2) \approx 2.2795\ldots$$

Implementation details

- Parallel form:

$$\frac{\mathrm{He}_n(x)}{n!} = \sum_{m=0}^{\left\lfloor \frac{n}{2} \right\rfloor} \frac{(-1)^m}{m!(n-2m)!} \frac{x^{n-2m}}{2^m}$$

- Recursive form:

$$\mathrm{He}_{n+1}(x) = x\,\mathrm{He}_n(x) - n\,\mathrm{He}_{n-1}(x)$$

- Derivative property:

$$\mathrm{He}'_k(x) = k\,\mathrm{He}_{k-1}(x)$$

Implementation details

- Cosine basis:

$$x \mapsto F(x) = a_0 + \sum_{k=1}^{n} \frac{(a_k \cos(kx) + b_k \sin(kx))}{k!} \longrightarrow x \mapsto F(x) = a_0 + \sqrt{2} \sum_{k=1}^{n} \frac{a_k \cos(f_k x - \phi_k)}{k!}$$

as it is less expensive in terms of FLOPs. The learnable parameters here are initialized as follows: $\forall k \in \mathbb{N}^* f_k = k, \phi_k = \frac{\pi}{4}$ and a_k and a_0 initialized as in 3.14. In our implementation of Fourier activation, not only were the coefficients learnable, but also the frequencies yielding to what is known as "cosine basis" Mallat (2009) rather than Fourier series.

- Unitary variance:

$$x \sim \mathcal{U}(-\sqrt{3}, \sqrt{3}).$$

$$x \mapsto F(x) = a_0 + \sum_{k=1}^{n} \frac{1}{k!} \left(a_k \cos(k \frac{\pi}{\sqrt{3}} x) + b_k \sin(k \frac{\pi}{\sqrt{3}} x) \right)$$

Deep polynomial neural networks are multivariate polynomial mappings

Deep MLPs are compositions of affine transformations and activation functions applied layer by layer. When the activation functions are polynomial, the entire network can be expressed as a polynomial mapping.

Definition F.1. Let $n, m \in \mathbb{N}$. A function $F : \mathbb{R}^n \to \mathbb{R}^m$ is called a *polynomial mapping* if each component function $F_i : \mathbb{R}^n \to \mathbb{R}$, for $i = 1, \ldots, m$, is a polynomial in n variables. Explicitly, this means that for each i, F_i has the form:

$$F_i(x_1, \ldots, x_n) = \sum_{|\alpha| \leq d_i} c_{i,\alpha} x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}, \tag{73}$$

where the sum is taken over all multi-indices $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$ such that $|\alpha| = \alpha_1 + \alpha_2 + \cdots + \alpha_n \leq d_i$, $c_{i,\alpha} \in \mathbb{R}$ are real coefficients, and $d_i \in \mathbb{N}$.

Deep polynomial neural networks are multivariate polynomial mappings

Definition F.2. A *deep neural network* with L layers, input dimension n, and output dimension m is a function $F : \mathbb{R}^n \to \mathbb{R}^m$ of the form:

$$F(x) = W_L\sigma(W_{L-1}\sigma(\cdots\sigma(W_1x + b_1)\cdots) + b_{L-1}) + b_L, \tag{74}$$

where $\forall i \in [\![1, L]\!]\ C_i \in \mathbb{N}^*$. Each $W_i \in \mathbb{R}^{C_i \times C_{i-1}}$ is a weight matrix, $b_i \in \mathbb{R}^{C_i}$ is a bias vector, and σ is an activation function applied element-wise.

Proposition F.3. *Let $F : \mathbb{R}^n \to \mathbb{R}^m$ be a deep neural network with polynomial activation functions of degree d. Then F is a polynomial mapping of degree at most d^L. Furthermore, any L-layer MLP could be collapsed into an equivalent 3-layer network with the middle layer being a polynomial mapping of degree at most d^L.*

Deep polynomial neural networks are multivariate polynomial mappings

Definition F.2. A *deep neural network* with L layers, input dimension n, and output dimension m is a function $F : \mathbb{R}^n \to \mathbb{R}^m$ of the form:

$$F(x) = W_L\sigma(W_{L-1}\sigma(\cdots\sigma(W_1x + b_1)\cdots) + b_{L-1}) + b_L, \tag{74}$$

where $\forall i \in [\![1, L]\!]\ C_i \in \mathbb{N}^*$. Each $W_i \in \mathbb{R}^{C_i \times C_{i-1}}$ is a weight matrix, $b_i \in \mathbb{R}^{C_i}$ is a bias vector, and σ is an activation function applied element-wise.

Proposition F.3. *Let $F : \mathbb{R}^n \to \mathbb{R}^m$ be a deep neural network with polynomial activation functions of degree d. Then F is a polynomial mapping of degree at most d^L. Furthermore, any L-layer MLP could be collapsed into an equivalent 3-layer network with the middle layer being a polynomial mapping of degree at most d^L.*

The total number of monomial terms in this mapping is $\binom{d^L+n}{d^L}$.

