An overview of MLP Alternatives to Transformer in Computer Vision

Her-Hsuan Lin
Department of Computer Science
Rice University
Houston, TX 77005
hl162@rice.edu

Sheena Bai
Department of Computer Science
Rice University
Houston, TX 77005
xb5@rice.edu

Hsiu-Chieh Lee
Department of Computer Science
Rice University
Houston, TX 77005
hl161@rice.edu

Abstract

Convolutional Neural Network (CNN) models have been the standard model for computer vision tasks such as image classification for a long period. In recent year, attention-based models including Vision Transformer (ViT) have becomes the new state-of-the-art model. While many researchers focus on improving the performance of those models, some research show that we can obtain a comparable result with Multilayer Perceptron (MLP) based models which is much simpler in structure. MLP models completetly replaces the attention based mechanism in attention-based models with multilayer perception (MLP). In this paper, we explore MLP-like models including MLP-Mixer, ResMLP, gMLP, and a generalized framework called Metaformer and compare these different architectures with Vision Transformer. We also tried to explore the structure of these models to gain some insights into how their structure affects performance.

1 Introduction

Convolutional Neural Network (CNN) models have been the standard for computer vision tasks for many years until attention based model such as Vision Transformer (ViT) models were developed and achieved the state-of-the-art result. While CNNs and attention based models are still leading in this field, in this paper, we explore multilayer Perceptron(MLP) based model, a type of model that achieved similar result with much simpler architectures, and compare them with attention based model.

Unlike CNNs or transformer, MLP models use neither convolutional neural network nor self-attention layers, two key components traditionally associated with the success of CNN and transformer models. In our paper, we mainly explore several Multilayer Perceptron(MLP) based models including MLP-Mixer, ResMLP, gMLP and compare them with vision transformer (ViT), a kind of attention based models. Then, we introduce metaformer, a general framework that proposes an architecture for image classification task. Metaformer can be viewed as a superset of both attention-based models and MLP-alternative models. MLP-Mixer(1) is the first paper that introduces a MLP alternative model and it replaces multihead self-attention with a MLP layer. It's highly inspired by vision transformer(ViT). ResMLP (2) is introduced after MLP-Mixer, and in addition to replacing attention

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

layers with MLP layers, it also replaces layer normalization with affine transformation. gMLP (3) introduces spatial gating unit to the model to extract information from cross-token interactions. Metaformer (4) summarizes the general architecture of transformer and MLP-like models and argues that all these models are similar in general structure, but use different mechanism to mix tokens. The metaformer paper argues the general architecture, instead of the specific design of token mixture layers such as attention and MLP, contributed more to the success of these models.

In this paper, we aim to provide a comprehensive overview of these models mentioned above, focusing on comparing and contrasting their structure and explore connection between their structure and performance. We also hope to compare attention-based models and MLP alternatives to transformer. We focus on accessing the performance of these models in image classification task. Our work is important as it give insights into new model designs aside from well-developed CNNs and transformers.

2 Model

In this section, we delve into the architecture of the Vision Transformer (ViT) and several Multilayer Perceptron based alternatives, including MLP-Mixer, ResMLP, gMLP. We also explore Metaformer, a general framework which abstracts away certain parts of the model. We aim to compare and contrast the structures of these models to gain some understanding of how various parts may influence performance. We will focus mainly on comparing the token mixture layer and the channel mixture layer of each model. Token mixture enables communication and interaction across pixels and channel mixture enables communication and interaction across channels.

2.1 Vision Transformer (ViT)

This section introduces the general structure of the vision transformer (ViT).

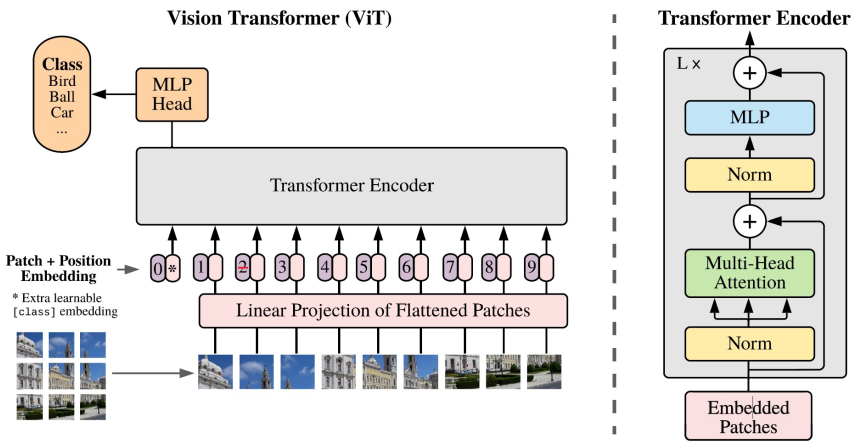


Figure 1: An overview of ViT structure.

As shown in Figure 1, the transformer encoder consists of L alternating layers of multiheaded self-attention and MLP blocks. Layernorm is applied before every attention/MLP blocks and residual connection after every attention/MLP blocks(5). The layers can be written using the following equation:

$$z'_\ell = \mathrm{MSA}(\mathrm{LN}(z'_{\ell-1})) + z'_{\ell-1}, \qquad \ell = 1, \ldots \tag{1}$$

$$z_\ell = \mathrm{MLP}(\mathrm{LN}(z'_\ell)) + z'_\ell, \qquad \ell = 1, \ldots \tag{2}$$

To drive deeper into the model, it's essential to understand the mechanism behind multihead attention. Self-attention is an important mechanism in various neural network architectures including transformers. Self attention computes a weighted sum of all values in the sequence for each element in an input sequence $z \in \mathbb{R}^{N \times D}$. These weights, denoted by A_{ij}, are derived from the pairwise similarity metrics between query q_i and key k_j for any two elements in the sequence. The following equations represent the self attention mechanism(5).

$$[q; k; v] = zU_{\text{qkv}}, \qquad U_{\text{qkv}} \in \mathbb{R}^{D \times 3D_h} \tag{3}$$

$$A = \text{softmax}\left(\frac{qk^T}{\sqrt{D_h}}\right), \qquad A \in \mathbb{R}^{N \times N} \tag{4}$$

$$\text{SA}(z) = Av \tag{5}$$

Multihead Self-Attention runs k parallel self-attention processes, known as 'heads.' The outputs of these heads are concatenated and projected. The following equations represent the multihead self attention mechanism(5). $\frac{D}{k}$.

$$\text{MSA}(z) = [SA_1(z); SA_2(z); \ldots; SA_k(z)]U_{\text{msa}}, \qquad U_{\text{msa}} \in \mathbb{R}^{k \times D_h \times D} \tag{6}$$

In this model, multiheaded self-attention serves primarily as a token mixture as it computes attention scores based on relationships between tokens and mix information from different tokens using these computed attentions. The MLP layer serves as channel mixture.

2.2 MLP-Mixer

This section introduces general structure of the mixer layer of a transformer alternative model, MLP-Mixer. MLP-Mixer consists of N mixer layer and each mixer Layer consists of two MLP blocks,

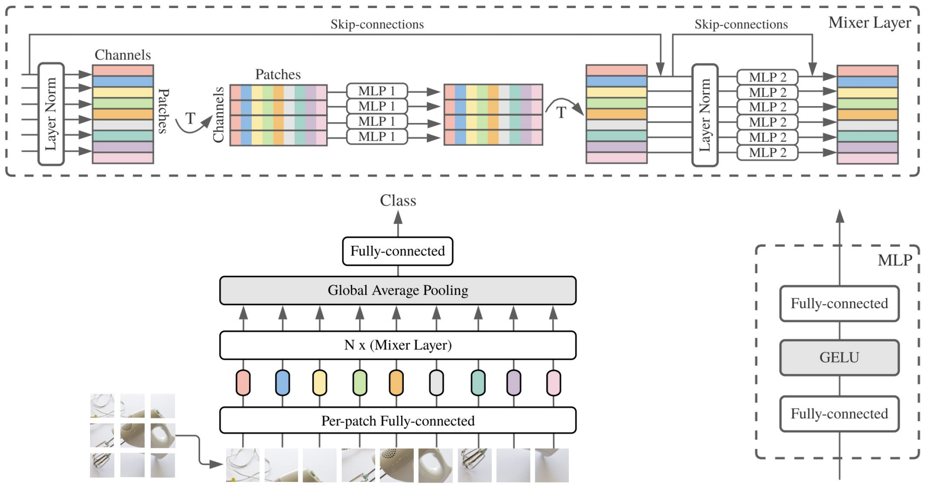


Figure 2: General structure of MLP-Mixer.

as shown in figure 2. The first one is token-mixing MLP and the second one is a channel-mixing MLP. Each MLP have two fully-connected layers and a nonlinearity applied to each row of the input data tensor. Residual connections are applied similar to that in vision transformer. The equations representing the mixer layers are shown below where input $X \in \mathbb{R}^{S \times C}$. Here σ is an element-wise nonlinearity GELU and where W_1, W_2,W_3 and, W_4 are the main learnable weight matrices of the layer.

$$U_{\ell,i} = X_{\ell,i} + W_2\sigma\left(W_1\text{LayerNorm}(X)_{\ell,i}\right), \qquad \text{for } i = 1, \ldots, C \tag{7}$$

$$Y_{j,\ell} = U_{j,\ell} + W_4\sigma\left(W_3\text{LayerNorm}(U)_{j,\ell}\right), \qquad \text{for } j = 1, \ldots, S \tag{8}$$

Compared to vision transformer, MLP-Mixer uses MLP for token-mixing, which is significantly simpler compared to multihead attention. While the cost of computing multihead attention is quadratic with respect to tokens because every token computes attention scores with every other token, the cost of MLP is linear with respect to tokens.

2.3 ResMLP

This section introduces the general structure of another transformer alternative model, ResMLP.

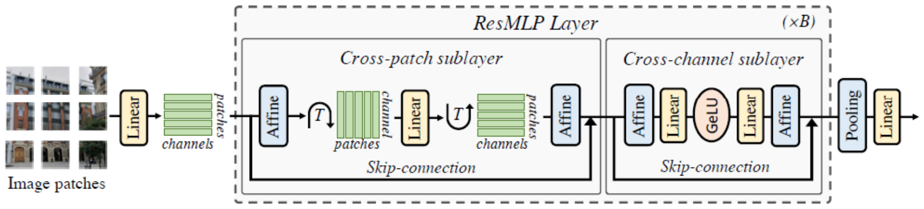


Figure 3: General structure for ResMLP.

Similar to MLP-Mixer, ResMLP also used MLP layers for both token mixing and channel mixing. The way it uses residual connection is similar to both ViT and MLP- Mixer. As shown in figure 3, the first affine-linear-affine section is a token mixer and the affine-linear-Gelu-Linear-Affine section is a channel mixer. The equations representing the ResMLP layer is shown below, where A, B and C are the main learnable weight matrices of the layer and σ is an element-wise nonlinearity GELU.

$$Z = X + \text{Aff}\left((\mathbf{A}\text{Aff}(X))^T\right) \tag{9}$$

$$Y = Z + \text{Aff}\left(\mathbf{C}\sigma\left(\mathbf{B}\text{Aff}(Z)\right)\right) \tag{10}$$

Compared to MLP-Mixer, ResMLP replaces layer normalization with simpler affine transformation. This operation only rescales and shifts the input element-wise and is superior to other normalization techniques because it has no cost at inference time and it does not depend on batch statistics.

2.4 gMLP

This section introduces the general structure of another transformer alternative model, gMLP.

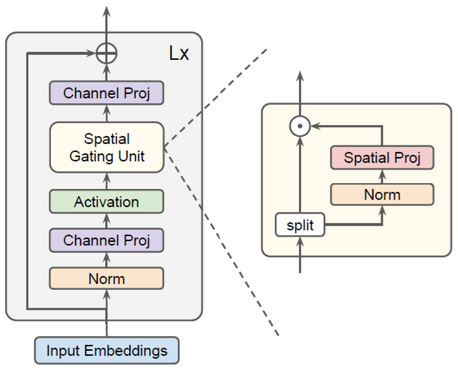


Figure 4: General structure of gMLP.

As shown in in figure 4, gMLP consists of L blocks. Each block is defined as:

$$Z = \sigma(XU) \tag{11}$$

$$\tilde{Z} = s(Z) \tag{12}$$

$$Y = \tilde{Z}V \tag{13}$$

where σ is an activation function such as GeLU . U and V define linear projections along the channel as in feed forward layers in transformers which include a normalization and a MLP layer. s represents a technique called spatial gating unit to enable cross-token interaction and is represented by the formula below.

$$s(Z) = Z_1 \oplus f_{W,b}(Z_2) \tag{14}$$

$\oplus$ denotes element-wise multiplication. $f_{W,b}(Z)$ is a linear projection represented by the formula below.

$$f_{W,b}(Z) = WZ + b \tag{15}$$

The authors found it is effective to split Z into two independent parts (Z1, Z2) along the channel dimension for the gating function and for the multiplicative bypass(3).

SGUs offer an alternative to capture high-order relationships besides self-attention. Specifically, SGUs contains up to 2nd-order interactions (e.g., $z_i z_j$) whereas output for self-attention (assuming no nonlinearity) contains up to 3rd-order interactions (e.g., $q_i k_j v_k$). Both SGUs and self attention have computation cost which is linear over the input channel size and quadratic over the token size(3).

2.5 Metaformer

This section introduces the structure of a framework called mataformer, which consists of a general structure consisting of token mixer and channel mixer, with normalization layers and residual connection as shwon in figure 5. The framework generalizes the structure of transformers and

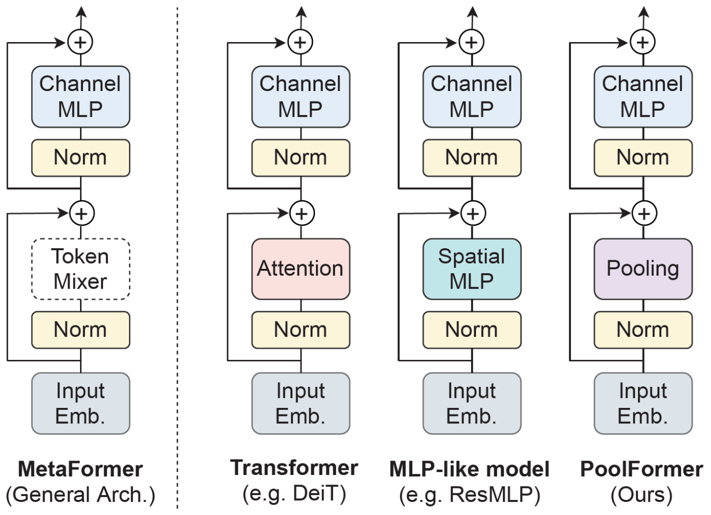


Figure 5: General framework for metaformer

MLP-like model described in the above sections and claims attributes the general structure, instead of the design of token mixer layer, to a significant portion of the success of these models. The author also proposes poolformer, which uses simple pooling as token-mixing technique and achieves good result. The pooling operator can be expressed with an input image T where K is the pooling size.

$$T'_{i,j} = \frac{1}{K \times K} \sum_{p,q=1}^{K} T_{i+p-\frac{K+1}{2},\, j+q-\frac{K+1}{2}} - T_{i,j} \tag{16}$$

Compared to other methods, In contrast, pooling's computational complexity is linear to the sequence length and it doesn't need without any learnable parameters, so it should be more time efficient (4).

3 Performance Comparison

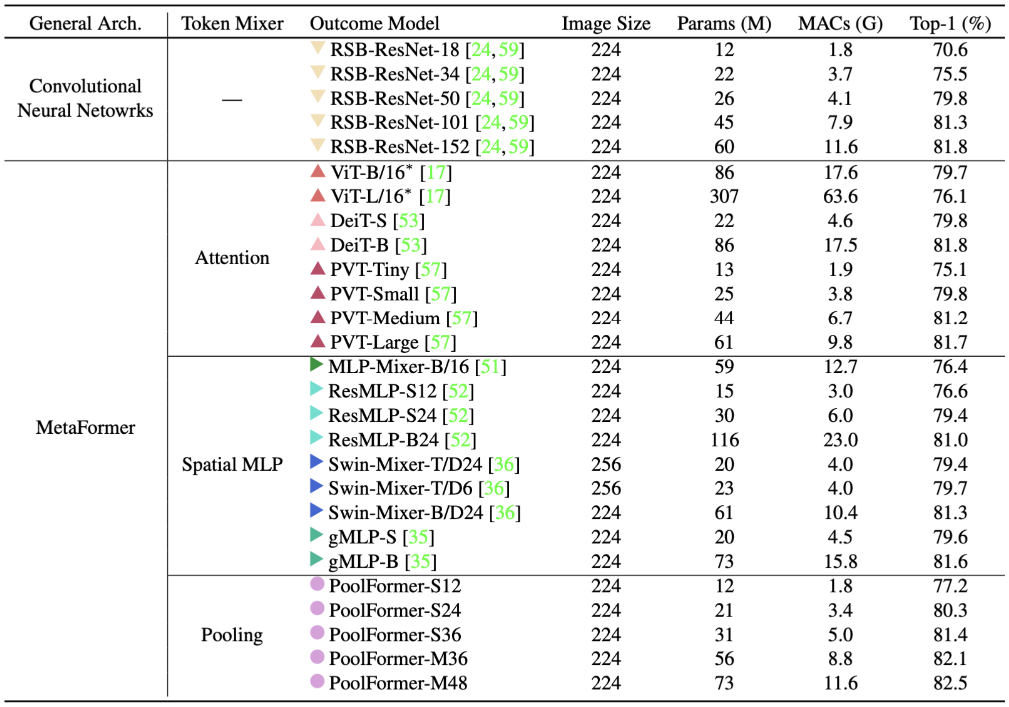
3.1 Datasets

In computer vision, when comparing the performance of image classification, a widely utilized method is evaluating results using common datasets. Not all of the pretrained datasets on the models covered in this review are the same. However, they do share some common ones such as ImageNet (6), which helps us evaluate and compare the performance of models. ImageNet is used as the base dataset for evaluating these models, and bigger datasets like ImageNet-21k or JFT-300M are used to analyze the scalability and verify the accuracy growth compared to the pre-training size.

Brief introduction of the mentioned datasets: ImageNet is a dataset with 1,000 classes and 1.3 million images. ImageNet-21k is the superset of ImageNet with the classes increased to 21,000 and images increased to 14 million. JFT-300M is a dataset with 18,000 classes and 303 million images. JFT-3B is the superset of JFT-300M with the classes increased to 30,000 and images increased to 3 billion.

3.2 Metrics

We use three important shared metrics for comparing the performance across different models: pre-training data size, pre-training computation cost, and accuracy. This ensure we get a holistic review of performance.



General Arch.	Token Mixer	Outcome Model	Image Size	Params (M)	MACs (G)	Top-1 (%)
Convolutional Neural Netowrks	—	▼ RSB-ResNet-18 [24,59]	224	12	1.8	70.6
		▼ RSB-ResNet-34 [24,59]	224	22	3.7	75.5
		▼ RSB-ResNet-50 [24,59]	224	26	4.1	79.8
		▼ RSB-ResNet-101 [24,59]	224	45	7.9	81.3
		▼ RSB-ResNet-152 [24,59]	224	60	11.6	81.8
MetaFormer	Attention	▲ ViT-B/16* [17]	224	86	17.6	79.7
		▲ ViT-L/16* [17]	224	307	63.6	76.1
		▲ DeiT-S [53]	224	22	4.6	79.8
		▲ DeiT-B [53]	224	86	17.5	81.8
		▲ PVT-Tiny [57]	224	13	1.9	75.1
		▲ PVT-Small [57]	224	25	3.8	79.8
		▲ PVT-Medium [57]	224	44	6.7	81.2
		▲ PVT-Large [57]	224	61	9.8	81.7
	Spatial MLP	▶ MLP-Mixer-B/16 [51]	224	59	12.7	76.4
		▶ ResMLP-S12 [52]	224	15	3.0	76.6
		▶ ResMLP-S24 [52]	224	30	6.0	79.4
		▶ ResMLP-B24 [52]	224	116	23.0	81.0
		▶ Swin-Mixer-T/D24 [36]	256	20	4.0	79.4
		▶ Swin-Mixer-T/D6 [36]	256	23	4.0	79.7
		▶ Swin-Mixer-B/D24 [36]	224	61	10.4	81.3
		▶ gMLP-S [35]	224	20	4.5	79.6
		▶ gMLP-B [35]	224	73	15.8	81.6
	Pooling	● PoolFormer-S12	224	12	1.8	77.2
		● PoolFormer-S24	224	21	3.4	80.3
		● PoolFormer-S36	224	31	5.0	81.4
		● PoolFormer-M36	224	56	8.8	82.1
		● PoolFormer-M48	224	73	11.6	82.5

Figure 6: Model performance on ImageNet classification. (image source (4))

We first compare the accuracy of each of the base models that are pretrained on ImageNet. We used ResNet, a classical CNN model, as a reference for comparison. Figure 6 is from the metaformer paper and displays four columns for each model. The first column shows the image size, indicating the input size of each model. The second column shows the parameter count in millions. The third column shows the Multiply-Accumulate Operations (MAC) in billions, a commonly used metric for calculating computational complexity. A model with a higher MAC count generally requires more computational resources to operate. Finally, the fourth column shows the top-1 accuracy, which is the correct ratio of the most confident classification. As depicted in the Figure 6, ViT demonstrates a decent top-1 accuracy but requires slightly more parameters and computational resources during training. Similarly, MLP-Mixer exhibits a slightly lower performance compared to ViT but also requires less parameters and computation. ResMLP achieve similar accuracy while requiring significantly fewer computations and parameters. gMLP performances slightly better than ViT and uses slightly fewer resources. Notably, PoolFormer achieves better accuracy with significantly less computational load compared to ViT. The difference in computation cost is consistent with the time complexity analysis of each model compared to transformer made in the model section. The performance of the model in terms of top-1 accuracy is decent in general, which may indicate that the general structure of metaformer is a good design. The MLP alternatives demonstrated comparable and sometimes even slightly better performance compared to ViT, demonstrating that MLP have the potential to substitute attention as a token-mixer.

In order to better understand the performance difference between models, we provide visualization to compare the models performance against their model size and MACs respectively.

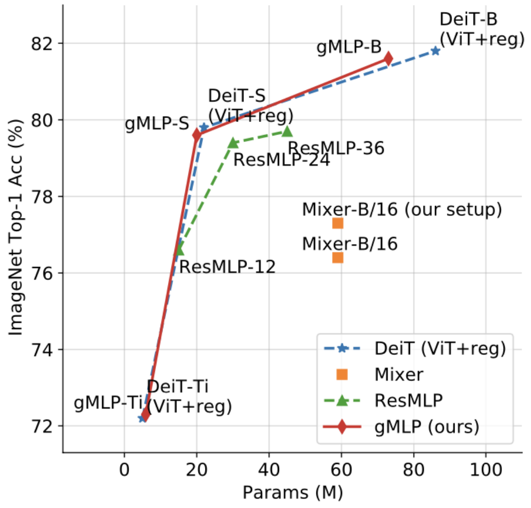


Figure 7: The accuracy of ViT and MLP-based models with different parameter numbers. (image source (3))

Figure 7 shows the parameters versus top-1 accuracy for MLP-based models compared to ViT provided in the gMLP paper. The general trend for all of the models is the same; as number of parameters increases, accuracy increases. However, the accuracy are different across models. As shown in the figure, MLP Mixer and ResMLP perform worse than ViT, and gMLP slightly outperformed ViT when the parameter number is larger than around 50M.

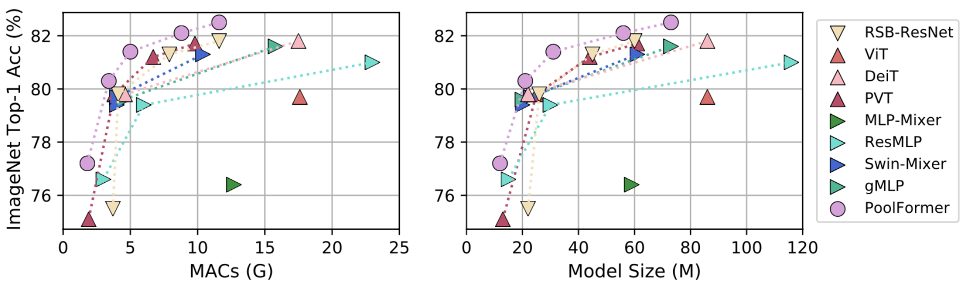


Figure 8: The accuracy of metaformer structured models with different pre-training cost and parameter numbers. The cost is measured in Multiply-Accumulate Operations (MACs). (image source (4))

Figure 8 shows the performance of models against MACs and model size respectively provided in the metaformer paper. Generally, the curves for the MLP models are above that of ViT, meaning that the MLP models achieved a better performance given a fix MACs and model size. We can also observe that PoolFormer model yields better performance than the attention-based or MLP-based models both when compared on same MACs basis and model size basis, meaning poolformer uses resources the most efficiently out of all the models.

The general trend of parameter/model size versus accuracy between the graph provided in the metaformer paper and gMLP are similar, with some slight difference in the relative performance between MLP models and ViT. This is possible due to the difference in implementation.

4 Conclusion and Evaluation

By reviewing and comparing MLP-Mixer, ResMLP, gMLP, and the concept of Metaformer with Vision Transformer, we can see that CNNs and attention based models are not the only few options for computer vision tasks. In fact, MLP-like structures and models that follow the architecture of Metaformer attain comparable results with lower cost compared to ViT in general as shown in the graphs.

By evaluating the structure of the respective models, we compared the relative time complexity of performing the token mixture step of MLP models and metaformer as compared to ViT. The MACs values in the result generally agrees with our analysis of our model. All these models achieve decent performance, indicating that the general metaformer structure may be important in the performance of the model. However, other factors aside from the structure might also play a role which can be an interesting topic for future exploration.

However, one limitation of our paper is that since we don't have the resources to train large models, we have to rely on the results in the original paper of these models, which might vary across implementation and be biased toward their own model.

Furthermore, the results we obtained are from image classification tasks in computer vision. In future works, it's meaningful to explore the performance of MLP models and attention-based model in other fields such as natural language processing and determine if similar conclusion holds.

References

[1] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy, "Mlp-mixer: An all-mlp architecture for vision," 2021.

[2] H. Touvron, P. Bojanowski, M. Caron, M. Cord, A. El-Nouby, E. Grave, G. Izacard, A. Joulin, G. Synnaeve, J. Verbeek, and H. Jégou, "Resmlp: Feedforward networks for image classification with data-efficient training," 2021.

[3] H. Liu, Z. Dai, D. R. So, and Q. V. Le, "Pay attention to mlps," 2021.

[4] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan, "Metaformer is actually what you need for vision," 2022.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

Learning to Generate from Corrupted Data

Moayed Haji Ali
Rice University
{mh155}@rice.edu

Abstract

Diffusion models have emerged as powerful generative tools for solving complex inverse problems in data-rich fields like imagery and video. However, their reliance on extensive, clean datasets limits their applicability in areas with noisy or scarce data, such as medical imaging. Existing training approaches of diffusion models in corrupted data often depend on unrealistic assumptions about data corruption. This paper introduces a novel methodology for training diffusion models using corrupted data alongside clean data from unrelated domains, to compensate for missing high-frequency details. We analyze the diffusion process, showing that early stages can utilize corrupted data effectively, while later stages require fine-tuning with clean data to restore high-frequency information. By exploring the use of blended datasets, such as combining blurred cat images with clean dog images, this study aims to reduce dependency on conventional strong assumptions and expand the practical use of diffusion models in various domains.

1. Introduction

Diffusion models represent a class of generative models that have recently emerged as a powerful framework for solving a variety of blind and nonlinear inverse problems [2, 3, 15]. The versatility and scalability of these models have made them the foundation of many recent breakthroughs in domains rich in data, such as natural imagery and video content. Training a diffusion model necessitates access to extensive datasets and substantial computational resources. For instance, state-of-the-art text-to-image models [16, 17] require hundreds of millions of images for training to achieve their impressive performance. This limits their usage in data scarce domains, such as medical imaging.

Subsequent research has focused on fine-tuning these foundational models for specialized domains, accomplishing tasks such as medical image analysis or enhancing personalized user experiences in image diffusion. This fine-tuning process still mandates a significant quantity of clean data that accurately reflects the target distribution. However, in many domains such as environmental monitoring. medical imaging, and astrophotography, data frequently exhibit high levels of noise—either inherently, as in environmental data affected by sensor inaccuracies, or by design, such as undersampling in diffusion MRI to reduce data acquisition times. In realistic settings, corruption typically exhibits a range from weak to strong spatial correlation. For instance, undersampling in MRI selectively omits information across specific frequency levels, impacting every pixel and thereby inducing a form of corruption with a global effect on the image. This prevalence of corrupted datasets presents substantial challenges in training generative models effectively for these domains, thereby restricting their applicability. One potential solution involves employing unsupervised denoising methods to preprocess the corrupted data for model training. However, these methods often rely on unrealistic assumptions, such as pixel-wise noise independence [1, 6, 9, 13, 19], necessitate knowledge of the specific corruption process [5], are limited to certain noise types [14], rely on unrealistic assumptions over the corruption process [5, 12], or are incapable of restoring high-frequency details [20]. AmbientDiffusion [5] explored training a diffusion model on corrupted images with missing pixels. To achieve this, they assume access to the corruption matrix of each sample in the training dataset and the distribution of the corruption parameters.

In this work, we focus on corruption processes that happen in the high-frequency domain (e.g blurring, low-resolution, noisy). Given that these corruption processes often eliminates critical, frequently high-frequency, details from the images, there's a need for a prior that accounts for these omissions. While previous efforts generally impose stringent assumptions about the nature of the corruption as a prior, our research explores whether it is feasible to derive a prior from a clean, unrelated data set from another domain. Specifically, we are investigating whether it is possible to train a generative model on blurred cat images by using a combination of *blurred cat* images and *clean dog* images, without imposing any additional assumptions.

To accomplish this, we analyze the generation process of diffusion models across different timesteps, distinguishing between the generation of low-frequency and high-frequency information. Specifically, we show that for high-frequency corruptions (*e.g.* blurring), the forward diffusion process removes the high-frequency information in the early steps, making a blurred and a clean image looks similar for these steps. We then show that these initial stages of diffusion primarily focus on generating low-frequency details, allowing for the effective training of the model using directly corrupted images during these early phases. For the later stages, which involve the generation of high-frequency details, we experiment with various fine-tuning approaches that utilize clean image distributions from a related domain to restore high-frequency information pertinent to the target domain.

2. Problem Formulation

Consider a dataset of corrupted image observations drawn from a distribution $\tilde{p}(y)$, where each observation $y \in \mathbb{R}^{3\times H\times W}$ is derived from a corresponding clean image $x \in \mathbb{R}^{3\times H\times W}$ that is drawn from an underlying clean data distribution $p(x)$. We assume that y results from a linear corruption operation of x.

$$x = Ay + \epsilon, \tag{1}$$

where $A \in \mathbb{R}^{(3\times H\times W)\times(3\times H\times W)}$ represents the linear forward operator, and ϵ denotes the additive noise, which typically follows a normal Gaussian distribution $\epsilon \sim \mathcal{N}(0, I)$.

Our objective is to learn a generative model $G : \mathbb{R}^d \to \mathbb{R}^{3\times H\times W}$, mapping a latent space variable $z \sim \mathcal{N}(0, I)$ to the space of clean images, such that the distribution of images generated by $G(z)$ approximates the clean data distribution $p(x)$ as closely as possible. The problem can be mathematically formulated as follows:

$$\min_G \mathbb{E}_{z\sim\mathcal{N}(0,I_d)} \left[\mathcal{L}\left(G(z), p(x)\right)\right], \tag{2}$$

where $\mathcal{L}$ is a loss function that measures the discrepancy between the distribution of generated images $G(z)$ and the clean data distribution $p(x)$. The goal of this work is to learn G such that it generates images which are indistinguishable from those drawn from the clean distribution $p(x)$, despite the model's access only to the corrupted data from distribution $\tilde{p}(y)$ and clean data from a similar distributing $p(z)$ during training.

3. Background on Diffusion Models

Denoising Diffusion Probabilistic Models [7] are a class of generative models that operates by gradually transforming a data distribution into a simple, typically Gaussian, noise distribution over a series of diffusion steps, and then learn to reverse this process. The forward diffusion process is described by a Markov chain that progressively adds Gaussian noise to the data over T steps, transforming the data distribution $p(\mathbf{x}_0)$ into a noise distribution $p(\mathbf{x}_T)$. This process can be expressed by the conditional distribution $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$, where β_t is a variance schedule that controls the noise level at each step t. Then, a reverse process is learnt to estimate $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ that can recover the original data from the noise. This is achieved by parameterizing the reverse process with a neural network with parameters θ, and training it to approximate the posterior distribution. The training of diffusion models is typically framed as a variational inference problem, where the goal is to minimize the KL divergence between the forward process and the reverse process across all timesteps. This is simplified as minimizing a weighted sum of squared errors between the noisy data at each timestep and the denoised estimates produced by the model:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x}_0,\epsilon,t}\left[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2\right],$$

where $\epsilon_\theta(\mathbf{x}_t, t)$ is the model's estimate of the noise added at step t, and ϵ is the actual noise.

4. Related Work

4.1. Unsupervised denoising methods

Unsupervised denoising methods aim to recover a noise-free data sample y given the noisy measurement x through a mapping f, which is given when optimizing the objective $\mathbb{E}\|f(x)-y\|^2$. Most work carry a strong, and rather unrealistic assumption that the noise is pixel-wise independent. That is $x = y + n$, where n is a noise that can come from various distributions such as Gaussian ($n \sim \mathcal{N}(\mu, I)$), Bernoulli ($n \sim$ Bernoulli(p)), or Poisson ($n \sim$ Poisson(λ)).

