Universal Approximation Theory: Foundations for Parallelism in Neural Networks

Wei Wang[1], **Qing Li**[1]

[1]The Hong Kong Polytechnic University
weiuat.wang@connect.polyu.hk, qing-prof.li@polyu.edu.hk

Abstract

Neural networks are increasingly evolving towards training large models with big data, a method that has demonstrated superior performance across many tasks. However, this approach introduces an urgent problem: current deep learning models are predominantly serial, meaning that as the number of network layers increases, so do the training and inference times. This is unacceptable if deep learning is to continue advancing. Therefore, this paper proposes a deep learning parallelization strategy based on the Universal Approximation Theorem (UAT). From this foundation, we designed a parallel network called Para-Former to test our theory. Unlike traditional serial models, the inference time of Para-Former does not increase with the number of layers, significantly accelerating the inference speed of multi-layer networks. Experimental results validate the effectiveness of this network.

1 Introduction

In today's era, deep learning models have undoubtedly taken the lead in the fields of computer vision (CV) and natural language processing (NLP), demonstrating exceptional performance across numerous tasks. In CV, models such as ResNet(He et al. 2015) and ViT(Vaswani et al. 2017) for image recognition, U-Net (Ronneberger, Fischer, and Brox 2015) for image segmentation, GAN (Goodfellow et al. 2014) for image generation, and diffusion models like SORA (Liu et al. 2024) for video generation highlight the extensive application of deep learning in CV. These models are primarily based on convolutional neural networks (CNNs), Transformers, or a combination of both. Simultaneously, large language models (LLMs) based on Transformers have become dominant in natural language processing. Overall, deep learning models have shown vast application potential and powerful capabilities so far.

However, this progress has brought forth some challenges: the development trend in deep learning is moving towards larger models and larger datasets. This trend is fundamentally driven by the UAT(Cybenko 2007; Hornik, Stinchcombe, and White 1989). From the papers UAT2LLM(Wang and Li 2024) and UAT2CV(Wang and Li 2024), we know that residual-based CNNs and Transformer-based models are specific implementations of UAT, both exhibiting the characteristic of dynamically approximating functions based on inputs. Larger models and datasets are indeed requirements of the UAT theory: deeper network layers can more precisely approximate target functions dynamically based on inputs, while larger and more diverse datasets help ensure the training data closely represents real-world conditions, making the model's approximated functions more reflective of real-world complexities.

However, this trend inevitably leads to increased demands for computational resources and extended training times. As more powerful models are needed, the UAT theory mathematically necessitates larger networks, which in turn require more computational resources. While technological advancements, especially in hardware performance, provide potential ways to alleviate this issue, the increase in time costs remains a fundamental obstacle to scaling serial networks. The improvement in computational resources such as GPUs and TPUs is limited by the speed of electrical signal transmission, whereas the theoretical scale of networks has no such upper limit. When networks reach tens or even hundreds of thousands of layers, inference times can extend to several minutes or even hours for each batch. Current solutions include model optimization techniques like quantization and pruning (Sun et al. 2023; Ma, Fang, and Wang 2023; Xia et al. 2023), hardware acceleration with specialized accelerators and edge computing (Gale et al. 2020), and parallel and distributed computing (Ben-Nun and Hoefler 2018) with model parallelization, data parallelization, and distributed computing frameworks. However, these do not fundamentally address the inference delay caused by the increasing number of serial network layers.

We believe the root cause lies in the serial computation mechanism of existing deep learning models, where the output of one layer serves as the input for the next. This design stems from the early development of deep learning networks in the field of CV, where image data exhibit local correlations and spatiotemporal invariance. To accommodate these features, early CV networks primarily employed convolutional layers with small receptive fields. To achieve a larger receptive field, deeper network layers were necessary, leading to early deep-learning architectures like AlexNet and VGG. The introduction of ResNet brought a paradigm shift, as residual networks have the ability to dynamically fit functions based on input, influencing almost all subsequent net-

work designs and solidifying the tradition of multi-layer stacking.

While this design was manageable with fewer layers, the increasing depth and number of parameters in modern networks have created a critical issue: slow computation efficiency due to the need for sequential layer-by-layer calculations during inference. This computation method inevitably leads to delays, which become more severe as the network depth increases, significantly constraining the progress of CV and NLP technologies.

Therefore, it is imperative to explore parallelization techniques for deep learning networks, enabling parallel computation across multiple layers. This approach is key to accelerating the inference process, breaking the time bottleneck, and advancing CV and NLP technologies towards greater efficiency and speed. To develop such parallel computation, we must start from the fundamental theory of deep learning: the UAT. The network must be designed in accordance with UAT principles and must satisfy the requirements outlined in the papers UAT2LLM and UAT2CV, which emphasize the ability to dynamically fit functions based on data. With this in mind and leveraging existing network structures, we designed a parallel network, Para-Former, to validate the feasibility of our theory. Our contributions include:

- Proposing the theory of parallel deep learning networks and experimentally validating its feasibility.
- Analyzing the characteristics, advantages, and disadvantages of both serial and parallel networks.
- Investigating the issues present in deep learning from a data perspective.

