TOKENUNIFY: SCALABLE AUTOREGRESSIVE PRE-TRAINING FOR LARGE SCALE EM IMAGE SEGMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive next-token prediction, a standard pretraining method for large-scale language models, excels in handling long sequential data. However, its application to complex visual tasks, particularly biological imaging, faces challenges due to the spatial continuity and high dimensionality of biological images. High-resolution 3D biological images, such as electron microscopy (EM) brain scans, offer ideal long-sequence data, but existing methods struggle to fully leverage this characteristic. To address these challenges, we introduce **TokenUnify**, a novel pretraining method that integrates random token prediction, next-token prediction, and next-all token prediction. We provide theoretical evidence demonstrating that TokenUnify mitigates cumulative errors in visual autoregression, particularly when dealing with complex three-dimensional anatomical structures. In conjunction with TokenUnify, we have assembled a large-scale, ultra-high-resolution EM brain image dataset comprising over 120 million finely annotated voxels. This dataset not only represents the largest neuron segmentation dataset to date but, more importantly, provides ideal long-sequence biological image data that fully exhibits spatial continuity. Leveraging the Mamba network, which is inherently suited for long-sequence modeling, TokenUnify capitalizes on the advantages of autoregressive methods in processing long-sequence data, achieving a 45% performance improvement on downstream EM neuron segmentation tasks compared to existing methods. Furthermore, TokenUnify demonstrates superior scalability over MAE and traditional autoregressive methods, effectively bridging the gap between pretraining strategies for language and vision models. Code is available at `https://anonymous.4open.science/r/TokenUnify-3DBF`.

1 INTRODUCTION

Large language models (LLMs) have demonstrated impressive scaling capabilities, reaching trillions of parameters through pretraining Achiam et al. (2023); Touvron et al. (2023a;b). This success is primarily attributed to high-quality data and effective autoregressive models. These models benefit from strong scaling laws due to the structured and sequential nature of text data, which allows unification into a single next-token prediction task. However, when extending to multimodal tasks such as Unified IO Lu et al. (2023) and Qwen VL Bai et al. (2023b), these models often fail to achieve state-of-the-art performance on fine-grained image tasks, particularly in biological imaging.

Unlike language, the complexity of visual signals, especially in biological images, has led to diverse approaches in visual pretraining. Contrastive learning methods like DINO v2 Oquab et al. (2023) excel in fine-grained representation, while masked reconstruction methods such as MAE He et al. (2022); Chen et al. (2023a) offer good scalability and zero-shot classification abilities. However, these methods exhibit poor scaling laws, where increasing model size does not yield expected performance gains Singh et al. (2023) (see Section A in the appendix). To achieve scaling laws similar to language models, approaches like AIM El-Nouby et al. (2024) and LVM Bai et al. (2023c) have introduced autoregressive tasks into the visual domain, showing promising scaling properties. However, image sequence disorder and error accumulation in autoregressive tasks often degrade performance in smaller models Bachmann & Nagarajan (2024). Additionally, the computational burden of

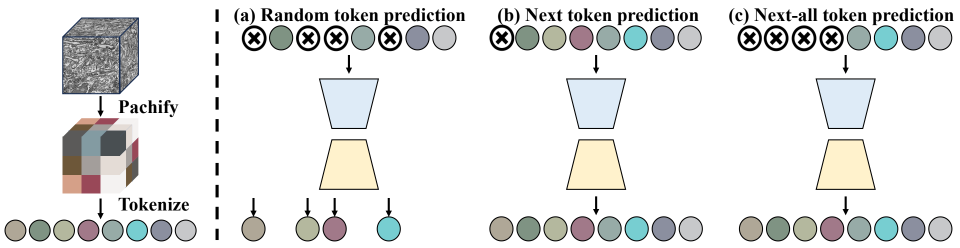
Pachify
Tokenize
(a) Random token prediction
(b) Next token prediction
(c) Next-all token prediction

Figure 1: TokenUnify prediction paradigms divide the 3D EM image into non-overlapping patches, which are tokenized into a sequence. Three tasks are performed for rich 3D image representations: (a) random token prediction, (b) next token prediction, and (c) next-all token prediction.

long-sequence images, particularly in high-resolution 3D biological scans, makes researching image autoregressive tasks particularly challenging.

We summarize the challenges of visual autoregressive tasks in biological imaging as follows: 1) *How to reduce error accumulation in visual autoregression to achieve stronger scaling laws, especially when dealing with complex anatomical structures?* 2) *How to develop more efficient computational methods to handle massive, high-dimensional biological image data?* 3) *How to construct spatially correlated long-sequence relationships in biological images?*

This paper aims to tackle the above three critical challenges in the context of biological imaging, particularly focusing on EM brain scans. 1) To address the issue of cumulative errors in visual autoregression, we propose **TokenUnify**, a novel mixed token prediction paradigm. TokenUnify integrates next-token prediction, next-all token prediction, and random token prediction (as illustrated in Fig. 1), leveraging global information to overcome the limitations of local receptive fields in complex anatomical structures. We theoretically demonstrate that this mixed approach reduces cumulative errors while maintaining favorable scaling laws. 2) To alleviate computational burdens, we introduce the Mamba architecture, which reduces the computational complexity of autoregressive tasks from quadratic (as in Transformers) to linear. This is particularly crucial for processing high-resolution 3D biological images. Detailed comparisons of the scaling properties between Mamba and Transformer architectures reveal that Mamba achieves superior performance and efficiency in large-scale autoregressive visual models, especially for biological imaging tasks. 3) To construct spatially correlated long-sequence relationships, we have collected large-scale, ultra-high-resolution 3D electron microscopy (EM) images of mouse brain slices. The ultra-high resolution allows for thousands of continuous image tokens, ensuring robust spatial continuity crucial for understanding complex neuronal structures. We have fully annotated six different functional regions within the mouse brain, totaling 120 million pixels, resulting in the largest manually annotated neuron dataset to date. This comprehensive dataset also serves as a unified benchmark for evaluating experimental performance in biological image analysis[1].