Noise2Noise [13] uses two noisy observations $x_1 = y + n_1$ and $x_2 = y + n_2$ to train a denosing network f that optimizes the objective $\mathbb{E}\|z_2 - f(x_1)\|$. The method relies on the noise independence assumption with the intuition that the network learns to reconstruct only the noise-free sample y while discarding the noise n_2 as it cannot be inferred from the image features x or the noise n_1.

Noise2Self and Bline-Spot Networks [1] drops the requirement of two noisy observations by using the image's inherent structure. Specifically, it defines a family of functions as $\mathcal{J} - Invarant$ if $f(x)_j$ does not depend on x_j for any j. In the context of images, it means that *we can* infer the value of a pixel from its neighboring pixels. The authors shows that for such functions $\mathbb{E}\|f(x) - x\|^2 = \mathbb{E}\|f(x) - y\|^2 + \mathbb{E}\|x - y\|^2$. In other words, optimizing $\mathbb{E}\|f(x) - y\|^2$ is equivalent to optimizing $\mathbb{E}\|f(x) - x\|^2$ with respect to f. Based on this observation, they propose to randomly mask pixel from the noisy measurement and predict their values from their surrounding ones. This way, the model can only reconstruct the image features while discarding the noise as the noise is pixel-wise independent. Many followup works [8, 12] improves the efficiency of this

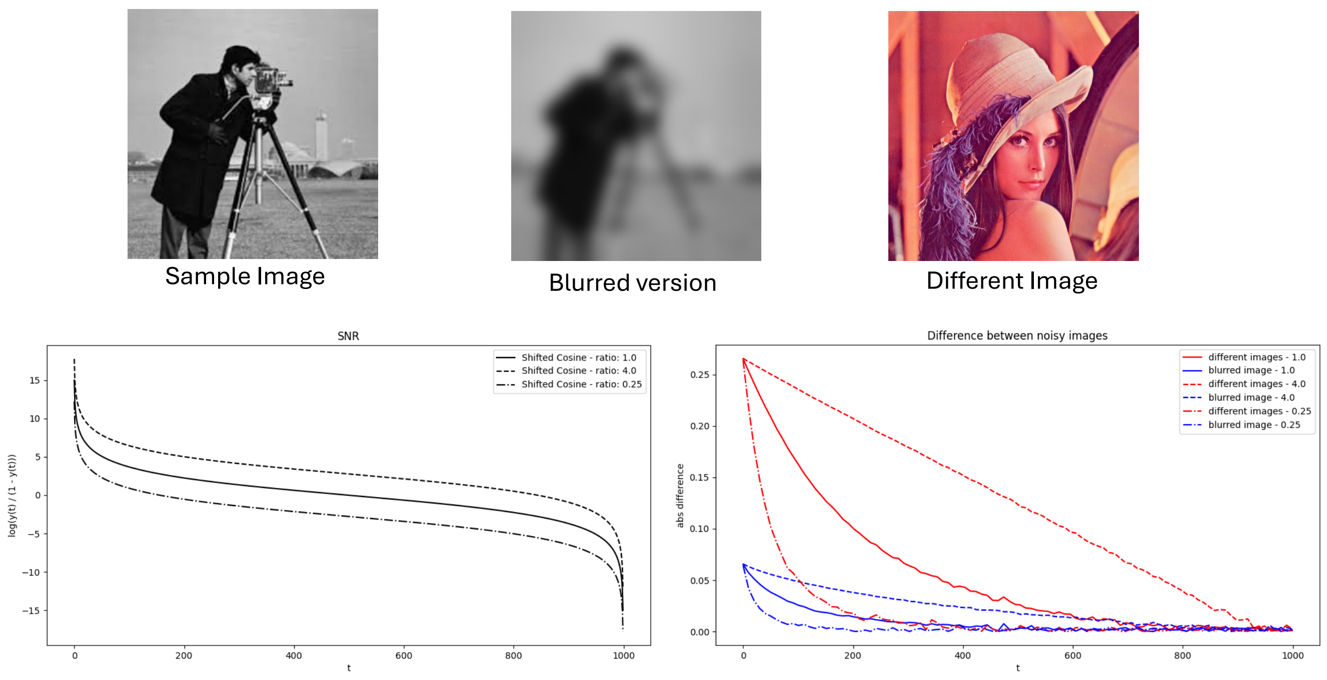


Figure 1. **Difference between corrupted and clean image at various diffusion steps.** we show that for a 11x11 blurred kernel, the difference between the blurred and clean image (in blue) becomes almost zero when the LogPSNR achieves zero (on the left figure) after 200 diffusion steps. This is compared to the difference with a different image (in red).

approach by performing the denoising operation from context pixels in a single forward pass, often using networks that are referred to as Blind-Spot Networks. One disadvantage of these networks is that they don't use the information of the masked pixel, leading to suboptimal results. Laine et al [11] assumes access to the noise distribution to compute analytically the posterior of the clean pixel given the noisy one. That is, the probability of a pixel y_i is given as:

$$p(y_i|x_{j\neq i}) = \int p(y_i|x_i)p(x_i|x_{j\neq i})d_{x_i}$$

Where $p(y_i|x_i)$ can be computed analytically knowing the noise distribution, and $p(x_i|x_{j\neq i})$ is estimated with a blind-spot network.

Neighbor2Neighbor [9] performs two downsampling operations with different kernels on the noisy observation y to obtain two noisy observations y_1 and y_2, then follows the Noise2Noise framework.

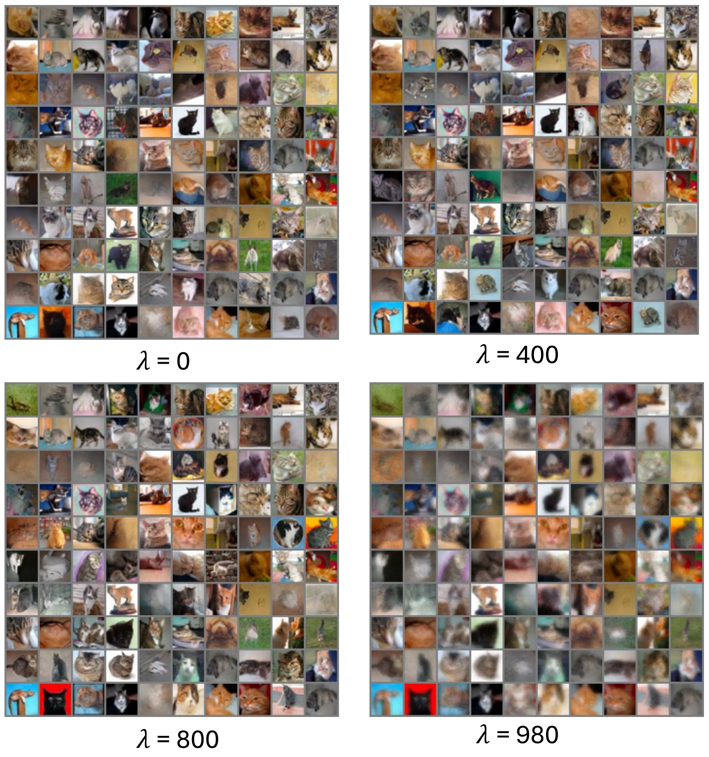
Patch2Self [6] extends the Noise2Self idea to 4D data such as MRI diffusion by exploiting the spatial and temporal redundancy of N over-sampled ($N \geq 20$) 3D volumes. Specifically, they formulate the denoising as prediction a 3D patch in the N_{th} volume from the remaining $N-1$ volumes. By repeating this operation for different volumes, the authors can significantly outperforms previous methods on these types of data.

Noise2Score [10] provides a unified perspective over the previously mentioned Noise2X methods by linking it to score model through the Tweedie's formula. Specifically, for a Gaussian noise distribution, the Tweedie's formula calculates the posterior of the clean sample y given the noisy x as:

$$\mathbb{E}[y|x] = x + \nabla_x \log p(x)$$

Where $p(x)$ is the marginal distribution of x and thus $\nabla_x \log p(x)$ denotes the score function. Thus, we may obtain the denoised image if we have access to a score network. A more general formulation for all exponential family of noise distributions can be obtained such that the posterior only depends on the score function, and the noise distribution parameters. The score can be estimated using a denoising network for very small noise levels, similar to the score-matching perspective of diffusion models [18]. Additionally, the authors draw a connection between their method and the SURE-based method as all of them essentially aims to estimate the score function, with the difference that in the Noise2Score formulation, the score is independent from the noise sampling and thus can be used for blind image denoising problems, whereas SURE-based model requires retraining in case of changing the noise distribution assumption.

DDM2 [19] attempts to improve the performance of Patch2Self by increasing its solution space through learning a diffusion model from the given corrupted data. While the natural naive method is to denoise the corrupted dataset and



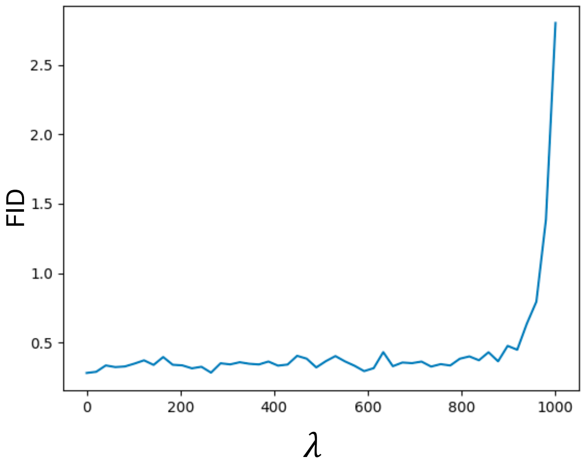


Figure 2. **Training early diffusion steps with blurred images.** On the left, we show generated sample when training early diffusion steps ($t \leq \lambda$) on blurred images. On the right, we show qualitatively how the FID score changes for different values of λ

then train a generative model, the authors shows that this constrains the solution space. Instead, they follow a three-stage approach. In the first stage, they train an improved version of Patch2Self to obtain a denoised version of each corrupted sample in the dataset and estimate its noise level as $\bar{\epsilon} = y - x$. In the second stage, they use the estimated noise to map the noisy sample into a specific timestamp in the diffusion process. Lastly, they train a diffusion model using the estimated noise calibrated with the matched state. They modify the training process of the diffusion model by first pixel-wise shuffle the noise as a way to augment the data. Additionally, they train to reconstruct the noisy samples instead of the denoised ones. This is to prevent the diffusion model from exactly mimicking the deonising network in stage 1. Under the pixel-wise noise independence, reconstructing the noisy same is equivalent to reconstructing the clean sample up to a constant.

Relaxing noise pixel-wise independence. Few follow-up work aimed to adapt previous methods to real noises by relaxing the assumption that the noise is spatially-independent. [20] breaks the noise dependence by performing pixel shuffling downsampling operation. Then uses a blind-spot network for denoising, trained with synthetic data. This assumes that the noise has a short-range correlation compared to the image. However, the pixel shuffling also breaks the image features, introducing aliasing artifacts. Carrying the same assumption, Li et al [14] proposes to process flat and textures regions differently. Specifically, it assumes that pixels in flat regions are correlated to other pixels that are further away. Therefore, it also masks neighboring pixels in blind spot networks so that the pixel noise cannot be inferred from the surrounding ones. The resulting image is then used to supervised the denoising of textured regions using a locally aware network in a fashion similar to Noise2Noise.

4.2. Generative models from corrupted data

SoftDiffusion [4] explores a non-trivial task of training diffusion models with a non-Markovian corruption processes other than Gaussian noises (e.g blurring, masking, etc). Specifically, they are interested in the forward diffusion corruption process that is defined as:

$$x_t = A_t x_0 + s_t \eta_t$$

Where A_t is the linear corruption matrix at timestamp t, s_t is a constant that controls the level of noise at time t, and η_t is the forward Wiener process. To train a generative model on such family of corruption, they redefine the training objective as the error between the *corrupted* version of the network prediction for the clean image $A_t * \epsilon_\theta(x_t)$ and the corrupted image $A_t * x_0$. Additionally, they update the sampling and

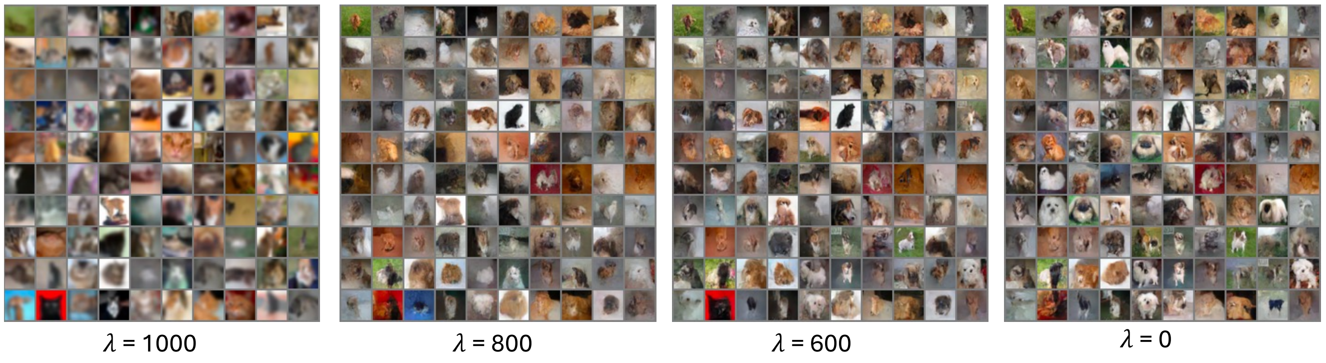


Figure 3. **Fine-tuning later diffusion steps with clean images from untreated domain.** We experiment with training diffusion model on blurred cat images and fine-tune on later steps ($t \leq \lambda$ on clean dog images. We show that compared to the reference blurred model ($\lambda = 1000$), fine-tuning on clean dog images for smaller values of λ decreases the blurriness, yet results in more distorted shapes.

noise scheduling process to accommodate the change in noise distribution.

Ambient Diffusion [5] is the first and only existing work that aims to learn a generative model from corrupted data. Specifically, the target is to learn the condition expectation $\mathbb{E}[x_0|Ax_0 + s_t * \eta_t]$ for all noise levels t given access to only corrupted sample $y_0 = Ax_0$ and use this expectation to recover $\mathbb{E}[x_0|x_t]$ for all t, which would allow us to sample from the clean samples distributions $p(x_0)$.

To make the problem settings easier, the authors assumes having access to the corruption matrix A_i and its distribution $p(A)$ for each sample in the dataset. Then, the authors propose to further corrupt the samples with $\tilde{A}$ that is very close to A and learn to remove this additional corruption. In the example of missing pixels corruption, this introduces few additional missing pixels and learn to recover them through the SoftDiffusion training framework. The authors further assumes that $\mathbb{E}[x_0|\tilde{A}x_0, \tilde{A}]$ is equivalent to $\mathbb{E}[x_0|Ax_0, A]$ when $\tilde{A}$ is close enough to A, and show that under the condition that $\mathbb{E}_{A|\tilde{A}}[A^T A]$ is a full-rank matrix, it is possible to recover $\mathbb{E}[x_0|\tilde{A}x_0 + s_t * \eta_t, \tilde{A}]$, Which estimates the clean sample x_0, given the corrupted sample. To accommodate this change, they also alter the sampling process such that at every diffusion step $t+1$, they predict the noise-free sample x_0, and corrupt it back with the noise level of t. This provides a framework to learn a generative model from corrupted data, yet it comes with huge assumption over the corruption process that is only met in few setting such as missing pixel and not met in a more realistic settings such as blurring. It also assumes access to a privileged information of the exact corruption matrix for each sample, which is unrealistic.

5. Method and Experiments

Training a diffusion model involves a forward diffusion process that removes information from a sample image by adding a high level of noise, and a reverse diffusion process that learn to generate information through predicting the added noise. Adding a small amount of noise, in the early diffusion steps, essentially destroys the high-frequency information. The means for high-frequency corruptions (*e.g.* blurring), both the a corrupted image and a clear version of it will be similar at early diffusion steps (*e.g.* $t \leq \lambda$). We validate this assumption in Fig. 1, where we show that at a low LogPSNR (*e.g.* ≤ 0), the difference between the corrupted and clean image becomes almost zero. This motivates us to distinguish between learning the generation of low-frequency information ($t \leq \lambda$), and high-frequency information ($t \geq \lambda$).

5.1. Stage I: Low-frequency Information

As the corrupted and clean images becomes similar when the high frequency information is removed in early diffusion steps, we propose to train these steps directly on corrupted images, following a standard diffusion model training. To validate this, we train a diffusion model using *blurred cat* images for ($t \leq \lambda$) and *clean cat* images for ($t \geq \lambda$). In Fig. 2, we show that despite training the first 800 steps on blurred images ($\lambda = 800$), the results are indistinguishable from training all diffusion steps in clean images ($\lambda = 0$). To measure this quantitatively, we employ Fréchet Inception Distance metric (FID), which measures the distance between the generated and ground truth distributions (lower is better). We show that the FID score remains similar $\lambda \leq 800$ and only gets exponentially worse when using $\lambda > 800$, thus validating our assumption that we may learn the early diffusion steps directly on the corrupted images.

5.2. Stage II: High-frequency Details

The corruption process predominantly affects the high-frequency information in our target domain (*e.g.*, cats), unlike the low-frequency details. To recover this high-frequency information, we propose imposing a prior using a dataset of clean images from an unrelated domain (*e.g.*, dogs). Although this approach does not replicate the exact distribution of the target domain, our objective is to estimate and approximate a distribution from which we can generate clean images of the target domain. This method focuses on achieving a practical approximation rather than exact replication, aiming to enhance the usability of diffusion models in the presence of corrupted data.

To achieve this, we experiment with a basic baseline of training the diffusion model on corrupted images from the target domain in stage I, and fine-tune the high-frequency steps (*i.e.* $t \geq \lambda$) on the clean images from the unrelated domain. In Fig. 3, we experiment with training a diffusion model on blurred cat images, followed by fine-tuning later diffusion steps $t \geq \lambda$ on clean dog images directly. We show that following this procedure decreased the blurriness significantly, yet it results in distorted shapes. The value of λ offers a trade-off between distortion and perception. A lower value of λ results in less blurred images but more distorted shapes.

6. Conclusion

In this study, we explored the innovative question of training generative models using only corrupted data from the target domain alongside clean data from a different domain. Our experiments demonstrate the feasibility of extracting clean low-frequency details from corrupted data within the diffusion model framework. However, our method falls short in generating high-frequency details. Moving forward, we aim to continue this research by exploring additional fine-tuning methods that can bridge the gap between the clean and corrupted domains, thereby producing less corrupted data without significant distortions.

References

[1] Joshua Batson and Loic Royer. Noise2self: Blind denoising by self-supervision, 2019. 1, 2

[2] Hyungjin Chung, Jeongsol Kim, Sehui Kim, and Jong Chul Ye. Parallel diffusion models of operator and image for blind inverse problems, 2022. 1

[3] Hyungjin Chung, Jeongsol Kim, Michael T. Mccann, Marc L. Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems, 2023. 1

[4] Giannis Daras, Mauricio Delbracio, Hossein Talebi, Alexandros G. Dimakis, and Peyman Milanfar. Soft diffusion: Score matching for general corruptions, 2022. 4

[5] Giannis Daras, Kulin Shah, Yuval Dagan, Aravind Gollakota, Alexandros G. Dimakis, and Adam Klivans. Ambient diffusion: Learning clean distributions from corrupted data, 2023. 1, 5

[6] Shreyas Fadnavis, Joshua Batson, and Eleftherios Garyfallidis. Patch2self: Denoising diffusion mri with self-supervised learning, 2020. 1, 3

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. 2

[8] David Honzatko, Siavash A. Bigdeli, Engin Turetken, and L. Andrea Dunbar. Efficient blind-spot neural network architecture for image denoising. In *2020 7th Swiss Conference on Data Science (SDS)*. IEEE, 2020. 2

[9] Tao Huang, Songjiang Li, Xu Jia, Huchuan Lu, and Jianzhuang Liu. Neighbor2neighbor: Self-supervised denoising from single noisy images, 2021. 1, 3

[10] Kwanyoung Kim and Jong Chul Ye. Noise2score: Tweedie's approach to self-supervised image denoising without clean images, 2021. 3

[11] Samuli Laine, Tero Karras, Jaakko Lehtinen, and Timo Aila. High-quality self-supervised deep image denoising, 2019. 3

[12] Wooseok Lee, Sanghyun Son, and Kyoung Mu Lee. Ap-bsn: Self-supervised denoising for real-world images via asymmetric pd and blind-spot network, 2022. 1, 2

[13] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2noise: Learning image restoration without clean data, 2018. 1, 2

[14] Junyi Li, Zhilu Zhang, Xiaoyu Liu, Chaoyu Feng, Xiaotao Wang, Lei Lei, and Wangmeng Zuo. Spatially adaptive self-supervised learning for real-world image denoising, 2023. 1, 4

[15] Morteza Mardani, Jiaming Song, Jan Kautz, and Arash Vahdat. A variational perspective on solving inverse problems with diffusion models, 2023. 1

[16] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis, 2023. 1

[17] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. 1

[18] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021. 3

[19] Tiange Xiang, Mahmut Yurt, Ali B Syed, Kawin Setsompop, and Akshay Chaudhari. Ddm^2: Self-supervised diffusion mri denoising with generative diffusion models, 2023. 1, 3

[20] Yuqian Zhou, Jianbo Jiao, Haibin Huang, Yang Wang, Jue Wang, Honghui Shi, and Thomas Huang. When awgn-based denoiser meets real noises, 2019. 1, 4

Literature Review: Parameter Efficient Fine-tuning for Large Language Models

Nikhil Chigali, Jeffrey J Sam
Department of Computer Science, Rice University
Houston, TX
`nc71@rice.edu, jj116@rice.edu`

Abstract

Large language models have achieved remarkable success in various natural language processing tasks, but fine-tuning these massive models for each new task is computationally expensive and memory-intensive. This has motivated research into parameter-efficient learning methods that adapt pre-trained models to new tasks with minimal parameter updates. This paper reviews several prominent approaches, including prefix-tuning, adapter methods, intrinsic dimensionality analysis, and low-rank adaptation techniques like Prefix Tuning, Adapters, and LoRA. We discuss their key ideas, advantages, and limitations, providing a comprehensive overview of this rapidly evolving field.

1 Introduction

Large pre-trained language models like GPT-3, PaLM, and LaMDA have revolutionized natural language processing (NLP) by achieving state-of-the-art performance on a wide range of tasks. However, these models have billions of parameters, making it computationally expensive and memory-intensive to fine-tune them for each new task. This has motivated research into parameter-efficient learning methods that can adapt these massive models to new tasks with minimal parameter updates. The key idea behind parameter-efficient learning is to introduce a small number of task-specific parameters that can be optimized during fine-tuning, while keeping the pre-trained model weights frozen. This not only reduces the computational and memory requirements but also enables better transfer learning and faster convergence, especially in low-data regimes.

In this paper, we review several prominent parameter-efficient learning methods for large language models, including prefix-tuning, adapter methods, intrinsic dimensionality analysis, and low-rank adaptation techniques like LoRA. We discuss their key ideas, advantages, and limitations, providing a comprehensive overview of this rapidly evolving field.

2 Problem

Fine-tuning large pre-trained language models for new tasks is computationally expensive and memory-intensive due to the vast number of parameters that need to be updated. This poses significant challenges for deploying these models in resource-constrained environments or for tasks with limited data. Moreover, fine-tuning requires storing a separate copy of the model for each task, leading to substantial storage and maintenance costs, especially for large models. This issue becomes more pronounced as the number of tasks and models increases, making it impractical to fine-tune and maintain multiple models for different tasks. To address these challenges, parameter-efficient learning methods aim to adapt pre-trained models to new tasks with minimal parameter updates, reducing computational and memory requirements while enabling better transfer learning and faster convergence. The key problems that parameter-efficient learning methods aim to solve are:

- **Computational and Memory Efficiency**: Fine-tuning large pre-trained language models requires updating a vast number of parameters, which can be computationally expensive and memory-intensive, especially in resource-constrained environments or for tasks with limited data.
- **Storage and Maintenance Costs**: Fine-tuning a pre-trained model for each new task requires storing a separate copy of the model, leading to substantial storage and maintenance costs, especially as the number of tasks and models increases.
- **Transfer Learning and Convergence**: Fine-tuning can be inefficient in terms of transfer learning and convergence, particularly in low-data regimes, as the model needs to learn task-specific parameters from scratch.

Parameter-efficient learning methods address these problems by introducing techniques that can adapt pre-trained models to new tasks with minimal parameter updates. This reduces the computational and memory requirements, enables better transfer learning and faster convergence, and allows for more efficient storage and maintenance of multiple models for different tasks.

3 Prefix-Tuning: Optimizing Continuous Prompts for Generation

The prefix-tuning method, proposed by Li and Liang (2021) [4], aims to achieve parameter-efficient transfer learning for natural language generation tasks. Instead of updating all the parameters of the pre-trained language model, prefix-tuning optimizes a small continuous task-specific vector called the "prefix".

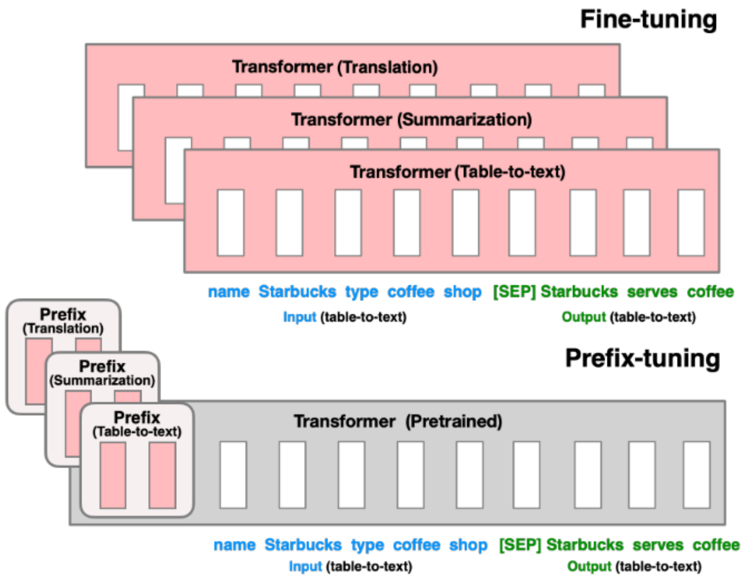

Fine-tuning
Transformer (Translation)
Transformer (Summarization)
Transformer (Table-to-text)
name Starbucks type coffee shop [SEP] Starbucks serves coffee
Input (table-to-text)
Output (table-to-text)
Prefix (Translation)
Prefix (Summarization)
Prefix (Table-to-text)
Prefix-tuning
Transformer (Pretrained)
name Starbucks type coffee shop [SEP] Starbucks serves coffee
Input (table-to-text)
Output (table-to-text)

Figure 1: Fine-tuning (top) updates all Transformer parameters (the red Transformer box) and requires storing a full model copy for each task. We propose prefix-tuning (bottom), which freezes the Transformer parameters and only optimizes the prefix (the red prefix blocks). Consequently, we only need to store the prefix for each task, making prefix-tuning modular and space-efficient. Note that each vertical block denote transformer activations at one time step.

3.1 Optimization Problem

The key idea of prefix-tuning is to learn a continuous prefix vector $\mathbf{p}$ that is prepended to the input sequence $\mathbf{x}$. This allows the language model to attend to the prefix as if it were "virtual tokens", enabling efficient adaptation to the target task.

3.2 Objective Function

The objective of prefix-tuning is to maximize the likelihood of the target sequence $\mathbf{y}$ given the input $\mathbf{x}$ and the prefix $\mathbf{p}$, while keeping the pre-trained model parameters θ frozen:

$$\max_{\mathbf{p}} \log P(\mathbf{y}|\mathbf{x}, \mathbf{p}; \theta) \tag{1}$$

where θ represents the parameters of the pre-trained language model.

3.3 Optimization Algorithm

To optimize the prefix vector $\mathbf{p}$, the authors use gradient descent. The gradients are computed with respect to the prefix vector $\mathbf{p}$, while the pre-trained model parameters θ are kept fixed.

3.4 Key Advantage

The key advantage of prefix-tuning is that it requires optimizing only a small number of parameters (the prefix vector $\mathbf{p}$), typically around 0.1% of the total model parameters. This leads to significant parameter efficiency compared to full fine-tuning, where all the model parameters need to be updated. The authors show that by learning only 0.1% of the parameters, prefix-tuning can achieve comparable performance to full fine-tuning in the high-data regime. Moreover, prefix-tuning outperforms fine-tuning in low-data settings and extrapolates better to examples with topics unseen during training. The modular nature of the prefix also enables efficient multi-task learning, as the prefix can be easily swapped for different tasks without interfering with the pre-trained model.

4 Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning

The paper by Aghajanyan et al. (2020) [1] provides valuable insights into the dynamics of fine-tuning large language models by analyzing the concept of intrinsic dimensionality.

4.1 Intrinsic Dimensionality

The key insight of this work is that common NLP tasks within the context of pre-trained representations have an intrinsic dimension several orders of magnitude smaller than the full parameterization of the language model.

The authors propose a new interpretation of intrinsic dimension as the downstream fine-tuning task's minimal description length within the framework of the pre-trained model. In other words, they argue that a low-dimensional reparameterization (e.g., a few thousand dimensions out of hundreds of millions) can effectively represent the fine-tuning task.

4.2 Empirical Analysis

Through extensive empirical analysis, the authors show that a low-dimensional reparameterization can represent the fine-tuning task effectively. Specifically, they demonstrate that a random projection of the pre-trained model's parameters to a subspace of just a few thousand dimensions (out of hundreds of millions) is enough to represent the problem of fine-tuning a RoBERTa model to within 90% of the performance of the full model.

Figure 2 illustrates the authors' findings on the intrinsic dimensionality of pre-trained language models. The figure shows that common NLP tasks have an intrinsic dimension several orders of magnitude smaller than the full parameterization of the pre-trained model.

4.3 Implications

The low intrinsic dimensionality of fine-tuning tasks explains why vanilla gradient descent algorithms can effectively fine-tune large language models on small datasets. It also motivates the development of parameter-efficient learning methods that exploit this low intrinsic dimensionality, such as adapter methods and low-rank adaptation techniques like LoRA.

	SAID		DID	
Model	MRPC	QQP	MRPC	QQP
BERT-Base	1608	8030	1861	9295
BERT-Large	1037	1200	2493	1389
RoBERTa-Base	896	896	1000	1389
RoBERTa-Large	**207**	**774**	322	**774**

Figure 2: Intrinsic dimensionality of pre-trained language models [1]

By understanding that the fine-tuning task can be effectively represented in a much lower-dimensional subspace, these methods can adapt pre-trained models to new tasks with minimal parameter updates, reducing computational and memory requirements while enabling better transfer learning and faster convergence.

4.4 Theoretical Connections

The authors also describe strong empirical and theoretical connections between intrinsic dimensionality, number of parameters, pre-training, and generalization. They provide intrinsic-dimension-based generalization bounds that are independent of the full parameter count, further highlighting the importance of understanding the intrinsic dimensionality of language tasks.

Overall, the insights from this work on the intrinsic dimensionality of language models have been highly influential in the development of parameter-efficient learning methods, as they provide a theoretical foundation for the effectiveness of these approaches.

4.5 Intrinsic-Dimension-Based Generalization Bounds

In addition to the empirical analysis of intrinsic dimensionality, Aghajanyan et al. (2020) [1] also provide theoretical insights into the generalization properties of language models based on their intrinsic dimensionality.

The authors derive intrinsic-dimension-based generalization bounds that are independent of the full parameter count of the language model. Specifically, they show that the generalization error can be bounded by a term that depends on the intrinsic dimension of the task, rather than the total number of parameters in the model.

Let $\mathcal{H}$ be the hypothesis class of the pre-trained language model, and d_{int} be the intrinsic dimension of the fine-tuning task. The authors prove the following generalization bound:

$$\mathbb{E}_{\mathcal{D}}\left[\sup_{h\in\mathcal{H}} |L(h) - \hat{L}(h)|\right] \leq \mathcal{O}\left(\sqrt{\frac{d_{\text{int}}\log(1/\delta)}{n}}\right) \tag{2}$$

where $L(h)$ is the true loss, $\hat{L}(h)$ is the empirical loss, n is the number of training examples, and δ is the failure probability.

This bound shows that the generalization error depends on the intrinsic dimension d_{int}, rather than the full parameter count of the language model. This provides a theoretical justification for the effectiveness of parameter-efficient learning methods, as they can exploit the low intrinsic dimensionality of language tasks to achieve efficient adaptation with minimal parameter updates.

The authors further discuss the connections between intrinsic dimensionality, pre-training, and generalization, highlighting the importance of understanding the underlying structure of language models for developing effective and efficient learning algorithms.

5 Parameter-Efficient Transfer Learning for NLP

The parameter-efficient transfer learning approach proposed by Houlsby et al. (2019) [2] aims to achieve efficient adaptation of pre-trained language models to new tasks by introducing small task-specific adapter modules.

5.1 Optimization Problem

The key idea of the adapter method is to learn task-specific adapter modules that are inserted between the layers of the pre-trained model, while keeping the original model parameters frozen. This allows the model to be efficiently adapted to new tasks without modifying the entire set of parameters.

