Comparing Kolmogorov Arnold Networks and Multilayer Perceptrons for Learning the Reverse Process in Diffusion Models

CS431.Q12

N. H. Vinh[1] M. T. Tuong[1]

[1]Department of Computer Science
VNU-UIT

Table of Contents

Table of Contents

Multilayer Perceptron

Definition: Multilayer Perceptron (MLP)

A **Multilayer Perceptron** is a feedforward neural network defined as:

$$\mathrm{MLP}(\mathbf{x}) = (\sigma_L \circ \mathbf{W}_L \circ \cdots \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$$

where:

- $\mathbf{x} \in \mathbb{R}^d$ is the input vector.
- $\mathbf{W}_\ell$ is the weight matrix of layer ℓ.
- σ_ℓ is the activation function of layer ℓ.
- L is the total number of layers.

Table of Contents

Kolmogorov-Arnold Networks

Kolmogorov-Arnold Representation Theorem

If f is a multivariate continuous function on a bounded domain, then f can be written as a finite composition of continuous functions of a single variable and the binary operation of addition:

$$f(\mathbf{x}) = f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \Phi_q\left(\sum_{p=1}^{n} \phi_{q,p}(x_p)\right).$$

Kolmogorov-Arnold Networks

Definition: Kolmogorov-Arnold (KAN)

A **Kolmogorov-Arnold** is a neural network based on Kolmogorov-Arnold Representation Theorem defined as:

$$\text{KAN}(\mathbf{x}) = (\Phi_L \circ \cdots \circ \Phi_1)(\mathbf{x})$$

Kolmogorov-Arnold Networks

Definition: Kolmogorov-Arnold (KAN)

where:

- L is the total number of layers.
- $$\Phi_i = \begin{pmatrix} \phi_{l,1,1}(\cdot) & \phi_{l,1,2}(\cdot) & \cdots & \phi_{l,1,n_l}(\cdot) \\ \phi_{l,2,1}(\cdot) & \phi_{l,2,2}(\cdot) & \cdots & \phi_{l,2,n_l}(\cdot) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{l,n_l+1,1}(\cdot) & \phi_{l,n_l+1,2}(\cdot) & \cdots & \phi_{l,n_l+1,n_l}(\cdot) \end{pmatrix}$$

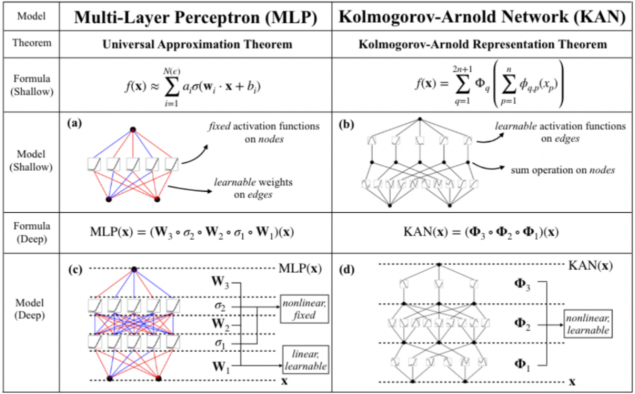
Kolmogorov-Arnold Networks

Definition: Kolmogorov-Arnold Network (KAN)

- $\phi_{l,i,j}(x) = w_b\, b(x) + w_s\, \text{Spline}(x)$
- $\text{Spline}(x) = \sum_i c_i B_{i,k}(x)$
- Cox-De Boor Formula to calculate Spline:

$$\begin{cases} B_{i,0}(x) = \begin{cases} 1 & \text{if } x \in [t_i, t_{i+1}) \\ 0 & \text{otherwise} \end{cases} \\ B_{i,p}(x) = \frac{x - t_i}{t_{i+p} - t_i} B_{i,p-1}(x) + \frac{t_{i+p+1} - x}{t_{i+p+1} - t_{i+1}} B_{i+1,p-1}(x) \end{cases}$$

Multilayer Perceptron vs Kolmogorov-Arnold Network



Model	Multi-Layer Perceptron (MLP)	Kolmogorov-Arnold Network (KAN)
Theorem	Universal Approximation Theorem	Kolmogorov-Arnold Representation Theorem
Formula (Shallow)	$f(\mathbf{x}) \approx \sum_{i=1}^{N(\epsilon)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$	$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$
Model (Shallow)	(a) *fixed* activation functions on *nodes*; *learnable* weights on *edges*	(b) *learnable* activation functions on *edges*; sum operation on *nodes*
Formula (Deep)	$\mathrm{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$	$\mathrm{KAN}(\mathbf{x}) = (\mathbf{\Phi}_3 \circ \mathbf{\Phi}_2 \circ \mathbf{\Phi}_1)(\mathbf{x})$
Model (Deep)	(c) MLP(x); $\mathbf{W}_3$; σ_2; $\mathbf{W}_2$; σ_1; $\mathbf{W}_1$; x; *nonlinear, fixed*; *linear, learnable*	(d) KAN(x); $\mathbf{\Phi}_3$; $\mathbf{\Phi}_2$; $\mathbf{\Phi}_1$; x; *nonlinear, learnable*

Figure: MLP vs KAN

Table of Contents

Denoising Diffusion Probabilistic Models

Forward Process

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

where:

- $\beta_1, ..., \beta_T$ is variance shedule

Denoising Diffusion Probabilistic Models

Reverse Process

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$$

We can implement reparameterization trick to make the model computationally tractable.