Pretraining with TokenUnify led to a **45%** improvement in performance on subsequent EM neuron segmentation tasks. The mixed training paradigm of TokenUnify outperformed MAE He et al. (2022) by **21%** in pretraining performance, even with fewer parameters. Furthermore, TokenUnify demonstrated superior scaling properties of autoregressive models, offering a promising approach for pretraining large-scale visual models in biological imaging.

Our contributions can be summarized as follows:

1. We introduce a novel pre-training paradigm, **TokenUnify**, which models visual pre-training tasks from multiple perspectives at the token level, specifically designed for biological imaging. This ensures sublinear growth of the scaling law while demonstrating superior fine-grained feature extraction capabilities compared to MAE in smaller model pre-training, crucial for detecting subtle anatomical features. We also provide a theoretical explanation for this phenomenon in the context of biological image analysis.

[1]We commit to open-sourcing the dataset and code of the paper to facilitate future research in biological imaging.

2. We achieve a **45%** performance improvement on the neuron segmentation task and, for the first time, validate the Mamba model with billion-level parameters on visual tasks, demonstrating the effectiveness and efficiency of TokenUnify in long-sequence visual autoregression for complex 3D biological images.

3. We provide a large-scale biological image dataset with 120 million annotated pixels, offering a long-sequence image dataset to validate the potential of autoregressive methods in biological imaging.

2 Related work

Recent advances in large language models (LLMs) have unified various NLP tasks under a single architecture, formulating them as generation tasks. This architecture can be categorized into BERT-like Devlin et al. (2018); Xia et al. (2020); Lee et al. (2020); Song et al. (2023); Mo et al. (2024) and GPT-like Brown et al. (2020); Radford et al. (2019); Li et al. (2024a); Zhou et al. (2023) models. The latter, decoder-only autoregressive structure, has been shown to be more effective, as validated by products like ChatGPT. Subsequent works have built upon GPT, introducing techniques like RMSNorm Zhang & Sennrich (2019), SwiGLU, and RoPE to ensure efficient and stable training. The LLaMA series Touvron et al. (2023a;b) has improved training efficiency, while the Qwen series Bai et al. (2023b;a); Xiang et al. (2024) has focused on data cleaning and filtering for Chinese language models. Currently, LLMs have surpassed human-level performance in many text processing tasks Achiam et al. (2023).

In multi-modal tasks, the CLIP Radford et al. (2021) and BLIP series Li et al. (2022; 2023a); Dai et al. (2024) have pioneered contrastive learning on image-text pairs, achieving remarkable zero-shot classification and generalization capabilities. Further works Zhang et al. (2023); Chen et al. (2023b); Wang et al. (2022); Zhou et al. (2024) have applied multi-modal models to specific domains. By processing arbitrary modalities into a unified token format Lu et al. (2023); Wang et al. (2023); Chen et al. (2024a), these models can generate outputs in any modality. However, there is still room for improvement in fine-grained visual tasks, and training large vision models remains an open problem.

Self-supervised pre-training has been a cornerstone for enhancing model representation capabilities. Approaches based on contrastive learning for representation extraction Chen et al. (2024b); Zbontar et al. (2021); He et al. (2020); Li et al. (2021) and masked reconstruction methods He et al. (2022); Chen et al. (2023a); Li et al. (2023c); Ding et al. (2022); Chen et al. (2024d; 2023d) have shown promise. However, current vision models have not exhibited the same sublinear scaling laws as language models. To address this issue, some tasks have adopted autoregressive pre-training paradigms similar to those used in language models Chen et al. (2020); Bai et al. (2023c); El-Nouby et al. (2024), though the training costs remain a significant concern. In this paper, we explore the potential of long visual token autoregressive pre-training and introduce a collaborative training scheme for long token prediction. Our approach aims to balance the scaling laws and training costs, demonstrating improvements in fine-grained visual tasks.

3 Method

3.1 Overview

Our theoretical contributions include proving the parameter independence of MAE performance (see Section A), establishing the strong correlation between autoregressive model performance and parameter count (see Section B), and demonstrating the advantages of next-all token prediction from both intuitive (see Section C.1) and theoretical perspectives (see Section C.2).

Our experimental framework comprises two stages: pre-training and fine-tuning. During the pre-training stage, we leverage only the unlabeled raw data $\mathcal{X}$ to learn a generic visual representation $f_{\theta_1}(\cdot)$, parameterized by θ_1. We employ a mixed token prediction strategy to capture both local and global dependencies in the data (see Section 3.2). Additionally, we utilize Mamba for efficient modeling of long sequences in autoregressive tasks, enhancing computational efficiency (see Section 3.3).

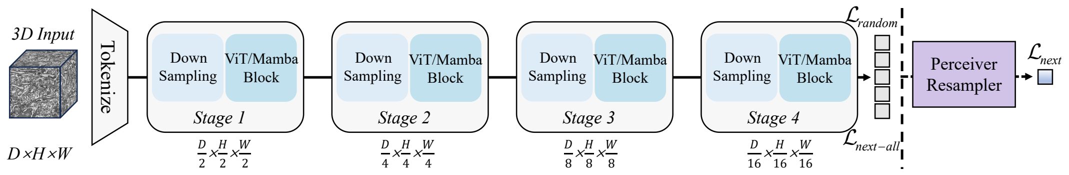

Figure 2: illustrates the main pretraining workflow of TokenUnify. The image $\mathbf{X}$ is fed into the Tokenizer, transforming it into a long sequence of tokens $\mathbf{x}_i|_{i=1}^{K}$. The predictions for the random token, next token, and next-all token are performed sequentially. The Perceiver Resampler is employed to convert varying-size large feature maps into a few visual tokens (see Section 3.2).

In the fine-tuning stage, we use both the raw data $\mathcal{X}$ and the corresponding labels $\mathcal{Y}$ to adapt the pre-trained representation to specific downstream tasks. Let $g_{\theta_2}(\cdot)$ be the task-specific model, parameterized by θ_2. We initialize θ_2 with the pre-trained weights θ_1 and optimize the task-specific objective. Further details are provided in Sections 3.4 and F.3.