5.2 Adapter Architecture

The adapter module proposed by Houlsby et al. consists of a bottleneck architecture with the following components (as shown in Figure 3):

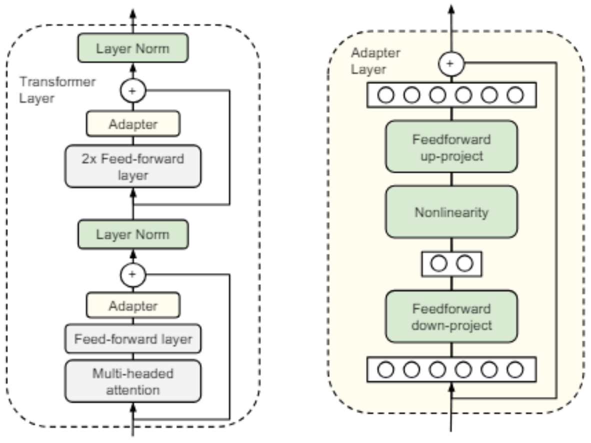


Figure 3: Adapter architecture proposed by Houlsby et al. [2]

A **down-projection layer** that projects the input representation to a lower-dimensional space. A **non-linear layer**, typically using a ReLU or GeLU activation function. An **up-projection layer** that projects the representation back to the original dimension. The adapter module is inserted between the layers of the pre-trained model, and its output is added to the output of the corresponding layer. This allows the adapter to modify the behavior of the pre-trained model while keeping the original parameters frozen.

5.3 Objective Function

The objective of the adapter method is to minimize the task-specific loss function (e.g., cross-entropy for classification tasks) by optimizing the adapter parameters, while keeping the pre-trained model parameters fixed:

$$\min_{\mathbf{w}_{\text{adapter}}} \mathcal{L}(\mathbf{x}, \mathbf{y}; \theta, \mathbf{w}_{\text{adapter}}) \tag{3}$$

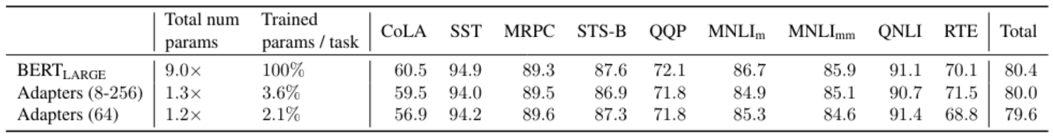
where $\mathbf{x}$ and $\mathbf{y}$ are the input and target output, respectively, θ represents the frozen pre-trained model parameters, and $\mathbf{w}_{\text{adapter}}$ are the trainable adapter parameters.

5.4 Optimization Algorithm

To optimize the adapter parameters $\mathbf{w}_{\text{adapter}}$, the authors use gradient descent. The gradients are computed with respect to the adapter parameters, while the pre-trained model parameters θ are kept fixed.

5.5 Key Advantage

The key advantage of the adapter method is that it adds only a few trainable parameters per task, typically less than 1% of the pre-trained model's parameters. This enables efficient transfer learning and multi-task learning, as new tasks can be added without revisiting previous ones. Houlsby et al. evaluated the adapter method on 26 diverse text classification tasks, including the GLUE benchmark [5]. They found that adapters attain near state-of-the-art performance while adding only a few parameters per task. On the GLUE benchmark, adapters achieved within 0.4% of the performance of full fine-tuning while adding only 3.6% parameters per task, as shown in Figure 4.



	Total num params	Trained params / task	CoLA	SST	MRPC	STS-B	QQP	MNLI_m	MNLI_{mm}	QNLI	RTE	Total
$\text{BERT}_{\text{LARGE}}$	9.0×	100%	60.5	94.9	89.3	87.6	72.1	86.7	85.9	91.1	70.1	80.4
Adapters (8-256)	1.3×	3.6%	59.5	94.0	89.5	86.9	71.8	84.9	85.1	90.7	71.5	80.0
Adapters (64)	1.2×	2.1%	56.9	94.2	89.6	87.3	71.8	85.3	84.6	91.4	68.8	79.6

Figure 4: Performance of adapter-based tuning compared to fine-tuning on the GLUE benchmark [2]

The authors also discussed the potential of adapters for multi-task learning and continual learning scenarios, as the task-specific adapters can be easily added or removed without interfering with other tasks.

6 LoRA: Low-Rank Adaptation of Large Language Models

The LoRA (Low-Rank Adaptation) method, proposed by Hu et al. (2022) [3], is a parameter-efficient fine-tuning technique for adapting large language models to new tasks. LoRA aims to achieve efficient adaptation by introducing low-rank updates to the pre-trained model weights, while keeping the original weights frozen.

6.1 Optimization Problem

The key idea of LoRA is to learn low-rank updates to the pre-trained model weights, rather than updating all the parameters. Specifically, LoRA introduces two small rank-decomposed matrices, $\mathbf{A}$ and $\mathbf{B}$, which are multiplied with the pre-trained weights $\mathbf{W}$ during fine-tuning:

$$\mathbf{W}' = \mathbf{W} + \mathbf{A}\mathbf{B}^\top \tag{4}$$

where $\mathbf{W}'$ represents the updated weights used during the forward pass.

6.2 Objective Function

The objective of LoRA is to minimize the task-specific loss function by optimizing the low-rank update parameters $\mathbf{A}$ and $\mathbf{B}$, while keeping the pre-trained weights $\mathbf{W}$ frozen:

$$\min_{\mathbf{A},\mathbf{B}} \mathcal{L}(\mathbf{x}, \mathbf{y}; \mathbf{W}, \mathbf{A}, \mathbf{B}) \tag{5}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the input and target output, respectively, and $\mathcal{L}$ is the task-specific loss function (e.g., cross-entropy for classification tasks).

6.3 Optimization Algorithm

To optimize the low-rank update parameters $\mathbf{A}$ and $\mathbf{B}$, the authors use gradient descent. The gradients are computed with respect to $\mathbf{A}$ and $\mathbf{B}$, while the pre-trained weights $\mathbf{W}$ are kept fixed.

6.4 LoRA Architecture

The LoRA adaptation can be viewed as a special case of the adapter architecture, where the down-projection, non-linear, and up-projection layers are replaced by the low-rank update matrices $\mathbf{A}$ and $\mathbf{B}$. This is illustrated in Figure 5.

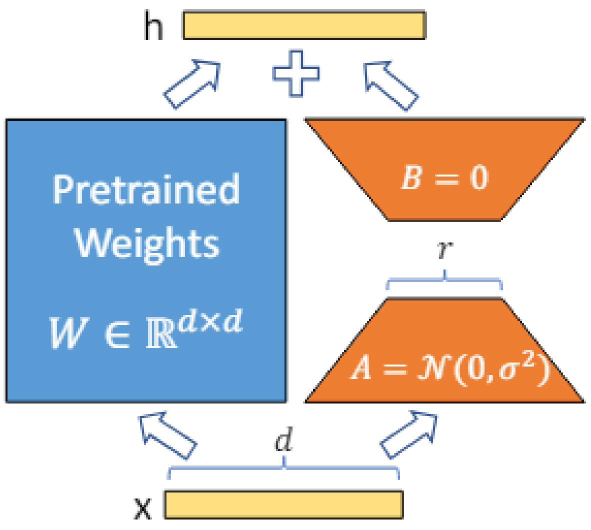

h
Pretrained Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

Figure 5: LoRA adaptation architecture [3]

6.5 Key Advantages

The key advantages of LoRA are:

1. **Parameter Efficiency**: LoRA requires only a small number of trainable parameters (typically less than 1% of the pre-trained model's parameters), making it highly parameter-efficient.

2. **Effective Adaptation**: LoRA achieves comparable or better performance than full fine-tuning on various tasks.

3. **Modularity**: LoRA modules can be easily swapped for different tasks, enabling efficient multi-task learning.

6.6 Expressive Power of Low-Rank Adaptation

The paper "The Expressive Power of Low-Rank Adaptation" by Zeng et al. (2024) [6] provides a theoretical analysis of the expressive power of the Low-Rank Adaptation (LoRA) method.

6.6.1 Fully Connected Neural Networks

For fully connected neural networks (FNNs), the authors show that LoRA can adapt any model f to accurately represent any smaller target model $\bar{f}$ if the LoRA rank is greater than or equal to:

$$\text{LoRA-rank} \geq (\text{width of } f) \times \frac{\text{depth of } \bar{f}}{\text{depth of } f} \tag{6}$$

This result holds under the mild assumption that the target model $\bar{f}$ can be well-approximated by a low-rank matrix.

Furthermore, the authors provide bounds on the approximation error when the LoRA rank is lower than the threshold. Specifically, they show that the approximation error between the fine-tuned model and the target model decreases as the LoRA rank increases.

6.6.2 Transformer Networks

For Transformer networks, the authors show that any model can be adapted to a target model of the same size with rank-$(\frac{\text{embedding size}}{2})$ LoRA adapters.

This result indicates that the expressive power of LoRA is nearly optimal for Transformer networks, as the effective expressive power of the LoRA-adapted model is close to the expressive power of the target model.

6.6.3 Implications

The theoretical insights provided in this work offer several important implications:

1. **Theoretical Guarantees**: The authors establish theoretical guarantees on the expressive power of LoRA, showing that it can effectively approximate the full fine-tuning solution with a small number of parameters.

2. **Rank Selection**: The results provide guidelines for selecting the rank of the low-rank updates based on the desired approximation error and the number of available parameters.

3. **Limitations**: The authors also discuss the limitations of low-rank adaptation, noting that it may not be effective for tasks that require significant changes to the pre-trained model's behavior, as the approximation error can be large in such cases.

Overall, this work provides the first known theoretical results on the expressive power of LoRA, offering valuable insights into the practical success of this parameter-efficient fine-tuning method.

6.7 Empirical Evaluation

The authors evaluate LoRA on various natural language processing tasks, including text classification, question answering, and language generation. They show that LoRA can achieve comparable or better performance than full fine-tuning while requiring only a small fraction of the parameters.

Model&Method	# Trainable Parameters	WikiSQL	MNLI-m	SAMSum
		Acc. (%)	Acc. (%)	R1/R2/RL
GPT-3 (FT)	175,255.8M	**73.8**	89.5	52.0/28.0/44.5
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 (Adapter^{H})	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 (Adapter^{H})	40.1M	73.2	**91.5**	53.2/29.0/45.1
GPT-3 (LoRA)	4.7M	73.4	**91.7**	**53.8/29.8/45.9**
GPT-3 (LoRA)	37.7M	**74.0**	**91.6**	53.4/29.2/45.1

Figure 6: Performance Comparison of LoRA with other fine-tuning methods

Overall, LoRA demonstrates the effectiveness of low-rank adaptation as a parameter-efficient learning method for adapting large language models to new tasks, with strong theoretical and empirical support.

7 Conclusion

Parameter-efficient learning methods have emerged as a promising solution to adapt large pre-trained language models to new tasks with minimal parameter updates, reducing computational and memory requirements while enabling better transfer learning and faster convergence.
This review has covered several prominent approaches, including prefix-tuning, adapter methods, and low-rank adaptation techniques like LoRA. Prefix-tuning optimizes a small continuous prefix vector that is prepended to the input, allowing the language model to attend to it as "virtual tokens." This method achieves significant parameter efficiency compared to full fine-tuning, particularly in high-data regimes.
Adapter methods introduce small task-specific modules within the pre-trained model, which are trained while keeping the original model parameters frozen. This enables efficient transfer learning

and multi-task learning, with adapters adding only a few trainable parameters per task.
The analysis of intrinsic dimensionality has provided valuable insights into the dynamics of fine-tuning large language models. The finding that common NLP tasks have an intrinsic dimension much smaller than the full model parameterization explains the effectiveness of simple gradient-based fine-tuning and motivates the development of parameter-efficient learning methods.
Low-rank adaptation techniques, such as LoRA, achieve parameter-efficient fine-tuning by introducing low-rank updates to the pre-trained model weights. LoRA can effectively approximate the full fine-tuning solution with a small number of parameters, as demonstrated by the theoretical and empirical results.
As large language models continue to grow in size and complexity, parameter-efficient learning methods will become increasingly important for practical deployment and efficient adaptation to new tasks. Future research directions may include developing more expressive and versatile parameter-efficient methods, exploring their theoretical properties, and investigating their application to emerging areas such as multi-modal and multi-task learning.

References

[1] Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning, 2020.

[2] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp, 2019.

[3] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[4] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation, 2021.

[5] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019.

[6] Yuchen Zeng and Kangwook Lee. The expressive power of low-rank adaptation, 2024.

Post-Training Pruning Methods for Neural Networks

Abbas Shaikh
COMP 414
Rice University
ats16@rice.edu

Abstract

The recent explosion in the parameter size of large-scale neural networks has made the use of such models computationally costly or prohibitive for resource-constrained systems. Post-training pruning provides a particularly promising methods to reduce the size and latency of such networks without loss in performance through the sparsification of network elements. While numerous methods have been proposed for post-training pruning, including those that rely on quadratic approximations of the loss function or alternative heuristics, it remains unclear how these methods compare, particularly in lieu of computational constraints. In this project, we explore various impact and heuristic-based methods for neural network pruning as well as experimentally verify how such methods compare on several neural network architectures.

1 Introduction

Modern large-scale neural networks utilize increasingly large number of parameters, resulting in slower and computationally costlier inference. Numerous methods have been proposed to mitigate the impact of increasing computational costs, including low-bit quantization, knowledge distillation, low-rank factorization, and pruning. Pruning provides a particularly promising method to reduce the size and latency of neural networks by aiming to identify sub-networks of larger, dense networks with minimal loss in performance. Pruning methods extract these sub-networks through sparsifying network elements.

Various classes of sparsity can be achieved in the context of neural network pruning, yielding disparate improvements in efficiency and cost. Unstructured sparsity, corresponds to the removal of individual weights from the network. While requiring the least constraints or knowledge of model structure to achieve, it is most difficult to leverage for improvements in latency. The removal of higher granularity structures such as neurons or layer, corresponding to contiguous blocks of parameters, yields structured sparsity, which can provide significant speedups in inference on nearly any system. The semi-structured variant of sparsity removes pre-defined fractions of weights in regularly interspaced patterns (i.e. 2 out of every 4 weights, or 2:4 sparsity), which select GPUs can leverage to improve latency.

Many approaches have been proposed to induce such sparsity patterns within neural networks. Sparse training or mask learning methods, for example, train a network with knowledge of a sparsity constraint. However, we focus particularly on post-training pruning methods which attempt to find sparse masks of neural networks after training while minimally impacting the network performance. Efficient post-training pruning methods allow for the pruning of large-scale neural networks without the need for costly and often infeasible model retraining. These pruning methods will allow for greater utility and democratization of neural networks on resource constrained systems without the computational capability to host large-scale models.

2 Approaches to Post-Training Pruning

2.1 Problem Formulation

Suppose we have a neural network of p trainable parameters represented as a vector $w \in \mathbb{R}^p$ with an empirical loss function $\mathcal{L}(w) \coloneqq \sum_{i=1}^{N} \ell_i(w)$, where N is the number of data samples and ℓ_i is a function on the i-th sample. The pruning problem (in the case of unstructured sparsity) is most often represented by the following sparse optimization objective:

$$\min_{w \in \mathbb{R}^p} \mathcal{L}(w) \quad s.t. \|w\|_0 \le k \tag{1}$$

where k denotes the target sparsity. In the post-training pruning scenario, we are given a pre-trained weight vector $\overline{w} \in \mathbb{R}^p$, and seek to find a k-sparse vector $w \in \mathbb{R}^p$ that maintains the model's original performance, i.e. $\mathcal{L}(w) \approx \mathcal{L}(\overline{w})$.

2.2 Impact-Based Pruning Methods

Impact-based pruning methods remove weights in accordance with how those weights impact the loss function, $\mathcal{L}$. Leveraging the information provided by the full loss function would require some sort sparse training regime, which is often computationally infeasible. Thus, most works use a quadratic approximation of $\mathcal{L}$ around the pre-trained weight vector $\overline{w}$:

$$\mathcal{L}(w) \approx \mathcal{L}(\overline{w}) + \nabla \mathcal{L}(\overline{w})^\top (w - \overline{w}) + \frac{1}{2}(w - \overline{w})^\top \nabla^2 \mathcal{L}(\overline{w})(w - \overline{w}) + \mathcal{O}(\|w - \overline{w}\|^3).$$

Given a gradient approximation $g \approx \nabla \mathcal{L}(\overline{w})$ and Hessian approximation $H \approx \nabla^2 \mathcal{L}(\overline{w})$, (1) can be reformulated as

$$\min_{w \in \mathbb{R}^p} Q(w) \coloneqq \mathcal{L}(\overline{w}) + g^\top (w - \overline{w}) + \frac{1}{2}(w - \overline{w})^\top H (w - \overline{w}) \quad s.t. \|w\|_0 \le k. \tag{2}$$

Most commonly in literature, the gradient is approximated by the stochastic gradient and the Hessian by the empirical Fisher information matrix:

$$g = \frac{1}{n}\sum_{i=1}^{n} \nabla \ell_i(\overline{w}) \qquad H = \frac{1}{n}\sum_{i=1}^{n} \nabla \ell_i(\overline{w}) \nabla \ell_i(\overline{w})^\top$$

2.2.1 Traditional Frameworks

Impact-based pruning methods were initially proposed in the Optimal Brain Damage (OBD) framework [1]. It is commonly assumed, as in the OBD framework, that $\overline{w}$ is a local optimum of the loss function, i.e. $g = 0$ and $\mathcal{L}(w)$ can be approximated simply using second-order information, i.e. $\mathcal{L}(\overline{w}) + \frac{1}{2}(w - \overline{w})^\top H (w - \overline{w})$. With this approximation, OBD searches for individual weights to prune assuming a diagonal Hessian H, deriving an expression for the saliency of each parameter w_i as $\frac{\overline{w}_i^2}{2H_{ii}^{-1}}$, which approximates the increase in the loss function when w_i is eliminated [1]. In order to prune the network to be k-sparse, the k most salient parameters are selected.

This approach is further expanded in the Optimal Brain Surgeon (OBS) framework, which utilizes the same local approximation of the loss function, without assuming a diagonal Hessian [2]. A critical extension of the OBS framework is the additional inclusion of a weight update $\delta w = -\frac{\overline{w}_i}{H_{ii}^{-1}} H^{-1} e_i$, which is the optimal perturbation of the weights to compensate for removing w_i [2].

Note: While this framework traditionally focus on unstructured sparsity, the notion of saliency can be extended easily to the case of structured sparsity by aggregating the saliency of the individual parameters that comprise a block [3].

2.2.2 Scaling to Large-Scale Neural Networks

Subsequent work has focused on scaling the traditional OBD/OBS frameworks to large-scale neural networks. Primarily, as p grows large, computing the Hessian or Hessian-inverse as a $p \times p$ matrix grows increasingly computationally infeasible.

To resolve this concern, numerous methods have been proposed, which introduce further approximations. For example, the layerwise-OBS algorithm adopts the traditional OBS framework and update rule but considers the objective of minimizing a quadratic approximation of the layer-wise reconstruction error rather than the global loss function, while still retaining theoretical guarantees for the accumulated error of the entire network [4]. Here, the objective is represented as

$$\min_{\hat{Z}^\ell} E(\hat{Z}^\ell) \coloneqq \frac{1}{n}\|\hat{Z}^\ell - Z^\ell\|_F^2 \tag{3}$$

where $\hat{Z}^\ell$ and Z^ℓ are the outputs of the network at layer l before applying an activation function.

More recently, the SparseGPT algorithm extends pruning methods to GPT-scale models by applying a modified OBS update rule in a similar layer-wise pruning framework [5]. Alternatively, other large-scale methods such as LLM-pruner, designed for structured pruning of LLMs, avoid Hessian computation entirely and simply rely on gradient information to estimate the importance or saliency of parameters [6].

Critically, we observe that, as model size grows, methods incur additional burdens to further approximate the loss or error of the model, beyond a simple quadratic approximation of the loss function.

2.2.3 Combinatorial Approaches

The previously described frameworks select each parameter or group of parameters individually by their saliency score calculated from an approximation of the loss function. However, the pruning problem is inherently combinatorial in nature, in that removing particular combinations weights may have significantly more salient effects on the loss than when only considering the impact of removing each weight individually. However, considering the full combinatorial nature of pruning p parameters grows exponentially in complexity, and is largely infeasible. Recent work such as the Combinatorial Brain Surgeon (CBS) algorithm, however, present tractable combinatorial extensions of the OBS framework, which disentangles the weight selection and update steps and proposes a greedy randomized algorithm for combinatorial weight selection [7]. While this algorithm still suffers from the computational constraints of computing a $p \times p$ Hessian matrix, alternative such as that Combinatorial Hessian-free Iterative Thresholding Algorithm (CHITA) algorithm have been proposed, which utilizes IHT to solve a sparse regression reformulation of the pruning objective, without any Hessian computation [8].

2.3 Heuristic-Based Methods

Additional pruning methods have been proposed that do not rely on information regarding the loss function information, but rather are based on alternative heuristics to select weights. Such heuristics provide simple metrics for pruning neural networks, especially in the absence of labeled data upon which to approximation information about the loss function as well as in the case that model size grows larger and gradient computation becomes infeasible.

Perhaps the most simple heuristic is to use the magnitude of weights as a selection criteria. This is equivalent to finding the weight vector w that satisfies the objective:

$$\min_{w \in \mathbb{R}^p} \|w - \overline{w}\|_2^2 \quad s.t. \|w\|_0 \leq k. \tag{4}$$

Magnitude pruning often serves as a baseline for alternative pruning methods, and experimentally can provide strong results, especially when integrated into gradual pruning schemes [9].

Numerous more sophisticated heuristics have been proposed, most notably pruning by weights and activations (Wanda). The Wanda metric quantifies the importance of a given weight w_{ij} is a weight matrix as the product of its magnitude $|w_{ij}|$ and its input feature norm $\|X_j\|_2$, which can be applied to select weights across the entire network or per layer [10]. Such heuristics, in addition to impact-based methods, can additionally be augmented based on particular sparsity allocation schemes, rather than simply layer-wise pruning [11].

3 Experiments

Ultimately, it can be observed that, as networks grow larger, computational constraints require the use of more significant approximations of the information provided by the loss function to prune neural networks. However, it remains unclear how these methods are connected, and whether there exists any practical hierarchy between sparse training, impact-based, and other heuristic-based methods. Thus, we seek to experimentally verify how sparse training algorithms compare to methods that leverage quadratic approximations of the loss function, layer-wise approximations of the reconstruction error, or that separate loss function information entirely from pruning and rely on alternative heuristics.

3.1 Experimental Setup

We evaluate these pruning methods on two pretrained models, a simple multilayer perception (MLP) trained on the MNIST dataset [12], and Resnet-20 [13] trained on the CIFAR-10 dataset [14]. For each model, the accuracy was calculated on a held out test set, using each pruning method to prune to multiple levels of sparsity from 0.3 to 0.99.

Sparse training was performed using standard mini-batch stochastic gradient descent, with a sparsity projection onto the larger k parameters in magnitude at each gradient step. Each model was trained for 50 epochs over the entire dataset, with a step size of 10^{-3} and batch size of 16.

To evaluate the effectiveness of using a Taylor approximation of the loss function to guide pruning, we observe, as in [8], that (2) can be reformulated as the following sparse regression problem, with an additional regularization term:

$$\min_{w \in \mathbb{R}^p} \hat{Q}(w) := \frac{1}{2}\|b - Aw\|^2 + \frac{n\lambda}{2}\|w - \overline{w}\|_2^2 \quad s.t. \|w\|_0 \leq k \tag{5}$$

where $A = [\nabla \ell_1(\overline{w}), ..., \nabla \ell_n(\overline{w})]^\top \in \mathbb{R}^{n \times p}$, $b := A\overline{w} - e \in \mathbb{R}^n$, and e is a vector of ones.

We use IHT to solve this objective, with an optimal step size selection scheme:

$$\eta(w) = \frac{\|\nabla \hat{Q}(w)\|_2^2}{\|A \nabla \hat{Q}(w)\|_2^2 + n\lambda \|\nabla \hat{Q}(w)\|_2^2}$$

and the addition of a debiasing step on each iteration, which computes in closed form the solution to the unconstrained least-squares problem, with A and $\overline{w}$ restricted on the current active set, $\mathcal{S}$.

As additional ablation studies, we also study the impact of the sample size, n, and first-order term, g, on the effectiveness of pruning using an approximation of the loss function:

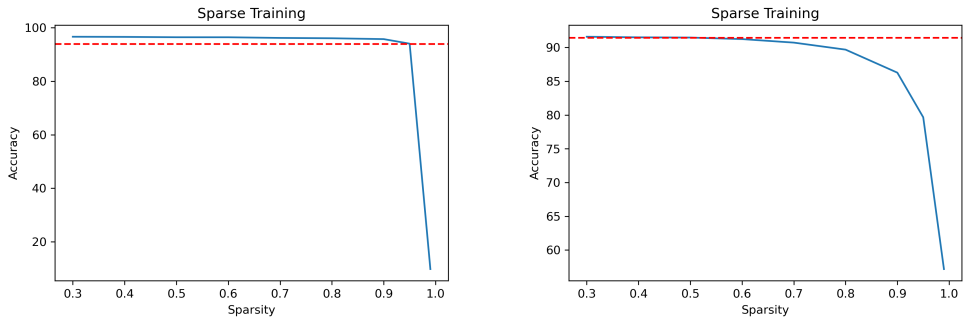
1. Recent works such as [3] and [7] argue that larger sample size, n, for approximating the Hessian and gradient approximation yields better accuracy. We validate how pruning performs at various choices of sample sizes, from 10^2 to 10^4 data samples.
2. As previously noted, a common assumption in literature is that the pre-trained network prior to pruning is at a local minimum of the loss function, and thus we take $g = 0$, which corresponds to taking $b = A\overline{w}$ in (5). We additionally evaluate the impact of this assumption on the corresponding pruned models.

Finally, to compare the utility of approximating the loss function with heuristic-based methods, we benchmark these methods against standard magnitude pruning, both with and without a debiasing step, for completeness.

3.2 Results

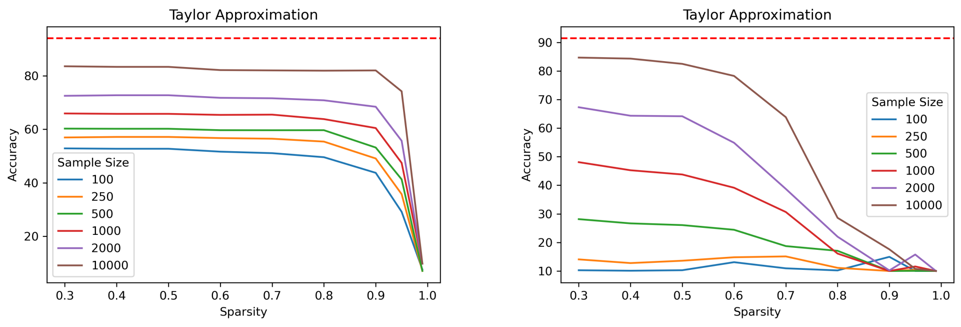
Sparse training yields the strongest results, meeting or exceeding the baseline dense accuracy for most levels of sparsity (Figure 1).

We observe that minimizing a Taylor approximation of the loss function yields notably reduced test accuracy as opposed to sparse training, where information about the loss function can be leveraged at every gradient step (Figure 2). However, the accuracy of pruned models approaches the dense accuracy as the sample size used to compute the Hessian and gradient approximations increases.



(a) (b)

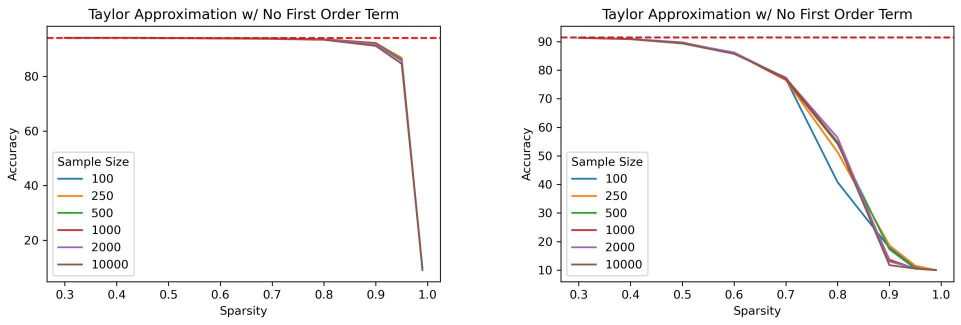
Figure 1: Accuracy of (a) MLP and (b) Resnet-20 after sparse training to various levels of sparsity. The baseline accuracy of the dense network is shown in red.



(a) (b)

Figure 2: Accuracy of (a) MLP and (b) Resnet-20 after pruning according to objective (5) to various levels of sparsity

Crucially, we find that the assumption that the network is trained to a local minimum and thus that the gradient of the loss function at $\overline{w}$ is 0 carries significant weight in the success of pruning methods. While failing to provide comparable results to sparse training, pruning with this method provides notably improved test accuracies as opposed to when the gradient is included in the approximation of the loss function, that meet or near the baseline dense accuracy for low levels of sparsity (Figure 3).



(a) (b)

Figure 3: Accuracy of (a) MLP and (b) Resnet-2 0 after pruning according to objective (5), with the assumption that $g = 0$, to various levels of sparsity

Finally, in the case of magnitude pruning, we observe that the magnitude heuristic yields comparable results across most levels of sparsity to pruning according to an approximation of the loss function (Figure 4). Debiasing the selected weights provides little to no improve model accuracy.

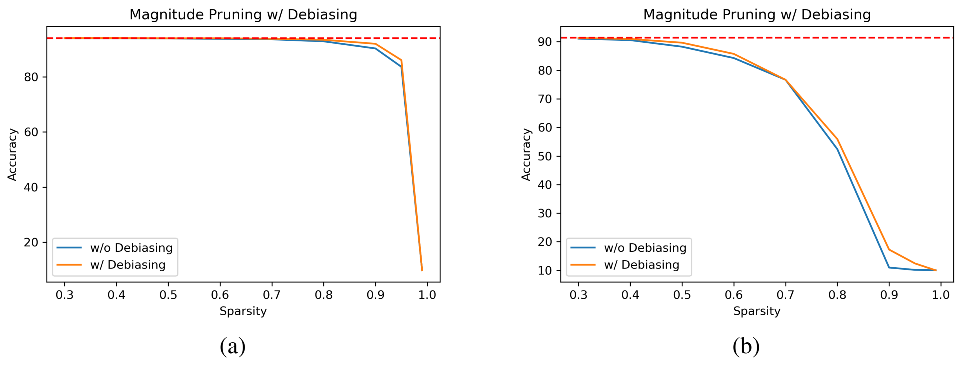


Figure 4: Accuracy of (a) MLP and (b) Resnet-20 after magnitude pruning to various levels of sparsity

4 Discussion and Future Work

While our results establish that pruning methods that incorporate model retraining outperform post-training pruning methods, such retraining is often computationally prohibitive. Thus, such sparse training methods can only be used in the case the model size remains small enough to be computationally feasible.

As an alternative we find that both impact-based methods and heuristic-based methods provide comparable results in pruning small-scale neural networks such as simple MLP and Resnet-20. In concurrence with previous work, we find that the sample size used to estimate the Hessian and gradient for impact-based methods is positively correlated with the resulting test accuracy of the pruned model. We additionally observe that a frequent and seemingly inconsequential assumption that the gradient of a pre-trained model is 0 when approximating the loss function in fact poses significant impacts on the effectiveness of impact-based pruning methods. It remains unclear why the inclusion of a gradient term in the Taylor approximation of the loss function has such an appreciable influence.

We further find that heuristic-based methods, particularly magnitude pruning perform nearly identically to more sophisticated methods that incorporate information about the loss function. However, literature suggests that the effectiveness of magnitude pruning methods declines as model size grows larger and the model's task grows more difficult. In such scenarios, small magnitude weights encode critical knowledge essential for tackling difficult downstream tasks, a phenomenon referred to as the junk DNA hypothesis [15].

It remains unclear how these pruning methods are connected and whether there exists a practical hierarchy between these methods based on what is prohibitive given monetary and computational constraints. To further explore the effectiveness of the proposed methods, future work will involve investigating how these methods compare as model size increases, and whether a larger discrepancy between heuristic and impact-based methods can be observed in this scenario. Furthermore, a more robust comparison of layer-wise approximations of the reconstruction error per layer as opposed to a approximation of the global loss over the entire network will be conducted.

References

[1] Yann LeCun, John Denker, and Sara Solla. “Optimal brain damage”. In: *Advances in neural information processing systems* 2 (1989).

[2] Babak Hassibi and David Stork. “Second order derivatives for network pruning: Optimal brain surgeon”. In: *Advances in neural information processing systems* 5 (1992).

[3] Sidak Pal Singh and Dan Alistarh. “Woodfisher: Efficient second-order approximation for neural network compression”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 18098–18109.

[4] Xin Dong, Shangyu Chen, and Sinno Pan. “Learning to prune deep neural networks via layer-wise optimal brain surgeon”. In: *Advances in neural information processing systems* 30 (2017).

[5] Elias Frantar and Dan Alistarh. “Sparsegpt: Massive language models can be accurately pruned in one-shot”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 10323–10337.

[6] Xinyin Ma, Gongfan Fang, and Xinchao Wang. “Llm-pruner: On the structural pruning of large language models”. In: *Advances in neural information processing systems* 36 (2023), pp. 21702–21720.

[7] Xin Yu et al. “The combinatorial brain surgeon: pruning weights that cancel one another in neural networks”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 25668–25683.

[8] Riade Benbaki et al. “Fast as chita: Neural network pruning with combinatorial optimization”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 2031–2049.