The structure of this paper is as follows: In Section 2, we first present the general form of the Universal Approximation Theorem (UAT). Based on the mathematical form of UAT, we define the parallel network Para-Former in Section 3. In Section 4, we validate the effectiveness of Para-Former in various ways: we assess its feasibility in Section 4.2, explore the impact of network depth in Section 4.3, and analyze the influence of different datasets in Section 4.4.

2 UAT

Deep learning networks have demonstrated their powerful capabilities in numerous CV and NLP tasks. In the papers UAT2LLMs and UAT2CVs, it has been proven that both Transformer-based networks and residual-based CNNs are specific implementations of the UAT. Therefore, we consider UAT to be the fundamental theory for deep learning. To design parallelized networks, we must start from the principles of UAT. Before formally designing parallel networks, we will first review the basic form of the UAT theory and then propose a network parallelization solution based on this mathematical theory. The UAT was initially proposed by Cybenko (Cybenko 2007), and its basic form is as follows:

$$G(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \sigma\left(\mathbf{W}_j^{\mathrm{T}}\mathbf{x} + \theta_j\right) \tag{1}$$

is dense in $C(\mathbf{I}_n)$. Here, $\mathbf{W}_j \in \mathbb{R}^n$ and $\alpha_j, \theta \in \mathbb{R}$ are parameters σ is a sigmoid function. For any $f \in C(\mathbf{I}_n)$ and $\varepsilon > 0$, there exists a function $G(\mathbf{x})$:

$$|G(\mathbf{x}) - f(\mathbf{x})| < \varepsilon \quad \text{for all} \quad \mathbf{x} \in \mathbf{I}_n. \tag{2}$$

This indicates that when N is sufficiently large, a neural network can approximate any continuous function on a closed interval. Hornik, Stinchcombe, and White (Hornik, Stinchcombe, and White 1989) further proved that this function can approximate any Borel measurable function. Observing Equation (1), where the function $G(\mathbf{x})$ produces scalar outputs in the real number field $\mathbb{R}$, the scenario naturally extends when $G(\mathbf{x})$ maps to $\mathbb{R}^m$, requiring approximation in each dimension. To accommodate such multi-dimensional outputs, we simply need to extend Equation (1): the transformation matrix $\mathbf{W}_j$ is revised to the space $\mathbb{R}^{n\times m}$, the bias term θ_j is recast as a vector in $\mathbb{R}^m$, and α_j is reshaped into a matrix. We call the parameters $\mathbf{W}_j$ and α_j in UAT as weight and θ_j as bias.

3 UAT for Parallel Network

3.1 Pre-request for Basic Block of Parallel Network

From Equation (1), we can observe a significant characteristic: if we define different indices j as the indices of layers and the corresponding layer's mathematical form is $\alpha_j\sigma\left(\mathbf{W}_j^{\mathrm{T}}\mathbf{x} + \theta_j\right)$, it is evident that the parameters between different layers do not interact with each other. Therefore, based on (1), we can design networks where parameters across multiple layers are independent, achieving parallel computation, like Figure 1. Figure 1.a represents a typical sequential network, which clearly requires computation to proceed in order from front to back. In contrast, Figure 1.b depicts a parallel network, where there is no parameter interaction between different blocks, allowing them to be considered independent and thus suitable for parallel computation. The blocks in this network should satisfy the formula given in Figure 1.b. Clearly, under this condition, the sum of the multiple blocks' outputs adheres to the UAT theory.

If we design the parallel network solely based on Eq. (1), the network will face a limitation once training is complete: the function that can be approximated by the network is fixed, as the parameters in the corresponding UAT are fixed. So to ensure the network can dynamically approximate functions based on input, some or all of the parameters α_j, W_j and θ_j must be influenced by the input (referencing to UAT2LLM and UAT2CV), which can be simply written as $\alpha_j(\mathbf{x})$, $W_j(\mathbf{x})$ and $\theta_j(\mathbf{x})$.

This allows the network to dynamically adapt and fit the function based on the input while enabling parallel computation. So we could design the basic block in Figure 1, like $\mathrm{Block}_1 \cdots \mathrm{Block}_n$ are the implementation of $\alpha_j(\mathbf{x})\sigma\left(\mathbf{W}_j^{\mathrm{T}}(\mathbf{x})\mathbf{x} + \theta_j(\mathbf{x})\right), i = 1 \cdots n$. Some or all the parameters α_j, W_j and θ_j will change according to input.

3.2 Para-former

Based on the previous request on the basic module, we designed the Para-Former which is a parallel network by us-