To illustrate the application of our framework, consider the modeling of EM images. Given a total of T EM images $\mathcal{X} = \{\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(T)}\}$, where each $\mathbf{X}^{(t)} \in \mathbb{R}^{D\times H\times W}$ represents a 3D image with depth D, height H, and width W, we aim to learn a meaningful representation of this high-dimensional, long-sequence visual data by leveraging its inherent spatial structure and continuity. To achieve this, we partition each large 3D image $\mathbf{X}$ into smaller patches $\mathbf{x} \in \mathbb{R}^{D'\times H'\times W'}$.

3.2 Mixed-mode autoregressive modeling

We theoretically demonstrate the effectiveness of MAE He et al. (2022) on smaller models, the scaling advantages of next token prediction, and the ability of next-all token prediction to mitigate cumulative errors in autoregressive models. Based on these insights, we propose a hybrid training paradigm that aims to combine the strengths of these three methods, as shown in Fig. 2.

Given an image $\mathbf{X}$, we first divide it into a sequence of K non-overlapping patches $\{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$. Standard autoregressive modeling typically adopts a fixed left-to-right factorization:

$$p(\mathbf{x}) = \prod\nolimits_{i=1}^{K} p(\mathbf{x}_i \mid \mathbf{x}_{<i}), \tag{1}$$

where $\mathbf{x}_{<i}$ denotes all patches preceding $\mathbf{x}_i$.

We introduce TokenUnify, a mixed-mode autoregressive modeling approach designed to enhance existing autoregressive image modeling techniques Chen et al. (2020); Bai et al. (2023c). TokenUnify combines three distinct prediction tasks: random token prediction, next token prediction, and next-all token prediction, instead of using the fixed factorization in Eq. 1.

Random Token Prediction. Given the full patch sequence $\mathbf{x}_{1:K}$, we randomly mask out a subset of patches $\mathcal{M} \subset \{1, \ldots, K\}$ and train the model to predict the masked patches conditioned on the remaining context:

$$\mathcal{L}_{\text{random}} = -\sum\nolimits_{i\in\mathcal{M}} \log p(\mathbf{x}_i|\mathbf{x}_{\bar{\mathcal{M}}}), \tag{2}$$

where $\mathbf{x}_{\bar{\mathcal{M}}} = \{\mathbf{x}_i \mid i \notin \mathcal{M}\}$ denotes the unmasked patches.

Next Token Prediction. We integrate the standard next token prediction loss into our task. In this setup, we use the Perceiver Resampler Alayrac et al. (2022) (see Section F.2) to convert the variable-sized feature maps generated by the Vision Encoder into a fixed number of visual tokens. This loss trains the model to predict the next patch $\mathbf{x}i$ given the preceding context $\mathbf{x}_{<i}$:

$$\mathcal{L}_{\text{next}} = -\sum\nolimits_{i=1}^{K} \log p(\mathbf{x}_i|\mathbf{x}_{<i}). \tag{3}$$

Next-All Token Prediction. To encourage the model to capture longer-range dependencies, we extend the next token prediction to a next-all token prediction task. For each patch $\mathbf{x}_i$, the model is trained to predict not only $\mathbf{x}_i$ but also all the subsequent patches $\mathbf{x}_{i:K}$ in the sequence:

$$\mathcal{L}_{\text{next-all}} = -\sum\nolimits_{i=1}^{K} \sum\nolimits_{j=i}^{K} \log p(\mathbf{x}_j|\mathbf{x}_{<i}). \tag{4}$$

Our pre-training algorithms are summarized in Algorithm 1. By alternating between these token prediction tasks every 100 epochs, we prevent the model from converging to a trivial solution and encourage it to learn meaningful representations from the input data. This alternating training strategy enables the model to capture both local and global dependencies within the patch sequence, thereby enhancing performance on downstream tasks. The workflow of TokenUnify is illustrated in Fig. 2.

Algorithm 1: TokenUnify Pre-training

```
Input  : Unlabeled image data $X = \{X^{(1)}, \ldots, X^{(T)}\}$
Input  : Model parameters $\theta_1$
Output: Pre-trained model $f_{\theta_1}(\cdot)$
1 for $t \leftarrow 1$ to $T$ do
2     Partition $X^{(t)}$ into patches $\{x_1, \ldots, x_K\}$
3     Tokenize patches: $\{x_1, \ldots, x_K\} \rightarrow$ tokens
4     Compute loss functions:
5     Random token prediction: $\mathcal{L}_{\text{random}} = -\sum_{i \in M} \log p(x_i \mid x_{\bar{M}})$
6     Next token prediction: $\mathcal{L}_{\text{next}} = -\sum_{i=1}^{K} \log p(x_i \mid x_{<i})$
7     Next-all token prediction: $\mathcal{L}_{\text{next-all}} = -\sum_{i=1}^{K} \sum_{j=i}^{K} \log p(x_j \mid x_{<i})$
8     Update $\theta_1$ to minimize $\mathcal{L}_{\text{random}}$, $\mathcal{L}_{\text{next}}$, $\mathcal{L}_{\text{next-all}}$
9 return $f_{\theta_1}(\cdot)$
```

3.3 Mamba ordering

While the aforementioned mix token prediction task improves sequence autoregressive modeling capabilities, it also introduces additional computational complexity for long sequences. Inspired by the *Mamba* strategy proposed by Gu & Dao (2023), we introduce an enhanced approach to effectively model long sequences in volumetric EM images. Traditional sequence modeling methods often struggle with capturing long-range dependencies due to their rigid sequential nature. Our enhanced Mamba ordering strategy addresses this by incorporating a more sophisticated and flexible sequence modeling approach.

The fundamental idea behind Mamba ordering is to dynamically prioritize regions of the sequence based on contextual significance rather than adhering to a fixed order. This is achieved through an adaptive mechanism that evaluates the importance of different patches within the sequence and adjusts the processing order accordingly. By doing so, Mamba ordering enhances the model's ability to capture intricate patterns and long-range dependencies more effectively.