Tropical polynomial activation

$$x \oplus y := \max\{x, y\} \quad \text{and} \quad x \otimes y := x + y$$

Tropical polynomial activation

$$x \oplus y := \max\{x, y\} \quad \text{and} \quad x \otimes y := x + y$$

$$x^{\otimes a} := x \otimes \cdots \otimes x = a \cdot x$$

Tropical polynomial activation

$$x \oplus y := \max\{x, y\} \quad \text{and} \quad x \otimes y := x + y$$

$$x^{\otimes a} := x \otimes \cdots \otimes x = a \cdot x$$

$$x \mapsto F(x) = \bigoplus_{k=0}^{n} a_k \otimes x^{\otimes k} := \max_{k=0}^{n} \{a_k + kx\}$$

With $\max_{k=0}^{n} \{a_k + kx\} := \max(a_0, a_1 + x, \ldots, a_n + nx)$.

Tropical polynomial activation

$$x \oplus y := \max\{x, y\} \quad \text{and} \quad x \otimes y := x + y$$

$$x^{\otimes a} := x \otimes \cdots \otimes x = a \cdot x$$

$$x \mapsto F(x) = \bigoplus_{k=0}^{n} a_k \otimes x^{\otimes k} := \max_{k=0}^{n} \{a_k + kx\}$$

$$x \mapsto F(x) = \frac{\sqrt{2}}{n} \max_{k=0}^{n} \{a_k + kx\}$$

Tropical polynomial activation

$$x \oplus y := \max\{x, y\} \quad \text{and} \quad x \otimes y := x + y$$

$$x^{\otimes a} := x \otimes \cdots \otimes x = a \cdot x$$

$$x \mapsto F(x) = \bigoplus_{k=0}^{n} a_k \otimes x^{\otimes k} := \max_{k=0}^{n} \{a_k + kx\}$$

$$x \mapsto F(x) = \frac{\sqrt{2}}{n} \max_{k=0}^{n} \{a_k + kx\}$$

$$\forall k \in [\![0, n]\!] \; a_k = 1$$

Tropical activations are convex conjugates

$$x \mapsto F(x) = \frac{\sqrt{2}}{n} \max_{k=0}^{n} \{a_k + kx\}$$

Definition 3.18. *Convex conjugate (Legendre-Fenchel).* Let $x \in \mathbb{R}$, $f^* : \mathbb{R} \to \mathbb{R}$ is the convex conjugate of $f : \mathbb{R} \to \mathbb{R}$ if and only if:

$$f^*(x) = \sup_{k \in \mathbb{R}} \{kx - f(k)\} \tag{19}$$

The tropical polynomial activation can be viewed as a generalization of the ReLU activation. Furthermore, it can be interpreted as a discrete version of the convex conjugate of a function f whose values at the natural integers $k \in \mathbb{N}$ are $f(k) = -a_k$ effectively encoding the convex hull of the epigraph of f, as illustrated in Figures 5 and 6.

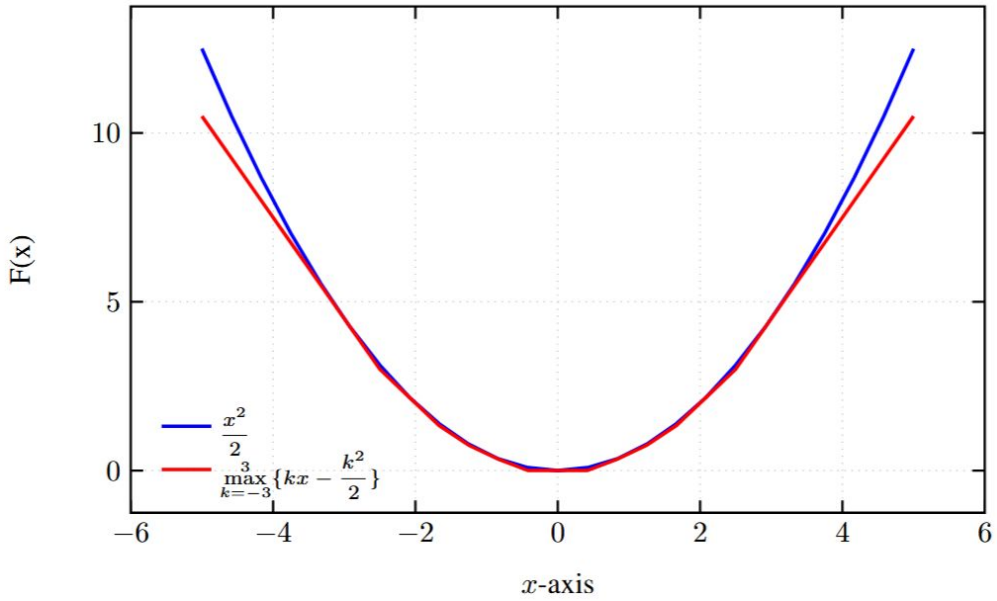
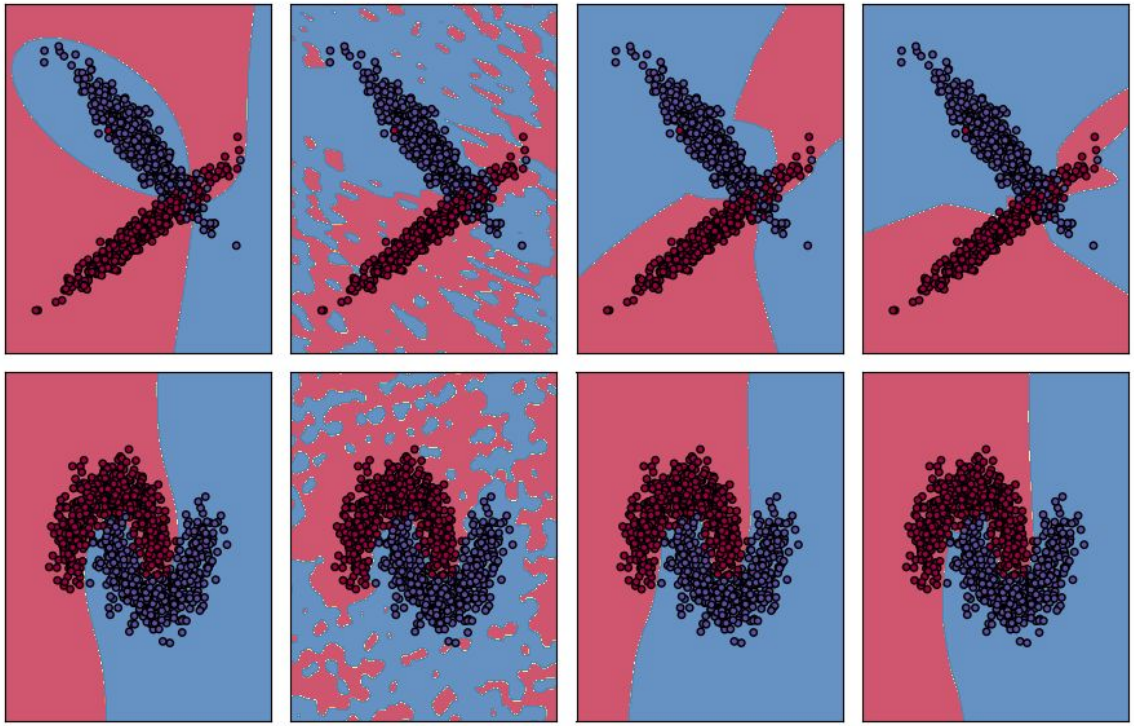