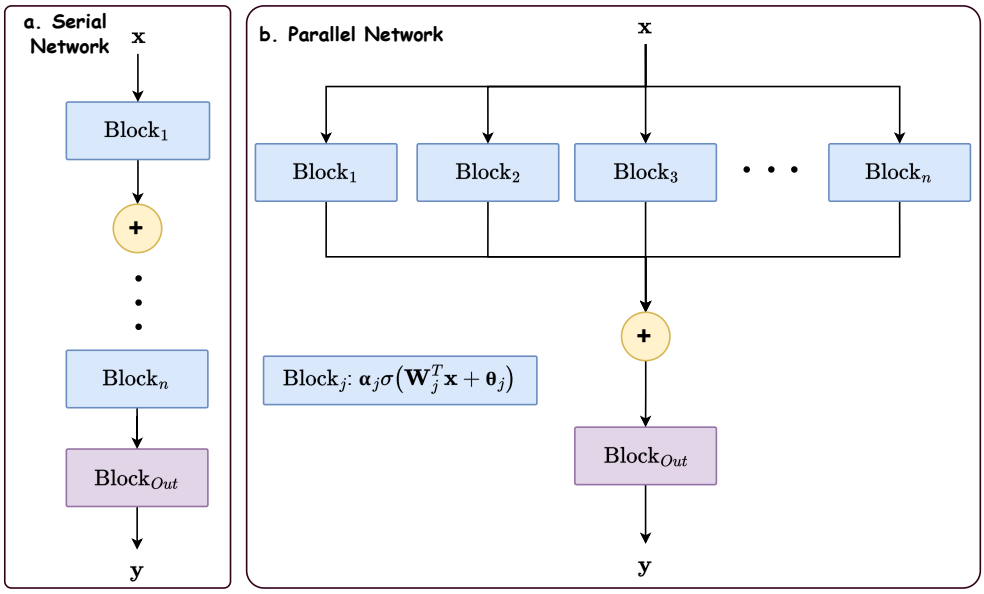

Figure 1: The general description of the serial network and parallel network.

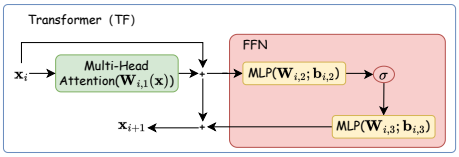

Figure 2: The process of Transformer module.

ing the Transformer module as the basic module. We chose Transformer for this design because it is a well-established and widely-used network architecture, excelling in both CV and NLP. Another reason is that its basic mathematical form aligns with the requirements of Section 3.1.

Before formally introducing the structure of Para-Former, let's briefly review the Transformer structure. The basic structure of a Transformer is shown in Figure 2, and for convenience, we refer to it as a Transformer module. Typically, it is represented as follows:

$$\begin{aligned}\mathring{\mathbf{H}} &= \text{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) \\ &= \text{Concat}\left(\hat{\mathbf{H}}_1,\ldots,\hat{\mathbf{H}}_h\right)\mathbf{W}_O\end{aligned} \tag{3}$$

$$\begin{aligned}\hat{\mathbf{H}}_i &= \text{Attention}(\mathbf{x}_i\mathbf{W}_{Qi}=\mathbf{Q}_i,\mathbf{x}_i\mathbf{W}_{Ki}=\mathbf{K}_i, \\ &\quad \mathbf{x}_i\mathbf{W}_{Vi}=\mathbf{V}_i) \\ &= \text{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^T}{\sqrt{M}}\right)\mathbf{V}_i \\ &= \mathbf{H}_i\mathbf{V}_i = \mathbf{H}_i[\mathbf{x}_i\mathbf{W}_{Vi}]\end{aligned} \tag{4}$$

$$FFN(\mathbf{x}_i) = \mathbf{W}_{i,3}\sigma(\mathbf{W}_{i,2}\mathbf{x}_i+\mathbf{b}_{i,2})+\mathbf{b}_{i,3} \tag{5}$$

The above representation mainly describes the engineering process rather than its true mathematical form. According to UAT2LLM, the Multi-Head Attention (MHA) and Feed-Forward Network (FFN) components of Transformers can be expressed in matrix-vector form. We denote the matrix-vector form of variables with a prime symbol (e.g., $\mathbf{x}_i$ becomes $\mathbf{x}'_i$). For more details, refer to UAT2LLM.

$$MHA(\mathbf{x}'_i) = \mathbf{W}'_{i,1}\mathbf{x}'_i \tag{6}$$

$$FFN(\mathbf{x}'_i) = \mathbf{W}'_{i,3}\sigma(\mathbf{W}'_{i,2}\mathbf{x}'_i+\mathbf{b}'_{i,2})+\mathbf{b}'_{i,3} \tag{7}$$

The parameter $\mathbf{W}'_{i,1}$ is influenced by the input. As shown in Figure 2, the output of a single Transformer layer, $\mathbf{x}'_{i+1}$, can be described by Eq. (8) (Details could be found in Section A):

$$\mathbf{x}'_{i+1} = \mathbf{W}'_{i,1}\mathbf{x}'_i + \mathbf{W}'_{i,3}\sigma(\hat{\mathbf{W}}'_{i,1}\mathbf{x}'_i+\mathbf{b}'_{i,2})+\mathbf{b}'_{i,3} \tag{8}$$

The mathematical form in Eq. (8) meets the requirements of a basic module in UAT. Therefore, we can use the structure in Figure 2 as the basic module in Para-Former, corresponding to Figure 3.a. Besides, we add a symbol of a hat on some parameters to represent the parameters will be affected by the input, like $\hat{\mathbf{W}}'_{i,1}$.

$$\begin{aligned}\mathbf{x}'_{i+2} &= \hat{\mathbf{W}'_{i+1,2}}\mathbf{x}'_i + \hat{\mathbf{W}'_{i+1,3}}\sigma(\hat{\mathbf{W}}'_{i,1}\mathbf{x}'_i+\mathbf{b}'_{i,2}) \\ &\quad + \mathbf{W}'_{i+1,3}\sigma\{\hat{\mathbf{W}'_{i+1,4}}\mathbf{x}'_i+\hat{\mathbf{b}'_{i+1,4}}\}+\hat{\mathbf{b}'_{i+1,5}}\end{aligned} \tag{9}$$