[9] Eldar Kurtic and Dan Alistarh. “GMP*: Well-Tuned Gradual Magnitude Pruning Can Outperform Most BERT-Pruning Methods”. In: *arXiv preprint arXiv:2210.06384* (2022).

[10] Mingjie Sun et al. “A simple and effective pruning approach for large language models”. In: *arXiv preprint arXiv:2306.11695* (2023).

[11] Peng Xu et al. “BESA: Pruning Large Language Models with Blockwise Parameter-Efficient Sparsity Allocation”. In: *arXiv preprint arXiv:2402.16880* (2024).

[12] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[13] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[14] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[15] Lu Yin et al. *Pruning Small Pre-Trained Weights Irreversibly and Monotonically Impairs "Difficult" Downstream Tasks in LLMs*. 2024. arXiv: `2310.02277 [cs.LG]`.

Overview of Homotopy Methods for Path Planning

Amy Danjul
Rice University
jad20@rice.edu

Noah Spector
Rice University
nls9@rice.edu

***Abstract*—Homotopy is a concept that often appears in the problem of robotics path planning, but is not well-studied. Most graph-based and potential field methods for collision-free path planning do not consider the homotopy class of a given path, which may yield useful information. In this literature review, we will investigate the Homotopy Continuation Method (HCM), which is one technique of solving complex systems of nonlinear equations by deforming a simple solvable system of equations into the final complex system. By representing the configuration space as a system of nonlinear equations, HCM can be adapted to the problem of robotics path planning. We will also briefly introduce additional graph-based path planning methods that use the concept of homotopy in path planning.**

I. Introduction

In general, finding collision-free paths is a hard problem. One must consider constraints on dynamics, environment, and time. As a result, many path-finding algorithms are only probabilistically complete, in that they are guaranteed to find a path asymptotically. Even fewer algorithms make a claim of asymptotic optimality. The upshot to this tradeoff is that the algorithms can run much quicker.

Graph algorithms and potential fields are two popular methods of collision-free path planning. Graph algorithms represent the configuration space as a graph and perform search algorithms to get from the start to goal nodes. Some landmark examples of graph algorithms include PRM, which samples points and constructs them into a roadmap, and RRT*, which builds a rapidly-exploring tree starting at the star and goal points and attempts to connect the trees. Newer works have introduced heuristics to encourage the exploration of more promising paths, but the fundamental principle of sampling and evaluating remains the same. While this approach allows for efficient exploration of high dimensional spaces, graph algorithms ultimately produce non-smooth paths that may not be optimal. On the other hand, potential fields methods assign a potential field to every point in the space. Points around obstacles are assigned a repulsive field, and the goal point is assigned an attractive field. Running a gradient descent algorithm on the field map then produces a valid collision-free path. It is easy to see that both graph-based and potential field algorithms fall prey to the local optima problem.

Homotopy path planning methods are an attempt to address this problem. Information about path homotopy classes is useful in a couple different ways. They can be used to more intelligently compute and consider paths in space, since it doesn't make much sense to iterate through many potential paths that all have the same homotopy class. Homotopy methods also combat the local minima problem since they can be used to explore new classes of paths and are far less prone to falling into local optima traps.

While homotopy arises in path planning very often, research in areas of homotopy path planning is relatively sparse. The method we will primarily investigate is Homotopy Continuation Methods (HCM), which represents the problem as a system of nonlinear equations, and "deforms" a simple system of equations into the target system of equations using homotopy. HCM has been used to solve problems in problems that involve scientific computing and solving systems of polynomial or partial differential equations, such as in control theory, fluid dynamics, and quantum theory. In this work, we investigate how HCM has been applied to the problem of path planning in robotics.

HCM Is just one way that researchers have incorporated concepts from mathematical homotopy to solve path planning problems. Other methods use homotopy information to more intelligently construct and search graph representations of the space, or to compute whether two valid paths in a graph structure are homotopically equivalent. In all cases, the use of homotopy concepts can produce higher quality motion trajectories.

II. Terminology

A. Algorithmic Robotics

One of the central problems in algorithmic robotics is path planning. In essence, given a workspace, or set of obstacles, plus the possible modes of movement of a robot, we want to find an efficient way to get from one point to another. In order to formalize this notion, we introduce the notion of a configuration space. Whereas a point in the workspace corresponds to a point in the environment, a point in the configuration space C specifies the exact position of each controllable part of the robot. For example, a robot that operates in 2D with free rotation would have a configuration space of $SE(2)$.

The final definition we must give is that of free space. Given a set of obstacles $\{O_i\}$, we define the free space to be

$$W_f = W - \cup O_i$$

, where W is the workspace. Now, suppose $f : C \to W$ is the transformation that takes configurations to positions in the workspace. Then, a configuration c is free in C if $f(c) \in W_f$, and we denote the free space in the configuration space as C_f.

Formalizing the problem, given a start point x_{start} and a goal point x_{goal}, we want to find a continuous path $\pi : [0, 1] -> C$ such that

$$\pi(0) \in f^{-1}(x_{start}), \tag{1}$$
$$\pi(1) \in f^{-1}(x_{goal}) \tag{2}$$

and for all $0 \leq x \leq 1$,

$$\pi(x) \in C_f$$

. However, the configuration space is often extraordinarily high dimensional. A freely translating and rotating robot in $\mathbb{R}^3$ has a configuration space of dimension 6, and actual implementations of robotic arms routinely exceed 6 dimensions.

In practice, we can define the path in the workspace and do inverse kinematics to recover a valid path in the configuration space. Formally, our path becomes

$$\pi : [0, 1] \to W$$

, and we require that $\pi(x) \in W_f$ for $0 \leq x \leq 1$. For a point robot with no rotation, these formulations are the same. In this review, we will assume that paths are in the workspace, rather than in the configuration space.

B. Sampling-based Path Planning

A good example of sampling-based path planning is the Probabilistic Road Map (PRM) method. The main idea is to randomly sample the configuration space and connect points if they are in the free space. Then, the problem becomes a graph search problem to find the shortest path between the start and goal points. PRM is probabilistically complete, in that as $t \to \infty$, the probability that it doesn't find a valid path, if one exists, goes to zero.

Other sampling-based path planning techniques fall under a similar idea of sampling nodes from a distribution of points in the configuration space, checking for validity, and appending to a graph representation of the configuration space. Ultimately, the goal is to adequately capture the complexity of the configuration space such that the optimal path is very likely to be found without excessive computation time, regardless of how difficult it is to find (i.e. paths that run through very narrow passages).

C. Homotopy

Two functions are *homotopic* if one can be continuously deformed into the other, where the deformation is called the homotopy.

More formally, let X and Y be two topological spaces, and f, g be two continuous functions from X to Y. A homotopy is a continuous function

$$H : X \times [0, 1] \to Y \tag{3}$$

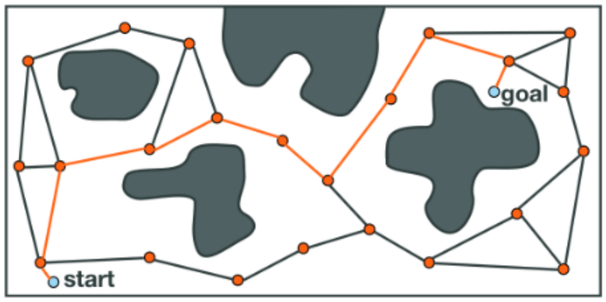


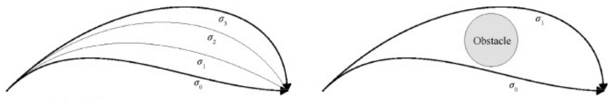
Fig. 1: Probabilistic Road Map

such that

$$H(x, 0) = f(x), \tag{4}$$
$$H(x, 1) = g(x) \tag{5}$$

for all $x \in X$.

Homotopy is an equivalence relation on the set of all continuous functions from X to Y. The composition of two homotopic functions is also homotopic.



(a) An example of paths (from σ_0 to σ_3) belonging to the same homotopy class

(b) An example of two paths (from σ_0 to σ_1) belonging to different homotopy classes

Let X and Y be two topological spaces, and f, g be two continuous functions from X to Y. f and g are a *homotopy equivalence* if $f \circ g$ is homotopic to id_X, or the identity map in X. Furthermore, X and Y are *homotopically equivalent*. More intuitively, X can be transformed into Y by stretching or shrinking the space without introducing new holes (no cutting is allowed).

III. Homotopy Continuation Methods

Homotopy Continuation Method (HCM) is a technique for solving systems of polynomial or nonlinear equations. Where n is the number of variables in our system of equations and variables x, we can write the system in the form

$$F(x) = 0, \tag{6}$$
$$F : \mathbb{R}^n \to \mathbb{R}^n \tag{7}$$

Define a homotopy map as

$$H(F(x), \lambda) = 0, \tag{8}$$
$$H : \mathbb{R}^{n+1} \to \mathbb{R}, 0 \leq \lambda \leq 1 \tag{9}$$

To solve a system F, we can create a homotopy map

$$H_\lambda = (1 - \lambda)G + \lambda F, \lambda \in [0, 1] \tag{10}$$

where the start system is $G = H_0$ and the target system (to be solved) is $F = H_1$.

We require

- H_0 is simple to solve in the sense that H_0^{-1} is easy to find numerically

- H_0^{-1} is continuous for all λ

By varying λ from 0 towards 1, we can continuously deform G into F, and thus find solutions of G that lead us towards solutions of F. We can choose G to get different homotopies. For example, we can get the Newton homotopy by choosing

$$G(x) = F(x) - F(x_0) \tag{11}$$

which produces the homotopy equation

$$H(F(x), \lambda) = F(x) - (1 - \lambda)F(x_0) = 0 \tag{12}$$

Observe that for $\lambda = 0$, we get

$$H(x, 0) = F(x) - F(x_0) = 0 \implies F(x) = F(x_0) \tag{13}$$

which is the trivial problem, and that when $\lambda = 1$, we get

$$H(x, 1) = F(x) = 0 \tag{14}$$

as desired. This process generates homotopy paths γ, such as below:

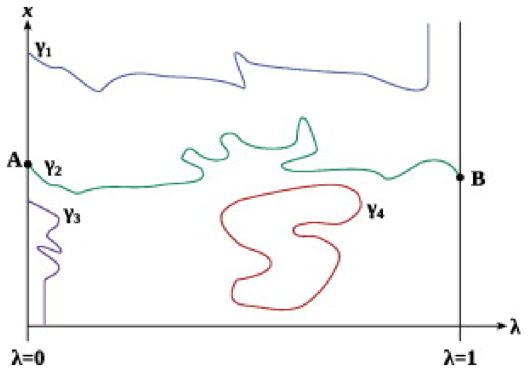


Fig. 2: Only γ_2 successfully finds a path from point A to point B. All other trajectories failed to find a feasible path.

To solve this problem numerically, we discretize the λ domain by choosing a step size $\Delta\lambda$ sufficiently small, and each mesh point $\lambda_i \in [0, 1]$ is such that

$$\lambda_i = \lambda_{i-1} + \Delta\lambda \tag{15}$$

We can choose $\Delta\lambda$ depending on the problem. For example, we can vary $\Delta\lambda$ depending on the curvature of the homotopy trajectory. We choose start point $(\lambda_0, x_0) = (0, x_0)$. Then, we solve $H(\lambda_i, x) = 0$ to get x_i for each i. This can be achieved by any choice of predictor-corrector method. Some popular choices are the Euler predictor method and the Newton corrector method.

HCM is useful for problems that can be modeled as a system of polynomial or nonlinear equations. In robotics path planning, the problem to be solved is finding the optimal collision-free path from a start to goal point in a configuration space. We will investigate how we can model the configuration space as a system of nonlinear equations.

IV. Homotopy Continuation Methods for Path Planning

One branch of recent work formulates the entire configuration space as a system of nonlinear equations and uses homotopy methods to construct a path from scratch. In 2013, Vazquez-Leal et al proposed the Homotopy Path Planning Method, one such method. The basis of this method is a special case of homotopy continuation methods called Newton's homotopy. Specifically, this case puts

$$G(x) = F(x) - F(x_0)$$

so that the entire homotopy method becomes

$$H_\lambda(x) = F(X) + (1 - \lambda)F(x_0)$$

When $\lambda = 0$, this homotopy has a trivial solution, and when $\lambda = 1$, we recover $F(X)$, as desired.

A. Homotopy Path Planning Method

The next step is to translate the set of obstacles and start and goal states into a system of nonlinear equations. For a point robot, this problem reduces to finding a path through the workspace. On a high level, one approach is to start with a linear system of equations whose solution is the goal point, add the obstacles in as repulsions, and use homotopy continuation methods to solve the resultant system of equations.

The 2013 paper deals first with point robots in the plane. Without loss of generality, we start with a linear system of equations whose solution is $(1, 1)$. The paper does not specify how values are chosen or how they affect the performance of the algorithm. Given these equations, the next step is to augment one with obstacle information.

In that paper, the authors provide the formulation for a circular object with center (x_i, y_i) and radius r_i as follows:

$$C_i = (x - x_i)^2 + (y - y_i)^2 - r_i^2$$

We note that this function is zero for points on the circle, negative for those inside, and positive for those outside. At first, this function does not seem to create a repulsion around the object. However, $1/C_i$ has function values that are very large in magnitude around the edge of the object and smaller as one goes further away. Since we assume that we are working in free space, the contribution from each obstacle will be positive. Unless the constants from the original linear system of equations are enormous, which normalizing should help abate, it is unlikely that the contribution from either line will be enough to overcome the asymptotic growth of functional values.

The paper also approximates rectangles by squircles (or 2D projections of $2n$ balls for arbitrary n). See the figure for a sample environment with squircles and circles. For a rectangle with side lengths $2a$ and $2b$ centered at (x_i, y_i), the paper proposes the following formulation:

$$R_i(x, y) = ((x - x_i)/a)^{2n} + ((x - y_i)/a)^{2n} - 1$$

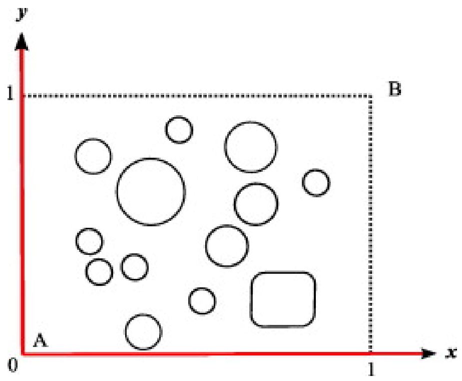


Fig. 3: 2D map environment

The paper then composes obstacles by multiplication of their reciprocals; call this product $1/F(x)$. The paper then subtracts off $1/(F(1,1)$ to ensure that there is still a solution to the system of equations at $(1,1)$ and scales the entire obstacle repulsion expression by an adjustable constant p.

By taking the product of obstacle repulsions rather than the sum, this method tightens the boundaries around each obstacle. For a given point x, if $1/C_i(X) < 1$, then there would be a positive contribution to the repulsion factor even though the obstacle is quite far away. By multiplying instead, obstacles that are far away reduce the value of $1/F(X)$. We are not concerned about the path clipping into the obstacle, as $1/F(X)$ still grows unboundedly as one approaches the boundary of the obstacle. We merely must set λ to be sufficiently small.

Putting it all together, we model the 2D workspace with the following set of equations, where ℓ_1 and ℓ_2 are the underlying set of linear equations.

$$f_1(x,y) = \ell_1(x,y) = 0$$

$$f_2(x,y) = \ell_2(x,y) + p(1/F(x,y) - 1/F(1,1)) = 0$$

We can then apply Newton's homotopy method to these two equations to recover a path. This method is called HPPM.

B. Generalizing Out

There are a number of ways to generalize HPPM. First, the original paper suggests setting an individual p_i for each obstacle, rather than one global p. This modification does make solving the equation harder, but it allows for one to handle variable environments. Specifically, when obstacles are close, we want p to be small, so that we don't accidentally shut off any paths. However, if the obstacles are sparse, we can let p be somewhat larger, so that the continuation method has more direction. Some algebraic manipulation allows for this modified version of HPPM to be solved by existing homotopy continuation methods.

The paper also suggests a way to generalize HPPM into 3 dimensions. We start with three linear equations with a solution of $(1,1,1)$, as in the 2D case. The circle approximations become sphere approximations with the addition of a z constraint, and we approximate rectangular prisms as the 3D equivalent of circles. We then get a system of three nonlinear equations, which we can solve using existing homotopy continuation methods.

One could generalize this method to $\mathbb{R}^n$ following the same steps, though the accumulation of error might render it more difficult in high-dimensional spaces. Specifically, the curse of dimensionality might strike in this application. Further, in robotics, spaces homeomorphic to $\mathbb{R}^n$ for $n > 3$ are uncommon. Any workspace will likely be $\mathbb{R}^2$ or $\mathbb{R}^3$, and if we decide to work in the configuration space instead, there is often some subspace which is not isomorphic to the reals due to rotational constraints. For example, a revolute joint introduces a subspace of S^1, which would require a nontrivial modification to the existing homotopy methods.

C. Applications

Finally, a recent work generalizes HPPM to planar robotic arms made of revolute joints. In prior work, HPPM implementations merely returned a path in the workspace, which assumes that the robot has arbitrary movement in the plane and is a point. However, a common embodiment that does not satisfy these constraints is a planar arm. This arm comprises links attached by joints that can rotate freely. A human arm restricted to the plane, for example, would have four links and three joints (shoulder, wrist, and elbow).

In order to apply HPPM to this environment, one must find a path in the C-Space. Any given path in the workspace might not be possible given the constraints of the arm, so instead, we attempt to find a path in the configuration space. The configuration space is messier, though; circular obstacles in the workspace can even become multiple obstacles in the C-Space. The paper approximates transformed obstacles by dividing each link into keypoints and observing coordinates where the arm is in collision with an obstacle.

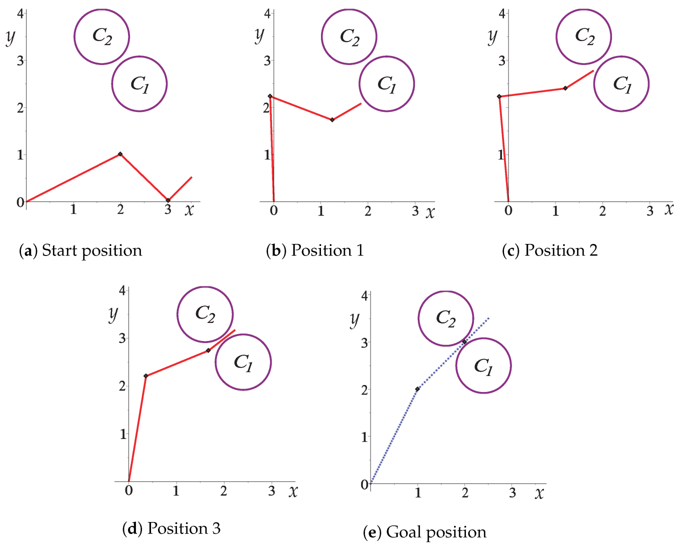
One shortcoming of the paper is that it models the configuration space as $\mathbb{R}^n$ for some n, which has a different topology than the actual $(S^1)^n$. If we assume that a link cannot intersect the previous link, then this approximation might be fine, since we can place the discontinuity in $\mathbb{R}$ on top of the previous link. However, in many actual embodiments, revolute joints have more than 360 degrees of freedom, meaning that this method could miss certain paths that only exist in $(S^1)^n$.

The paper tackles a major challenge for probabilistic planners like PRM and RRT: narrow paths. Probabilistic planners struggle to find paths in narrow environments because the probability that one samples a point within the narrow gap is small. Further, the probability that one can then connect that point to the existing graph is even smaller. In contrast, the homotopy method finds a narrow path between two circular obstacles consistently in a handful of milliseconds for simple arms, whereas probabilistic planners most likely would take longer to find the same path.

HPPM was tested in three different case studies, each of which tested the planner's ability to navigate complex environments whose goals require finding narrow passages with a wide variation on the robot's complexity. Case study

1 uses a three-link arm, while case study 2 uses a six-link arm with a gripper attached to one end. Case study 3 uses a 20-link arm to test HPPM's performance on hyper-redundant robots. In all three studies, HPPM successfully arrived at the final position.

This work demonstrates the versatility of HPPM, as it can be adapted to work with different numbers of robotic arms, shapes of obstacles, widths of passages, and even with added grippers. Thre is work to be done to explore the extension of HCM methods in 3D spaces.



(a) Start position (b) Position 1 (c) Position 2

(d) Position 3 (e) Goal position

Fig. 4: Case study 1. A three-link planar robotic arm must pass through the very narrow passage between C_1 and C_2 to reach the goal point.

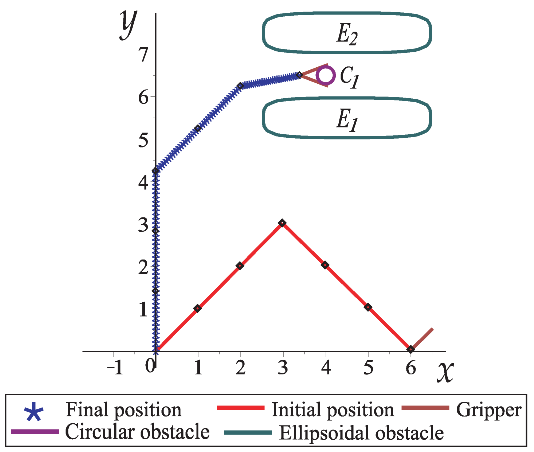


Fig. 5: Case study 2. A six-link planar robotic arm, equipped with a gripper, must navigate between two ellipsoidal obstacles.

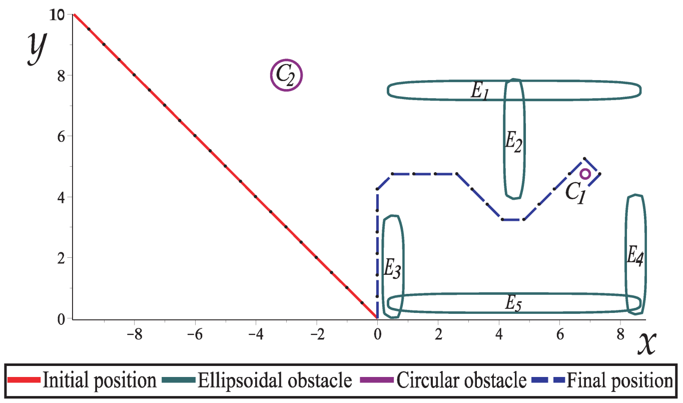
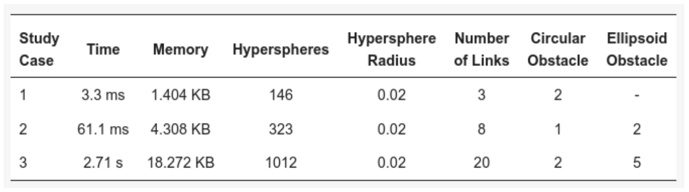


Fig. 6: Case study 3. This tests HPPM on a 20-link hyper-redundant robot.



Study Case	Time	Memory	Hyperspheres	Hypersphere Radius	Number of Links	Circular Obstacle	Ellipsoid Obstacle
1	3.3 ms	1.404 KB	146	0.02	3	2	-
2	61.1 ms	4.308 KB	323	0.02	8	1	2
3	2.71 s	18.272 KB	1012	0.02	20	2	5

Fig. 7: HPPM case study results.

V. Other Uses of Homotopy in Path Planning

A. *Topologically-informed Roadmaps*

One recent work, Clustering Topological PRM (CTopPRM), modifies the probabilistically complete algorithm, Informed-PRM, to create a roadmap that consists only of topologically-distinct paths. Homotopy information does not extend well to 3D in that it does not capture a sufficient number of useful paths, so Visibility Deformation (VD) is often used to capture the same ideas in 3D; however, performing VD is computationally expensive. The authors use Uniform Visibility Deformation (UVD), which is a more efficient extension of VD, to decrease computation time, and compute a set of paths that belong to distinct UVD classes. CTopPRM is probabilistically complete in finding a valid path, if it exists.

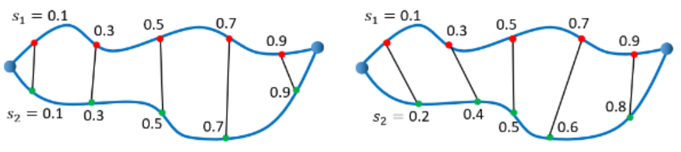
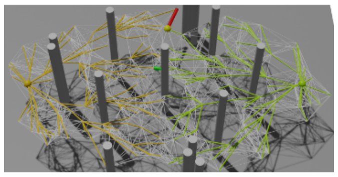


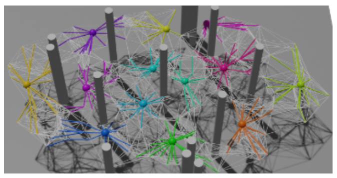
Fig. 8: Figure from RAPTOR. Comparison between UVD (left) and VD (right). Each red point on one path is transformed to a green point on the other path. Any two associated points correspond to the same parameters in UVD, but not in VD

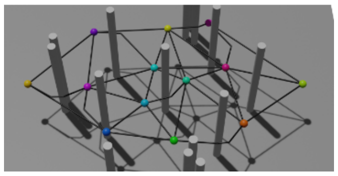
In summary, CTopPRM begins with a dense roadmap created by running Informed-PRM and dividing the nodes into two initial clusters that contain x_{start} and x_{goal}, respectively.

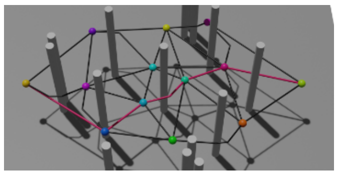
Continue to iteratively create clusters. Between two neighboring clusters, define a cluster centroid. Using these centroids as nodes, construct a new sparse roadmap containing only these vertices. Considering how to intelligently place these new centroids is imperative for defining an algorithm that minimizes the order of the graph while capturing all UVD classes. The algorithm compares the paths between every pair of connected clusters. If two paths are not deformable by UVD, then a new centroid is added at the border of the two clusters. Lastly, CTopPRM shortens found paths, prunes sub-optimal ones, and filters paths that belong to the same UVD class. It has a high experimental success rate in finding the percentage of distinct paths in a windows environment (environment where the robot navigates through a series of windows) not only in comparison to other methods, but in general as well, while taking a similar computation time.

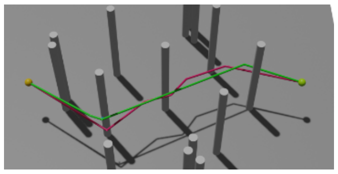
CTopPRM represents a sampling-based approach to incorporating homotopy information into path planning by augmenting existing road map path planning methods, and exemplifies how homotopy information can improve the quality of paths in sampling-based planners.


(a) roadmap clustered with initial centroids


(b) fully clustered roadmap


(a) connections between clusters


(b) one of the found paths


(a) shortening of the found path

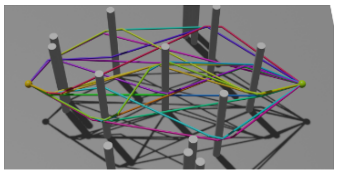

(b) filtered distinct paths

Fig. 9: CTopPRM Algorithm

VI. Conclusion

In robotics path planning, it would seem natural to consider homotopy in some way, but this area is surprisingly understudied despite its usefulness. As we discussed in this review, HCM is used in many other fields to solve polynomial or nonlinear systems of equations. HCM can be applied to robotics path planning to solve for a smooth optimal trajectory by modeling the configuration space as a system of nonlinear equations. Different works have investigated HCM's practical applications, notably in the manipulation of planar robotic arms of varying degrees of complexity. Others have taken a different approach to applying homotopy to robotics by augmenting existing sampling-based methods. CTopPRM is one such work that takes a dense Informed-PRM, and strategically constructs a sparse road map that captures distinct UVD classes.

There remains work to be done on how homotopy methods explicitly compare to more conventional methods, and how well they generalize to 3D. Homotopy seems like a simple problem in the 2D plane, but things that are true in 2D are not necessarily true in 3D. This introduces much complexity in the problem, which raises questions about computational tractability. While it remains to be seen whether the computation involved in homotopy methods is realistic in real-world applications, this is an area that has yet to be thoroughly investigated, and it is clear that incorporating homotopy information is able to yield a higher quality trajectory in robotics path planning.

References

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," IEEE Transactions on Robotics and Automation, vol. 12, no. 4, pp. 566–580, 1996.

[3] Boyu Zhou, Jie Pan, Fei Gao, Shaojie Shen: RAPTOR: Robust and Perception-aware Trajectory Replanning for Quadrotor Fast Flight. CoRR abs/2007.03465 (2020)

[4] Velez-Lopez, G.C.; Vazquez-Leal, H.; Hernandez-Martinez, L.; Sarmiento-Reyes, A.; Diaz-Arango, G.; Huerta-Chua, J.; Rico-Aniles, H.D.; Jimenez-Fernandez, V.M. A Novel Collision-Free Homotopy Path Planning for Planar Robotic Arms. Sensors 2022, 22, 4022. https://doi.org/10.3390/s22114022

[5] H. Vazquez-Leal, A. Marin-Hernandez, Y. Khan, A. Yıldırım, U. Filobello-Nino, R. Castaneda-Sheissa, V.M. Jimenez-Fernandez, Exploring collision-free path planning by using homotopy continuation methods, Applied Mathematics and Computation, Volume 219, Issue 14, 2013, Pages 7514-7532, ISSN 0096-3003, https://doi.org/10.1016/j.amc.2013.01.038.

[6] M. Novosad, R. Penicka and V. Vonasek, "CTopPRM: Clustering Topological PRM for Planning Multiple Distinct Paths in 3D Environments," in IEEE Robotics and Automation Letters, vol. 8, no. 11, pp. 7336-7343, Nov. 2023, doi: 10.1109/LRA.2023.3315539.

[7] G. Diaz-Arango, L. Hernandez-Martinez, A. Sarmiento-Reyes and H. Vazquez-Leal, "Fast and robust homotopy path planning method for mobile robotics," 2016 IEEE International Symposium on Circuits and Systems (ISCAS), Montreal, QC, Canada, 2016, pp. 2579-2582, doi: 10.1109/ISCAS.2016.7539120.

[8] Yang LIU, Zheng ZHENG, Fangyun QIN, Homotopy based optimal configuration space reduction for anytime robotic motion planning, Chinese Journal of Aeronautics, Volume 34, Issue 1, 2021, Pages 364-379, ISSN 1000-9361, https://doi.org/10.1016/j.cja.2020.09.036.

PIDformer: Transformer Meets Control Theory

Tam Nguyen [1] **César A. Uribe** [1] **Tan M. Nguyen** [† 2] **Richard G. Baraniuk** [† 1]

Abstract

In this work, we address two main shortcomings of transformer architectures: input corruption and rank collapse in their output representation. We unveil self-attention as an autonomous state-space model that inherently promotes smoothness in its solutions, leading to lower-rank outputs and diminished representation capacity. Moreover, the steady-state solution of the model is sensitive to input perturbations. We incorporate a Proportional-Integral-Derivative (PID) closed-loop feedback control system with a reference point into the model to improve robustness and representation capacity. This integration aims to preserve high-frequency details while bolstering model stability, rendering it more noise-resilient. The resulting controlled state-space model is theoretically proven robust and adept at addressing the rank collapse. Motivated by this control framework, we derive a novel class of transformers, PID-controlled Transformer (PIDformer), aimed at improving robustness and mitigating the rank-collapse issue inherent in softmax transformers. We empirically evaluate the model for advantages and robustness against baseline transformers across various practical tasks, including object classification, image segmentation, and language modeling.

1. Introduction

Transformer models (Vaswani et al., 2017) have shown remarkable achievements across various domains such as reinforcement learning (Chen et al., 2021; Janner et al., 2021), computer vision (Dosovitskiy et al., 2021b; Touvron et al., 2021; Zhao et al., 2021; Guo et al., 2021), natural language processing (Devlin et al., 2018; Al-Rfou et al., 2019; Child et al., 2019; Raffel et al., 2020) and other practical applications (Zhang et al., 2019; Gulati et al., 2020). At the core of transformers lies the self-attention mechanism, which computes weighted averages of token representations within a sequence based on the similarity scores between pairs of tokens, thus capturing diverse syntactic and semantic relationships effectively (Cho et al., 2014; Parikh et al., 2016). This flexibility in capturing relationships has been identified as a key factor contributing to the success of transformers.

1.1. Background: Self-Attention

Given a sequence of tokens $\mathbf{X}^\ell := [\boldsymbol{x}^\ell(1), \cdots, \boldsymbol{x}^\ell(N)]^\top$, $\mathbf{X}^\ell \in \mathbb{R}^{N \times D_x}$, the query, key and value matrices at layer ℓ-th are $\mathbf{Q}^\ell = \mathbf{X}{\mathbf{W}_Q^\ell}^\top$; $\mathbf{K}^\ell = \mathbf{X}{\mathbf{W}_K^\ell}^\top$; and $\mathbf{V}^\ell = \mathbf{X}{\mathbf{W}_V^\ell}^\top$, respectively. The weight matrix $\mathbf{W}_Q^\ell, \mathbf{W}_K^\ell \in \mathbb{R}^{D_{qk} \times D_x}$ and $\mathbf{W}_V^\ell \in \mathbb{R}^{D \times D_x}$. The attention mechanism computes the output of token i at layer ℓ-th as follows

$$\boldsymbol{u}^\ell(i) = \sum_{j=1}^{N} \mathrm{softmax}\Big(\boldsymbol{q}^\ell(i)^\top \boldsymbol{k}^\ell(j)/\sqrt{D_{qk}}\Big)\boldsymbol{v}^\ell(j), \quad (1)$$

where $\boldsymbol{q}^\ell(i)$ is the row i-th of $\mathbf{Q}^\ell$ and $\boldsymbol{k}^\ell(j), \boldsymbol{v}^\ell(j)$ are the row j-th of $\mathbf{K}^\ell, \mathbf{V}^\ell$, respectively. The softmax function computes the attention score between token i and j, for all $i, j = 1, \dots, N$. The self-attention (1) is referred to as softmax attention. Our work refers to a transformer that uses softmax attention as a softmax transformer.