Figure 6: Interpolation of $\frac{x^2}{2}$ function by the Tropical-Laurent polynomial (with potentially negative powers) $\max_{k=-3}^{3}\{kx - \frac{k^2}{2}\}$ of degree 6.

Decision boundaries



Evaluation accuracy of ConvNeXt-T on ImageNet1k

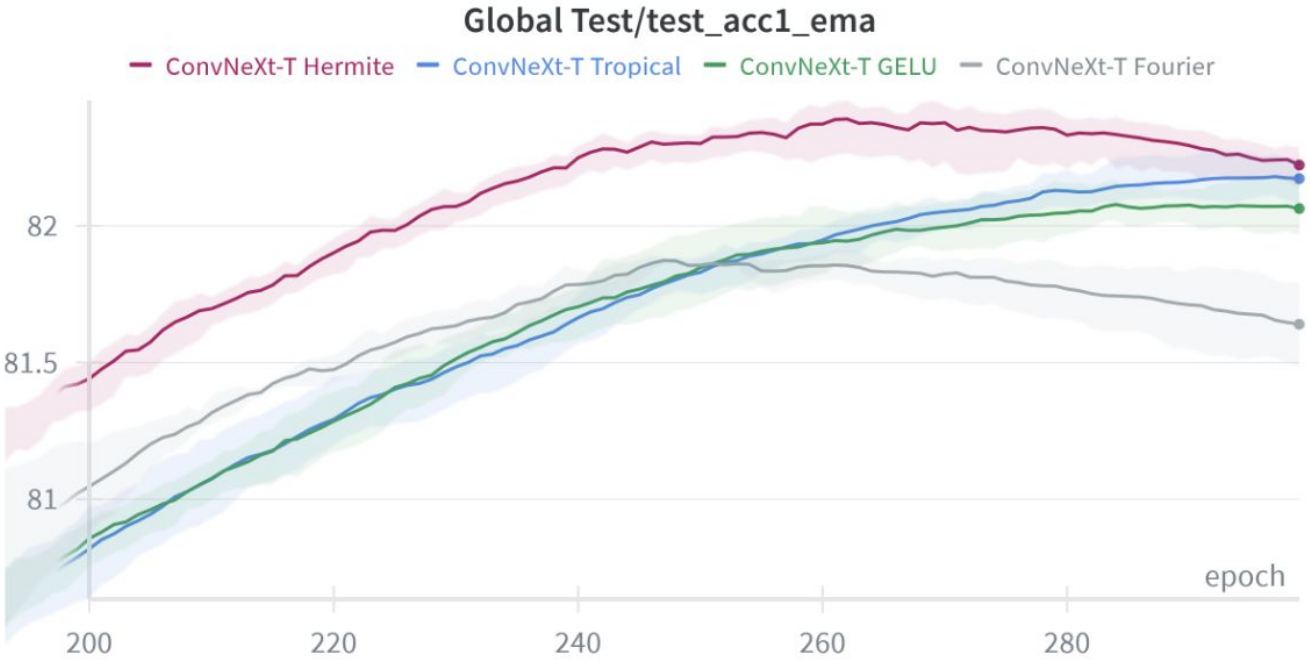


Figure 12: Top1 evaluation accuracy for ConvNeXt-T on ImageNet1k. The solid lines represent the mean of the metric over 5 different seeds, and the shaded areas show the range (min to max).