The output of a two-layer Transformer is shown in Eq. (9) (Details could be found in Section A). From Eq. (9), it is evident that a serial configuration of multi-layer Transformers also conforms to the mathematical form of UAT. Therefore, we can extend the Para-Former structure. That is, within one block in Figure 1, there can be multiple serially

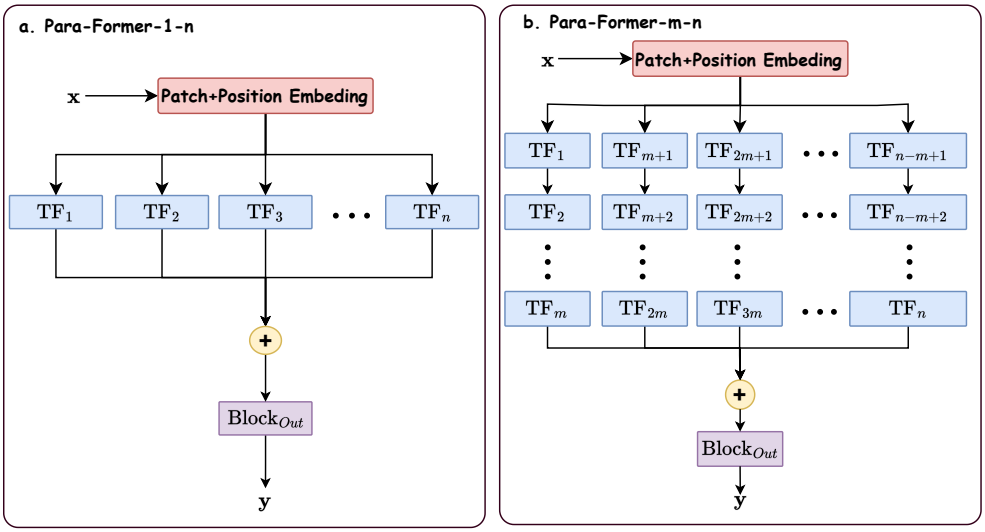

Figure 3: The structure of the Para-Former network. a.Para-Former-1-n and b.Para-Former-m-n represent Para-Former with 1 depth of n layers and m depth of n layers, respectively.

connected Transformer modules, while multiple blocks operate in parallel, as shown in Figure 3.b. We refer to the number of Transformer modules within a block as the depth and the total number of Transformer modules across all blocks as the network layer count.

3.3 Speed-up Ratio

We have proposed a network parallelization scheme, and in this section, we will compare the running speed of this parallelized network with that of traditional serial networks. It is important to note that due to differences in hardware and implementation methods, providing an objective comparison through experiments can be challenging. Therefore, we will focus on a theoretical analysis. Based on the network structure shown in Figure 2, the inference time of Para-Former is only affected by its network depth.

Assume we use a Para-Former with a depth of M and a layer count of L, compared to a traditional Transformer network with a depth of N layers, where $L >> N$. In this case, the running speed of the parallel network would be N/M times faster than that of the serial network. This is because the speed of the parallel network is solely related to the depth of Para-Former. As the network depth increases, the acceleration effect becomes more pronounced.

4 Experiments

Our experimental approach is to first verify the feasibility of the parallel network theory and its adherence to the UAT approximation properties using several commonly used public image classification datasets (Section 4.1). We then explore the impact of network depth on the UAT theory (Section 4.3), followed by an investigation into the influence of the dataset (Section 4.4).

4.1 Dataset

In pursuit of a comprehensive understanding of the efficacy and practicality of parallel models, we meticulously selected several widely-acknowledged and representative image classification datasets for an integrated comparative analysis. These datasets encompass a diverse array of visual content and span various levels of difficulty, thereby furnishing us with a fertile ground to probe the performance boundaries of parallel models. Specifically, our study centers on the following pivotal datasets:

1. CIFAR-10 & CIFAR-100 (Krizhevsky 2009): These were constructed by Alex Krizhevsky et al. CIFAR-10 encompasses 60,000 32x32 pixel color images across 10 classes, while CIFAR-100 extends this to 100 categories.

2. STL-10 (Coates, Ng, and Lee 2011): STL-10 comprises 5,000 training and 8,000 testing images distributed among 10 classes, each at a 96x96 pixel resolution.

3. Flowers-102 (Nilsback and Zisserman 2008): Focused on floral categorization, this dataset houses 102 distinct flower species, totalling 8,189 images with a per-species count ranging from 40 to 258.

4. Oxford-IIIT Pets (Parkhi et al. 2012): Tailored for the recognition of pet breeds (cats and dogs), this dataset contains over 7,300 images across 37 categories, characterized by varied animal poses, backgrounds, and lighting conditions.

In Table 1, we provide more details for these dataset.

Through experimentation on these meticulously chosen datasets, we aspire to thoroughly validate the feasibility and strengths of parallel models across a spectrum of scales, complexities, and domains in image classification tasks. Simultaneously, we seek to elucidate potential limitations and chart avenues for future research endeavors.

Table 1: This table presents the sizes of the training, validation, and test sets for the datasets CIFAR-10, CIFAR-100, STL-10, OxfordIIITPet, and Flower-102. It also provides the number of categories (Classes) and the average number of images per category for each dataset (Per class).