Despite their remarkable success, transformers exhibit practical performance issues in their robustness and representation capacity. For example, recent studies (Mahmood et al., 2021; Madry et al., 2017; Zhou et al., 2022) have provided empirical evidence of Vision Transformer's susceptibility to adversarial attacks and common input perturbations, such as noise or blur. Additionally, deep transformer-based models have been observed to suffer from rank-collapse in their outputs, wherein token embeddings become increasingly similar as the model depth increases (Shi et al., 2022; Dong et al., 2021; Wang et al., 2022). This issue severely constrains the representation capacity of transformers, hindering their performance in various tasks. Addressing these issues is crucial for ensuring the reliability and effectiveness of transformer models across different applications.

1.2. Contribution

We introduce self-attention as a self-evolving state-space model (SSM) and provide insights into the non-robustness and rank-collapse issues inherent in transformers. Specif-

[†]Co-last authors [1]Department of Electrical & Computer Engineering, Rice University, Houston, USA [2]Department of Mathematics, National University of Singapore, Singapore. Correspondence to: Tam Nguyen <mn72@rice.edu>.

ically, we demonstrate that self-attention can be seen as a discretization of an SSM from a gradient flow, minimizing the nonlocal total variation (Gilboa & Osher, 2008) of an input signal and promoting smoothness. This characteristic leads to rank collapse and diminishes the output's representation capacity. Additionally, the steady-state solution of the SSM is sensitive to input perturbation. Motivated by this novel understanding, we propose the Proportional-Integral-Derivative (PID) control transformer, PIDformer, as a new transformer class that mitigates both issues. PIDformer is derived as a discretization of a PID-control integrated SSM proven to enhance the model's stability and representation capacity. Our contributions are four-fold.

1. We present a novel control framework for self-attention mechanisms, unveiling the connection between self-attention and the state-space model. Our analysis sheds light on the shortcomings of transformers, which exhibit non-robust behavior to input perturbations and are prone to rank collapse.

2. Motivated by these analyses, we propose PIDformer, a new class of transformers, that integrates a Proportional-Integral-Derivative (PID) controller into transformers. PIDformer enhances model robustness and effectively mitigates the rank-collapse issue.

3. We demonstrate how the connection between energy optimization and our controlled SSMs enhances the understanding of these models.

4. We theoretically prove that employing softmax self-attention is inherently sensitive to noise and tends to produce low-rank outputs. In contrast, our controlled SSM is guaranteed to exhibit superior robustness and avoid the rank-collapse issue.

We empirically demonstrate the advantages of PIDformers on various large-scale applications, including the ImageNet object classification (Deng et al., 2009) (under diverse input perturbations and robustness benchmarks), ADE20K image segmentation (Zhou et al., 2018), and WikiText-103 language modeling (Merity et al., 2017). tasks.

Organization. We structure our paper as follows: In Section 2, we introduce a control framework for self-attention, offering insights into the non-robustness and rank-collapse issues in transformer-based models. In Section 3, we incorporate a PID controller into the SSM, providing theoretical guarantees of its stability and ability to mitigate the rank-collapse issue. Subsequently, we developed PIDformer, a discretization of the PID-controlled SSM, and established the connection between these dynamics and energy optimization for further understanding. In Section 4, we empirically validate the benefits of PIDformer. We review related work in Section 5. Finally, we summarize our main contributions and provide additional results, details, and proofs in the Appendix.

2. A Control Framework for Self-Attention

Consider the value matrix of layer ℓ-th $\mathbf{V}^\ell := [\boldsymbol{v}^\ell(1), \cdots, \boldsymbol{v}^\ell(N)]^\top \in \mathbb{R}^{N\times D}$ in Section 1.1. Let $\Omega \subset \mathbb{R}$, $x \in \Omega$, and $\boldsymbol{v}(x,t) := [v_1(x,t), \dots, v_D(x,t)]^T$ be a real vector-valued function, $\boldsymbol{v} : \Omega \times [0,\infty) \to \mathbb{R}^D$, $\boldsymbol{v} \in L^2(\Omega \times [0,\infty))$. Assume the value matrix $\mathbf{V}^\ell$ discretizes the function $\boldsymbol{v}(x,t)$ on the spatial and time dimension. In the context of a control system, $\boldsymbol{v}(x)$ can be considered as the state signal of the following state-space model:

$$\frac{d\boldsymbol{v}(x,t)}{dt} = \int_\Omega (\boldsymbol{v}(y,t) - \boldsymbol{v}(x,t))K(x,y,t)dy + \boldsymbol{z}(x,t)$$
$$\boldsymbol{v}(x,0) = \boldsymbol{v}^0(x), \boldsymbol{z}(x,t) = \mathbf{0}, \forall x \in \Omega, \forall t \geq 0 \quad (2)$$

where $\boldsymbol{z} \in L^2(\Omega \times [0,\infty))$ is a control input and $\boldsymbol{v}^0$ is the initial state. The function $K(x,y,t)$ is the kernel function that captures the proximity of the signal $\boldsymbol{v}$ at positions x, y at time t. Here, the SSM is autonomous, as no control inputs or feedback are fed into the system. In this section, we illustrate that system in (2) induces smoothness to the signal by minimizing the nonlocal total variation (Gilboa & Osher, 2008) of the signal, hence losing detailed information as it evolves. Subsequently, we show that self-attention serves as a discretization of this dynamic. Lastly, we theoretically demonstrate that the SSM in 2 is vulnerable to input perturbation and representation collapse.

2.1. Connection between State Space Model and Nonlocal Variational Minimization

We show that the gradient flow aimed at minimizing the following nonlocal functional is a case of our SSM described in (2)

$$J(\boldsymbol{v}) = \frac{1}{2}\int_{\Omega\times\Omega} \|\boldsymbol{v}(x) - \boldsymbol{v}(y)\|_2^2 k(x,y)dxdy. \quad (3)$$

Here, $J(\boldsymbol{v})$, the sum of the square of the nonlocal derivative on the spatial dimension $\partial_y \boldsymbol{v}(x) = \big(\boldsymbol{v}(x) - \boldsymbol{v}(y)\big)\sqrt{k(x,y)}$ (Gilboa & Osher, 2008) , represents the non-local variation of the signal $\boldsymbol{v}$. $k(x,y)$ captures the proximity between position x and y in the signal. Minimizing $J(\boldsymbol{v})$ promotes the smoothness of $\boldsymbol{v}$ and penalizes high-frequency in the signal.

The gradient of J with respect to $\boldsymbol{v}$ is then given by

$$\nabla_{\boldsymbol{v}} J(\boldsymbol{v}) = \left[\frac{\partial J}{\partial v_1}, \frac{\partial J}{\partial v_2}, \dots, \frac{\partial J}{\partial v_D}\right]^T. \quad (4)$$

As shown in the Appendix B.10, the Frechet derivative of J with respect to v_j is

$$\frac{\partial J}{\partial v_j} = \int_\Omega (v_j(x) - v_j(y)(k(x,y) + k(y,x))dy. \quad (5)$$

Substituting the formula for $\partial J/\partial v_j$ in (5) into (4) for $\nabla_{\boldsymbol{v}} J(\boldsymbol{v})(x)$, we obtain the following gradient flow

$$\begin{aligned}\frac{d\boldsymbol{v}(x,t)}{dt} &= -\nabla_{\boldsymbol{v}} J(\boldsymbol{v}) \\ &= \int_\Omega \big(\boldsymbol{v}(y,t) - \boldsymbol{v}(x,t)\big)\big(k(x,y)+k(y,x)\big)dy,\end{aligned} \tag{6}$$

The autonomous state-space representation in (2) simplifies to this dynamic when $K(x,y,t) := k(x,y)+k(y,x)$, which is symmetric and time-invariant. In this scenario, the model reduces the total nonlocal variance of the signal, resulting in a smoother solution. This renders the model susceptible to rank collapse in the output representation. In Section 2.2, we prove that the model suffers from rank collapse regardless of whether $K(x,y,t)$ is symmetric.

Connection between SSM and self-attention. We show that a discretization of our SSM recovers the self-attention mechanism. Let $\boldsymbol{q},\boldsymbol{k} : \Omega\times[0,\infty) \to \mathbb{R}^{D_{qk}}$, $\boldsymbol{q},\boldsymbol{k} \in L^2(\Omega\times[0,\infty))$ be real vector-valued functions. Similar to $\boldsymbol{v}(x,t)$, we can discretize $\boldsymbol{q}(x,t),\boldsymbol{k}(x,t)$ on spatial dimension to attain the query vectors $\boldsymbol{q}^\ell(1),\dots,\boldsymbol{q}^\ell(N) \in \mathbb{R}^{D_{qk}}$, and the key vectors $\boldsymbol{k}^\ell(1),\dots,\boldsymbol{k}^\ell(N) \in \mathbb{R}^{D_{qk}}$ of layer ℓ-th.

Applying the Euler method to discretize (2) with the time step $\Delta t(x) := 1$, the update step of the system becomes

$$\begin{aligned}\boldsymbol{v}(x,t+1) &\approx \boldsymbol{v}(x,t) + \int_\Omega (\boldsymbol{v}(y,t)-\boldsymbol{v}(x,t))K(x,y,t)dy + \boldsymbol{z}(x,t) \\ &\approx \boldsymbol{v}(x,t) + \int_\Omega K(x,y,t)\boldsymbol{v}(y,t)dy - \boldsymbol{v}(x,t)\int_\Omega K(x,y,t)dy + \boldsymbol{z}(x,t)\end{aligned} \tag{7}$$

We define the proximity kernel as

$$K(x,y,t) := \frac{\exp\big(\boldsymbol{q}(x,t)^T\boldsymbol{k}(y,t)/\sqrt{D_{qk}}\big)}{\int_\Omega \exp\big(\boldsymbol{q}(x,t)^T\boldsymbol{k}(y',t)/\sqrt{D_{qk}}\big)dy'}.$$

, therefore

$$\int_\Omega K(x,y,t)dy = \int_\Omega \frac{\exp\big(\boldsymbol{q}(x,t)^T\boldsymbol{k}(y,t)/\sqrt{D_{qk}}\big)}{\int_\Omega \exp\big(\boldsymbol{q}(x,t)^T\boldsymbol{k}(y',t)/\sqrt{D_{qk}}\big)dy'}dy = 1$$

In addition, since in (2) $\boldsymbol{z}(x,t) = 0$ for all t, we obtain (7) as

$$\begin{aligned}&\boldsymbol{v}(x,t+1) \\ &\approx \int_\Omega \frac{\exp\big(\boldsymbol{q}(x,t)^T\boldsymbol{k}(y,t)/\sqrt{D_{qk}}\big)}{\int_\Omega \exp\big(\boldsymbol{q}(x,t)^T\boldsymbol{k}(y',t)/\sqrt{D_{qk}}\big)dy'}\boldsymbol{v}(y,t)dy.\end{aligned} \tag{8}$$

Using the Monte-Carlo method (Metropolis & Ulam, 1949) to approximate the integrals in (8) using the key vectors $\boldsymbol{k}(1),\dots,\boldsymbol{k}(N)$ and value vectors $\boldsymbol{v}(1),\dots,\boldsymbol{v}(N)$, we obtain

$$\boldsymbol{v}(x,t+1) \approx \sum_{j=1}^N \frac{\exp\big(\boldsymbol{q}(x)^T\boldsymbol{k}(j)/\sqrt{D_{qk}}\big)}{\sum_{j'=1}^N \exp\big(\boldsymbol{q}(x)^T\boldsymbol{k}(j')/\sqrt{D_{qk}}\big)}\boldsymbol{v}(j).$$

Discretizing $\boldsymbol{v}(x,t+1)$ on another 1-D grid, and as correspond time $t+1$ with layer $l+1$ we attain

$$\begin{aligned}\boldsymbol{v}^{\ell+1}(i) &\approx \sum_{j=1}^N \frac{\exp\big(\boldsymbol{q}(i)^T\boldsymbol{k}(j)/\sqrt{D_{qk}}\big)}{\sum_{j'=1}^N \exp\big(\boldsymbol{q}(i)^T\boldsymbol{k}(j')/\sqrt{D_{qk}}\big)}\boldsymbol{v}(j) \\ &\sum_{j=1}^N \mathrm{softmax}\Big(\boldsymbol{q}^\ell(i)^\top\boldsymbol{k}^\ell(j)/\sqrt{D_{qk}}\Big)\boldsymbol{v}^\ell(j).\end{aligned}$$

which recovers $\boldsymbol{u}^\ell(i)$, the output token i of self-attention at layer ℓ-th as in (1). As self-attention discretizes the SSM outlined in (2), it inherits the characteristics of the model, making it susceptible to input corruption and output rank collapse. These properties are theoretically demonstrated in Section 2.2.

2.2. Stability and Representation Collapse of the State Space Model

Model robustness is its ability to maintain high performance despite encountering uncertain or challenging scenarios such as noisy data, distribution shifts, or adversarial attacks (Wang & Bansal, 2018; Dong et al., 2020). Robustness also entails stability, wherein the model's output remains relatively unchanged even when the input is perturbed.

For the theoretical analysis of our SSMs, we assume that the kernel K is time-invariant, i.e., $K(x,y,t) = K(x,y)$. This assumption is practical in the context of transformers, particularly in deep transformer models, where the attention matrix tends to remain similar after the initial layers (Shi et al., 2022). The discretization of model in (2) on the spatial dimension gives

$$\frac{d\boldsymbol{v}(i,t)}{dt} = \sum_{j=1}^N (\boldsymbol{v}(j,t)-\boldsymbol{v}(i,t))K(i,j),$$

for $i,j = 1,2,\dots,N$ By choosing $K(i,j) := \mathrm{softmax}\big(\boldsymbol{q}(i)^T\boldsymbol{k}(j)/\sqrt{D_{qk}}\big)$, its corresponding matrix representation is obtained as

$$\mathbf{V}'(t)dt = \mathbf{K}\mathbf{V}(t) - \mathbf{V}(t), \mathbf{V}(0) = \mathbf{V}^0, \tag{9}$$

where $\boldsymbol{K}$ is a right-stochastic matrix with all positive entries. In the context of transformer, $\boldsymbol{K}$ is the attention matrix and $\mathbf{V} = [\boldsymbol{v}^0(1),\dots,\boldsymbol{v}^0(N)]^T$ is the value matrix at the first layer. Lemma 1 sheds light on the stability and representation collapse of the solution for the SSM in (2).

Lemma 1. *Given $\{\alpha_1,\alpha_2,\dots,\alpha_M\}, M \le N$, is the complex spectrum of $\mathbf{K}-\mathbf{I} \in \mathbb{R}^{N\times N}$. The solution of the ordinary differential equation (ODE) (9) is given by*

$$\mathbf{V}(t) = \boldsymbol{P}\exp(\boldsymbol{J}t)\boldsymbol{P}^{-1}\mathbf{V}^0, \tag{10}$$

where $\boldsymbol{P}\boldsymbol{J}\boldsymbol{P}^{-1}$ is the Jordan decomposition of $\boldsymbol{K}-\boldsymbol{I}$, $\boldsymbol{P}$ is invertible and contains the generalized eigenvectors of

$\boldsymbol{K}-\boldsymbol{I}$, and $\boldsymbol{J}=\mathbf{diag}(\boldsymbol{J}_{\alpha_1,m_1},\boldsymbol{J}_{\alpha_2,m_2},\dots,\boldsymbol{J}_{\alpha_M,m_M})$ is the Jordan form of matrix $\boldsymbol{K}-\boldsymbol{I}$ with, $\boldsymbol{J}_{\alpha_i,m_i}=\begin{bmatrix}\alpha_i & 1 & \dots & 0\\ \vdots & \ddots & & \vdots\\ & & \alpha_i & 1\\ 0 & \dots & & \alpha_i\end{bmatrix}\in\mathbb{R}^{m_i\times m_i}$, *for $i=1,\dots,M$ are Jordan blocks. Here, $\sum_{i=1}^{M}m_i=N$.*

The proof of Lemma 1 is shown in the Appendix B.2. Since $\mathbf{K}$ is a positive right-stochastic matrix, its largest and unique eigenvalue α_1 is 1 and $|\alpha_i|<1$ (see Theorem 4.1 in (Bandeira et al., 2020)), meaning $\mathrm{Re}(\alpha_i)\in[-1,1)$, for $i=2,\dots,M$. Hence, the matrix $\mathbf{K}-\mathbf{I}$, whose eigenvalues are $\alpha_1-1,\dots,\alpha_M-1$, has a unique largest eigenvalue of 0 and the real part of other eigenvalues in $[-2,0)$. This leads to the rank collapse of the steady-state solution, as stated in the following Lemma 2.

Lemma 2. $\lim_{t\to\infty}\mathbf{V}(t)=\begin{bmatrix}c_{1,1}\boldsymbol{p_1}, & \dots, & c_{1,D_x}\boldsymbol{p_1}\end{bmatrix}$, *where $\boldsymbol{p}_1$ is the eigenvector corresponds with the eigenvalue $(\alpha_1-1)=0$ of $\boldsymbol{K}-\boldsymbol{I}$, and $c_{1,1},\dots,c_{1,D_x}$ are the coefficients w.r.t $\boldsymbol{p}_1$ of the decomposition of $\boldsymbol{V}^0$'s columns in the Jordan basis (column vectors of $\boldsymbol{P}$).*

The proof of Lemma 2 is shown in the Appendix B.3. This yields two insights. Firstly, the steady-state solution of the system depends on the initial $\boldsymbol{V}^0$, implying that any perturbation in the input results in changes in the output. Secondly, the solution experiences rank collapse, with the rank of its steady state solution being 1 as $t\to\infty$. This indicates that our SSM in (2) not only yields a non-robust solution but also experiences information loss (low-rank output representation). As the self-attention mechanism discretizes the model in (2), it inherently exhibits both issues.

3. Transformer with PID-Controller for State-Space Representation

To counteract the loss of detailed information caused by smoothness and to bolster model stability, a PID controller is integrated into the state-space representation as follows:

$$\frac{d\boldsymbol{v}(x,t)}{dt}=\int_{\Omega}(\boldsymbol{v}(y,t)-\boldsymbol{v}(x,t))K(x,y,t)dy+\boldsymbol{z}(x,t)$$
$$\boldsymbol{z}(x,t)=\lambda_P\boldsymbol{e}(x,t)+\lambda_I\int_0^t\boldsymbol{e}(x,t)+\lambda_D\frac{d\boldsymbol{e}(x,t)}{dt}$$
$$\boldsymbol{v}(x,0)=\boldsymbol{v}^0(x),\boldsymbol{z}(x,0)=\mathbf{0}. \tag{11}$$

The regularizer term, denoted as $\boldsymbol{e}(x,t)=\boldsymbol{f}(x)-\boldsymbol{v}(x,t)$, encapsulates the loss of information as $\boldsymbol{v}(x,t)$ becomes smoother over time. Here, the reference function $\boldsymbol{f}(x)$ represents a high-frequency signal containing detailed information about the original inputs. We select $\boldsymbol{f}(x)$ as the scaled initial value function, denoted as $\beta\boldsymbol{v}(x,0)$. In the context of a transformer, we set $\boldsymbol{f}(i)=\beta\boldsymbol{v}^0(i)$, representing the value vector embedding at token index i of the first layer. This choice is motivated by our desire to have flexibility in determining the detailed information from the input signal we wish to preserve. This flexibility is governed by the parameter $\beta\in(0,1]$. The regularizer $\boldsymbol{e}(x,t)$ is fed back into the system, guiding the model to reintegrate the lost information while maintaining stability through three components: (P), (I), and (D).

- The (P) term is directly proportional to the regularizer, $e(x,t)$. In cases of substantial information loss, the control input $\boldsymbol{z}(x,t)$ should be proportionately large, determined by the gain factor λ_P, to reintroduce the lost information into the system. A small choice of λ_P results in slow convergence, while a large choice may lead to overshooting issues, causing instability in reaching the reference point.

- The (I) term accumulates all past errors, given by $\lambda_I\int_0^t\boldsymbol{e}(x,t)$. This component aids in reintroducing any persistent, long-term loss of information that might persist despite proportional control.

- Finally, the (D) term, $\lambda_D\frac{d\boldsymbol{e}(x,t)}{dt}$, anticipates future losses of information by considering the rate at which the error is changing. A more rapid change in error prompts a greater control effect, and the derivative term proves beneficial in enhancing the stability and responsiveness of the control system.

In this section, we unveil a connection between the two components, (P) and (I), of the SSM in (11) and different optimization methods applied to minimize a regularized functional. This functional is tailored to preserve the detailed information of the solution. Moreover, we show that the P-control (where $\lambda_I=\lambda_D=0$), PD-control ($\lambda_I=0$), and PID-controlled SSM in (11) are theoretically guaranteed to be more robust and mitigate the issue of rank collapse. Subsequently, we introduce the PID-controlled transformer (PIDformer), a novel architecture that enhances performance and robustness.

3.1. Connection between (P) and (I) Components with Different Optimization Methods

In Section 2.1, we have shown that the SSM in (2) implicitly performs a gradient descent to minimize the nonlocal variation $J(\boldsymbol{v})$, which leads to the loss of signal information. Now, we illustrate that the feedback-controlled state-space in (11), without the derivative (D) ($\lambda_D=0$), implicitly minimizes the following functional:

$$\begin{aligned} E(\boldsymbol{v}, \boldsymbol{f}) &= J(\boldsymbol{v}) + G(\boldsymbol{v}, \boldsymbol{f}) \\ &= \frac{1}{2} \int_{\Omega \times \Omega} \|\boldsymbol{v}(x) - \boldsymbol{v}(y)\|_2^2 k(x, y) dx dy \\ &+ \frac{\lambda}{2} \int_{\Omega} \|\boldsymbol{v}(x) - \boldsymbol{f}(x)\|_2^2 dx. \end{aligned} \tag{12}$$

where the data fidelity term $G(\boldsymbol{v}, \boldsymbol{f}) = \frac{\lambda}{2} \int_{\Omega} \|\boldsymbol{v}(x) - \boldsymbol{f}(x)\|_2^2 dx$ (Gilboa & Osher, 2008; 2007) is introduced to penalize significant information loss. This observation further validates that systems in (11) retains relevant information from the reference signal $\boldsymbol{f}$.

P-controlled SSM as gradient descent to minimize $E(\boldsymbol{v}, \boldsymbol{f})$. The gradient of E w.r.t $\boldsymbol{v}$ is $\nabla_{\boldsymbol{v}} E(\boldsymbol{v}) = \nabla_{\boldsymbol{v}} J(\boldsymbol{v}) + \lambda\big(\boldsymbol{v}(x) - \boldsymbol{f}(x)\big)$. The derivation of the derivative is given in Appendix B.10. Using the gradient descent method, we obtain the gradient flow:

$$\begin{aligned} &\frac{d\boldsymbol{v}(x, t)}{dt} = -\nabla_{\boldsymbol{u}} E(\boldsymbol{v}) \\ &= \int_{\Omega} \big(\boldsymbol{v}(y, t) - \boldsymbol{v}(x, t)\big)\big(k(x, y) + k(y, x)\big) dy \\ &+ \lambda\big(\boldsymbol{f}(x) - \boldsymbol{v}(x, t)\big). \end{aligned} \tag{13}$$

If we set $K(x, y, t) := k(x, y) + k(y, x)$ to be symmetric and time-invariant, and $\lambda_P = \lambda, \lambda_I = \lambda_D = 0$, the controlled system in (11) simplifies to the gradient flow of E in (13). It suggests that integrating the (P) component into the system in (2) minimizes the functional E and reintroduces the lost information to the system.

PI-controlled SSM as Bregman iteration to minimize $E(\boldsymbol{v}, \boldsymbol{f})$. An alternative to gradient descent, Bregman iteration (Yin et al., 2008; Zhang et al., 2010) iteratively refines the solution by minimizing a Bregman divergence, which measures the discrepancy between the current solution and a reference point. Given the convex functional $J(\boldsymbol{v})$, the Bregman divergence of J between $\boldsymbol{v}$ and $\boldsymbol{s} \in L^2(\Omega)$ is $D_J^p(\boldsymbol{v}, \boldsymbol{s}) := J(\boldsymbol{v}) - J(\boldsymbol{s}) - \langle \boldsymbol{p}, \boldsymbol{v} - \boldsymbol{s}\rangle$, where $\boldsymbol{p} \in \partial J(\boldsymbol{s})$, the subgradient of J at $\boldsymbol{s}$. $D_J^p(\boldsymbol{v}, \boldsymbol{s})$ captures the difference between $J(\boldsymbol{v})$ and the tangent plane $J(\boldsymbol{s}) - \langle \boldsymbol{p}, \boldsymbol{v} - \boldsymbol{s}\rangle$. The $\ell + 1$-th Bregman iteration to minimize $\min_{\boldsymbol{v}} J(\boldsymbol{v})$ with the contraint $G(\boldsymbol{v}, \boldsymbol{f})$ is given by:

$$\boldsymbol{v}^{\ell+1} = \arg\min_{\boldsymbol{v}} D_J^{p^\ell}(\boldsymbol{v}, \boldsymbol{v}^\ell) + G(\boldsymbol{v}, \boldsymbol{f}), p^\ell \in \partial J(\boldsymbol{v}^\ell) \tag{14}$$

The following Lemma 3 shows that the optimization problem in (14) can be turned into solving iterative subproblems.

Lemma 3. *Applying Bregman iteration to minimize $E(\boldsymbol{v}, \boldsymbol{f})$ involves solving iterative subproblems:*

$$\begin{aligned} &\boldsymbol{v}^{\ell+1} = \arg\min_{\boldsymbol{v}} J(\boldsymbol{v}) + \frac{\lambda}{2} \int_{\Omega} \|\boldsymbol{v}(x) - \boldsymbol{f}(x) - \boldsymbol{e}_{\mathrm{a}}^\ell(x)\|_2^2 dx \\ &\boldsymbol{e}_{\mathrm{a}}^\ell(x) = \sum_{m=1}^{\ell} \boldsymbol{e}^m(x) = \sum_{m=1}^{\ell} \big(\boldsymbol{f}(x) - \boldsymbol{v}^m(x)\big), \end{aligned} \tag{15}$$

The proof of Lemma 3 is in Appendix B.4. Here, the term $\boldsymbol{e}_{\mathrm{a}}^\ell(x)$ captures the accumulated information loss between the original and the smoothed signals $\boldsymbol{v}^m(x)$ of each iteration $m = 1, \dots, \ell$. Taking a one-step update in the direction of gradient descent (see Appendix B.11), we obtain

$$\begin{aligned} \boldsymbol{v}^{\ell+1}(x) &= \int_{\Omega} \big(\boldsymbol{v}^\ell(y) - \boldsymbol{v}^\ell(x)\big)\big(k(x, y) + k(y, x)\big) dy \\ &+ \boldsymbol{v}^\ell(x) + \lambda \boldsymbol{e}^\ell(x) + \lambda \boldsymbol{e}_{\mathrm{a}}^\ell(x). \end{aligned} \tag{16}$$

On the other hand, the Euler discretization with $\Delta t = 1$ of the PI-controlled state-space in (11) (as $\lambda_D = 0$) is:

$$\begin{aligned} \boldsymbol{v}^{\ell+1}(x) &= \boldsymbol{v}^\ell(x) + \int_{\Omega} \big(\boldsymbol{v}^\ell(y) - \boldsymbol{v}^\ell(x)\big) K(x, y) dy \\ &+ \lambda_P \boldsymbol{e}^\ell(x) + \lambda_I \sum_{m=1}^{\ell} \boldsymbol{e}^m(x). \end{aligned} \tag{17}$$

By selecting a time-independent $K(x, y, t) := k(x, y) + k(y, x)$ and setting $\lambda_P = \lambda_I = \lambda$, the update step of the PI-controlled model in (17) simplifies to the update step of Bregman iteration in (16). This connection suggests that the PI-controlled SSM minimizes $E(\boldsymbol{v}, \boldsymbol{f})$.

3.2. Stability and Representation Collapse of PID-Controlled State Space Model

In this section, we aim to show that: (i) Integrating the (P) term enhances robustness against input perturbations and mitigates rank collapse of the output representation; (ii) Adding the (D) term in PD-control further stabilizes the system by mitigating rapid and unstable changes of $\mathbf{V}(t)$, (iii) finally, integrating the (I) term in the PID-controlled SSM described in (11) guarantees system stability, making it robust to input corruption. Following the same assumption in Section 2.2, we assume that $K(x, y, t)$ is time-invariant for our theoretical analysis in this section.

3.2.1. Analysis of P-control SSM

Robustness of P-controlled SSM. From the SSM in (11), by choosing $\lambda_I = \lambda_D = 0$, and applying Euler discretization on the spatial domain, the P-control model is given as:

$$\begin{aligned} \frac{d\boldsymbol{v}(i, t)}{dt} &= \sum_{j=1}^{N} (\boldsymbol{v}(j, t) - \boldsymbol{v}(i, t)) K(i, j) \\ &+ \lambda_P\big(\boldsymbol{f}(i) - \boldsymbol{v}(i, t)\big), \end{aligned} \tag{18}$$

for $i,j = 1,2,\dots,N$, and $K(i,j) := \mathrm{softmax}\big(\boldsymbol{q}(i)^T\boldsymbol{k}(j)/\sqrt{D_{qk}}\big)$. The corresponding matrix representation is given as

$$\frac{d\mathbf{V}(t)}{dt} = \mathbf{K}\mathbf{V}(t) - (\lambda_P+1)\mathbf{V}(t) + \lambda_P\boldsymbol{F}, \mathbf{V}(0) = \mathbf{V}^0. \tag{19}$$

where $\boldsymbol{F} = [\boldsymbol{f}(1),\dots,\boldsymbol{f}(N)]^T$. The following Lemma 4 help us analyze the stability and representation collapse of the solution for the SSM in (19). Here, since the eigenvalues of $\mathbf{K}$ the has the real part in $[0,1]$, $\lambda_P + 1$ ($\lambda_P > 0$) can not be one of them. This implies that $\det(K - (\lambda_P+1)I) \neq 0$ hence the matrix is non-singular.

Lemma 4. *Let $\boldsymbol{B} := \mathbf{K} - (\lambda_P+1)\mathbf{I} \in \mathbb{R}^{N\times N}$, the solution of the ordinary differential equation (19) is*

$$\mathbf{V}(t) = \exp(\boldsymbol{B}t)(\mathbf{V}^0 + \boldsymbol{B}^{-1}\boldsymbol{F}) - \lambda_P\boldsymbol{B}^{-1}\boldsymbol{F}. \tag{20}$$

If $\boldsymbol{B}$ has only eigenvalues with negative real parts, then $\lim_{t\to\infty}\boldsymbol{V}(t) = -\lambda_P\boldsymbol{B}^{-1}\boldsymbol{F}$.

The proof of Lemma 4 is shown in the Appendix B.5. As shown in Section 2.2, since the eigenvalues of $\boldsymbol{K}$ has $\mathrm{Re}(\alpha_i) \in [-1,1], i = 1,\dots,M$, therefore the real parts of eigenvalues of $\boldsymbol{B}$ must be in the range $[-2-\lambda_P, -\lambda_p]$, which are all negative. As the result of 4, the steady state solution in (20) $\lim_{t\to\infty}\boldsymbol{V}(t) = -\lambda_P\boldsymbol{B}^{-1}\boldsymbol{F}$. Therefore, adding any perturbation to the initial state $\mathbf{V}^0$ does not change the steady state solution. However, in our context of a transformer, the perturbation also affects the reference point $\boldsymbol{F}$, which is chosen to be a scaled $\beta\mathbf{V}^0$, leading to the steady state solution becomes $-\lambda_P\beta\boldsymbol{B}^{-1}\mathbf{V}^0$. Fortunately, the P-control system allows the error caused by perturbation to be as neglectable as desired. The following Proposition 1 confirms the robustness of the P-control SSM.

Proposition 1. *Given the coefficient $\lambda_P > 0$ in (11), and any arbitrary $\bar{\epsilon}, \delta > 0$, by adding the perturbation $\boldsymbol{\epsilon} \in \mathbb{R}^{N\times D}$, $\|\boldsymbol{\epsilon}\|_\infty \le \bar{\epsilon}$ to $\mathbf{V}^0$, the corresponding change in the steady state solution of the system in (19) is independent of λ_P and becomes negligible with an amount of at most δ if*

$$\beta \le \delta/\bar{\epsilon}. \tag{21}$$

The proof of Proposition 1 is shown in the Appendix B.6. Proposition 1 suggests that we can select the hyper-parameter β to make the impact of input perturbation on the output as small as desired.

P-controlled SSM on representation collapse. Since $\boldsymbol{B}^{-1}$ is full rank ($\boldsymbol{B}$ is non-singular), hence $\mathrm{rank}(-\lambda_P\boldsymbol{B}^{-1}\boldsymbol{F}) = \mathrm{rank}(\boldsymbol{F})$ (Strang, 2006). In the case of a transformer, when choosing $\boldsymbol{F} = \beta\mathbf{V}^0$, the rank of the steady state solution equals the rank of the input $\mathbf{V}^0$. This implies that the P-control dynamic in (19) prevents rank collapse.