Mathematically, let $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K\}$ represent the sequence of patches. Instead of processing these patches in a fixed order, we define a dynamic ordering function $\sigma : \{1, 2, \ldots, K\} \rightarrow \{1, 2, \ldots, K\}$ that determines the sequence in which patches are processed. The Mamba ordering objective can be formulated as:

$$\mathcal{L}_{\text{mamba}} = -\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[\sum_{i=1}^{K} \log p(\mathbf{x}_{\sigma(i)} | \mathbf{x}_{\sigma(<i)}) \right], \tag{5}$$

where $\mathbf{x}_{\sigma(<i)}$ represents the context preceding the i-th patch in the dynamically determined order.

To optimize this objective, we introduce a context-aware attention mechanism that assesses the relevance of each patch with respect to the overall sequence. This mechanism outputs a relevance score for each patch, guiding the dynamic ordering function σ to prioritize patches that are most informative for subsequent predictions. By iteratively updating the relevance scores and reordering the patches, Mamba ordering ensures that the model focuses on the most crucial aspects of the sequence at each step.

Consider the state-space model representation where the hidden state $\mathbf{h}(t)$ evolves dynamically based on the input $\mathbf{x}(t)$. The state-space equations are given by:

$$\mathbf{h}'(t) = \mathbf{A}(t)\mathbf{h}(t) + \mathbf{B}(t)\mathbf{x}(t), \quad \mathbf{y}(t) = \mathbf{C}(t)\mathbf{h}(t), \tag{6}$$

where $\mathbf{A}(t)$, $\mathbf{B}(t)$, and $\mathbf{C}(t)$ are time-dependent matrices. Specifically, $\mathbf{B}(t)$ and $\mathbf{C}(t)$ are parameterized by the input $\mathbf{x}(t)$ as follows:

$$\mathbf{B}(t) = s_B(\mathbf{x}(t)), \quad \mathbf{C}(t) = s_C(\mathbf{x}(t)), \quad \mathbf{\Delta}(t) = \tau_\Delta(\text{Linear}(\mathbf{x}(t))), \tag{7}$$

where s_B, s_C, and τ_Δ are functions that map the input to the respective parameters.

The benefits of our enhanced Mamba ordering are twofold. First, it mitigates error accumulation by allowing the model to refine its predictions based on a globally coherent understanding of the sequence. Second, it enhances the model's capacity to capture long-range dependencies, as the dynamic ordering can adapt to the inherent structure and complexity of the data.

Empirical results demonstrate that our enhanced Mamba ordering significantly improves the performance of sequence modeling tasks in volumetric EM images, particularly for long sequences. By enabling a more adaptive and context-aware approach to sequence processing, our enhanced Mamba ordering represents a substantial advancement over traditional methods, offering a robust and scalable solution for high-dimensional visual data.

3.4 Finetuning and Segmentation

The segmentation network, denoted as $g_{seg}(\cdot)$, consists of an encoder $g_e(\cdot)$ and a decoder $g_d(\cdot)$:

$$g_{seg}(x; \theta_s) = g_d(g_e(x)), \quad \theta_s = \{\theta_e, \theta_d\}, \tag{8}$$

where θ_s represents the parameters of the entire segmentation network, and θ_e and θ_d are the parameters of the encoder and decoder, respectively.

The encoder $g_e(\cdot)$ gradually downsamples the input volume and extracts high-level semantic features, while the decoder $g_d(\cdot)$ upsamples the encoded features back to the original resolution. Meanwhile, the output of each downsampling layer in the encoder is connected to the corresponding layer in the decoder via skip connections to fuse local and global multi-scale information. We adopt 3D ResUNet/ViT/Mamba as the backbone network, respectively.

The output of the segmentation network $\hat{y} = g_{seg}(x) \in \mathbb{R}^{C \times D \times H \times W}$ represents the predicted affinity map Li et al. (2018; 2023b), corresponding to the connectivity probability of each voxel in three directions. During training, the loss function for labeled samples is the mean squared error between the predicted and ground-truth affinity maps:

$$L_{seg} = \frac{1}{|D_l|} \sum\nolimits_{i=1}^{|D_l|} |\hat{y}_i - y_i|^2 = \frac{1}{|D_l|} \sum\nolimits_{i=1}^{|D_l|} \left| g_{seg}(x_i^l) - y_i \right|^2 . \tag{9}$$

Our fine-tuning algorithm is summarized in Algorithm 2. During inference, for any new test sample x_t, forward propagation through $g_{seg}(x_t)$ yields its predicted affinity map. This predicted affinity map is then post-processed using a seeded watershed algorithm and a structure-aware region agglomeration algorithm Funke et al. (2018); Beier et al. (2017) to obtain the final neuron instance segmentation. Detailed information on our segmentation process can be found in Section F.3, and the segmentation pipeline is illustrated in Fig. 7.

Algorithm 2: TokenUnify Fine-tuning

```
Input  : Labeled data $\mathcal{D}_l = \{(x_i^l, y_i)\}_{i=1}^{|\mathcal{D}_l|}$
Input  : Pre-trained model $f_{\theta_1}(\cdot)$
Input  : Segmentation model $g_{\theta_2}(\cdot)$
Output: Fine-tuned segmentation model $g_{\theta_2}(\cdot)$
1 Initialize $\theta_2$ with $\theta_1$
2 for $i \leftarrow 1$ to $|\mathcal{D}_l|$ do
3     $\hat{y}_i = g_{\theta_2}(f_{\theta_1}(x_i^l))$
4     $\mathcal{L}_{\text{seg}} = \frac{1}{|\mathcal{D}_l|} \sum_{i=1}^{|\mathcal{D}_l|} |\hat{y}_i - y_i|^2$
5     Update $\theta_2$ to minimize $\mathcal{L}_{\text{seg}}$
6 return $g_{\theta_2}(\cdot)$
```

4 Dataset and Metrics

Dataset. For the pretraining phase of TokenUnify, we collect a vast amount of publicly available unlabeled EM imaging data, from four large-scale electron microscopy (EM) datasets: Full Adult Fly Brain (FAFB) Schlegel et al. (2021), MitoEM Wei et al. (2020), FIB-25 Takemura et al. (2017), and Kasthuri15 Kasthuri et al. (2015). These datasets cover a diverse range of organisms, including Drosophila, mouse, rat, and human samples, totaling over 1 TB. The details of the pretraining datasets can be found in Table 3. We sample from the datasets with equal probability and ensure that each brain region has an equal chance of being sampled, guaranteeing the diversity of the pretraining data.