Dataset	Cifar-10	Cifar-100	STL-10	OxfordIIITPet	Flower-102
Classes	10	100	10	37	102
Train set	47500	47500	4750	3,515	969
Val set	2500	47500	250	185	51
Test set	10000	10000	800	3669	6149
Per class	4750	4750	475	95	9.5

Table 2: This table shows the predictions of Para-former-1-1, Para-former-1-6, Para-former-1-12, Para-former-1-24 on the datasets CIFAR-10, CIFAR-100, STL-10, OxfordIIITPet, and Flower-102. We simply represent them as Para-F-1-1, Para-F-1-6, Para-F-1-12 and Para-F-1-24, simply.

Dataset	Para-F-1-1	Para-F-1-6	Para-F-1-12	Para-F-1-24
Cifar-10	55.95	64.90	66.88	**71.53**
Cifar-100	30.67	38.84	41.41	**46.96**
STL-10	44.41	51.57	55.96	**59.53**
OxfordIIITPet	19.62	20.95	22.07	**22.73**
Flower-102	27.33	32.65	**37.42**	35.84

4.2 The Feasibility of Para-Former

To validate the feasibility and characteristics of Para-Former, we conducted training and validation on several commonly used image classification datasets, with all models trained for 500 epochs. The experimental design involved preprocessing inputs using the common ViT techniques of patch extraction and positional encoding. We then used Transformer as the basic model architecture, setting all network depths to 1 and configuring Para-Former from single-layer to multi-layer setups, as shown in Figure 3.a. A single-layer setup corresponds to a ViT with a depth of 1. The results of the models are presented in Table 2.

It is evident from Table 2 that as the number of parallel network layers increases, the model accuracy improves progressively (Except for the result of Para-F-1-24 being worse than Para-F-1-12 on the Flower-102 dataset, this variation is within a reasonable range.). According to UAT theory, increasing the network layers is equivalent to increasing N in Equation (1), thereby enhancing the approximation capability of UAT. The results in Table 2 are in complete agreement with UAT theory. Therefore, we believe that this parallel network approach is entirely feasible.

However, the overall prediction accuracy in Table 2 is quite low. What could be the reasons for this? We believe there are three main factors affecting the model's predictive performance:

1. Model's Fitting Capability: This is influenced by the model's size and design. Specifically, the value of N in UAT and the degrees of freedom in the parameters. The larger the N value, the stronger the UAT's fitting capability. Parameter freedom refers to the number of parameters affected by the input in UAT, such as a freedom of 1 in Eq. 8 and a freedom of 6 in Eq. 9. The degrees of freedom correspond to the model's ability to dynamically fit based on input. Merely increasing N enhances the model's ability to fit specific functions, but images are often diverse. Therefore, the network must have the ability to dynamically fit the corresponding function based on the input. Higher degrees of freedom mean the model can fit a wide range of features with significant differences. Additionally, the bias term in the network can also dynamically fit based on input using UAT, which we refer to as bias freedom. Models with bias freedom have a stronger dynamic fitting capability. The mathematical model generated by the residual network is an example of UAT with bias freedom, achieving the fitting capability of a multi-layer network with just shallow layers. We will talk about this problem in Section 4.3.

2. Data Influence: Data is another critical factor in deep learning training. Since natural images are usually diverse, limited data can lead to overfitting. For example, in a cat-dog classification task, if dogs in the training set mostly appear against backgrounds of blue skies and grass, these backgrounds might be mistakenly considered part of the dog's features, a phenomenon known as feature coupling. If cats rarely appear in such backgrounds in the training set but do in the test set, they might be misclassified as dogs. Training on a large, diverse dataset means that common backgrounds frequently appear with various objects or animals, such as cats, dogs, and elephants against blue skies, white clouds, houses, and trees. Thus, even when classifying objects, the model inherently learns background information. In this context, fine-tuning on the cat-dog dataset yields better results because the background does not interfere with dog feature recognition. Without sufficient data, even a network with strong fitting capabilities cannot perform well and is prone to overfitting. This problem will be considered in Section 4.4.

Considering these factors, it is clear that simply improving the model architecture is not enough. A robust training

dataset is equally important for achieving high predictive accuracy.

4.3 Model Depth Effect

Based on the conclusions from Section 4.2, we will explore the impact of network depth on predictive performance in this section. According to Eq. 8 and Eq. 9, we know that model depth primarily affects the degrees of freedom of the model parameters in UAT. Therefore, we designed Para-Former models with 24 layers and varying depths of 2, 3, and 6, and compared their predictive performance with that of a ViT with a depth of 6. The results are shown in Table 3.

It is evident that all results surpass those of Para-F-1-24 in Table 2 because the networks in Table 3 have higher degrees of freedom in their parameters. Notably, Para-F-6-6 consistently outperforms Para-F-2-24 and Para-F-3-24, indicating that the higher the degrees of freedom of the parameters, the stronger the model's performance. However, the model cannot be too deep, so we must balance network depth and layer count. Another issue is that almost all models hit a performance bottleneck. Particularly, the prediction accuracy for OxfordIIITPet and Flower-102 is very poor. This is because these datasets are relatively small and exhibit high data diversity.