3.2.2. Analysis of PD-controlled SSM

Since $\lambda_D\frac{d\boldsymbol{e}(x,t)}{dt} = \lambda_D\frac{d}{dt}(\boldsymbol{f}(x) - \boldsymbol{v}(x,t)) = -\lambda_D\frac{d\boldsymbol{v}(x,t)}{dt}$, from the SSM in (11), by choosing $\lambda_I = 0$, $K(i,j) := \mathrm{softmax}\big(\boldsymbol{q}(i)^T\boldsymbol{k}(j)/\sqrt{D_{qk}}\big)$ for $i,j = 1,2,\dots,N$, and applying Euler discretization on the spatial domain, the PD-control model can be represented in the matrix form:

$$\begin{aligned}\mathbf{V}'(t) &= \mathbf{K}\mathbf{V}(t) - (\lambda_P+1)\mathbf{V}(t) + \lambda_P\boldsymbol{F} - \lambda_D\mathbf{V}'(t)\\ &= \frac{1}{1+\lambda_D}\big(\boldsymbol{K} - (\lambda_P+1)\boldsymbol{I}\big)\mathbf{V}(t) + \frac{\lambda_P}{1+\lambda_D}\boldsymbol{F},\end{aligned} \tag{22}$$

with $\mathbf{V}(0) = \mathbf{V}^0$. The solution of (22) is provided in the following Lemma 5.

Lemma 5. *Let $\boldsymbol{B} := \mathbf{K} - (\lambda_P+1)\mathbf{I} \in \mathbb{R}^{N\times N}$, the solution of the ordinary differential equation (22) is*

$$\mathbf{V}(t) = \exp\Big(\frac{1}{1+\lambda_D}\boldsymbol{B}t\Big)(\mathbf{V}^0 + \boldsymbol{B}^{-1}\boldsymbol{F}) - \lambda_P\boldsymbol{B}^{-1}\boldsymbol{F}.$$

and $\lim_{t\to\infty}\boldsymbol{V}(t) = -\lambda_P\boldsymbol{B}^{-1}\boldsymbol{F}$.

The proof of Lemma 5 is provided in Appendix B.7. This intriguing result suggests two key insights. Firstly, incorporating the (D) component into the P-control system does not alter the steady state of the solution. Consequently, the solution of the PD-controlled SSM retains the advantages of a P-control model, including avoiding rank collapse and ensuring bounded error. Secondly, the derivative term offers an additional benefit of further stabilizing the system by decreasing the eigenvalue by a factor of $1/(1+\lambda_D)$, thereby mitigating rapid changes in $\mathbf{V}(t)$.

3.2.3. Analysis of PID-controlled SSM

Following the same analysis in Section 3.2.1, by choosing $K(i,j) := \mathrm{softmax}\big(\boldsymbol{q}(i)^T\boldsymbol{k}(j)/\sqrt{D_{qk}}\big)$ and discretizing on the spatial domain, the matrix representation of the PID-controlled SSM reduced from (11) becomes

$$\begin{aligned}\mathbf{V}'(t) = &\frac{1}{\lambda_D+1}\Bigg(\big(\mathbf{K} - (\lambda_P+1)\boldsymbol{I}\big)\mathbf{V}(t)\\ &+ \lambda_I\int_0^t(\boldsymbol{F} - \mathbf{V}(t))dt + \lambda_P\boldsymbol{F}\Bigg),\end{aligned} \tag{23}$$

where $\mathbf{V}(0) = \mathbf{V}^0$. To deal with the integral in (23), we take the derivative of both sides, the equation becomes $\mathbf{V}''(t) = \frac{1}{\lambda_D+1}\big(\big(\mathbf{K} - (\lambda_P+1)I\big)\mathbf{V}'(t) - \lambda_I\mathbf{V}(t)\big)$, which is turned into a system of 1-st order differential equation:

$$\begin{bmatrix}\mathbf{V}'(t)\\ \mathbf{V}''(t)\end{bmatrix} = \begin{bmatrix}\mathbf{0} & \boldsymbol{I}\\ -\dfrac{\lambda_I\boldsymbol{I}}{\lambda_D+1} & \dfrac{\boldsymbol{K} - (\lambda_P+1)\boldsymbol{I}}{\lambda_D+1}\end{bmatrix}\begin{bmatrix}\mathbf{V}(t)\\ \mathbf{V}'(t)\end{bmatrix}, \tag{24}$$

where $\mathbf{V}(0) = \mathbf{V}^0$, and $\mathbf{V}'(0) = \frac{1}{\lambda_D + 1}\big((\boldsymbol{K} - (\lambda_P + 1))\mathbf{V}^0 + \lambda_P \boldsymbol{F}\big)$. To gain robustness, the steady state solution of the model should be independent of any perturbation of the input $\mathbf{V}_0$ The following Proposition 2 illustrates the stability of the system.

Proposition 2. *For any $\lambda_P, \lambda_I, \lambda_D > 0$, the system in (24) has a stable solution.*

The proof of Proposition 2 is in the Appendix B.8. The Proposition implies that the PID-controlled SSM in (11) remains robust and stable for any selection of positive values for $\lambda_P, \lambda_I, \lambda_D$.

3.3. Transformer with PID Control

By applying the Euler discretization with time step $\Delta t = 1$, initializing $\boldsymbol{v}$ at $t = 0$ as $\boldsymbol{v}(x, 0) = v^0(x)$, and choosing

$$K(x, y, t) := \frac{\exp\big(\boldsymbol{q}(x,t)^T \boldsymbol{k}(y,t)/\sqrt{D_{qk}}\big)}{\int_\Omega \exp\big(\boldsymbol{q}(x,t)^T \boldsymbol{k}(y',t)/\sqrt{D_{qk}}\big)dy'},$$

the update step of PID-controlled SSM in (11) becomes:

$$\begin{aligned}
&\boldsymbol{v}^{\ell+1}(x) \\
&\approx \int_\Omega \big(\boldsymbol{v}^\ell(y) - \boldsymbol{v}^\ell(x)\big)\frac{\exp\big(\boldsymbol{q}^\ell(x)^T \boldsymbol{k}^\ell(y)/\sqrt{D_{qk}}\big)}{\int_\Omega \exp\big(\boldsymbol{q}^\ell(x)^T \boldsymbol{k}^\ell(y')/\sqrt{D_{qk}}\big)dy'}dy \\
&+ \boldsymbol{v}^\ell(x) + \lambda_P \boldsymbol{e}^\ell(x) + \lambda_I \sum_{m=1}^{\ell} \boldsymbol{e}^m(x) + \lambda_D(\boldsymbol{e}^\ell(x) - \boldsymbol{e}^{\ell-1}(x)),
\end{aligned} \tag{25}$$

where $\boldsymbol{e}^m(x) = \boldsymbol{f}(x) - \boldsymbol{v}^m(x)$ for $m = 1, \ldots, \ell$. Applying the Monte-Carlo method to approximate the integrals in (25) and discretizing $\boldsymbol{v}^{l+1}(x)$, $\boldsymbol{v}^m(x)$, and $\boldsymbol{v}_0(x)$ on a spatial dimension, and by choosing $\boldsymbol{f}(x) = \boldsymbol{v}(x)$, we attain the output of the following novel PID-attention at layer ℓ-th is defined as

Definition 1 (PID-control Transformer (PIDformer))**.** *Given a set of key and value vectors $\{\boldsymbol{k}^\ell(j), \boldsymbol{v}^\ell(j)\}_{j=1}^N$ in each layer ℓ, $\ell = 1, \ldots, L$, for each query vector $\mathbf{q}^\ell(i)$, $i = 1, \ldots, N$, in the same layer, the self-attention unit at layer ℓ in a PID-control Transformer (PIDformer) computes the corresponding output vector $\boldsymbol{u}^\ell(i)$ of the query $\boldsymbol{q}^\ell(i)$ by the following attention formula:*

$$\begin{aligned}
\boldsymbol{u}^\ell(i) = &\sum_{j=1}^{N} \mathrm{softmax}\Big(\boldsymbol{q}^\ell(i)^\top \boldsymbol{k}^\ell(j)/\sqrt{D_{qk}}\Big)\boldsymbol{v}^\ell(y) \\
&+ \lambda_P \boldsymbol{e}^\ell(i) + \lambda_I \sum_{m=1}^{\ell} \boldsymbol{e}^m(i) + \lambda_D(\boldsymbol{e}^\ell(i) - \boldsymbol{e}^{\ell-1}(i)),
\end{aligned} \tag{26}$$

where $\boldsymbol{e}^\ell = \boldsymbol{v}^0 - \boldsymbol{v}^\ell$, $\boldsymbol{v}^0(1), \ldots, \boldsymbol{v}^0(N) \in \mathbb{R}^D$ are the value vectors in the first layer of PIDformer.

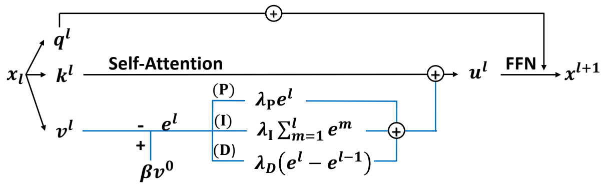


Figure 1. Our proposed PIDformer model at each layer.

Since PID-attention is a discretization of the controlled SSM in (11), it is inherently a more robust attention mechanism. Fig. 1 illustrates the architecture of PIDformer.

4. Experimental Results

In this section, we empirically demonstrate the advantages of our proposed PIDformer approach across multiple tasks, including ImageNet classification (Deng et al., 2009), ADE20K image segmentation (Zhou et al., 2018), and language modeling on WikiText-103 (Merity et al., 2017). Our objectives are to: (i) illustrate that PIDformer significantly outperforms the transformer baseline with softmax-attention across diverse tasks, (ii) highlight that the PID DeiT model exhibits significantly higher robustness than softmax attention under various adversarial attacks, and for out-of-distribution generalization, (iii) demonstrate that PID DeiT does not suffer from rank collapse in output representation. Throughout our experiments, we compare the performance of our proposed models with baselines of the same configuration. For additional details regarding datasets, models, and training procedures, please refer to Appendix A.

Object Classification on ImageNet. To demonstrate the advantage of our PIDformer, we compare it with the DeiT baseline (Touvron et al., 2021) on the ImageNet image classification task. Our PID DeiT surpasses the DeiT baseline, as shown in Table 1. Notably, our model performs significantly better than the baseline under white-box attacks, including fast gradient sign method (FGSM) (Dong et al., 2020), projected gradient descent method (PGD) (Tramer & Boneh, 2019b); score-based black-box attack method SPSA (Uesato et al., 2018); and sparse L_1 descent (SLD) (Tramer & Boneh, 2019a) method as well as noise-adding attack. Moreover, the last four rows of Table 1 demonstrate that PID DeiT is consistently more robust than the DeiT baseline under other adversarial examples and out-of-distribution dataset, including the Imagenet-C (common data corruption and perturbations, such as adding noise and blurring the images) (Hendrycks & Dietterich, 2019), Imagenet-A (adversarial examples) (Hendrycks et al., 2021b), Imagenet-R (out of distribution generalization) (Hendrycks et al., 2021a), and Imagenet-O(out-of-distribution detection) (Hendrycks et al., 2021b) datasets. Furthermore, in Appendix C.1, we visualize the performance gap between PID DeiT and the baseline DeiT model under attacks with escalating perturbation levels. This result demonstrates the significant advantages PIDformer has over the baseline model in terms of

Table 1. Evaluation of PID DeiT versus Softmax DeiT on the clean ImageNet validation set, as well as under various adversarial attacks and out-of-distribution datasets.

Attack	Metric/Model	*Softmax DeiT*	PID DeiT (%)
Clean	Top-1 Acc (%)	72.17	**73.13**
	Top-5 Acc (%)	91.02	**91.76**
FGSM	Top-1 Acc (%)	33.64	**38.52**
	Top-5 Acc (%)	68.18	**72.53**
PGD	Top-1 Acc (%)	12.02	**15.08**
	Top-5 Acc (%)	34.99	**39.69**
SPSA	Top-1 Acc (%)	65.75	**67.98**
	Top-5 Acc (%)	90.07	**90.58**
SLD	Top-1 Acc (%)	69.32	**70.84**
	Top-5 Acc (%)	90.8	**91.43**
Noise	Top-1 Acc (%)	69.2	**70.87**
	Top-5 Acc (%)	89.67	**90.77**
Imagenet-A	Top-1 Acc (%)	6.90	**8.82**
Imagenet-R	Top-1 Acc (%)	32.83	**34.89**
Imagenet-C	mCE (↓)	71.20	**68.41**
Imagenet-O	AUPR	17.47	**19.22**

Table 2. Single-scale (SS) MIoU and multi-scale MIoU (MS) of the PID DeiT vs. the DeiT on the ADE20K image segmentation.

Model/Metric	SS MIoU	MS MIoU (%)
Softmax DeiT	35.72	36.68
PID DeiT	**37.42**	**38.28**

Table 3. Test and valid perplexity (Test PPL and Valid PPL) on WikiText-103 of PIDformer compared to the softmax transformer.

Method/Metric	Valid PPL	Test PPL
Softmax Transformer	33.15	34.29
PIDformer	**32.44**	**33.45**

robustness, further confirming the benefits of our model.

Image Segmentation on ADE20K dataset. We evaluate the performance of Segmenter models (Strudel et al., 2021) using both PID DeiT and DeiT backbones on the ADE20K image segmentation task (Zhou et al., 2017), as outlined in Table 2. The outcomes illustrate significant performance enhancements obtained by employing the PID DeiT backbone instead of the DeiT backbone across both single-scale (SS) and multi-scale (MS) Mean Intersection over Union (MIoU) metrics.

Language Model on WikiText-103. In addition to computer vision tasks, we evaluate our model's performance in the language modeling task on the WikiText-103 dataset (Table 3). Our PIDformer language model surpasses the softmax transformer model (Xiong et al., 2021) in test and valid perplexity. These results, combined with findings across various tasks, empirically demonstrate the significant advantages of PIDformer models.

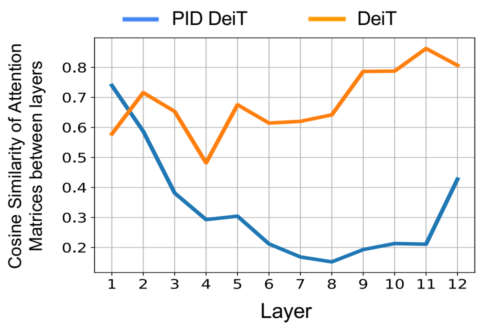


Figure 2. The cosine *similarity* of token representations in PID DeiT compared to baseline DeiT models across layers for ImageNet classification. The DeiT baseline demonstrates representation rank collapse as tokens become increasingly similar as depth increases. In contrast, PID DeiT models exhibit significantly greater diversity in tokens, indicating a mitigation in rank-collapse.

Representation Collapse Analysis. We empirically show PIDformer's effectiveness in addressing rank collapse in transformers. In Fig. 2, we compare token representation cosine similarity across layers in PID DeiT and softmax baseline models pretrained on Imagenet. PID DeiT exhibits significantly lower similarity, especially in later layers, alleviating rank collapse and enhancing token embedding diversity. Further details are in Appendix A.6.

5. Related Work

Robust transformer. Ensuring the generalization and robustness of both vision transformer and language model remains an ongoing research focus. Large language models are vulnerable to input corruption (Wang et al., 2021; Peyrard et al., 2022; Jin et al., 2020; Zang et al., 2019), posing a challenge in developing robust real-world applications that can withstand unforeseen adversarial threats. For ViTs, investigations into model robustness against adversarial attacks, domain shifts, and out-of-distribution data are crucial for real-world deployment. Techniques such as data augmentation, regularization, and adversarial training are actively explored to enhance the robustness and generalization capabilities of ViTs. Many investigations (e.g., (Yuan et al., 2023; Paul & Chen, 2022; Mahmood et al., 2021; Bhojanapalli et al., 2021; Madry et al., 2017; Mao et al., 2022; Zhou et al., 2022)) have attempted to explain and improve the resilience of ViT models against typical adversarial attacks. For example, (Mahmood et al., 2021) empirically mitigates ViT's vulnerability to white-box adversarial attacks by introducing a simple ensemble defense strategy that notably enhanced robustness without sacrificing accuracy on clean data.

Rank-collapse in transformer. Rank collapse in deep transformers, observed across domains from natural language processing (Shi et al., 2022) to computer vision (Wang et al., 2022; Dong et al., 2021), is evident. In computer vision, Zhou et al. (2021) find that adding more layers to the Vision Transformer (ViT) (Dosovitskiy et al., 2021a) quickly saturates its performance. Moreover, their experiments show that a 32-layer ViT performs worse than a 24-layer ViT, attributed to token representations becoming identical with increasing model depth. To address this matter, (Wang et al., 2022) discovers that self-attention functions as a low-pass filter, causing token representations in ViTs to be smoothed. Furthermore, (Shi et al., 2022) identifies a similar phenomenon in BERT (Devlin et al., 2018), and investigates rank-collapse from a graph perspective. The study employs hierarchical fusion techniques to retain the output of self-attention across all layers. Our work is orthogonal to the existing method as we develop a control framework to tackle the non-robustness and rank-collapse issues in transformers.

6. Concluding Remarks

In this paper, we present a novel control framework for self-attention mechanisms, revealing their inherent non-robustness and susceptibility to rank collapse in token representation. Leveraging this control perspective, we introduce the PIDformer, a novel PID-control Transformer designed to enhance robustness and mitigate the rank-collapse issue. Empirical validation across a range of large-scale applications, including ImageNet object classification (under various input perturbations and robustness benchmarks), ADE20K object segmentation, and WikiText-103 language modeling, confirms PIDformer's benefits. A limitation of our paper is the oversight regarding the privacy-preserving aspects of PIDformer. Exploring the potential of controlled transformers in enhancing privacy-preserving techniques is an intriguing avenue for future research.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Al-Rfou, R., Choe, D., Constant, N., Guo, M., and Jones, L. Character-level language modeling with deeper self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3159–3166, 2019.

Bandeira, A. S., Singer, A., and Strohmer, T. *Mathematics of Data Science*. 2020. URL https://people.math.ethz.ch/~abandeira/BandeiraSingerStrohmer-MDS-draft.pdf.

Bhojanapalli, S., Chakrabarti, A., Glasner, D., Li, D., Unterthiner, T., and Veit, A. Understanding robustness of transformers for image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10231–10241, 2021.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://www.aclweb.org/anthology/D14-1179.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dong, Y., Fu, Q.-A., Yang, X., Pang, T., Su, H., Xiao, Z., and Zhu, J. Benchmarking adversarial robustness on image classification. In *proceedings of the IEEE/CVF* *conference on computer vision and pattern recognition*, pp. 321–331, 2020.

Dong, Y., Cordonnier, J.-B., and Loukas, A. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning*, pp. 2793–2803. PMLR, 2021.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021a. URL https://openreview.net/forum?id=YicbFdNTTy.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=YicbFdNTTy.

Gilboa, G. and Osher, S. Nonlocal linear image regularization and supervised segmentation. *Multiscale Model. Simul.*, 6:595–630, 2007.

Gilboa, G. and Osher, S. Nonlocal operators with applications to image processing. *Multiscale Model. Simul.*, 7: 1005–1028, 2008.

Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

Guo, M.-H., Cai, J.-X., Liu, Z.-N., Mu, T.-J., Martin, R. R., and Hu, S.-M. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8340–8349, 2021a.

Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., and Song, D. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15262–15271, 2021b.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34: 1273–1286, 2021.

Jin, D., Jin, Z., Zhou, J. T., and Szolovits, P. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 8018–8025, 2020.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Mahmood, K., Mahmood, R., and Van Dijk, M. On the robustness of vision transformers to adversarial examples. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7838–7847, 2021.

Mao, X., Qi, G., Chen, Y., Li, X., Duan, R., Ye, S., He, Y., and Xue, H. Towards robust vision transformer. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 12042–12051, 2022.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Metropolis, N. and Ulam, S. The monte carlo method. *Journal of the American Statistical Association*, 44(247): 335–341, 1949. ISSN 01621459. URL http://www.jstor.org/stable/2280232.

Morača, N. Upper bounds for the infinity norm of the inverse of sdd and s-sdd matrices. *Journal of Computational and Applied Mathematics*, 206(2):666–678, 2007. ISSN 0377-0427. doi: https://doi.org/10.1016/j.cam.2006.08.013. URL https://www.sciencedirect.com/science/article/pii/S0377042706005139.

Parikh, A., Täckström, O., Das, D., and Uszkoreit, J. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2249–2255, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1244. URL https://www.aclweb.org/anthology/D16-1244.

Paul, S. and Chen, P.-Y. Vision transformers are robust learners. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 36, pp. 2071–2081, 2022.

Peyrard, M., Ghotra, S., Josifoski, M., Agarwal, V., Patra, B., Carignan, D., Kiciman, E., Tiwary, S., and West, R. Invariant language modeling. In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5728–5743, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.387. URL https://aclanthology.org/2022.emnlp-main.387.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3): 211–252, 2015.

Schlag, I., Irie, K., and Schmidhuber, J. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pp. 9355–9366. PMLR, 2021.

Shi, H., GAO, J., Xu, H., Liang, X., Li, Z., Kong, L., Lee, S. M. S., and Kwok, J. Revisiting over-smoothing in BERT from the perspective of graph. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=dUV91uaXm3.

Silvester, J. R. Determinants of block matrices. *The Mathematical Gazette*, 84(501):460–467, 2000. ISSN 00255572. URL http://www.jstor.org/stable/3620776.

Strang, G. *Linear algebra and its applications*. Thomson, Brooks/Cole, Belmont, CA, 2006. ISBN 0030105676 9780030105678 0534422004 9780534422004. URL http://www.amazon.com/Linear-Algebra-Its-Applications-Edition/dp/0030105676.

Strudel, R., Garcia, R., Laptev, I., and Schmid, C. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 7262–7272, 2021.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jegou, H. Training data-efficient image transformers distillation through attention. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10347–10357. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/touvron21a.html.

Tramer, F. and Boneh, D. Adversarial training and robustness for multiple perturbations. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019a. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/5d4ae76f053f8f2516ad12961ef7fe97-Paper.pdf.

Tramer, F. and Boneh, D. Adversarial training and robustness for multiple perturbations. *Advances in neural information processing systems*, 32, 2019b.

Uesato, J., O'Donoghue, B., Kohli, P., and van den Oord, A. Adversarial risk and the dangers of evaluating against weak attacks. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5025–5034. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/uesato18a.html.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Wang, B., Wang, S., Cheng, Y., Gan, Z., Jia, R., Li, B., and Liu, J. Infobert: Improving robustness of language models from an information theoretic perspective. 2021. Publisher Copyright: © 2021 ICLR 2021 - 9th International Conference on Learning Representations. All rights reserved.; 9th International Conference on Learning Representations, ICLR 2021 ; Conference date: 03-05-2021 Through 07-05-2021.

Wang, P., Zheng, W., Chen, T., and Wang, Z. Anti-oversmoothing in deep vision transformers via the fourier domain analysis: From theory to practice. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=O476oWmiNNp.

Wang, Y. and Bansal, M. Robust machine comprehension models via adversarial training. In Walker, M., Ji, H., and Stent, A. (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 575–581, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2091. URL https://aclanthology.org/N18-2091.

Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G., Li, Y., and Singh, V. Nyströmformer: A Nyström-based Algorithm for Approximating Self-Attention. 2021.

Yin, W., Osher, S., Goldfarb, D., and Darbon, J. Bregman iterative algorithms for l(1)-minimization with applications to compressed sensing. *Siam Journal on Imaging Sciences - SIAM J IMAGING SCI*, 1, 01 2008. doi: 10.1137/070703983.

Yuan, Z., Zhou, P., Zou, K., and Cheng, Y. You are catching my attention: Are vision transformers bad learners under backdoor attacks? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24605–24615, 2023.

Zang, Y., Qi, F., Yang, C., Liu, Z., Zhang, M., Liu, Q., and Sun, M. Word-level textual adversarial attacking as combinatorial optimization. *arXiv preprint arXiv:1910.12196*, 2019.

Zhang, S., Yao, L., Sun, A., and Tay, Y. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

Zhang, X., Burger, M., Bresson, X., and Osher, S. Bregmanized nonlocal regularization for deconvolution and sparse reconstruction. *SIAM Journal on Imaging Sciences*, 3(3):253–276, 2010. doi: 10.1137/090746379. URL https://doi.org/10.1137/090746379.

Zhao, H., Jiang, L., Jia, J., Torr, P. H., and Koltun, V. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16259–16268, 2021.

Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., and Torralba, A. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 633–641, 2017.

Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., and Torralba, A. Semantic understanding of scenes through the ade20k dataset, 2018.

Zhou, D., Kang, B., Jin, X., Yang, L., Lian, X., Jiang, Z., Hou, Q., and Feng, J. Deepvit: Towards deeper vision transformer, 2021.

Zhou, D., Yu, Z., Xie, E., Xiao, C., Anandkumar, A., Feng, J., and Alvarez, J. M. Understanding the robustness in vision transformers. In *International Conference on Machine Learning*, pp. 27378–27394. PMLR, 2022.

A. Additional Details on the Experiments in Section 4

This section provides datasets, models, and training details for experiments in Section 4.

A.1. Image Classification on Imagenet

Datasets and Metrics. The ImageNet dataset, as described in (Deng et al., 2009; Russakovsky et al., 2015), consists of 1.28 million images for training and $50,000$ images for validation, covering the classification of 1000 different categories. Performance evaluation is based on top-1 and top-5 accuracies.

Models and Baselines. Our baseline model is the DeiT-tiny model (Touvron et al., 2021), which consists of 12 transformer layers, 3 attention heads per layer, and a model dimension of 192. For model setting and setting and configuration, we follow (Touvron et al., 2021). Their implementation is available at https://github.com/facebookresearch/deit. The $\lambda_P, \lambda_I, \lambda_D$, and β used for our PID DeiT method is $0.8, 0.5, 0.05$, and 0.1, respectively.

A.2. Image Segmentation on ADK20 dataset

Datasets and Metrics. The ADE20K dataset is renowned for its incorporation of complex scenes featuring detailed labels, establishing it as one of the most rigorous semantic segmentation datasets. It comprises a training set of $20,210$ images covering 150 semantic classes. Furthermore, the dataset includes $2,000$ images in the validation set and $3,352$ images in the test set. Performance in this task is evaluated using the Single-scale mean Intersection over Union (SS mIoU) and Multi-scale (MS mIoU) metrics.

Models and baselines. The training configuration and setting for our models are followed by (Strudel et al., 2021). The baseline model is finetuned with the pretrained DeiT-tiny backbone while our segmenter model used the pretrained PID DeiT-tiny, with $\lambda_P, \lambda_I, \lambda_D$, and β are $0.5, 0.3, 0.05$, and 1, respectively.

A.3. Language Modeling on WikiText-103

Datasets and Metrics. The WikiText-103 dataset is composed of Wikipedia articles tailored to capture extensive contextual dependencies. Its training set includes roughly $28,000$ articles, totaling around 103 million words. Each article consists of text blocks averaging about $3,600$ words. The validation and test sets contain $218,000$ and $246,000$ words, respectively, divided into 60 articles per set and approximately $268,000$ words each. Our experiment adheres to the standard setup outlined in (Merity et al., 2017; Schlag et al., 2021), which entails segmenting the training data into independent long segments of length L words. For evaluation, we utilize a batch size of 1 and process the text sequence using a sliding window of size L. When calculating perplexity (PPL), we only consider the last position, except for the first segment where all positions are evaluated, consistent with the methodology in (Al-Rfou et al., 2019; Schlag et al., 2021).

Models and baselines. For our language modeling implementation, we rely on the publicly available code https://github.com/IDSIA/lmtool-fwp developed by (Schlag et al., 2021). In our experiments, we set the dimensions of keys, values, and queries to 128, while the training and evaluation context length is set to 256. In this experiment, $\lambda_P, \lambda_I, \lambda_D$, and β being set to $0.4, 0.5, 0.1$ and 0.3, respectively, yields the best performance of PIDformer language model.

A.4. Adversarial Examples and Out-of-distribution datasets

Imagenet-C To assess robustness against typical image corruptions, we employ the ImageNet-C dataset (Hendrycks & Dietterich, 2019), which comprises 15 categories of artificially generated corruptions spanning five levels of severity. ImageNet-C evaluates models using the mean corruption error (mCE) metric, where a lower mCE value indicates greater resilience to corruption.

Imagenet-A ImageNet-A (Hendrycks et al., 2021b) is a dataset consisting of authentic images that have been filtered to deceive ImageNet classifiers. Specifically, it focuses on a subset of 200 classes chosen from the original 1000 classes in ImageNet-1K. Errors made within these 200 classes are regarded as significant, encompassing a wide range of categories represented in ImageNet-1K.

Imagenet-O This dataset comprises examples that have been adversarially filtered to challenge out-of-distribution detectors for ImageNet (Hendrycks et al., 2021b). It includes samples from the larger ImageNet-22K dataset but excludes those from ImageNet1K. Specifically, samples are chosen if they are confidently misclassified as an ImageNet-1K class by a ResNet-50 model. The evaluation metric utilized is the area under the precision-recall curve (AUPR).

Imagenet-R This dataset comprises diverse artistic interpretations of object classes found in the original ImageNet dataset, a practice discouraged by the creators of the original ImageNet (Hendrycks et al., 2021a). ImageNet-R encompasses 30,000 renditions of images representing 200 classes from the ImageNet dataset, with a selection made from a subset of the ImageNet-1K classes.

A.5. Adversarial Attacks

We employ fast gradient sign method (FGSM) (Dong et al., 2020), projected gradient descent method (PGD) (Tramer & Boneh, 2019b); and Sparse L_1 descent method as well as noise-adding attack These attacks were applied to the entire validation set of ImageNet. FGSM and PGD attacks distort the input image with a perturbation budget $\epsilon = 3/255$, and $\epsilon = 0.1$ for SPSA, under l_∞ norm, while the PGD attack uses 20 steps with a step size of $\alpha = 0.15$. For the SLD and noise attack, we follow the same setting in https://github.com/cleverhans-lab

A.6. Rank-collapse Analysis

The average cosine similarity between all pairs of token's representations $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in a sequence is computed as

$$\frac{1}{N(N-1)} \sum_{i \neq j} \frac{\boldsymbol{x}_i^T \boldsymbol{x}_j}{\|\boldsymbol{x}_i\|_2 \|\boldsymbol{x}_j\|_2}.$$

The result is then averaged over 1000 randomly chosen test data in ImageNet. The result is then reported for each layer, as in Fig. 2.

B. Technical Proofs

B.1. Solution of the first order ODE

Given $\boldsymbol{Q} \in \mathbb{R}^{n \times n}$, $\boldsymbol{Y}(t) \in \mathbb{R}^{N \times P}, t > 0$, we are interested in the solution of the first order ODE stated as:

$$\boldsymbol{Y}'(t) = \boldsymbol{Q}\boldsymbol{Y}(t), \boldsymbol{Y}(0) = \boldsymbol{Y}^0. \tag{27}$$

The general solution of (27) is $\boldsymbol{Y}(t) = \exp(\boldsymbol{Q}t)\boldsymbol{C}$, where $\boldsymbol{C} \in \mathbb{R}^{n \times p}$ is an any constant matrix. Indeed,

$$\begin{aligned}\boldsymbol{Y}'(t) &= (\boldsymbol{I} + \boldsymbol{Q}t + \frac{\boldsymbol{Q}^2 t^2}{2!} + \frac{\boldsymbol{Q}^3 t^3}{3!} + \dots)'\boldsymbol{C} \\ &= (\boldsymbol{Q} + \boldsymbol{Q}^2 t + \frac{\boldsymbol{Q}^3 t}{2!} + \dots)\boldsymbol{C} \\ &= \boldsymbol{Q}\exp(\boldsymbol{Q}t)\boldsymbol{C} = \boldsymbol{Q}\boldsymbol{Y}(t).\end{aligned} \tag{28}$$

To satisfy the intitial condition, $\boldsymbol{Y}(0) = \boldsymbol{Q}\exp(\boldsymbol{Q}0)\boldsymbol{C} = \boldsymbol{Y}^0$. Hence, $\boldsymbol{C} = \boldsymbol{Y}^0$ and the solution for the intial value problem in (27) is $\exp(\boldsymbol{Q}t)\boldsymbol{Y}^0$.