For downstream fine-tuning and segmentation, we employ two datasets: a smaller dataset, AC3/4, and a larger dataset, MEC, for algorithm validation. The AC3/4 dataset Kasthuri et al. (2015) consists of mouse somatosensory cortex datasets with 256 and 100 successive EM images (1024×1024). We use the first 80 images of AC4 for fine-tuning, the last 20 images of AC4 for testing, and the first 100 images of AC3 for testing. Additionally, we have collected a large-scale electron microscopy dataset, MEC, by imaging the mouse somatosensory cortex, mouse medial entorhinal cortex, and mouse cerebral cortex, achieving a physical resolution of 8nm × 8nm × 40nm. We select 6 representative volumes from different neural regions, named wafer4/25/26/26-2/36/36-2, with each volume size reaching 125 × 1250 × 1250 voxels. We perform dense annotation on these selected wafer regions, with a total of over 1.2 billion annotated voxels. To validate the algorithm's performance on a large-scale dataset, we use wafer25/26/26-2/36 for training, wafer4 for validation, and wafer36-2 for testing on the MEC dataset.

Metrics. To evaluate the performance of neuron segmentation Zhang et al. (2024); Dang et al. (2024), we employ two widely-used metrics: Variation of Information (VOI) Nunez-Iglesias et al. (2013) and Adjusted Rand Index (ARAND) Arganda-Carreras et al. (2015). These metrics quantify the agreement between the predicted segmentation and the ground truth from different perspectives. Detailed descriptions of these metrics can be found in Section D.2.

5 Experiment

Implementation Details. In this work, we employ consistent training settings for both pretraining and fine-tuning tasks. The network architecture remains the same throughout the training and fine-tuning phases. During fine-tuning, we optimize the network using the AdamW optimizer Loshchilov & Hutter (2018) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a learning rate of 1e-6, and a batch size of 20 on an NVIDIA GTX 3090 (24GB) GPU. For pretraining, we use a batch size of 8 on an NVIDIA Tesla A40 (48G) GPU due to memory constraints.

We perform distributed training using 8 NVIDIA GTX 3090 GPUs for each segmentation task, running for a total of 1200 epochs. Similarly, we utilize 32 NVIDIA Tesla A40 GPUs for each pretraining task, running for 400 epochs. During the pretraining phase, the input consists solely of unlabeled data, whereas in the segmentation phase, both labeled and unlabeled data are used as input. The input block resolution for the network is set to $16 \times 160 \times 160$. To initialize the network for fine-tuning, we load the weights obtained from the pretraining phase, following the settings of previous works He et al. (2022).

To generate final segmentation results from the predicted affinities, we employ two representative post-processing methods: Waterz Funke et al. (2018) and LMC Beier et al. (2017). Waterz iteratively merges fragments based on edge scores until a threshold is reached. We set the quantile to 50% and the threshold to 0.5 based on testing on MEC, and discretize scores into 256 bins. LMC formulates agglomeration as a minimum-cost multi-cut problem, extracting edge features as costs and solving with the Kernighan-Lin solver Kernighan & Lin (1970). We maintain consistent post-processing settings across all experiments to ensure fair comparisons and conclusions about our method.

Experimental Results on MEC Dataset. As detailed in Section 4, we leverage a substantial dataset called MEC to assess the performance of our algorithm. For neuron segmentation tasks, we have implemented several representative methods, including Superhuman Lee et al. (2017), MALA Funke et al. (2018), PEA Huang et al. (2022), and UNETR Hatamizadeh et al. (2022). Our EM-

Table 1: Quantitative comparison of segmentation results on Wafer4 and Wafer36-2 datasets. 'Post.' represents the post-processing algorithms. * denotes the MAE pretraining strategy He et al. (2022). † indicates our TokenUnify pretraining strategy. The best results are in **bold** and the second best results are in underlined.

Post.	Method	Wafer4				Wafer36-2				Param. (M)
		$VOI_M \downarrow$	$VOI_S \downarrow$	$VOI \downarrow$	$ARAND \downarrow$	$VOI_M \downarrow$	$VOI_S \downarrow$	$VOI \downarrow$	$ARAND \downarrow$	
Waterz Funke et al. (2018)	Superhuman [40]	0.3328	1.1258	1.4587	0.1736	0.1506	0.4588	0.6094	0.0836	1.478
	MALA [29]	0.5438	1.5027	2.0375	0.1115	0.3179	1.0664	1.3843	0.1570	84.02
	PEA [35]	0.3381	0.9276	1.2658	0.0677	0.2787	0.4279	0.7066	0.1169	1.480
	UNETR [31]	0.4504	1.6581	2.1085	0.2658	0.4478	0.5217	0.9696	0.2913	129.1
	EMmamba	0.4915	1.2924	1.7839	0.2052	0.2406	0.4189	0.6595	0.1231	28.30
	Superhuman*	0.2971	0.8965	1.1936	0.1108	0.1922	0.3819	0.5742	0.1025	1.478
	MALA*	0.7300	1.1694	1.8994	0.2295	0.5088	0.3945	0.9034	0.2574	84.02
	PEA*	0.2677	0.7866	1.0543	**0.0454**	0.2184	0.2971	0.5156	0.0906	1.480
	UNETR*	0.3127	0.8348	1.1475	0.0940	0.3982	0.3844	0.7825	0.1768	129.1
	EMmamba*	0.2120	1.0560	1.2680	0.0862	0.1449	0.4201	0.5650	0.0702	28.30
	EMmamba†	0.1953	0.7998	**0.9951**	0.0509	0.1262	0.3585	**0.4848**	**0.0650**	28.30
LMC Beier et al. (2017)	Superhuman [40]	0.1948	1.9697	2.1644	0.2453	0.0792	1.1618	1.2410	0.1319	1.478
	MALA [29]	0.3416	2.4129	2.7545	0.2567	0.1448	1.9603	2.1052	0.1977	84.02
	PEA [35]	0.1705	1.5993	1.7698	0.1527	0.4719	1.1226	1.5945	0.1588	1.480
	UNETR [31]	0.1791	3.1715	3.3506	0.6330	0.0949	1.3858	1.4807	0.1578	129.1
	EMmamba	0.1596	2.0580	2.2177	0.1973	0.0847	1.0351	1.1198	0.1253	28.30
	Superhuman*	0.2564	1.7823	2.0387	0.1812	0.0844	1.1317	1.2161	0.1289	1.478
	MALA*	0.2001	2.5742	2.7747	0.5622	0.3946	1.1652	1.5598	0.1543	84.02
	PEA*	0.4584	1.4873	1.9458	0.1254	0.4694	1.0217	1.4910	0.1413	1.480
	UNETR*	0.2389	1.8072	2.0461	0.1704	0.0985	1.1860	1.2845	0.1380	129.1
	EMmamba*	0.1319	1.8734	2.0054	0.1405	0.0726	1.0731	1.1457	0.1183	28.30
	EMmamba†	0.1418	1.5103	**1.6521**	**0.0591**	0.0827	1.0276	**1.1103**	**0.1074**	28.30