4.4 Dataset Effect

Current deep learning models generally perform well on the aforementioned datasets. However, as shown in Table 3, the prediction accuracy of Para-Former is not very high, especially on the OxfordIIITPet and Flower-102 datasets. What is the fundamental reason for this poor performance? In this section, we will explain the relationship between deep learning and data.

First, the results were obtained using the validated ViT model (Para-F-1-1 and Para-Former-6-6), which performed similarly to Para-Former. Given that the ViT model has been extensively validated and is a commonly used model architecture, the effectiveness of the models is unquestionable. So, why is the performance so poor? The root cause lies in the data itself: the model's performance is generally directly proportional to the amount of available data. Take the Flower-102 dataset, for example, which includes 102 classes of flowers but has only 1,020 images in the training set, with some reserved for validation. This is far from sufficient to train deep learning models. Why, then, does the OxfordIIIT-Pet dataset, which has more data per class on average than Flower-102, perform even worse? This is due to the diversity of the data (i.e., the similarity of features within the same class) and the variety of backgrounds.

Figures 4 and 5 show some results from the OxfordIIIT-Pet and Flower-102 datasets. Both figures indicate that prediction results are worse for categories with diverse backgrounds or inherent diversity, as seen in Figures 4.b and 5.b. Conversely, results are relatively better when the background is consistent but the species vary, as shown in Figures 4.a and 5.a. The best results are achieved when both the background and species variation are minimal, as shown in Figures 4.c and 5.c. Although the Flower-102 dataset has fewer images, its backgrounds are more consistent and there is less variation among the species. In contrast, animals in the OxfordIIITPet dataset can appear in various forms, while flowers in the Flower-102 dataset are usually photographed from relatively fixed angles, leading to better performance.

Does this mean that deep learning models are not powerful enough or lack generalization and reasoning capabilities? On the contrary, achieving around 25% accuracy on OxfordIIITPet and about 40% on Flower-102 with such limited data actually demonstrates the strength and generalization capability of deep learning models. A common misconception about deep learning is that humans, even if unfamiliar with certain species, can accurately distinguish OxfordIIITPet and Flower-102 images by carefully examining them, thus believing that deep learning lacks generalization or human-like inferencing ability. This viewpoint is flawed. Assuming the human brain functions as a UAT model, it can automatically disregard background information before seeing these images, as this background is frequently encountered in daily life, and the UAT in the human brain has already fitted this information. Therefore, humans only need to focus on specific information and then use it to train their brains to find similarities and differences. For instance, a 10-year-old child's brain, observing the world at 60 frames per second, would have processed about 1×10^{10} high-resolution images by the age of 10, many of which include common objects and backgrounds in daily life. Therefore, criticizing deep learning models for being unable to train on small datasets and lacking universality and generalization is unfair.

Thus, we should train on large datasets and then fine-tune on smaller ones. This is also the current deep learning training method, showcasing the powerful generalization capability of deep learning models, especially when there is some correlation between datasets. Following this fine-tuning approach, we fine-tuned the pre-trained ViT model on the aforementioned datasets for just 10 epochs. The results, as shown in Table 4, indicate a significant improvement in prediction performance.

Figure 6 shows the rationality behind fine-tuning. Since we essentially adjust parameters based on the input to produce target results using UAT, if there is a correlation in the data, the corresponding parameters derived from the input ($\alpha_i, \mathbf{W}_i$, and $\theta_i; i = 1, \cdot, N$) will be similar. Therefore, fine-tuning is feasible.

5 Conclusion

In this paper, we propose a parallel network based on the Universal Approximation Theorem (UAT) and validate its effectiveness on large datasets. We analyze the characteristics of both serial and parallel networks. Serial networks, especially those based on residual connections, often have higher degrees of parameter freedom, leading to stronger generalization capabilities. However, as the network depth increases, the inference time also becomes longer. On the other hand, parallel networks trade computational resources for time, meaning their runtime is not affected by the number of network layers. This allows for the design of multi-layer parallel networks that can replace serial networks to achieve fast inference. Additionally, we analyze the impact and reasons data have on deep learning. From the UAT perspective,

Table 3: This table shows the predictions of Para-former-6-6, Para-former-2-24, Para-former-3-24, Para-former-6-24 on the datasets CIFAR-10, CIFAR-100, STL-10, OxfordIIITPet, and Flower-102.

Dataset	Para-F-6-6	Para-F-2-24	Para-F-3-24	Para-F-6-24
Cifar-10	79.39	74.75	77.21	**79.92**
Cifar-100	55.18	51.04	52.65	**56.15**
STL-10	62.81	61.60	62.75	**64.30**
OxfordIIITPet	**26.05**	23.65	26.02	25.92
Flower-102	37.97	39.17	40.12	**40.51**

Figure 4: OxfordIIITPet dataset's accuracy on different classes.