Every square matrix can be Jordan decomposed as the form of $\boldsymbol{Q} = \boldsymbol{S}\boldsymbol{J}\boldsymbol{S}^{-1}$, where $\boldsymbol{S}$ is invertible and contains the generalized eigenvectors of $\boldsymbol{Q}$, and $\boldsymbol{J} = \mathbf{diag}(\boldsymbol{J}_{\eta_1,m_1}, \boldsymbol{J}_{\eta_2,m_2}, \dots, \boldsymbol{J}_{\eta_M,m_M})$ is the Jordan form of matrix $\boldsymbol{Q}$ with,
$\boldsymbol{J}_{\eta_i,m_i} = \begin{bmatrix} \eta_i & 1 & \dots & 0 \\ \vdots & \ddots & & \vdots \\ & & \eta_i & 1 \\ 0 & \dots & & \eta_i \end{bmatrix} \in \mathbb{R}^{m_i \times m_i}$, for $i = 1, \dots, M$ are Jordan blocks and $\eta_1, \dots \eta_M$ are eigenvalues of $\boldsymbol{Q}$.
We rewrite the solution of (27) using the Jordan decomposition as

$$\begin{aligned}\boldsymbol{Y}(t) &= \exp(\boldsymbol{Q}t)\boldsymbol{Y}^0 = \exp(\boldsymbol{S}\boldsymbol{J}\boldsymbol{S}^{-1}t)\boldsymbol{Y}^0 \\ &= (\boldsymbol{S}\boldsymbol{S}^{-1} + \boldsymbol{S}\boldsymbol{J}\boldsymbol{S}^{-1}t + \frac{(\boldsymbol{S}\boldsymbol{J}\boldsymbol{S}^{-1})^2 t^2}{2!} + \dots)\boldsymbol{Y}^0 \\ &= \boldsymbol{S}\exp(\boldsymbol{J}t)\boldsymbol{S}^{-1}\boldsymbol{Y}^0.\end{aligned} \tag{29}$$

We are now interested in the asymptotic behavior of the solution in (29) as $t \to \infty$.
When Q only has eigenvalues negative real parts. As $\eta_1, \dots \eta_M < 0$, we consider

$$\begin{aligned}
\exp(\boldsymbol{J}_{\eta_i,m_i} t) &= \sum_{k=0}^{\infty} \frac{(\boldsymbol{J}_{\eta_i,m_i} t)^k}{k!} \\
&= \begin{bmatrix}
\sum_{k=0}^{\infty} \frac{t^k \eta_i^k}{k!} & \sum_{k=1}^{\infty} \frac{t^k \eta_i^{k-1}}{(k-1)!} & \dots & \sum_{k=m_i}^{\infty} \frac{t^k \eta_i^{k-m_i+1}}{(k-m_i+1)!} \\
\vdots & \ddots & & \\
0 & \dots & \sum_{k=0}^{\infty} \frac{t^k \eta_i^k}{k!} & \sum_{k=1}^{\infty} \frac{t^k \eta_i^{k-1}}{(k-1)!} \\
0 & \dots & 0 & \sum_{k=0}^{\infty} \frac{t^k \eta_i^k}{k!}
\end{bmatrix} \\
&= \begin{bmatrix}
e^{\eta_i t} & t e^{\eta_i t} & \dots & t^{m_i-1} e^{\eta_i t} \\
\vdots & \ddots & & \\
0 & \dots & e^{\eta_i t} & t e^{\eta_i t} \\
0 & \dots & 0 & e^{\eta_i t}
\end{bmatrix}
\end{aligned} \tag{30}$$

which is derived from the result $\boldsymbol{J}^k_{\eta_i,m_i} = \begin{bmatrix}
\eta_i^k & \binom{j}{1} \eta_i^{k-1} & \dots & \binom{j}{m_i-1} \eta_i^{k-m_i+1} \\
\vdots & \ddots & & \\
0 & \dots & \eta_i^k & \binom{j}{1} \eta_i^{k-1} \\
0 & \dots & 0 & \eta_i^k
\end{bmatrix}$
Therefore, when $t \to 0, \exp(\boldsymbol{J}_{\eta_i,m_i} t) \to \mathbf{0}$, making $\exp(\boldsymbol{J}t) \to \mathbf{0}$ and hence the solution in (29) will goes to $\mathbf{0}$ or being stable.
When Q only has at least one eigenvalue with positive real part. Without the loss of generalization, let $\mathrm{Re}(\eta_1) > 0$. Hence $\|\exp(\boldsymbol{J}_{\eta_1,m_i} t)\| \to \infty$ when $t \to \infty$. In other words, the solution of (27) will explode or unstable.

B.2. Proof of Lemma 1

The proof of Lemma 1 is the direct result in Appendix B.1. The solution of the ordinary differential equation (ODE) in (9) is $\mathbf{V}(t) = \boldsymbol{P}\exp(\boldsymbol{J}t)\boldsymbol{P}^{-1}\mathbf{V}^0$, where $\boldsymbol{P}\boldsymbol{J}\boldsymbol{P}^{-1}$ if the Jordan decomposition of $\boldsymbol{K} - \boldsymbol{I}$, $\boldsymbol{J} = \mathbf{diag}(\boldsymbol{J}_{\alpha_1,m_1}, \boldsymbol{J}_{\alpha_2,m_2}, \dots, \boldsymbol{J}_{\alpha_M,m_M})$ and $\alpha_1 \geq \alpha_2 \geq \dots, \geq \alpha_M, M \leq N$ are eigenvalues $\mathbf{K} - \mathbf{I}$. Consequently, we have proved the Lemma 1

B.3. Proof of Lemma 2

In Section 2.2, we have shown that $\boldsymbol{K} - \boldsymbol{I}$ has a unique largest eigenvalue $\lambda_1 = 0$. This means that the Jordan blocks corresponding with other eigenvalues which has negative real parts will approach $\exp(\boldsymbol{J}_{\eta_i,m_i} t) \to \mathbf{0}$, for $i = 1, \dots, M; , i \neq 1$, as $t \to \infty$. As the consequence, $\exp(\boldsymbol{J}t)$ are fill with 0 for all entries except for the first entry $\exp(\boldsymbol{J}t)(0,0) = 1$. Hence, the solution in (10) becomes

$$\begin{bmatrix} c_{1,1}\boldsymbol{p_1}, & \dots, & c_{1,D_x}\boldsymbol{p_1} \end{bmatrix}.$$

This concludes the proof.

B.4. Proof of Lemma 3

For $\boldsymbol{v}^{\ell+1}$ to be the solution of the optimization problem in (14), since $\mathbf{0} \in \partial J(\boldsymbol{v}^{\ell+1}) - \boldsymbol{p}^{\ell} + \partial G(\boldsymbol{v}^{\ell+1}, \boldsymbol{f})$, hence the iteration becomes:

$$\begin{cases} \boldsymbol{v}^{\ell+1} = \arg\min\limits_{\boldsymbol{v}} J(\boldsymbol{v}) - < \boldsymbol{p}^{\ell}, \boldsymbol{v} > + G(\boldsymbol{v}, \boldsymbol{f}) \\ \boldsymbol{p}^{\ell+1} \in \boldsymbol{p}^{\ell} - \partial G(\boldsymbol{v}^{\ell+1}, \boldsymbol{f}). \end{cases}$$

When $G(\boldsymbol{v}, \boldsymbol{f}) = \frac{\lambda}{2}\int_{\Omega} \|\boldsymbol{v}(x) - \boldsymbol{f}(x)\|_2^2 dx$,

$$\begin{aligned} G(\boldsymbol{v}, \boldsymbol{f}) - \langle \boldsymbol{p}^{\ell}, \boldsymbol{v}\rangle &= \frac{\lambda}{2}\int_{\Omega} \left((\|\boldsymbol{v}(x)\|_2^2 - 2\langle \boldsymbol{v}(x), \boldsymbol{f}(x)\rangle + \|\,\boldsymbol{f}(x)\|_2^2) + \lambda \langle \sum_{m=1}^{\ell} \left(\boldsymbol{v}^m(x) - \boldsymbol{f}(x)\right), \boldsymbol{v}(x)\rangle \right) dx \\ &= \frac{\lambda}{2}\int_{\Omega} \left(\|\boldsymbol{v}(x)\|_2^2 - \lambda\langle \boldsymbol{f}(x) - \sum_{m=1}^{\ell} \Big(\boldsymbol{v}^m(x) - \boldsymbol{f}(x)\Big), \boldsymbol{v}(x)\rangle \right) dx + \text{constant} \\ &= \frac{\lambda}{2}\int_{\Omega} \|\boldsymbol{v}(x) - \boldsymbol{f}^{\ell}(x)\|_2^2 dx + \text{constant}, \end{aligned}$$

where $\boldsymbol{f}^{\ell}(x) = \boldsymbol{f}^{\ell-1}(x) + \boldsymbol{f}(x) - \boldsymbol{v}^{\ell}(x)$.
Substituting $G(\boldsymbol{v}, \boldsymbol{f}) - \langle \boldsymbol{p}^{\ell}, \boldsymbol{v}\rangle$ into the iteration, it becomes

$$\begin{cases} \boldsymbol{v}^{\ell+1} = \arg\min\limits_{\boldsymbol{v}} J(\boldsymbol{v}) + \frac{\lambda}{2}\int_{\Omega} \|\boldsymbol{v}(x) - \boldsymbol{f}^{\ell}(x)\|_2^2 dx \\ \boldsymbol{f}^{\ell}(x) = \boldsymbol{f}^{\ell-1}(x) + \boldsymbol{f}(x) - \boldsymbol{v}^{\ell}(x). \end{cases} \tag{31}$$

The iteration in Lemma 3 can be reformulated as:

$$\boldsymbol{v}^{\ell+1} = \arg\min_{\boldsymbol{v}} J(\boldsymbol{v}) + \frac{\lambda}{2}\int_{\Omega} \|\boldsymbol{v}(x) - \boldsymbol{f}(x) - \boldsymbol{e}_{\mathrm{a}}^{\ell}(x)\|_2^2 dx$$

where $\boldsymbol{e}_{\mathrm{a}}^{\ell}(x) = \sum_{m=1}^{\ell} \boldsymbol{e}^m(x) = \sum_{m=1}^{\ell} \big(\boldsymbol{f}(x) - \boldsymbol{v}^m(x)\big)$ we conclude the proof for Lemma 3.

B.5. Proof of Lemma 4

To find the solution of Eqn 19, firstly, we find the solution for the homogenous ODE:

$$\mathbf{V}^{(h)\prime}(t) = \big(\mathbf{K} - (\lambda_P + 1)\boldsymbol{I}\big)\mathbf{V}^{(h)}(t) \tag{32}$$

From the result in Appendix B.1, the solution for this ODE is $\exp(\boldsymbol{B}t)\boldsymbol{C}$ where $\boldsymbol{B} = \mathbf{K} - (\lambda_P + 1)\boldsymbol{I}$ and $\boldsymbol{C} \in \mathbb{R}^{N\times D_x}$ is any constant matrix. Secondly, we find a particular solution for (19) by solving $\mathbf{V}^{(p)\prime}(t) = \boldsymbol{B}\mathbf{V}(t)^{(p)} + \lambda_P \boldsymbol{F} = \mathbf{0}$. Since $\boldsymbol{B}$ is invertible, the solution for this equation is $\mathbf{V}^{(p)}(t) = -\lambda_P \boldsymbol{B}^{-1}\boldsymbol{F}$. It is easy to check that $\mathbf{V}(t) = \mathbf{V}^{(h)}(t) + \mathbf{V}^{(p)}(t)$ is the solution of the $\mathbf{V}'(t) = \boldsymbol{B}\mathbf{V}(t) + \lambda_P \boldsymbol{F}$. Applying the initial condition, $\mathbf{V}(0) = \boldsymbol{C} - \lambda_P \boldsymbol{B}^{-1}\boldsymbol{F} = \mathbf{V}^0$, we find $\boldsymbol{C} = \mathbf{V}^0 + \lambda_P \boldsymbol{B}^{-1}\boldsymbol{F}$. Therefore, we have proved that the solution for the IVP problem in (19) is indeed $\mathbf{V}(t) = \exp(\boldsymbol{B}t)(\mathbf{V}^0 + \boldsymbol{B}^{-1}\boldsymbol{F}) - \lambda_P \boldsymbol{B}^{-1}\boldsymbol{F}$.

In Section 3.2.1, we show that $\boldsymbol{B}$ has only eigenvalues with negative real parts. As the result in Appendix B.1, when $t \to 0$, the $\exp(\boldsymbol{B}t) \to \mathbf{0}$, leading to the vanishing of the $\mathbf{V}^{(h)}(t)$. Hence the steady state solution for the ODE in (19) becomes $-\lambda_P \boldsymbol{B}^{-1}\boldsymbol{F}$.

This concludes the proof.

B.6. Proof of Proposition 1

We first show that $\boldsymbol{B}$ is a strictly diagonal dominant (SDD) matrix, i.e., $|\boldsymbol{B}(i,i)| > |\sum_{j\neq i}^{N} \boldsymbol{B}(i,j)|$, for $i, j = 1, \ldots, N$. In fact, $|\boldsymbol{B}(i,i)| = |\boldsymbol{K}(i,i) - \lambda_p - 1| > |1 - \boldsymbol{K}(i,i)| = |\sum_{j\neq i}^{N} \boldsymbol{K}(i,j)| = |\sum_{j\neq i}^{N} \boldsymbol{B}(i,j)|$ because $\boldsymbol{K}$ is a right-stochastic

matrix with all entries in $(0,1]$ and sum of each row is 1.
Hence, following (Morača, 2007), the upper bound of $\|\boldsymbol{B}^{-1}\|_\infty$, when $\boldsymbol{B}$ is an SDD matrix, is given as

$$\|\boldsymbol{B}^{-1}\|_\infty \leq \frac{1}{\min\limits_{i\in N}(|\boldsymbol{B}(i,i)| - |\sum\limits_{j\neq i}^{N}\boldsymbol{B}(i,j)|)} \tag{33}$$

$$= \frac{1}{|\boldsymbol{K}(i,i) - \lambda_p - 1| - |1 - \boldsymbol{K}(i,i)|} = \frac{1}{\lambda_P}, \tag{34}$$

where $\|\boldsymbol{B}^{-1}\|_\infty = \max\limits_{i=1}^{N}\sum\limits_{j=1}^{N}|\boldsymbol{B}^{-1}(i,j)|$.
On the other hand,

$$\begin{aligned}\|-\lambda_P\beta\boldsymbol{B}^{-1}\boldsymbol{\epsilon}\|_\infty &\leq \lambda_P\beta\|\boldsymbol{B}^{-1}\|_\infty\|\boldsymbol{\epsilon}\|_\infty \\ &= \lambda_P\beta\frac{1}{\lambda_P}\bar{\epsilon} = \beta\bar{\epsilon}\end{aligned} \tag{35}$$

For the bounded error get arbitrarily small, we constraint $\beta\bar{\epsilon} \leq \delta$, making $\beta \leq \frac{\delta}{\bar{\epsilon}}$.
Here in the proof, we used the submultiplicity property of $\|.\|_\infty$ norm of matrices, which is proved as follow:

$$\begin{aligned}\|\boldsymbol{B}^{-1}\boldsymbol{\epsilon}\|_\infty = \sup_{\boldsymbol{x}}\frac{\|\boldsymbol{B}^{-1}\boldsymbol{\epsilon x}\|_\infty}{\|\boldsymbol{x}\|_\infty} &= \sup_{\boldsymbol{x}}\frac{\|\boldsymbol{B}^{-1}\boldsymbol{\epsilon x}\|_\infty\|\boldsymbol{\epsilon x}\|_\infty}{\|\boldsymbol{\epsilon x}\|_\infty\|\boldsymbol{x}\|_\infty} \\ &\leq \sup_{\boldsymbol{x}}\frac{\|\boldsymbol{B}^{-1}\boldsymbol{\epsilon x}\|_\infty}{\|\boldsymbol{\epsilon x}\|_\infty}\sup_{\boldsymbol{x}}\frac{\|\boldsymbol{\epsilon x}\|_\infty}{\|\boldsymbol{x}\|_\infty} \\ &\leq \sup_{\boldsymbol{x}}\frac{\|\boldsymbol{B}^{-1}\boldsymbol{x}\|_\infty}{\|\boldsymbol{x}\|_\infty}\sup_{\boldsymbol{x}}\frac{\|\boldsymbol{\epsilon x}\|_\infty}{\|\boldsymbol{x}\|_\infty} \\ &= \|\boldsymbol{B}^{-1}\|_\infty\|\boldsymbol{\epsilon}\|_\infty\end{aligned}$$

With this, we conclude the proof of Proposition 1

B.7. Proof of Lemma 5

To find the solution of (22), firstly, we find the solution for the homogenous ODE:

$$\mathbf{V}^{(h)\prime}(t) = \frac{1}{1+\lambda_D}\big(\mathbf{K} - (\lambda_P+1)\boldsymbol{I}\big)\mathbf{V}^{(h)}(t)$$

From the result in Appendix B.1, the solution for this ODE is $\exp(\frac{1}{\lambda_D+1}\boldsymbol{B}t)\boldsymbol{C}$ where $\boldsymbol{B} = \mathbf{K} - (\lambda_P+1)\boldsymbol{I}$ and $\boldsymbol{C} \in \mathbb{R}^{N\times D_x}$ is any constant matrix. Secondly, we find a particular solution for (22) by solving $\mathbf{V}^{(p)\prime}(t) = \frac{1}{\lambda_D+1}(\boldsymbol{B}\mathbf{V}(t)^{(p)} + \lambda_P\boldsymbol{F}) = \mathbf{0}$. Since $\boldsymbol{B}$ is invertible, the solution for this equation is $\mathbf{V}^{(p)}(t) = -\lambda_P\boldsymbol{B}^{-1}\boldsymbol{F}$. The solution is $\mathbf{V}(t) = \mathbf{V}^{(h)}(t) + \mathbf{V}^{(p)}(t)$. Applying the initial condition, $\mathbf{V}(0) = \boldsymbol{C} - \lambda_P\boldsymbol{B}^{-1}\boldsymbol{F} = \mathbf{V}^0$, we find $\boldsymbol{C} = \mathbf{V}^0 + \lambda_P\boldsymbol{B}^{-1}\boldsymbol{F}$. Therefore, we have proved that the solution for the IVP problem in (22) is indeed $\mathbf{V}(t) = \exp(\frac{1}{\lambda_D+1}\boldsymbol{B}t)(\mathbf{V}^0 + \boldsymbol{B}^{-1}\boldsymbol{F}) - \lambda_P\boldsymbol{B}^{-1}\boldsymbol{F}$.

In Section 3.2.1, we show that $\boldsymbol{B}$ has only eigenvalues with negative real parts. As the result in Appendix B.1, when $t \to 0$, the $\exp(\frac{1}{\lambda_D+1}\boldsymbol{B}t) \to \mathbf{0}$, leading to the vanishing of the $\mathbf{V}^{(h)}(t)$. Hence the steady state solution for the ODE in (22) becomes $-\lambda_P\boldsymbol{B}^{-1}\boldsymbol{F}$. We have proved Lemma 5.

B.8. Proof of Proposition 2

Let

$$\boldsymbol{M} = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} \\ -\frac{\lambda_I\boldsymbol{I}}{\lambda_D+1} & \frac{\boldsymbol{K}-(\lambda_P+1)\boldsymbol{I}}{\lambda_D+1}\end{bmatrix} \tag{36}$$

For the solution of (24) to be stable, the real part of eigenvalues of $\boldsymbol{M}$ must be all negative. Let $\boldsymbol{B} := \boldsymbol{K} - (\lambda_P + 1)\boldsymbol{I}$, for any eigenvalue γ of $\boldsymbol{M}$

$$\begin{aligned}\det(\boldsymbol{M} - \gamma\boldsymbol{I}) &= \det\left(\begin{bmatrix} -\gamma\boldsymbol{I} & \boldsymbol{I} \\ -\frac{\lambda_I}{\lambda_D+1}\boldsymbol{I} & \frac{1}{\lambda_D+1}(\boldsymbol{B} - \gamma\boldsymbol{I})\end{bmatrix}\right) \\ &= \det\Big(\frac{1}{\lambda_D+1}(-\gamma\boldsymbol{B} + \gamma^2\boldsymbol{I} + \lambda_I\boldsymbol{I})\Big), \qquad \text{(since } \boldsymbol{B} - \gamma\boldsymbol{I} \text{ and } -\lambda_I\boldsymbol{I} \text{ are commute, see (Silvester, 2000))} \\ &= 0\end{aligned} \tag{37}$$

Notice that $\gamma = 0$ is not a solution of (37). This fact is proved by contradiction. If $\gamma = 0$ is a solution, $\det(-\gamma\boldsymbol{B} + \gamma^2\boldsymbol{I} + \lambda_I\boldsymbol{I}) = \det(\lambda_I\boldsymbol{I}) = (\lambda_I)^N\det(\boldsymbol{I}) = (\lambda_I)^N > 0$ because $\lambda_I > 0$. This is contradict to (37). Since $\gamma \neq 0$, we can rewrite (37) as:

$$(-\frac{\gamma}{\lambda_D+1})^N\det(\boldsymbol{B} - (\gamma + \frac{\lambda_I}{\gamma})\boldsymbol{I}) = 0 \tag{38}$$

$$\iff \det(\boldsymbol{B} - (\gamma + \frac{\lambda_I}{\gamma})\boldsymbol{I}) = 0. \tag{39}$$

Therefore, $\gamma + \dfrac{\lambda_I}{\gamma}$ are eigenvalues of $\boldsymbol{B}$. Given κ_i, for $i = 1, \dots, m; m \leq N$ are eigenvalues of $\boldsymbol{B}$. For each i, we find the solution of

$$\gamma_i + \frac{\lambda_I}{\gamma_i} = \kappa_i \tag{40}$$

$$\iff \gamma_i^2 - \kappa\gamma_i + \lambda_I = 0 \tag{41}$$

Let $\gamma_{i,1}, \gamma_{i,1}$ are the solution of (40), and then

$$\begin{cases}\gamma_{i,1} + \gamma_{i,2} = \kappa_i \\ \gamma_{i,1}\gamma_{i,2} = \lambda_I\end{cases} \iff \begin{cases}\mathrm{Re}(\gamma_{i,1}) + \mathrm{Re}(\gamma_{i,2}) = \mathrm{Re}(\kappa_i) \\ \mathrm{Im}(\gamma_{i,1}) + \mathrm{Im}(\gamma_{i,2}) = \mathrm{Im}(\kappa_i) \\ \mathrm{Re}(\gamma_{i,1})\mathrm{Re}(\gamma_{i,2}) - \mathrm{Im}(\gamma_{i,1})\mathrm{Im}(\gamma_{i,2}) = \lambda_I \\ \mathrm{Re}(\gamma_{i,1})\mathrm{Im}(\gamma_{i,2}) + \mathrm{Im}(\gamma_{i,1})\mathrm{Re}(\gamma_{i,2}) = 0\end{cases} \tag{42}$$

In Section 3.2.1, we show that $\boldsymbol{B}$ has only eigenvalues with negative real parts. Hence, $\mathrm{Re}(\kappa_i) < 0$. Firstly, without any loss of generalization, suppose that $\mathrm{Re}(\gamma_{i,1}) = 0$. This means

$$\begin{cases}\mathrm{Re}(\gamma_{i,2}) = \mathrm{Re}(\kappa_i) < 0 \\ -\mathrm{Im}(\gamma_{i,1})\mathrm{Im}(\gamma_{i,2}) = \lambda_I \\ \mathrm{Im}(\gamma_{i,1})\mathrm{Re}(\gamma_{i,2}) = 0\end{cases} \Rightarrow \begin{cases}\mathrm{Im}(\gamma_{i,1}) = 0 \\ -\mathrm{Im}(\gamma_{i,1})\mathrm{Im}(\gamma_{i,2}) = 0 \neq \lambda_I > 0\end{cases} \tag{43}$$

which causes contradiction. Therefore, $\mathrm{Re}(\gamma_{i,1}) \neq 0$. As the result, $\mathrm{Im}(\gamma_{i,2}) = -\dfrac{\mathrm{Im}(\gamma_{i,1})\mathrm{Re}(\gamma_{i,2})}{\mathrm{Re}(\gamma_{i,1})}$, substituting to (42), we obtain

$$\mathrm{Re}(\gamma_{i,1})\mathrm{Re}(\gamma_{i,2}) = \lambda_I - \mathrm{Im}(\gamma_{i,1})^2\frac{\mathrm{Re}(\gamma_{i,2})}{\mathrm{Re}(\gamma_{i,1})}. \tag{44}$$

Suppose that $\mathrm{Re}(\gamma_{i,1})\mathrm{Re}(\gamma_{i,2}) < 0$, hence $\dfrac{\mathrm{Re}(\gamma_{i,2})}{\mathrm{Re}(\gamma_{i,1})} < 0$ leading to $-\mathrm{Im}(\gamma_{i,1})^2\dfrac{\mathrm{Re}(\gamma_{i,2})}{\mathrm{Re}(\gamma_{i,1})} > 0$, (because $\mathrm{Im}(\gamma_{i,1})^2 > 0$). Therefore the RHS of (44) is greater than 0 (since λ_I also greater than 0), which contradicts our assumption that $\mathrm{Re}(\gamma_{i,1})\mathrm{Re}(\gamma_{i,2}) < 0$. As a consequence, we obattain the following result:

$$\begin{cases}\mathrm{Re}(\gamma_{i,1}) + \mathrm{Re}(\gamma_{i,2}) = \mathrm{Re}(\kappa_i) < 0 \\ \mathrm{Re}(\gamma_{i,1})\mathrm{Re}(\gamma_{i,2}) > 0\end{cases} \iff \begin{cases}\mathrm{Re}(\gamma_{i,1}) < 0 \\ \mathrm{Re}(\gamma_{i,2}) < 0,\end{cases} \tag{45}$$

for $i = 1, \dots, m$. Therefore, all eigenvalues of $\boldsymbol{M}$ as negative real parts. Combined with result in Appendix B.1, we have the system described by (24) has stable solution when $t \to 0$, for all $\lambda_P, \lambda_I, \lambda_D > 0$. This concludes our proof.

B.9. The Fretchet derivation of the derivative of J w.r.t v_j.

The partial derivative $\partial J/\partial v_j,\ j = 1, 2, \dots, D$, is defined through its dot product with an arbitrary function $h_j \in L^2(\Omega \times [0, \infty))$ as follows

$$\begin{aligned}
\frac{\partial J}{\partial v_j} \cdot h_j(x,t) &= \frac{d}{d\tau} J(v_j + \tau h_j)\big|_{\tau=0} \\
&= \frac{1}{2}\left(\frac{d}{d\tau}\int_{\Omega\times\Omega} (v_j(x) - v_j(y) + \tau h_j(x) - \tau h_j(y))^2 k(x,y) dx dy\right)\Bigg|_{\tau=0} \\
&= \left(\int_{\Omega\times\Omega} (v_j(x,t) - v_j(y) + \tau h_j(x) - \tau h_j(y,t))(h_j(x) - h_j(y)) k(x,y) dx dy\right)\Bigg|_{\tau=0} \\
&= \int_{\Omega\times\Omega} (v_j(x) - v_j(y))(h_j(x) - h_j(y)) k(x,y) dx dy \\
&= \int_{\Omega\times\Omega} (v_j(x) - v_j(y)) h_j(x) k(x,y) dx dy - \int_{\Omega\times\Omega} (v_j(x) - v_j(y)) h_j(y) k(x,y) dx dy
\end{aligned}$$

Applying a change of variables $(x, y) \to (y, x)$ to the second term of the above integral, we have

$$\begin{aligned}
\frac{\partial J}{\partial v_j} \cdot h_j(x) &= \int_{\Omega\times\Omega} (v_j(x) - v_j(y)) h_j(x) k(x,y) dx dy - \int_{\Omega\times\Omega} (v_j(y) - v_j(x)) h_j(x,t) k(y,x) dx dy \\
&= \int_{\Omega\times\Omega} (v_j(x) - v_j(y)(k(x,y) + k(y,x)) dy h_j(x) dx
\end{aligned}$$

Thus, the Frechet derivative of J with respect to v_j is given by

$$\frac{\partial J}{\partial v_j} = \int_{\Omega} (v_j(x) - v_j(y)(k(x,y) + k(y,x)) dy.$$

B.10. The derivation of the gradient flow of $E(v, f)$

Taking the gradient of $E(\boldsymbol{v}, \boldsymbol{f})$ with respect to $\boldsymbol{v}$, we obtain

$$\nabla_{\boldsymbol{v}} E = \nabla_{\boldsymbol{v}} J + \left[\frac{\partial G}{\partial u_1}, \frac{\partial G}{\partial u_2}, \dots, \frac{\partial G}{\partial u_D}\right]^T. \tag{46}$$

The partial derivative $\partial G/\partial v_j$, $j = 1, 2, \dots, D$, is defined through its dot product with an arbitrary function $h_j \in L^2(\Omega)$ as follows

$$\begin{aligned}
\frac{\partial G}{\partial v_j} \cdot h_j(x) &= \frac{d}{d\tau} G(v_j + \tau h_j)\big|_{\tau=0} \\
&= \frac{\lambda}{2}\left(\frac{d}{d\tau}\int_{\Omega} (v_j(x) - f_j(x) + \tau h_j(x))^2 dx\right)\Bigg|_{\tau=0} \\
&= \lambda \int_{\Omega} (v_j(x) - f_j(x)) h_j(x) dx.
\end{aligned}$$

Thus, the Frechet derivative of F with respect to v_j is given by

$$\frac{\partial G}{\partial v_j} = \lambda(v_j(x) - f_j(x)) \tag{47}$$

Substituting the formula for $\partial G/\partial v_j$ in (47) into (46) for $\nabla_{\boldsymbol{v}} E(\boldsymbol{v}, \boldsymbol{f})$, we obtain the following gradient flow

$$\frac{d\boldsymbol{v}(x,t)}{dt} = -\nabla_{\boldsymbol{v}} E(\boldsymbol{v}, \boldsymbol{f}) = -\nabla_{\boldsymbol{v}} J(\boldsymbol{v})(x) + \lambda\big(\boldsymbol{f}(x) - \boldsymbol{v}(x)\big). \tag{48}$$

This concludes the derivation.

B.11. The derivation of (16)

Denote $H(\boldsymbol{v}, \boldsymbol{f}) := \frac{\lambda}{2}\int_{\Omega} \|\boldsymbol{v}(x) - \boldsymbol{f}(x) - \boldsymbol{e}^{\ell}(x)\|_2^2 dx$. Taking the gradient of $J(\boldsymbol{v}) + H(\boldsymbol{v}, \boldsymbol{f})$ with respect to $\boldsymbol{v}$, we obtain

$$\nabla_{\boldsymbol{v}} E = \nabla_{\boldsymbol{v}} J + \left[\frac{\partial H}{\partial v_1}, \frac{\partial H}{\partial v_2}, \dots, \frac{\partial H}{\partial v_D}\right]^T. \tag{49}$$

The partial derivative $\partial H/\partial v_j$, $j = 1, 2, \dots, D$, is defined through its dot product with an arbitrary function $h_j \in L^2(\Omega)$ as follows

$$\begin{aligned}
\frac{\partial H}{\partial v_j} \cdot h_j(x) &= \frac{d}{d\tau} H(v_j + \tau h_j)\big|_{\tau=0} \\
&= \frac{\lambda}{2}\left(\frac{d}{d\tau}\int_{\Omega}(v_j(x) - f_j(x) - e_j^{\ell}(x) + \tau h_j(x))^2 dx\right)\Bigg|_{\tau=0} \\
&= \lambda \int_{\Omega} (v_j(x) - f_j(x) - e_j^{\ell}) h_j(x) dx.
\end{aligned}$$

Thus, the Frechet derivative of F with respect to v_j is given by

$$\frac{\partial H}{\partial v_j} = \lambda(v_j(x) - f_j(x) - e_j^{\ell}) \tag{50}$$

Substituting the formula for $\partial H/\partial v_j$ in (50) into (49) for $\nabla_{\boldsymbol{v}} E(\boldsymbol{v}, \boldsymbol{f})$, we obtain the following gradient flow at iteration $\ell + 1$

$$\begin{aligned}
\frac{d\boldsymbol{v}(x,t)}{dt} &= \int_{\Omega} \big(\boldsymbol{v}(y,t) - \boldsymbol{v}(x,t)\big)\big(k(x,y) + k(y,x)\big) dy \\
&\quad + \lambda\big(\boldsymbol{f}(x) - \boldsymbol{v}(x,t) + \boldsymbol{e}^{\ell}(x)\big).
\end{aligned} \tag{51}$$

Applying Euler method to discretize (51) with $\Delta t = 1$ and $\boldsymbol{v}(x, 0) = \boldsymbol{v}^{\ell}(x)$, we approximate the $\boldsymbol{v}^{\ell+1}$ with one-step gradient descent:

$$\begin{aligned}
\boldsymbol{v}^{\ell+1}(x) &= \int_{\Omega} \big(\boldsymbol{v}^{\ell}(y) - \boldsymbol{v}^{\ell}(x)\big)\big(k(x,y) + k(y,x)\big) dy \\
&\quad + \boldsymbol{v}^{\ell}(x) + \lambda \boldsymbol{e}^{\ell}(x) + \lambda \boldsymbol{e}_{\mathrm{a}}^{\ell}(x).
\end{aligned}$$

This concludes the derivation.

C. Additional Experiment results

C.1. PID DeiT and softmax DeiT under escalating perturbation attacks.

We evaluate PID DeiT and softmax DeiT under FGSM and PGD attack methods with increasing perturbation budgets (see Fig. 3) (scaled by 255). The proposed PID DeiT exhibits stronger defense in both attack methods and various perturbation budgets.

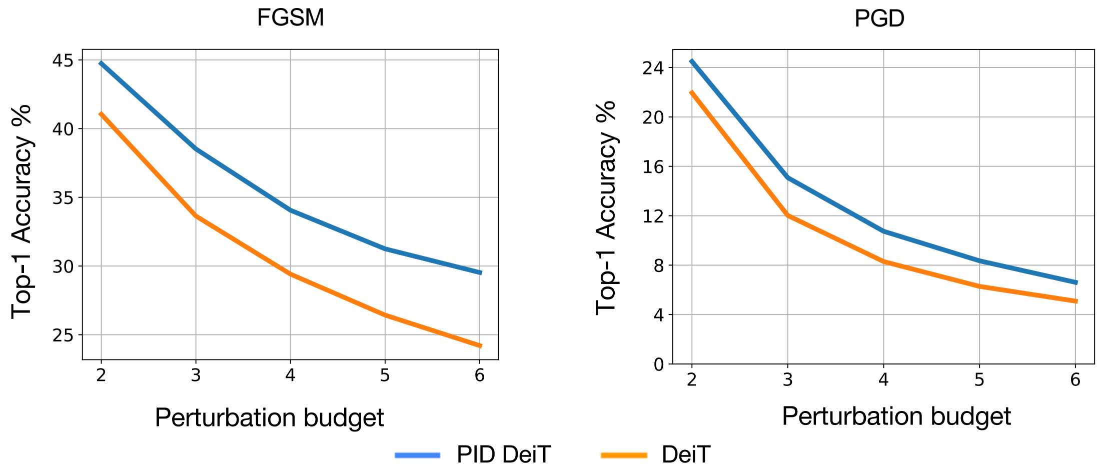


Figure 3. The top-1 classification accuracy curves on ImageNet against FGSM and PGD attack methods, plotted against perturbation budgets (scaled by 255).