Evaluation accuracy of ConvNeXt-T on ImageNet1k

Table 1: Training and validation results of ConvNeXt-T (28M) model on ImageNet-1k classification. Values are reported as mean ± standard deviation over 5 seeds. p-values (two-tailed Student's t-test assuming equal variances) are for Val Top-1 accuracy compared to GELU.

Act.	Deg.	Train Loss ↓	Val Top-1(%) ↑	Val Top-5(%) ↑	FLOP	FLOP/Act.	p-value (Top-1)
GELU	-	2.824 ± 0.0051	82.06 ± 0.072	95.92 ± 0.038	4.57G	12	-
Tropical	6	2.854 ± 0.0080	82.17 ± 0.063	95.95 ± 0.072	4.62G	3d + 1 = 19	0.0345 (*)
Fourier	6	**2.759 ± 0.0167**	81.64 ± 0.153	95.47 ± 0.049	4.83G	7d + 1 = 43	0.0005 (***)
Hermite	3	2.788 ± 0.0072	**82.22 ± 0.064**	**95.97 ± 0.045**	4.58G	4d + 1 = 13	0.0062 (**)

Validation loss of the GPT2 model on OpenWebText

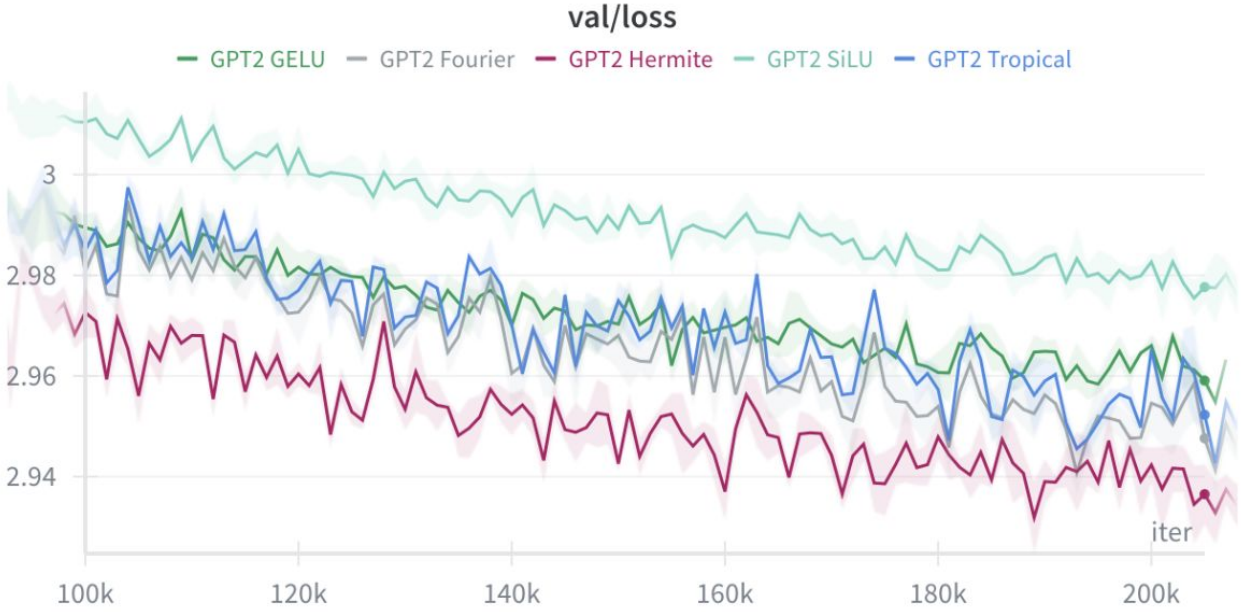


Figure 15: Comparison of the validation losses of the GPT2 model (124M) on OpenWebText with GELU, SiLU, Hermite, Fourier, and Tropical activations. The solid lines represent the mean of the metric over 5 different seeds, and the shaded areas show the range (min to max).

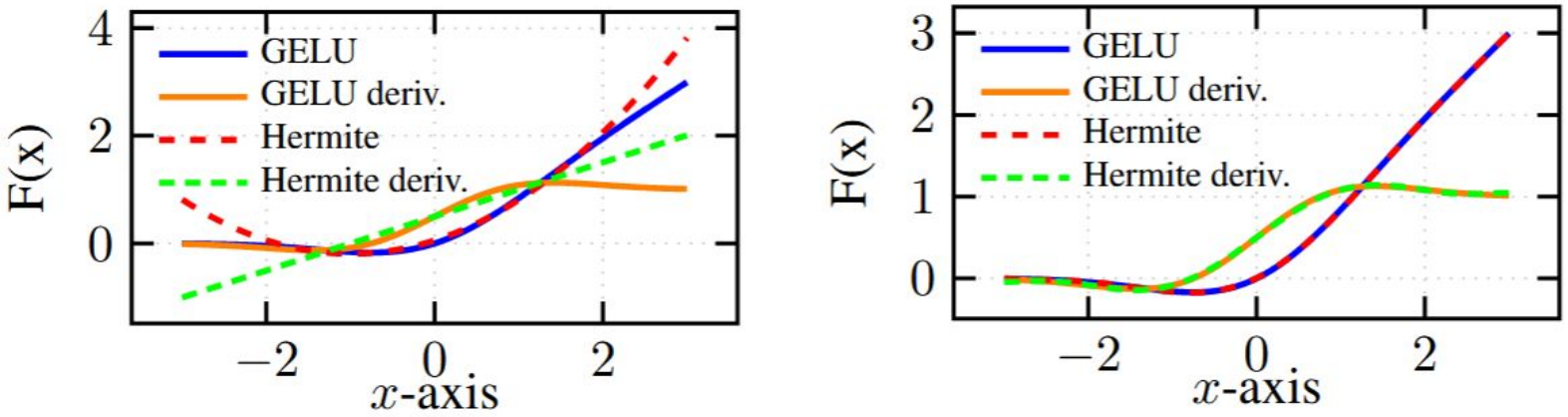
Validation loss of the GPT2 model on OpenWebText