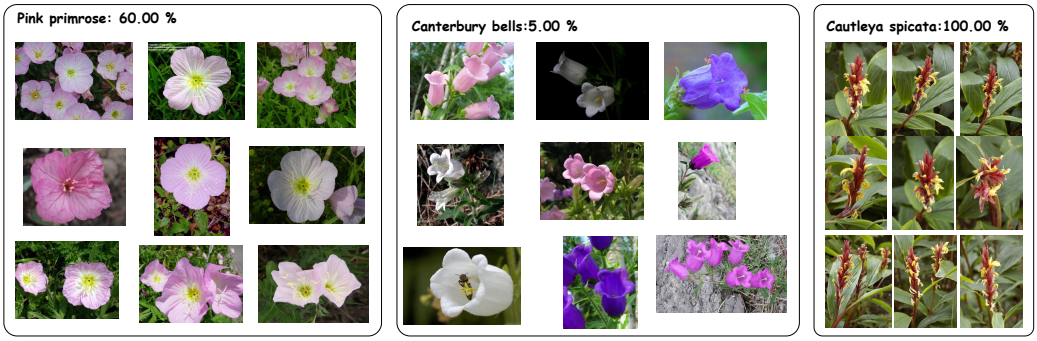

Figure 5: Flower-102 dataset's accuracy on different classes.

Table 4: This table shows the results of fine-tuning using Google's official pre-trained parameters. The pre-trained parameters were obtained from training the ViT model on the ImageNet dataset.

Dataset	Cifar-10	Cifar-100	STL-10	OxfordIIITPet	Flower-102
Para-F-12-12	98.62	91.36	99.47	91.87	96.89

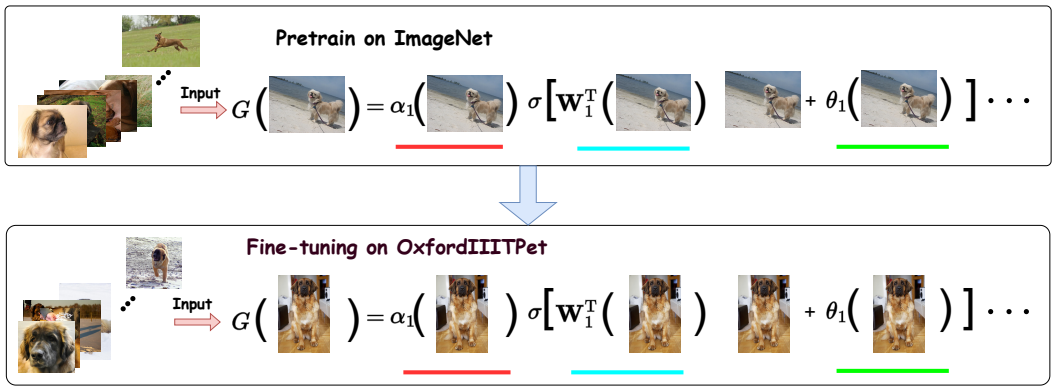

Figure 6: The fine-tuning process from the perspective of UAT.

training on large datasets essentially enhances the model's ability to fit data.

References

Ben-Nun, T.; and Hoefler, T. 2018. Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis. arXiv:1802.09941.

Coates, A.; Ng, A.; and Lee, H. 2011. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *International Conference on Artificial Intelligence and Statistics*.

Cybenko, G. 2007. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 303–314.

Gale, T.; Zaharia, M. A.; Young, C.; and Elsen, E. 2020. Sparse GPU Kernels for Deep Learning. *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–14.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Hornik, K.; Stinchcombe, M. B.; and White, H. L. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2: 359–366.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images.

Liu, Y.; Zhang, K.; Li, Y.; Yan, Z.; Gao, C.; Chen, R.; Yuan, Z.; Huang, Y.; Sun, H.; Gao, J.; et al. 2024. Sora: A review on background, technology, limitations, and opportunities of large vision models. *arXiv preprint arXiv:2402.17177*.

Ma, X.; Fang, G.; and Wang, X. 2023. LLM-Pruner: On the Structural Pruning of Large Language Models. *ArXiv*, abs/2305.11627.

Nilsback, M.-E.; and Zisserman, A. 2008. Automated Flower Classification over a Large Number of Classes. *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, 722–729.

Parkhi, O. M.; Vedaldi, A.; Zisserman, A.; and Jawahar, C. V. 2012. Cats and dogs. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 3498–3505.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. *ArXiv*, abs/1505.04597.

Sun, M.; Liu, Z.; Bair, A.; and Kolter, J. Z. 2023. A Simple and Effective Pruning Approach for Large Language Models. *ArXiv*, abs/2306.11695.

Vaswani, A.; Shazeer, N. M.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *Neural Information Processing Systems*.

Wang, W.; and Li, Q. 2024. Universal Approximation Theory: The basic theory for deep learning-based computer vision models. arXiv:2407.17480.

Xia, M.; Gao, T.; Zeng, Z.; and Chen, D. 2023. Sheared LLaMA: Accelerating Language Model Pre-training via Structured Pruning. *ArXiv*, abs/2310.06694.

A The UAT Format of Transformer

A.1 The UAT Format of Transformer-based ViTs

Eq.(1) and (2) give an example of the matrix-vector format of MHA and FFN. Eq. (3) and Eq. (4) give the corresponding UAT format of $\mathbf{x}_{i+1}$ and $\mathbf{x}_{i+2}$. It is obvious that the bias and weight terms in the UAT corresponding to Transformed can change dynamically based on the input.