Machine Learning Models in Weather Forecasting

Roger Dai [1] John Ho [1] Eddie Kim [1] Andrew Sun [1]

Abstract

This report presents an in-depth analysis of machine learning models in weather forecasting, focusing on GraphCast, a foundational model developed by Google. It provides detailed insights into GraphCast's architecture, optimization methods, and its role in improving forecasting accuracy and efficiency. Furthermore, the report offers a comparative analysis, evaluating GraphCast against other ML-based weather forecasting tools. By examining the evolution and impact of ML in weather prediction, this report aims to contribute valuable insights to the field.

1. Introduction

Weather forecasting is important in modern society as it influences many decisions that range from daily activities to emergency responses. Traditional numerical and physical methods such as the Integrated Forecasting System (IFS) have long been the standard for weather prediction. The techniques employed by these methods rely on supercomputers to solve complex equations for atmospheric dynamics, making these highly effective. However, these methods often fail to fully utilize historical data to improve and enhance forecasting precision. Rather, they rely heavily on expert-driven innovations, limiting the scalability and adaptability of forecast models to evolving weather patterns.

In contrast, the emergence of machine learning-based weather prediction (MLWP) presents a paradigm shift, offering a direct approach to model training from historical data. MLWP holds promise in capturing intricate patterns inherent in weather data, potentially surpassing the limitations of traditional numerical methods. Recent advancements in MLWP have demonstrated superior performance in scenarios where traditional methods lack, such as subseasonal heat wave prediction and precipitation nowcasting–providing short-range (0-6 hours) forecast of the rainfall intensity.

Amidst the rapid evolution of MLWP, the relevance and importance of integrating machine learning into weather forecasting become increasingly evident. MLWP not only offers avenues for enhanced accuracy but also presents opportunities for more efficient forecasting, potentially revolutionizing the field. Thus, exploring and understanding the implications of ML models in weather forecasting hold significant implications for advancing prediction capabilities and mitigating the impacts of weather-related events on society.

This report will focus on GraphCast, a model created by Google, as it stands as one of the most foundational MLWP tools. It will delve into an in-depth exploration of GraphCast, dissecting its architecture and optimization methods. Furthermore, the report will undertake a comparative analysis, juxtaposing GraphCast against other ML-based weather forecasting tools, both preceding and succeeding its inception. Through this comparative lens, the report aims to provide comprehensive insights into the efficacy and potential of GraphCast in revolutionizing weather forecasting methodologies.

2. Model Architecture

2.1. Model overview

The GraphCast model processes input weather states defined on a 0.25° latitude-longitude grid, encompassing a total of 721 × 1440 = 1,038,240 points(a). For each grid point, the model takes into account 5 surface variables and 6 atmospheric variables across 37 pressure levels (5 + 6 × 37 = 227 variables), culminating in a graph containing 235,680,480 values representing a weather state. By analyzing two consecutive weather states—six hours prior and the current state—the GraphCast model can accurately predict the weather conditions for the subsequent six-hour interval.

$$x^{t+1} = \text{GraphCast}\left(x_t, x_{t-1}\right)$$

Then, by taking the newly predicted state and the previous state as input, we can generate results for 12 hours later. By repeating this process iteratively, we can generate predictions along an arbitrary trajectory. This process is known as autoregression, which is also utilized in traditional numerical methods.

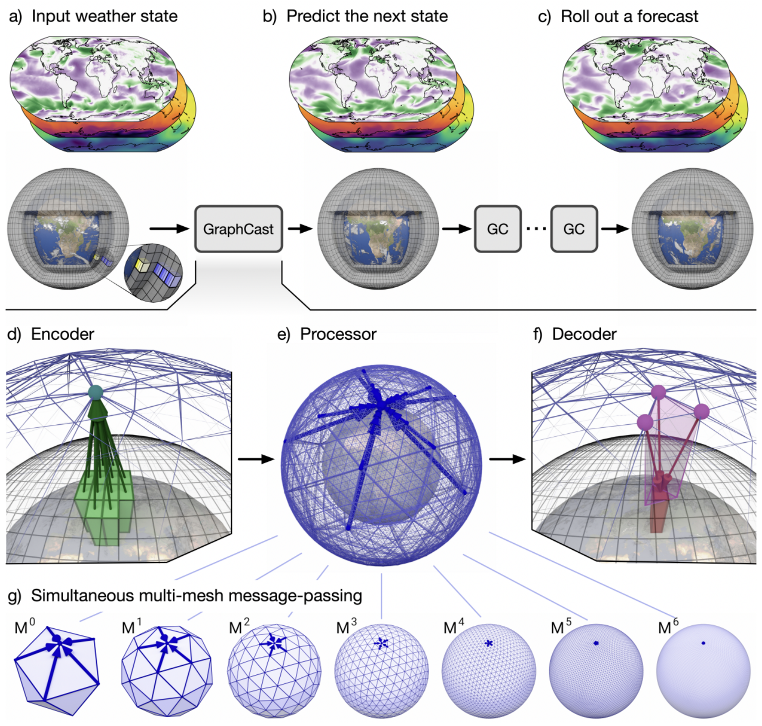


Figure 1. GraphCast Model Architecture

$$x^{t+1:t+T} = \left(\text{GraphCast}\left(x^t, x^{t-1}\right), \text{GraphCast}\left(x^{t+1}, x^t\right), \ldots, \ldots, \text{GraphCast}\left(x^{t+T-1}, x^{t+T-2}\right)\right) \quad (1)$$

However, due to limitations in accuracy, GraphCast prediction is typically reserved for mid-range forecasts spanning 10 to 14 days.

GraphCast's architecture is based on GNN(graphical neural network), specifically comprising three parts: encoder(d); processor(e); decoder(f). The paper uses GNN since it has been proven to be effective in learning parietal differential equations. Also, GNNs have a significant advantage over other neural network architectures because they enable interactions among the data representations based on the structure of the input graph. This allows for the modeling of arbitrary spatial interactions, unlike Convolutional Neural Networks (CNNs), which are limited to local or regularly spaced interactions, and Transformers, which are inefficient for very large inputs due to their quadratic memory complexity.

2.2. Multi-mesh structure

The development of the multi-mesh structure involves a series of increasingly refined meshes, starting from a base mesh, M_0, which is a unit-radius icosahedron. This base mesh has 12 nodes and 20 triangular faces.

The refinement process from one mesh, M_r, to the next, M_{r+1}, involves subdividing each triangular face of the mesh into four smaller triangles. This subdivision adds an extra node at the midpoint of each edge of the triangle. After adding these new nodes, they are re-projected onto the surface of the unit sphere to maintain the shape and uniformity of the sphere.

This iterative process increases the number of nodes and edges with each refinement step such that at the final step (M_6), for each node, the structure includes thousands of small edges that we can pass information for local weather prediction, and it also retains several large edges so that information can be also be passed continentally(g).

2.3. Encoder

In the GraphCast model, the encoder maps input data from the latitude-longitude grid of two previous weather states onto a learned node representation using the multi-mesh structure. This process utilizes one layer of a Graph Neural Network (GNN). Specifically, it first embeds the features of each grid node, mesh node, mesh edge, grid-to-mesh edge, and mesh-to-grid edge using Multi-Layer Perceptrons (MLP). Then, it performs a single message-passing step by first updating edges using the information from adjacent

nodes, then updating mesh nodes by aggregating information from all incoming edges, and finally updating the grid nodes. After updating all three elements, the model incorporates a residual connection by reassigning every node and edge with the sum of the new and old values.
The residual connection can be expressed as below, where V_i^G represents the grid node, $V_i'^G$ is the new grid node; V_i^M are the mesh nodes, $V_i'^M$ is the new grid node; e^{G2M} are the edges that are connecting grid to mesh nodes and e'^{G2M} is the updated one.

$$\begin{aligned} V_i^G &\leftarrow V_i^G + V_i'^G, \\ V_i^M &\leftarrow V_i^M + V_i'^M, \\ e^{G2M} &\leftarrow e^{G2M} + e'^{G2M}. \end{aligned}$$

2.4. Processor

The processor uses 16 layers of unshared GNN. By employing deep GNN layers and the multi-mesh structure, which retains edges of different lengths for each node, each node in the model can access information from far away, solving the traditional limitation of GNNs – the inability to reach distant nodes.
For each layer of the Mesh GNN, the message passing process is similar to that in the encoder: it first updates mesh edges using information from adjacent nodes, then updates mesh nodes by aggregating information from incoming edges, and finally applies a residual connection. The key difference in the message passing between the processor and the encoder is that the processor only deals with mesh nodes and edges and does not update grid nodes.

2.5. Decoder

Similar to the encoder, using one layer of GNN, the decoder performs a single message pass from the mesh to the grid. The process first updates the grid-to-mesh edges using information from adjacent nodes. Then it updates grid nodes using incoming information. Since the focus at the end is on grid information, there is no need to update the mesh nodes again. A residual connection is added here as well, and the results are output using another MLP.

3. Model Training

3.1. Training Data Split

Data used to develop GraphCast was split into a training set, validation set, and test set, as is standard among Machine Learning practices. The training set comprised data from the years 1979-2016, the validation set included 2016-2017, and the test set included 2018-2021.

GraphCast was trained according to an objective function developed by the researchers. They used 12-step forecasts in their training, which corresponded to 3 days, and the objective was the MSE between the target output and the predicted output.

$$\mathcal{L}_{\text{MSE}} = \underbrace{\frac{1}{|D_{\text{batch}}|} \sum_{d_0 \in D_{\text{batch}}}}_{\text{forecast date-time}} \underbrace{\frac{1}{T_{\text{train}}} \sum_{\tau \in 1:T_{\text{train}}}}_{\text{lead time}} \underbrace{\frac{1}{|G_{0.25^\circ}|} \sum_{i \in G_{0.25^\circ}}}_{\text{spatial location}} \underbrace{\sum_{j \in J}}_{\text{variable-level}} s_j w_j a_i \underbrace{(\hat{x}_{i,j}^{d_0+\tau} - x_{i,j}^{d_0+\tau})^2}_{\text{squared error}}$$

- $\tau \in 1 : T_{train}$ are the lead times that correspond to the T_{train} autoregressive steps
- $d_0 \in D_{batch}$ represent forecast initialization date-times in a batch of forecasts in the training set
- $j \in J$ indexes the variable, and for atmospheric variables the pressure level.
- $i \in G_{0.25^\circ}$ are the location (latitude and longitude coordinates in the grid
- $\hat{x}_{j,i}^{d_0+\tau}$ and $x_{j,i}^{d_0+\tau}$ are predicted and target values for some variable-level, location, and lead time
- s_j is the per-variable-level inverse variance of time differences
- w_j is the per-variable-level loss weight
- a_i is the area of the latitude-longitude grid cell, which varies with latitude and is normalized to unit mean over the grid

3.2. Autoregressive Training

The researchers used an autoregressive training scheme for GraphCast, meaning that the model's predictions were reused as inputs for new predictions. At each step of the autoregressive forecast, loss was computed with respect to the ground truth corresponding to the step. Error gradients with respect to the model parameters were back-propagated through all iterations of the model.
Model training followed the schedule highlighted in Figure 2. It consisted of three phases:
Phase 1: 1,000 gradient descent updates, 1 autoregressive step, linearly increasing scheduled learning rate from 0 to 1e-3.
Phase 2: 299,000 gradient descent updates, 1 autoregressive step, half-cosine decay scheduled learning rate from 1e-3 to 0.
Phase 3: 11,000 gradient descent updates, 2-12 autoregressive steps (+1 every 1000 updates), fixed learning rate of 3e-7.

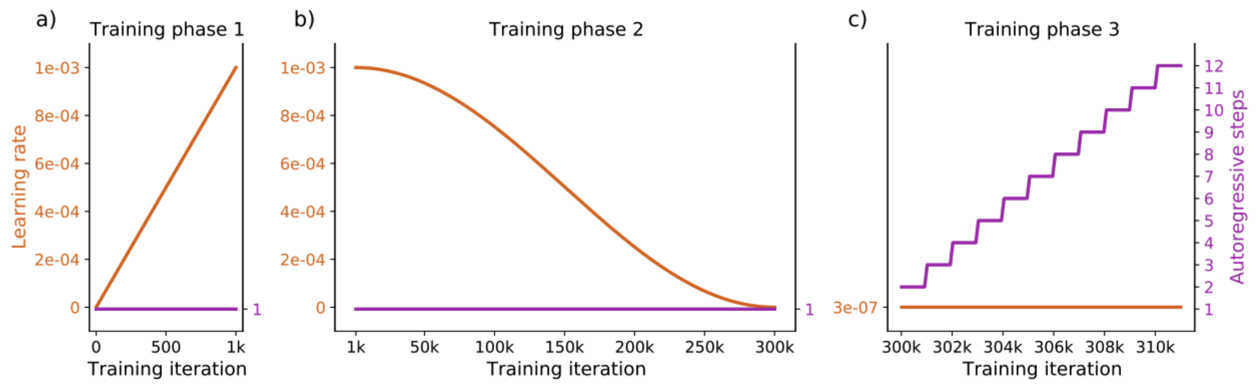


Figure 2. Training Schedule. Plots correspond to the three phases of model training designed by the researchers.

3.3. AdamW Optimization

The training objective function was minimized using the AdamW optimizer. Adam is the extension of gradient descent that implements adaptive learning rates by taking estimates of the first and second moments of the gradients. This allows for dynamic adjustments of learning rates, which generally leads to more efficient convergence when training machine learning models. AdamW is an adaptation of Adam that includes weight decay to penalize large weights, instead of L2 regularization. In AdamW, weight decay is only added after controlling the parameter-wise step size, which one can observe in step 12 of Figure 3. Ilya Loshchilov and Frank Hutter show in their research proposing AdamW that this approach leads to generally better results for convergence, and leads to a version of Adam that is much more competitive with SGD with momentum [5].

4. GraphCast Verification

4.1. Verification Methods

GraphCast's forecasting accuracy was evaluated by comparing its predictions to those of HRES, the high-resolution model from ECMWF (European Centre for Medium-Range Weather Forecasts), across various model configurations. To quantify these accuracies, researchers used Root Mean Square Error and Anomaly Correlation Coefficient.
GraphCast produces 227 predictions of different variables at each grid point on the map, and 69 of these predictions were chosen (based on WeatherBench and ECMWF's Forecast Scorecard) for comparison against HRES. Two main factors were considered in establishing how skill was measured: (1) Selection of Ground Truth, and (2) Accounting for the data assimilation windows used to ground data with observations. Since ERA5 was the dataset used to train GraphCast, it would work fine as a ground truth comparison for the model. HRES however, is not predicted from ERA5, and so to prevent initial forecasting error, the researchers compiled an HRES forecast inputs dataset, which essentially relabeled future prediction inputs as current ground truth. The researchers also only evaluated GraphCast forecasts from inputs that carried +3h of future observations, in order to match the information amount that HRES uses.

4.2. Verification Results

The researchers found that GraphCast did produce better 10-day forecasts than HRES at a horizontal resolution of 0.25°. As Figure 4 shows, Both the RMSE graphs and the ACC graphs show GraphCast predictions outperforming HRES in terms of prediction skill, especially as lead times increase. The skill score improvement margins lay at around 7%-14% better for GraphCast compared to HRES.

It is important to note that according to the Scorecards in 4d, HRES performs better at the lowest pressure level. However, these regions were disproportionately localized in the stratosphere and had the lowest training loss weight.

4.3. Training Data Recency

The researchers proposed the idea of retraining GraphCat as new weather data is collected to capture weather patterns that may change over time. To test this theory, they trained GraphCast with data that began in 1979, but ended in 2017, 2018, 2019, and 2020, and compared the performances of these models with HRES when predicting 2021 test data. Figure 5 shows the results of testing these models. Researchers found that their theory proved to be correct, as prediction accuracy increased gradually as training data was more recent to the prediction time.

Algorithm 2 Adam with L_2 regularization and Adam with decoupled weight decay (AdamW)

```
1: given $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda \in \mathbb{R}$
2: initialize time step $t \leftarrow 0$, parameter vector $\boldsymbol{\theta}_{t=0} \in \mathbb{R}^n$, first moment vector $\boldsymbol{m}_{t=0} \leftarrow \boldsymbol{0}$, second moment vector $\boldsymbol{v}_{t=0} \leftarrow \boldsymbol{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
3: repeat
4:     $t \leftarrow t + 1$
5:     $\nabla f_t(\boldsymbol{\theta}_{t-1}) \leftarrow \text{SelectBatch}(\boldsymbol{\theta}_{t-1})$    ▷ select batch and return the corresponding gradient
6:     $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{\theta}_{t-1})$ $+\lambda\boldsymbol{\theta}_{t-1}$
7:     $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t$    ▷ here and below all operations are element-wise
8:     $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2)\boldsymbol{g}_t^2$
9:     $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t/(1 - \beta_1^t)$    ▷ $\beta_1$ is taken to the power of $t$
10:    $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t/(1 - \beta_2^t)$    ▷ $\beta_2$ is taken to the power of $t$
11:    $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$    ▷ can be fixed, decay, or also be used for warm restarts
12:    $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t \left( \alpha \hat{\boldsymbol{m}}_t / (\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon) +\lambda\boldsymbol{\theta}_{t-1} \right)$
13: until stopping criterion is met
14: return optimized parameters $\boldsymbol{\theta}_t$
```

Figure 3. Adam and AdamW Algorithm Comparison. A step-by-step outline of the Adam and AdamW algorithms, where the added term unique to Adam is highlighted in purple, and the term unique to AdamW is highlighted in green. For the purpose of GraphCast, optimization was initialized with parameters: $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\lambda = 0.1$.

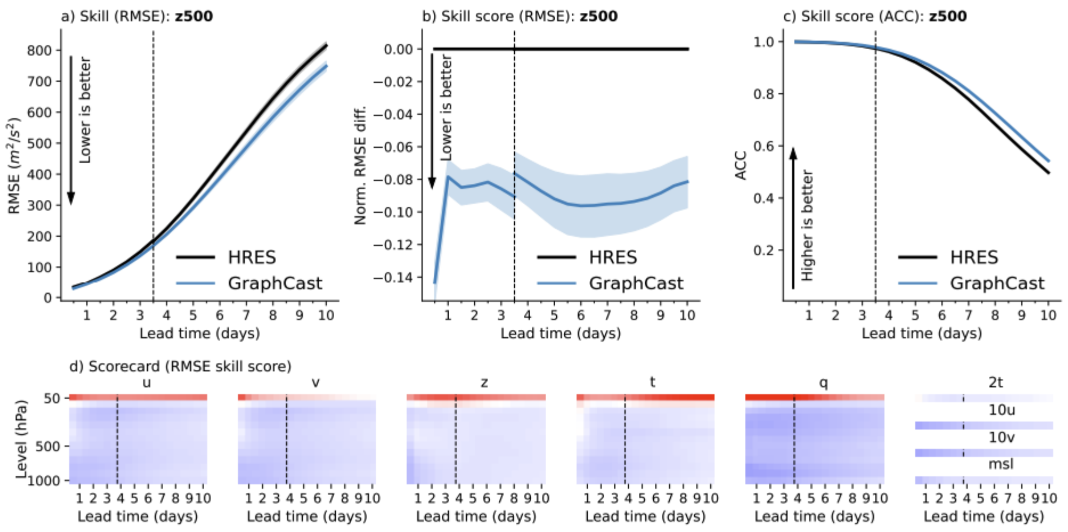


Figure 4. Plots of Forecast Skill for GraphCast and HRES a) Plot of RMSE for GraphCast and HRES, as a function of lead time, with 95% Confidence Intervals. A lower score means less error, which is preferred. b) Plot of RMSE for HRES and GraphCast, normalized with respect to HRES forecast skill. c) Plot of ACC for GraphCast and HRES as a function of lead time. d) Scorecard in ECMWF format, for GraphCast, with respect to HRES. The rows of each map correspond to physical levels in the atmosphere, and each map corresponds to a different predicted variable. Red represents a better forecast by HRES, and blue represents a better forecast by GraphCast.

Type	Variable name	Role
Atmospheric	Geopotential	Input/Predicted
Atmospheric	Specific humidity	Input/Predicted
Atmospheric	Temperature	Input/Predicted
Atmospheric	U component of wind	Input/Predicted
Atmospheric	V component of wind	Input/Predicted
Atmospheric	Vertical velocity	Input/Predicted
Single	2 metre temperature	Input/Predicted
Single	10 metre u wind component	Input/Predicted
Single	10 metre v wind component	Input/Predicted
Single	Mean sea level pressure	Input/Predicted
Single	Total precipitation	Input/Predicted
Single	TOA incident solar radiation	Input (1h)
Static	Geopotential at surface	Input
Static	Land-sea mask	Input
Static	Latitude	Input
Static	Longitude	Input
Clock	Local time of day	Input
Clock	Elapsed year progress	Input

Table 1. List of ECMWF variables used in the datasets. The Type column indicates whether the variable is a time-varying *atmospheric* property, a time-varying *single-level* property, or a *static* property (e.g., not time-varying).

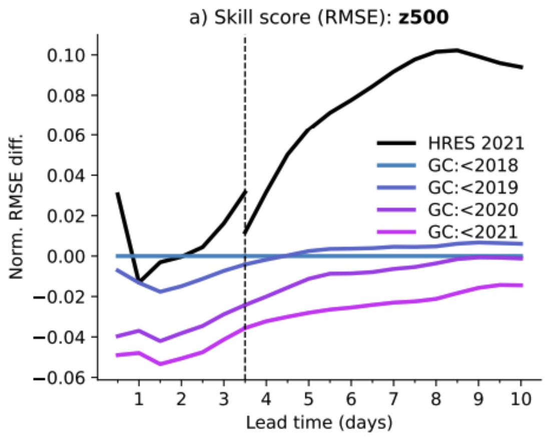


Figure 5. Retraining GraphCast. Colored lines represent the RMSE of GraphCast trained on a different number of years, with respect to the lead time. The black line represents the RMSE of HRES.

5. Model Comparisons

5.1. Comparison of GraphCast and ClimaX

5.1.1. Architecture Comparison

GraphCast and ClimaX are both advanced models designed for weather and climate forecasting but differ significantly in their architectural approach and underlying technology.

GraphCast primarily leverages graph neural networks to model the complex interactions between various atmospheric variables. This model type is particularly adept at capturing spatial relationships and dependencies within data, making it highly suitable for tasks where interaction dynamics are complex and non-linear. GraphCast's architecture allows for the dynamic incorporation of nodes and edges, representing different geographical locations and their interconnections, respectively. This structure is inherently flexible and can adapt to various scales of input data.

ClimaX, on the other hand, extends the Transformer architecture, which is fundamentally different from graph-based models. ClimaX introduces novel encoding and aggregation blocks that enhance the Transformer's ability to handle spatial and temporal data effectively. This model is designed to be a foundation model, meaning it can be pre-trained on a wide range of climate datasets and fine-tuned for specific tasks. This versatility is achieved through its ability to handle variable tokenization and aggregation, which optimizes compute resources while maintaining utility across different spatiotemporal scales.

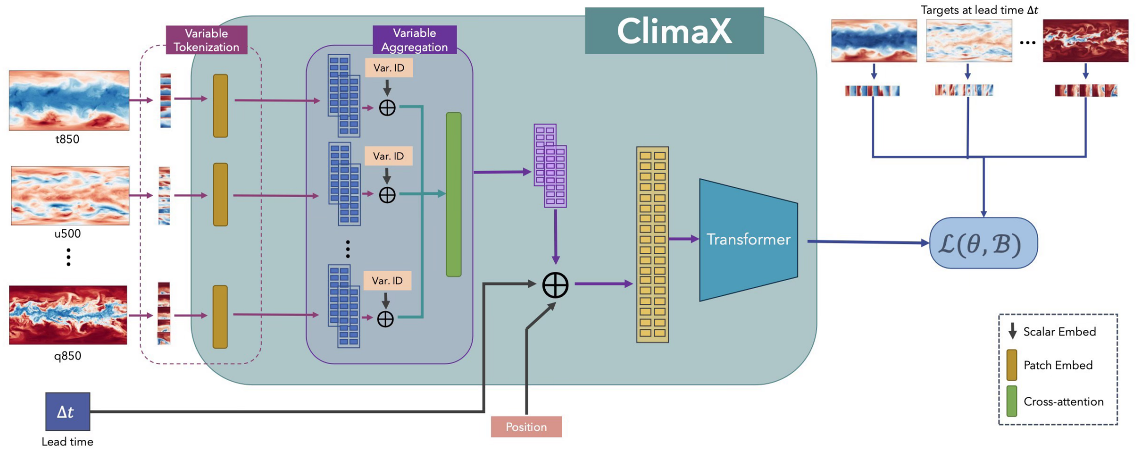


Figure 6. Pretraining phase of ClimaX. Variables are encoded using variable-separate tokenization and subsequently aggregated using variable aggregation. Together with position embedding and lead time embedding those are fed to the ViT backbone.

5.1.2. Prediction Results

GraphCast is tailored for medium-range forecasting, typically providing predictions for up to a week ahead. Its graph-based structure allows it to maintain high accuracy over this time frame by effectively capturing the dynamics between atmospheric variables at different locations. The resolution of predictions depends on the granularity of the input data and the configuration of the graph nodes, which can be adjusted to balance detail and computational efficiency.

ClimaX demonstrates superior performance in both weather forecasting and long-term climate projections. It can handle various forecasting tasks with different lead times, from hours to years, due to its robust pre-training and fine-tuning methodology. It's designed to handle resolutions from about 10 km for regional models to 50 km for global models in pre-training, with the ability to fine-tune for higher resolutions. Temporal resolution can vary widely as it is suitable for both short-term forecasting from hours to days, to long-term climate projections like months to years in advance. The model's ability to perform well even when pre-trained at lower resolutions and compute budgets highlights its efficiency and scalability. ClimaX can also be used for downscaling, which improves the resolution of output predictions from coarser model outputs, making it highly effective for detailed regional climate analyses.

5.2. Comparison of GraphCast and FourCastNet

5.2.1. Architecture Comparison

While GraphCast uses a graph-based approach, FourCastNet employs the Adaptive Fourier Neural Operator (AFNO), which integrates Fourier transformations into deep learning frameworks. This method allows FourCastNet to perform global convolutions efficiently, which is crucial for capturing the wide-ranging spatial dependencies inherent in atmospheric data.

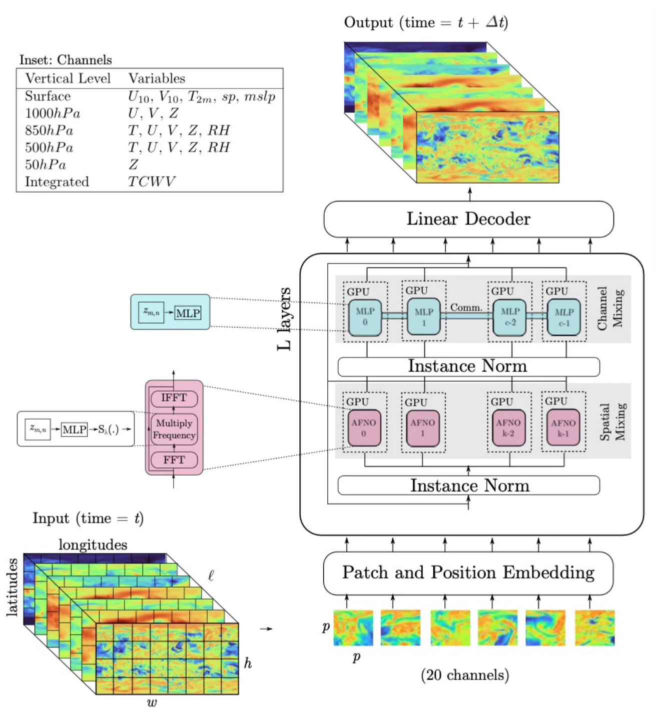
FourCastNet's architecture is highly optimized for parallel computing environments and is capable of scaling across thousands of GPUs, a feature that supports extremely high-resolution modeling and rapid computation speeds. This scaling capability is essential for handling the complex, high-dimensional data involved in global weather forecasting.

5.2.2. Prediction Results

FourCastNet has been demonstrated to generate accurate weather predictions globally with a lead time of up to a week, similar to GraphCast. However, FourCastNet stands out due to its ability to produce these predictions at a much higher resolution up to 1 km for global weather simulations, which is exceptionally detailed compared to traditional models, at speeds significantly faster than traditional numerical weather prediction models due to its use of Fourier transformations. For instance, FourCastNet can perform an 80,000 times faster inference compared to state-of-the-art numerical models, with the ability to handle resolutions that are much finer, thanks to its use of AFNO and the massive parallel computing power it leverages.

6. Conclusion

In conclusion, our discussion of GraphCast, an ML-based weather prediction tool, highlights the potential of machine learning in the field of meteorology. By using advanced architectures like Graph Neural Networks, models like GraphCast can approach weather forecasting and significantly ben-


Inset: Channels

Vertical Level	Variables
Surface	U_{10}, V_{10}, T_{2m}, sp, $mslp$
$1000hPa$	U, V, Z
$850hPa$	T, U, V, Z, RH
$500hPa$	T, U, V, Z, RH
$50hPa$	Z
Integrated	$TCWV$

Figure 7. The Fourcast Net architecture showing the key operations per- formed on the input tensor with dimensions (20×720×1440) to produce a 6-hour single-time step forecast with the same dimensions. Model parallelism is implemented by splitting the channels (feature maps) across GPUs. Channel mixing MLP operations require communication across the model parallel ranks, while the FFT-based spatial-mixing operates on disjoint blocks that are embarrassingly parallel.

efit from enhanced accuracy and efficiency. This report has demonstrated that GraphCast's ability to learn from historical weather data and atmospheric patterns offers a considerable improvement over traditional forecasting methods. Our comparative analysis further indicates that while newer ML-based models continue to evolve, GraphCast remains a very strong option in this rapidly growing field, providing critical insights and methodologies that continue to influence emerging weather prediction technology. Future research should focus on refining these models, expanding their predictive capabilities, and integrating more real-time data adaptation to ensure more reliable and actionable forecasts, ultimately aiding in better preparedness for weather-related challenges.

References

1. Battaglia, Peter W., et al. *Relational Inductive Biases, Deep Learning, and Graph Networks*. arXiv:1806.01261, arXiv, 17 Oct. 2018. https://doi.org/10.48550/arXiv.1806.01261.

2. Bi, Kaifeng, et al. *Pangu-Weather: A 3D High-Resolution Model for Fast and Accurate Global Weather Forecast*. arXiv:2211.02556, arXiv, 3 Nov. 2022. https://doi.org/10.48550/arXiv.2211.02556.

3. Chen, Kang, et al. *FengWu: Pushing the Skillful Global Medium-Range Weather Forecast beyond 10 Days Lead*. arXiv:2304.02948, arXiv, 6 Apr. 2023. https://doi.org/10.48550/arXiv.2304.02948.

4. ECMWF. ECMWF, 6 Mar. 2024. https://www.ecmwf.int/.

5. Kingma, Diederik P., and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. arXiv:1412.6980, arXiv, 29 Jan. 2017. https://doi.org/10.48550/arXiv.1412.6980.

6. Keisler, Ryan. *Forecasting Global Weather with Graph Neural Networks*. arXiv:2202.07575, arXiv, 15 Feb. 2022. https://doi.org/10.48550/arXiv.2202.07575.

7. Lam, Remi, et al. *GraphCast: Learning Skillful Medium-Range Global Weather Forecasting*. arXiv:2212.12794, arXiv, 4 Aug. 2023. https://doi.org/10.48550/arXiv.2212.12794.

8. Lavers, David A., et al. "An Evaluation of ERA5 Precipitation for Climate Monitoring." Quarterly Journal of the Royal Meteorological Society, vol. 148, no. 748, Oct. 2022, pp. 3152–65. DOI.org (Crossref), https://doi.org/10.1002/qj.4351.

9. Loshchilov, Ilya, and Frank Hutter. *Decoupled Weight Decay Regularization*. arXiv:1711.05101, arXiv, 4 Jan. 2019. https://doi.org/10.48550/arXiv.1711.05101.

10. Nguyen, Tung, et al. *ClimaX: A Foundation Model for Weather and Climate*. arXiv:2301.10343, arXiv, 18 Dec. 2023. https://doi.org/10.48550/arXiv.2301.10343.

11. Pathak, Jaideep, et al. *FourCastNet: A Global Data-Driven High-Resolution Weather Model Using Adaptive Fourier Neural Operators*. arXiv:2202.11214, arXiv, 22 Feb. 2022. https://doi.org/10.48550/arXiv.2202.11214.

12. Sønderby, Casper Kaae, et al. *MetNet: A Neural Weather Model for Precipitation Forecasting*. arXiv:2003.12140, arXiv, 30 Mar. 2020. https://doi.org/10.48550/arXiv.2003.12140.

13. Zhang, Guodong, et al. *Three Mechanisms of Weight Decay Regularization*. arXiv:1810.12281, arXiv, 29 Oct. 2018. https://doi.org/10.48550/arXiv.1810.12281.