Table 2: Training and validation results for next-token prediction using GPT-2 (124M) model with different activations. Values are reported as mean ± standard deviation over 5 different seeds. Perplexity is computed as exp(loss). p-values (two-tailed Student's t-test assuming equal variances) compare each activation's validation loss against GELU.

Act.	Deg.	Train PPL ↓	Train Loss ↓	Val PPL ↓	Val Loss ↓	FLOP	p-value (Val Loss)
GELU	-	19.003 ± 0.156	2.944 ± 0.0082	19.319 ± 0.076	2.961 ± 0.0039	87.52G	-
SiLU	-	19.324 ± 0.106	2.962 ± 0.0055	19.664 ± 0.088	2.979 ± 0.0045	87.37G	0.0001 (***)
Tropical	6	18.840 ± 0.107	2.936 ± 0.0057	19.027 ± 0.055	2.946 ± 0.0029	87.75G	0.0001 (***)
Fourier	6	18.761 ± 0.071	2.930 ± 0.0038	18.965 ± 0.154	2.941 ± 0.0086	88.69G	0.0014 (**)
Hermite	3	**18.678 ± 0.093**	**2.926 ± 0.0049**	**18.821 ± 0.293**	**2.932 ± 0.0175**	87.56G	0.0067 (**)

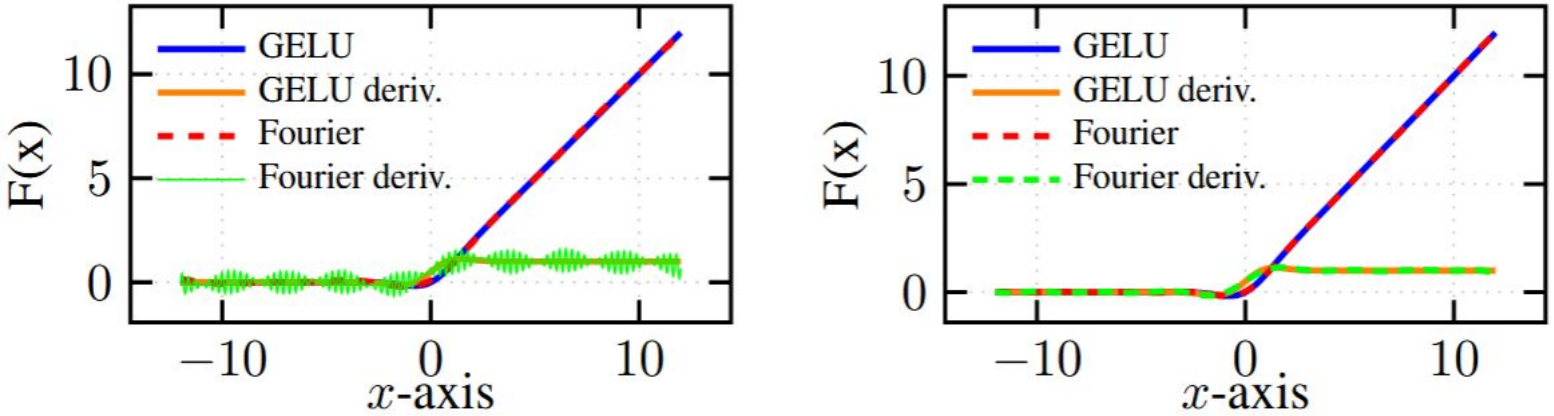
All experiments were conducted under fixed configurations to ensure that any observed differences were solely due to the choice of activation function, allowing for fair and reproducible comparisons[2].

Fitting a classical activation



Fitting a GELU with a Hermite Activation of degree 3 (left) and of degree 8 (right).

Fitting a classical activation



Lagrange interpolation (left) and Hermite interpolation (right) of a GELU with a Fourier Activation of degree 6