Due to the complexity of the variables involved, we will combine some terms to align the resulting equations with the standard UAT form. To distinguish these parameter variables, we will keep the original variables unchanged. If a parameter variable is influenced only by other variables, we will denote it with a bar on top, such as $\bar{\mathbf{W}}$. If it is influenced by the input $\mathbf{x}_i$, we will mark it with a hat, such as $\hat{\mathbf{W}}$.

$$MHA(\mathbf{x}'_i) = \mathbf{W}'_{i,1}\mathbf{x}'_i \tag{1}$$

$$FFN(\mathbf{x}'_i) = \mathbf{W}'_{i,3}\sigma(\mathbf{W}'_{i,2}\mathbf{x}'_i + \mathbf{b}'_{i,2}) + \mathbf{b}'_{i,3} \tag{2}$$

$$\begin{aligned}
\mathbf{x}'_{i+1} &= \mathbf{W}'_{i,1}\mathbf{x}'_i + \mathbf{W}'_{i,3}\sigma[\mathbf{W}'_{i,2}(\mathbf{W}'_{i,1}\mathbf{x}'_i) + \mathbf{b}'_{i,2}] + \mathbf{b}'_{i,3} \\
&= \mathbf{W}'_{i,1}\mathbf{x}'_i + \mathbf{W}'_{i,3}\sigma(\underline{\mathbf{W}'_{i,2}\mathbf{W}'_{i,1}}\mathbf{x}'_i + \mathbf{b}'_{i,2}) + \mathbf{b}'_{i,3} \\
&= \mathbf{W}'_{i,1}\mathbf{x}'_i + \mathbf{W}'_{i,3}\sigma(\hat{\mathbf{W}}'_{i,1}\mathbf{x}'_i + \mathbf{b}'_{i,2}) + \mathbf{b}'_{i,3}
\end{aligned} \tag{3}$$

$$\begin{aligned}
\mathbf{x}'_{i+2} &= \mathbf{W}'_{i+1,1}\mathbf{x}'_{i+1} + \mathbf{W}'_{i+1,3}\sigma(\hat{\mathbf{W}'}_{i+1,1}\mathbf{x}'_{i+1} + \mathbf{b}'_{i+1,2}) + \mathbf{b}'_{i+1,3} \\
&= \mathbf{W}'_{i+1,1}[\mathbf{W}'_{i,1}\mathbf{x}'_i + \mathbf{W}'_{i,3}\sigma(\hat{\mathbf{W}}'_{i,1}\mathbf{x}'_i + \mathbf{b}'_{i,2}) + \mathbf{b}'_{i,3}] \\
&+ \mathbf{W}'_{i+1,3}\sigma\{\hat{\mathbf{W}'}_{i+1,1}[\mathbf{W}'_{i,1}\mathbf{x}'_i + \mathbf{W}'_{i,3}\sigma(\hat{\mathbf{W}}'_{i,1}\mathbf{x}'_i + \mathbf{b}'_{i,2}) + \mathbf{b}'_{i,3}] + \mathbf{b}'_{i+1,2}\} + \mathbf{b}'_{i+1,3} \\
&= \underline{\mathbf{W}'_{i+1,1}\mathbf{W}'_{i,1}}\mathbf{x}'_i + \underline{\mathbf{W}'_{i+1,1}\mathbf{W}'_{i,3}}\sigma(\hat{\mathbf{W}}'_{i,1}\mathbf{x}'_i + \mathbf{b}'_{i,2}) + \underline{\mathbf{W}'_{i+1,1}\mathbf{b}'_{i,3}} \\
&+ \mathbf{W}'_{i+1,3}\sigma\{\underline{\hat{\mathbf{W}'}_{i+1,1}\mathbf{W}'_{i,1}}\mathbf{x}'_i + \underline{\hat{\mathbf{W}'}_{i+1,1}\mathbf{W}'_{i,3}}\sigma(\hat{\mathbf{W}}'_{i,1}\mathbf{x}'_i + \mathbf{b}'_{i,2}) + \underline{\hat{\mathbf{W}'}_{i+1,1}\mathbf{b}'_{i,3} + \mathbf{b}'_{i+1,2}}\} + \mathbf{b}'_{i+1,3} \\
&= \hat{\mathbf{W}'}_{i+1,2}\mathbf{x}'_i + \hat{\mathbf{W}'}_{i+1,3}\sigma(\hat{\mathbf{W}}'_{i,1}\mathbf{x}'_i + \mathbf{b}'_{i,2}) + \mathbf{W}'_{i+1,3}\sigma\{\hat{\mathbf{W}'}_{i+1,4}\mathbf{x}'_i + \underline{\hat{\mathbf{W}'}_{i+1,5}\sigma(\hat{\mathbf{W}}'_{i,1}\mathbf{x}'_i + \mathbf{b}'_{i,2}) + \hat{\mathbf{b}'}_{i+1,2}}\} + \underline{\hat{\mathbf{b}'}_{i,3} + \mathbf{b}'_{i+1,3}} \\
&= \hat{\mathbf{W}'}_{i+1,2}\mathbf{x}'_i + \hat{\mathbf{W}'}_{i+1,3}\sigma(\hat{\mathbf{W}}'_{i,1}\mathbf{x}'_i + \mathbf{b}'_{i,2}) + \mathbf{W}'_{i+1,3}\sigma\{\hat{\mathbf{W}'}_{i+1,4}\mathbf{x}'_i + \hat{\mathbf{b}'}_{i+1,4}\} + \hat{\mathbf{b}'}_{i+1,5}
\end{aligned} \tag{4}$$