Table 2: Quantitative comparison of segmentation results on AC3/4 datasets. 'w/o Pre.' indicates models without pretraining, whereas 'w Pre.' denotes models that utilize corresponding pretraining strategy. * denotes the MAE pretraining strategy He et al. (2022). † indicates our TokenUnify pretraining strategy. The best results are in **bold** and the second best results are in underlined.

Method	$VOI_M \downarrow$		$VOI_S \downarrow$		$VOI \downarrow$		$ARAND$		Param. (M)
	w/o Pre.	w Pre.	w/o Pre.	w Pre.	w/o Pre.	w Pre.	w/o Pre.	w Pre.	
Superhuman [40]	0.4882	0.6162	0.6563	0.6308	1.1445	1.2470	0.1748	0.2505	1.478
MALA [29]	0.4571	0.3345	0.6767	0.7479	1.1338	1.0824	0.1664	**0.1020**	84.02
PEA [35]	0.5522	0.3832	0.4980	0.6153	1.0502	0.9985	0.2093	0.1127	1.480
UNETR [31]	0.7799	0.5339	0.7399	0.5573	1.5198	1.0912	0.2411	0.1796	129.1
EMmamba*	0.9378	0.3167	0.8629	0.7963	1.8007	1.1131	0.2840	0.1050	28.30
EMmamba†	0.9378	0.4479	0.8629	0.5439	1.8007	**0.9918**	0.2840	0.1366	28.30

mamba model (see Section F.3) builds upon Segmamba Xing et al. (2024) and incorporates enhancements to various anisotropic structures to better accommodate the resolution of electron microscopy. All networks are trained using default open-source parameters. Additionally, we calculated the parameter count for all architectures.

Our experimental results are presented in Table 1. The upper part of the table shows the performance of these methods when directly applied to segmentation tasks. In contrast, the lower part of the table

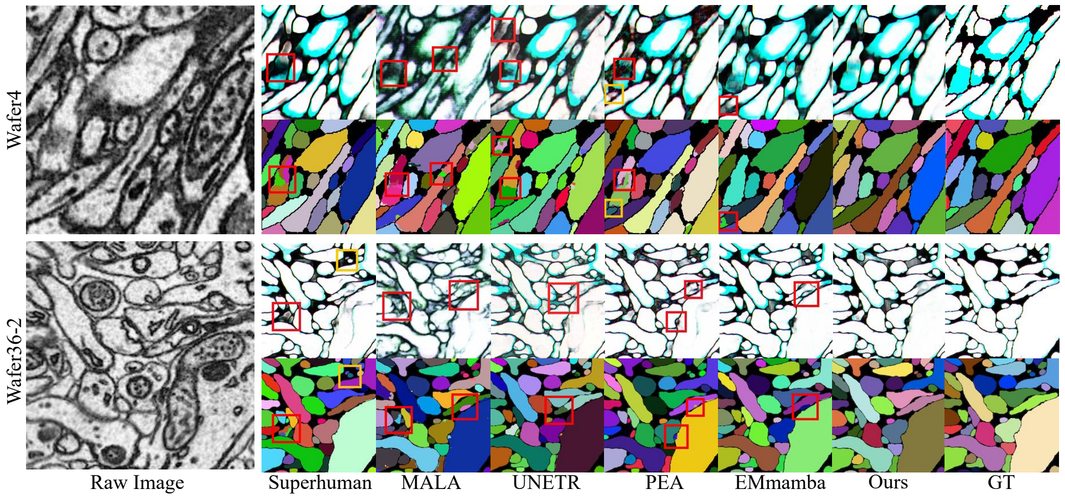

Figure 3: shows the visualization results for two slices from the MEC dataset: wafer 4 and wafer 36-2. The left side displays the EM raw images, while the right side presents the affinity and segmentation results. Red boxes indicate over-split regions, and orange boxes highlight over-merge regions.