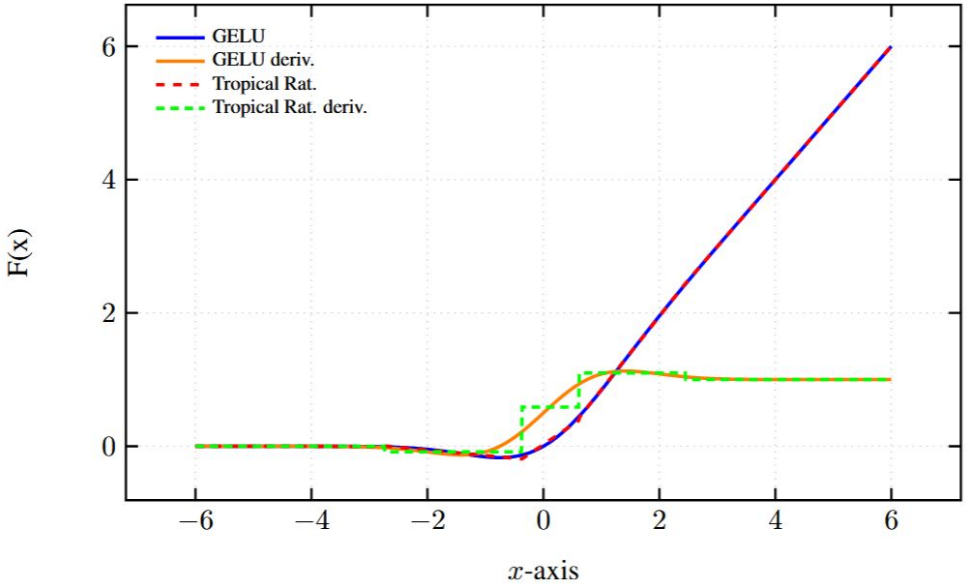
Fitting a classical activation

The tropical quotient $\oslash$ of x over y is defined as:

$$x \oslash y := x - y$$

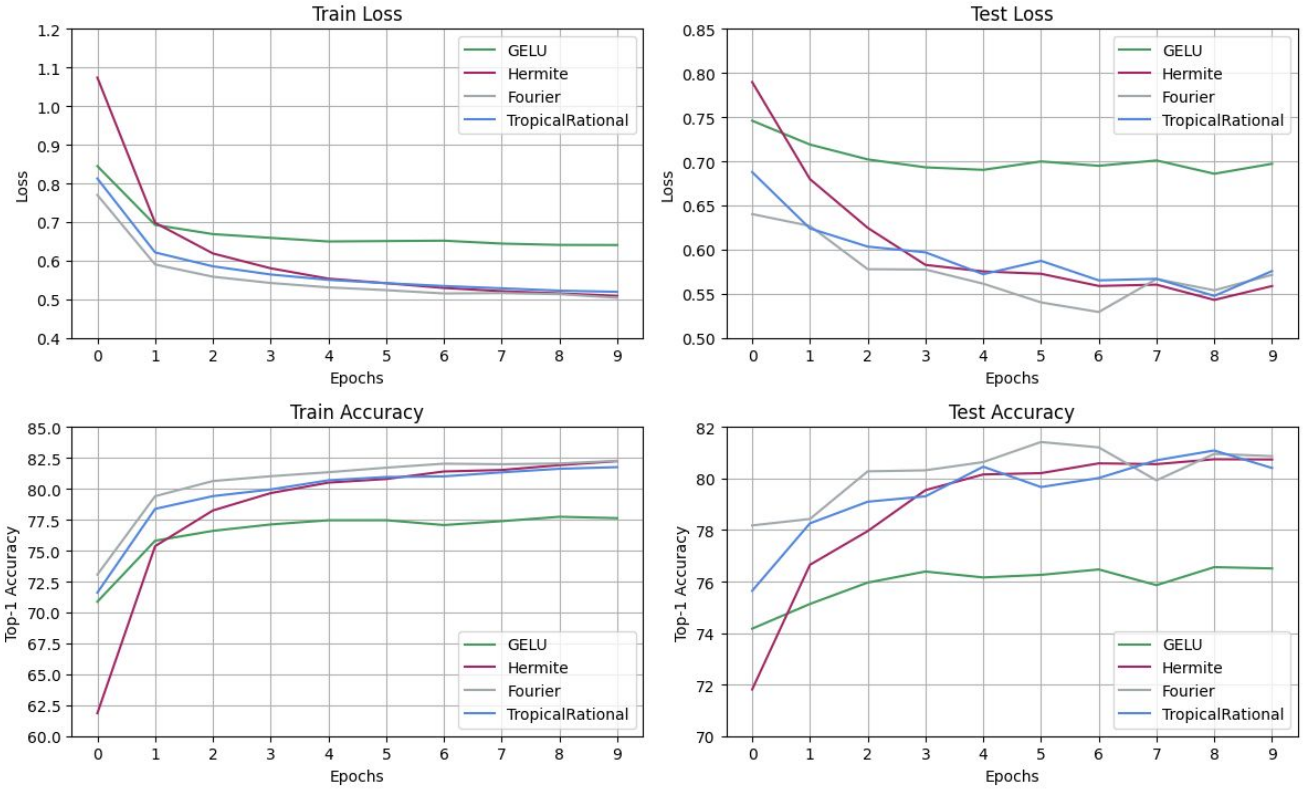
$$F : \mathbb{R} \to \mathbb{R}$$
$$F(x) \mapsto F_1(x) \oslash F_2(x) := F_1(x) - F_2(x)$$

- Tropical Geometry of Deep Neural Networks [Liwen Zhang et al.]



Hermite interpolation of a GELU with a Tropical Rational Activation of degree 6 in both the numerator and the denominator.