Denoising Diffusion Implicit Models

Forward Process

Consider a family of $\mathcal{Q}$ of distributions, indexed by a real vector $\sigma \in \mathbb{R}^T$:

$$q_\sigma(\mathbf{x}_{1:T}|\mathbf{x}_0) = q_\sigma(\mathbf{x}_T|\mathbf{x}_0)\prod_{t=2}^{T} q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$$

where:

- $q_\sigma(\mathbf{x}_T|\mathbf{x}_0) = \mathcal{N}(\sqrt{\alpha_T}\mathbf{x}_0, (1-\alpha_T)\mathbf{I})$
- $q_\sigma(x_{t-1} \mid x_t, x_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\, x_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \frac{x_t - \sqrt{\alpha_t} x_0}{\sqrt{1-\alpha_t}},\ \sigma_t^2 I\right)$

Denoising Diffusion Implicit Models

Reverse Process

We define the generative process:

$$p_\theta^{(t)}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \begin{cases} \mathcal{N}(f_\theta^{(1)}(\mathbf{x}_1), \sigma_1^2 \mathbf{I}) & \text{if } t = 1 \\ q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, f_\theta^{(t)}(\mathbf{x}_t)) & \text{otherwise,} \end{cases}$$

with $f_\theta^{(t)} = (\mathbf{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta^{(t)}(\mathbf{x}_t))/\sqrt{\alpha_t}$

DDIM vs DDPM

Intuitively, the forward process will look like this:

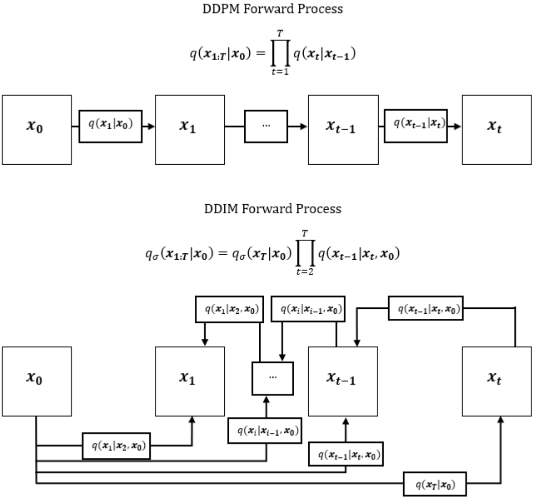


Table of Contents

Introduction

- To compare the effects of MLPs and KANs in diffusion models, we use a naive implementation in which every Dense and Convolution layer is replaced by a KAN layer and a Convolutional KAN layer, respectively.

Configuration

- We train the model for **78,000 steps** with a batch size of **32**.
- We use the **DDIM** model for accelerated generation.
- We adopt a **quadratic timestep subsequence** for the reverse process.
- Training and sampling are performed on an **RTX 4090 GPU**.
- The architecture follows the original DDIM design but with **reduced depth** due to limited computational resources.
- For the diffusion KAN-based model, we set the B-spline degree to **2** due to memory constraints, even though this may significantly reduce model effectiveness.

Datasets

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

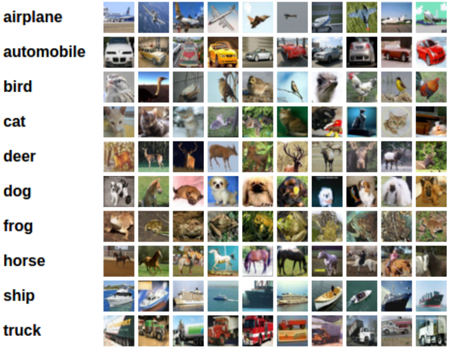


Figure: Cifar Dataset

Metrics

Inception Score (IS)

The Inception Score measures both the **quality** and **diversity** of images generated by a model.

- **Mathematical definition:**

$$\text{IS} = \exp\left(\mathbb{E}_x\, D_{\text{KL}}\big(p(y|\mathbf{x}) \,||\, p(y)\big)\right) \approx \exp\left(\frac{1}{N}\sum_{i=1}^{N} D_{\mathrm{KL}}\big(p(y|\mathbf{x}_i) \,\|\, p(y)\big)\right)$$

  where $p(y|x)$ is the predicted label distribution for image x, and $p(y)$ is the marginal distribution over all generated images.
- **Interpretation:** Higher IS indicates better quality and diversity.

Metrics

Frchet Inception Distance (FID)

The FID measures the **distance between the distributions** of real and generated images in the feature space of a pretrained Inception network.

- **Mathematical definition:**

$$\text{FID} = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}\Big(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\Big)$$

  where μ_r, Σ_r are the mean and covariance of real images' features, and μ_g, Σ_g for generated images.
- **Interpretation:** Lower FID indicates generated images are closer to real images in feature distribution.

Results

Metric	MLP	KAN
Inception Score (IS)	5.7340 ± 0.1215	1.2647 ± 0.0057
Frchet Inception Distance (FID)	34.7710	498.4735

Table: Comparison between MLP-based and KAN-based diffusion models.

Results

Metric	MLP	KAN
Parameter	$23,220,867$	$132,531,642$
Time for 1563 steps	$0m41s$	$8m33s$

Table: Comparison between MLP-based and KAN-based diffusion models.

Limitations

- Unable to reproduce the original results from the paper due to limited computational resources.
- B-Spline is implemented only up to degree 2, which is lower than the standard for most KAN layers.
- Only a naive implementation was performed; further research and optimization may be needed.

Thank You!

Thank you for listening!