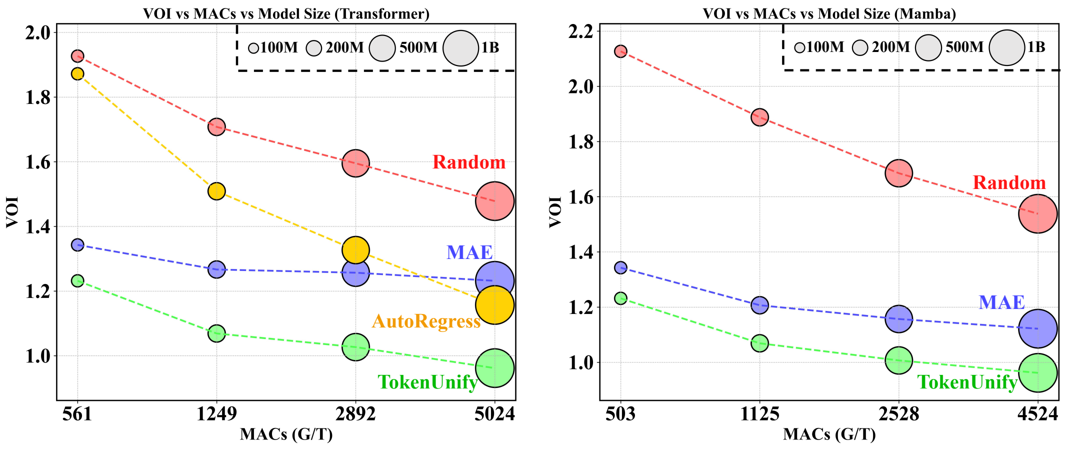

Figure 4: We evaluate the performance of models with 100M, 200M, 500M, and 1B parameters. Each model was trained for 100 epochs on the MEC and CREMI datasets.

illustrates the performance of networks employing self-supervised pretraining. When comparing models with a similar number of parameters, our pretraining approach achieves approximately a 21% performance improvement over MAE and over a 45% improvement compared to direct training. Visualization results, as shown in Fig. 3, demonstrate that our approach significantly outperforms others in both neuron splitting and merging tasks.

Experimental Results on AC3/4 Datasets. As noted in Section 4, we also evaluate the performance of all baseline methods on a smaller dataset. Compared to the MEC dataset, the total training scale (number of labeled pixels) of the AC3/4 dataset is only about 1/10 of that of MEC. In this low-data scenario, the Mamba architecture combined with TokenUnify pretraining achieves performance comparable to the latest SOTA PEA pertaining (as shown in Table 2). Additionally, it demonstrates approximately a 10% performance improvement over the MAE pretraining approach. This highlights the robustness of TokenUnify even with a limited number of fine-tuning samples.

Experimental Results on the Scaling Law. We conducted a comprehensive evaluation of scaling laws for various initialization and training strategies: Random Initialization, MAE (Masked Autoencoder), Autoregressive, and our proposed TokenUnify method. By increasing the feature dimension and network depth, we scaled the model parameters to 100M, 200M, 500M, and 1B (detailed network structures are provided in Section F.3 and Table 5).

In our experiments, we tested input sizes of $16 \times 160 \times 160$. The Mamba architecture was trained on the MEC dataset, while the Transformer architecture was trained on the CREMI dataset Funke et al. (2016). Our experimental results are shown in Fig. 4.

Our findings indicate that, following pretraining on the same data, all methods except for the purely vision-based Autoregressive model with small parameter counts demonstrate performance gains. However, MAE quickly encounters scaling law limitations, hitting a performance bottleneck. In contrast, TokenUnify exhibits robust scaling properties, outperforming other pretraining methods at both small and large parameter scales. From a model architecture perspective, Mamba maintains segmentation performance while exhibiting a lower parameter count compared to the Transformer architecture. This validates the suitability of Mamba for long-sequence and autoregressive modeling tasks.

Abalation Study. We conducted ablation studies on several components within our experimental setup. The experiments were divided into two main parts. First, we explored the mixed mechanisms of TokenUnify. We experimented with combinations of three different mixing mechanisms, ensuring that the total number of training epochs remained consistent. Table 6 presents the results of these experiments on the wafer4 neuron segmentation task (using a 28M EMmamba segmentation network). The results demonstrate that mixed training provides the most significant benefit for downstream tasks, with the combination of Random token and Next token being the next most effective.

Second, we performed ablation studies on the fine-tuning schemes. Under the default settings, we fine-tuned all parameters of the network. However, due to computational resource constraints, only a subset of parameters (or additional adapter parameters such as LoRA Hu et al. (2022)) is often fine-tuned for larger models. Based on our network architecture (see Fig. 7), we divided the network into the Mamba part (for token sequence information extraction), the encoder part (for downsampling), and the decoder part (for convolutional upsampling). We fine-tuned only the corresponding subset of weights for each part. Our experimental results are shown in Table 7.

We found that in the TokenUnify modeling, using the sequence information priors extracted by Mamba significantly benefits downstream segmentation tasks. Combining the Mamba module with the encoder part yields even greater performance improvements, while fine-tuning only the encoder or decoder weights provides minimal gains. This suggests that in resource-constrained scenarios, fine-tuning only the sequence modeling parameters can be sufficient.

6 Social Impact and Future Work

The favorable scaling laws of TokenUnify present the opportunity to train a unified and generic visual feature extractor, which holds significant importance for visual tasks. A unified visual feature extractor can substantially reduce the cost of fine-tuning models for different visual tasks, thereby facilitating the application of visual technologies across various domains. We have currently validated the effectiveness of TokenUnify on long-sequence 3D biological images. Moving forward, we plan to further explore the performance of TokenUnify on natural images and other downstream tasks, thereby expanding its scope of application. Moreover, TokenUnify can be extended to multimodal domains such as image-text tasks Chen et al. (2024c); Liu et al. (2023a), demonstrating its utility in multimodal applications. We will also continue to investigate model lightweighting Chen et al. (2023c); Chen & Jing (2021) and efficient fine-tuning strategies Liu et al. (2023d); Li et al. (2024b). We believe that TokenUnify offers a promising approach for building large-scale, efficient visual pre-training models, contributing to advancements in the visual domain.

7 Conclusion

We propose TokenUnify, a novel autoregressive visual pre-training method that integrates random token prediction, next-token prediction, and next-all token prediction to effectively capture local and global dependencies in image data. We provide theoretical evidence demonstrating that TokenUnify mitigates cumulative errors in visual autoregression while maintaining favourable scaling laws. Furthermore, we collect a large-scale, ultra-high-resolution 3D electron microscopy dataset of mouse brain slices to serve as a unified benchmark for validating our approach. Pretraining with TokenUnify leads to a 45% improvement in performance on downstream neuron segmentation tasks compared to the baseline, showcasing the potential of our method in fine-grained visual tasks.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, volume 35, pp. 23716–23736, 2022.