Fine-tuning of a pretrained ConvNeXt-T on CIFAR10



FLOPS, Parameters, Memory and Throughput

The proposed activation functions introduce a negligible number of additional parameters. For example, Hermite activations of degree $d = 3$ add only 72 parameters to ConvNeXt-Tiny (28M total), corresponding to 0.0002%, with similarly minimal overheads for Tropical and Fourier activations. Hermite activations leverage a recursive formulation (Alg. 3) that reduces both FLOP and required

FLOPS, Parameters, Memory and Throughput

The proposed activation functions introduce a negligible number of additional parameters. For example, Hermite activations of degree $d = 3$ add only 72 parameters to ConvNeXt-Tiny (28M total), corresponding to 0.0002%, with similarly minimal overheads for Tropical and Fourier activations. Hermite activations leverage a recursive formulation (Alg. 3) that reduces both FLOP and required memory (vRAM) complexity from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$, requiring only simple arithmetic per term. Fourier and Tropical activations also scale linearly with degree ($\mathcal{O}(d)$), as illustrated in Figure 17 and Table 9, measured on CPU. On GPUs, smaller degrees benefit from vectorized computation, leading to reduced runtime and near-constant $\mathcal{O}(1)$ scaling for low degrees (Figure 18, Table 10).

FLOPS, Parameters, Memory and Throughput

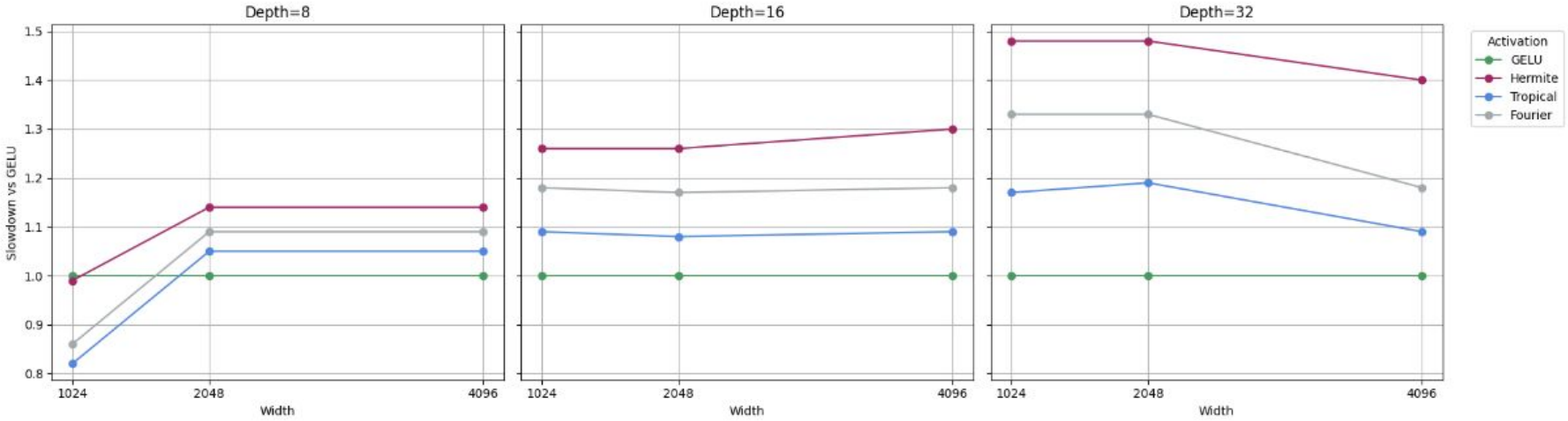


Figure 19: Relative slowdowns of Hermite, Tropical, and Fourier activations compared to GELU across different widths.

FLOPS, Parameters, Memory and Throughput

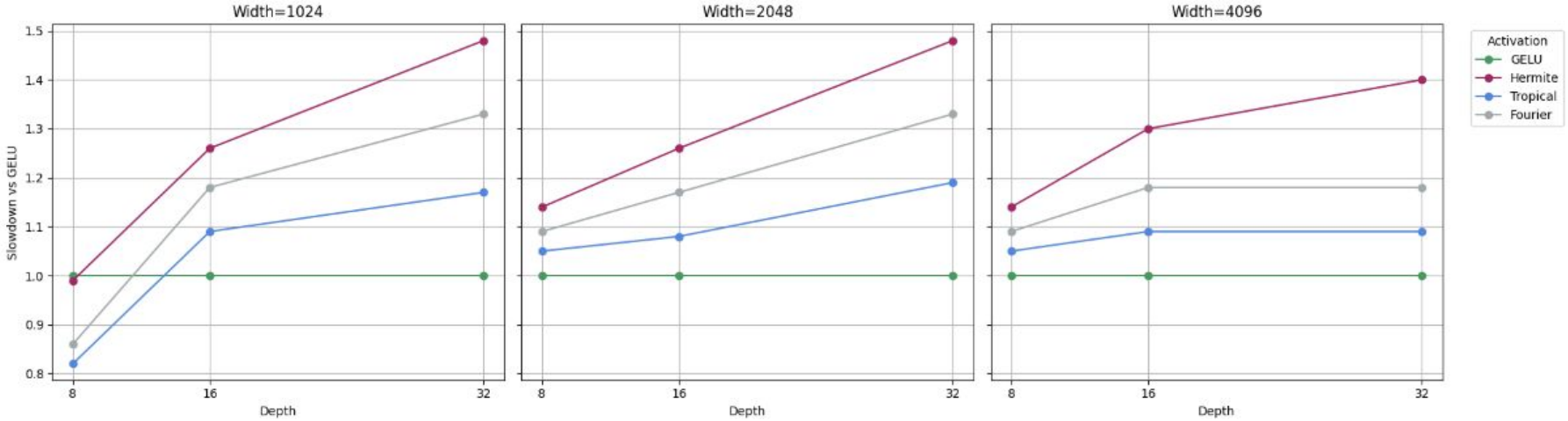


Figure 20: Relative slowdowns of Hermite, Tropical, and Fourier activations compared to GELU across different depths.

FLOPS, Parameters, Memory and Throughput

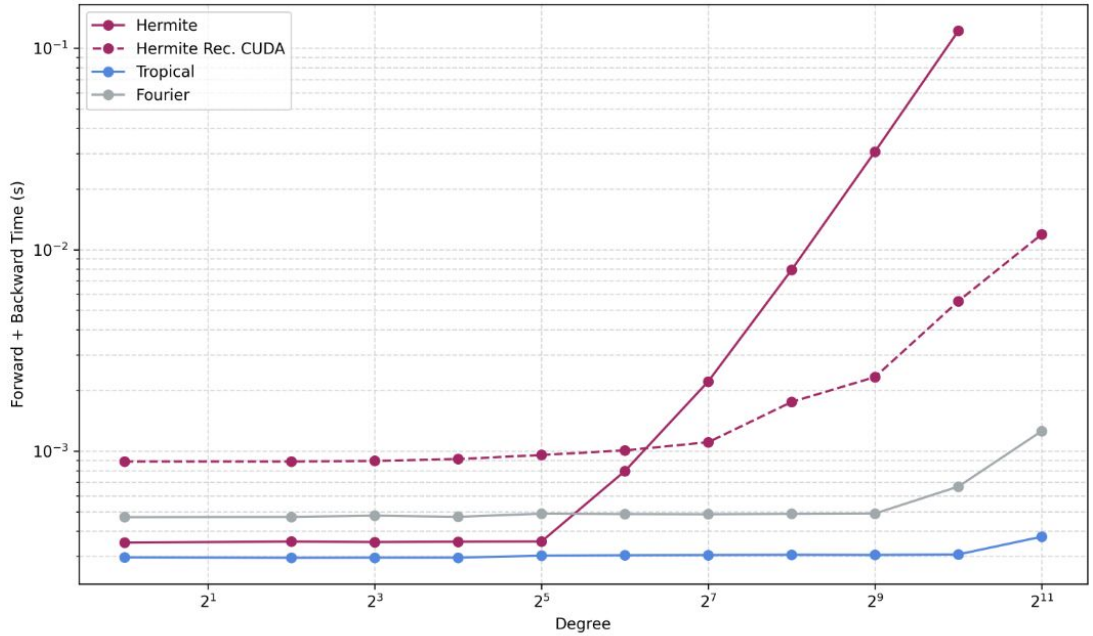


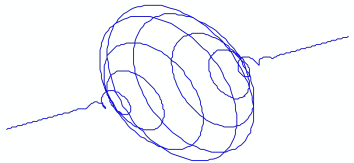
Figure 18: Forward + backward pass averaged times (in seconds) for Hermite (explicit Alg. 2, Eq. 22 and recurrence-based CUDA implementation Alg. 3, Eq. 23), Tropical, and Fourier activations across varying degrees as benchmarked on a single NVIDIA A100 GPU/40GB.

```python
import torch
from torchortho import HermiteActivation

# Define a learnable Hermite activation
degree = 5
activation = HermiteActivation(degree)

# Forward pass
x = torch.rand(7, 4, 3, 2)
y = activation(x)

# Compute gradients
loss = y.sum()
loss.backward()
```

pip install torchortho

Conclusion and Future Directions

- Orthonormal activations (Hermite, Fourier, Tropical) can match or outperform standard ones in large models (e.g., language and vision).
- Proper variance-preserving initialization ensures stable training.
- Networks gain clear mathematical structure and remain practical.
- Other bases (wavelets), rational functions [Yang & Wang, 2024]...
- Towards norm removal.
- MLPs and Attention as a high degree mapping.

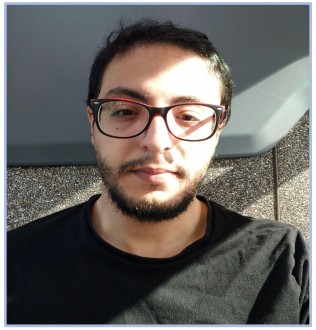


$$\Psi_n(x) = (2n)^{-\frac{n}{2}} c_n \operatorname{He}_n(x) e^{-\frac{1}{2}x^2}$$

ICLR

ICLR

HELMHOLTZ AI Artificial Intelligence Cooperation Unit



Ismail Khalfaoui Hassani
Forschungszentrum Jülich
i.khalfaoui@fz-juelich.de

Stefan Kesselheim
Forschungszentrum Jülich
s.kesselheim@fz-juelich.de

Simulation and Data Laboratory (SDL)
Applied Machine Learning (AML)
Forschungszentrum Jülich (FZJ)

github.com/K-H-Ismail/torchortho
pip install torchortho

Polynomial, trigonometric, and tropical activations
https://openreview.net/forum?id=QywpTFx86x

Thanks!

Any questions?