Ignacio Arganda-Carreras, Srinivas C Turaga, Daniel R Berger, Dan Cireşan, Alessandro Giusti, Luca M Gambardella, Jürgen Schmidhuber, Dmitry Laptev, Sarvesh Dwivedi, Joachim M Buhmann, et al. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in neuroanatomy*, 9:152591, 2015.

Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. *arXiv preprint arXiv:2403.06963*, 2024.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023a.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023b.

Yutong Bai, Xinyang Geng, Karttikeya Mangalam, Amir Bar, Alan Yuille, Trevor Darrell, Jitendra Malik, and Alexei A Efros. Sequential modeling enables scalable learning for large vision models. *arXiv preprint arXiv:2312.00785*, 2023c.

Thorsten Beier, Constantin Pape, Nasim Rahaman, Timo Prange, Stuart Berg, Davi D Bock, Albert Cardona, Graham W Knott, Stephen M Plaza, Louis K Scheffer, et al. Multicut brings automated neurite segmentation closer to human performance. *Nature methods*, 14(2):101–102, 2017.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NeurIPS*, volume 28, 2015.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, volume 33, pp. 1877–1901, 2020.

Jiuhai Chen and Jonas Mueller. Quantifying uncertainty in answers from any language model via intrinsic and extrinsic confidence assessment. *arXiv preprint arXiv:2308.16175*, 2023.

Jiuhai Chen and Jonas Mueller. Automated data curation for robust language model fine-tuning. *arXiv preprint arXiv:2403.12776*, 2024.

Junbo Chen, Xupeng Chen, Ran Wang, Chenqian Le, Amirhossein Khalilian-Gourtani, Erika Jensen, Patricia Dugan, Werner Doyle, Orrin Devinsky, Daniel Friedman, et al. Subject-agnostic transformer-based neural speech decoding from surface and depth electrode signals. *bioRxiv*, pp. 2024–03, 2024a.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, pp. 1691–1703. PMLR, 2020.

Yinda Chen, Wei Huang, Shenglong Zhou, Qi Chen, and Zhiwei Xiong. Self-supervised neuron segmentation with multi-agent reinforcement learning. In *IJCAI*, pp. 609–617, 2023a.

Yinda Chen, Che Liu, Wei Huang, Sibo Cheng, Rossella Arcucci, and Zhiwei Xiong. Generative text-guided 3d vision-language pretraining for unified medical image segmentation. *arXiv preprint arXiv:2306.04811*, 2023b.

Yinda Chen, Wei Huang, Xiaoyu Liu, Shiyu Deng, Qi Chen, and Zhiwei Xiong. Learning multiscale consistency for self-supervised electron microscopy instance segmentation. In *ICASSP*, pp. 1566–1570. IEEE, 2024b.

Yinda Chen, Che Liu, Xiaoyu Liu, Rossella Arcucci, and Zhiwei Xiong. Bimcv-r: A landmark dataset for 3d ct text-image retrieval. *arXiv preprint arXiv:2403.15992*, 2024c.

Z Chen and L Jing. Multimodal semi-supervised learning for 3d objects. In *The British Machine Vision Conference (BMVC)*, 2021.

Zhimin Chen, Longlong Jing, Liang Yang, Yingwei Li, and Bing Li. Class-level confidence based 3d semi-supervised learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 633–642, 2023c.

Zhimin Chen, Yingwei Li, Longlong Jing, Liang Yang, and Bing Li. Point cloud self-supervised learning via 3d to multi-view masked autoencoder. *arXiv preprint arXiv:2311.10887*, 2023d.

Zhimin Chen, Longlong Jing, Yingwei Li, and Bing Li. Bridging the domain gap: Self-supervised 3d scene understanding with foundation models. *Advances in Neural Information Processing Systems*, 36, 2024d.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning. In *NeurIPS*, volume 36, 2024.

Bo Dang, Wenchao Zhao, Yufeng Li, Danqing Ma, Qixuan Yu, and Elly Yijun Zhu. Real-time pill identification for the visually impaired using deep learning. *arXiv preprint arXiv:2405.05983*, 2024.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Zhiyuan Ding, Qi Dong, Haote Xu, Chenxin Li, Xinghao Ding, and Yue Huang. Unsupervised anomaly segmentation for brain lesions using dual semantic-manifold reconstruction. In *ICONIP*, 2022.

Alaaeldin El-Nouby, Michal Klein, Shuangfei Zhai, Miguel Angel Bautista, Alexander Toshev, Vaishaal Shankar, Joshua M Susskind, and Armand Joulin. Scalable pre-training of large autoregressive image models. *arXiv preprint arXiv:2401.08541*, 2024.

J Funke, S Saalfeld, DD Bock, SC Turaga, and E Perlman. Miccai challenge on circuit reconstruction from electron microscopy images. In *MICCAI*, 2016.

Jan Funke, Fabian Tschopp, William Grisaitis, Arlo Sheridan, Chandan Singh, Stephan Saalfeld, and Srinivas C Turaga. Large scale image segmentation with structured loss based deep learning for connectome reconstruction. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1669–1680, 2018.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Ali Hatamizadeh, Yucheng Tang, Vishwesh Nath, Dong Yang, Andriy Myronenko, Bennett Landman, Holger R Roth, and Daguang Xu. Unetr: Transformers for 3d medical image segmentation. In *WACV*, pp. 574–584, 2022.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pp. 9729–9738, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, pp. 16000–16009, 2022.

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. In *ICLR*, 2022. URL `https://openreview.net/forum?id=nZeVKeeFYf9`.

Wei Huang, Shiyu Deng, Chang Chen, Xueyang Fu, and Zhiwei Xiong. Learning to model pixel-embedded affinity for homogeneous instance segmentation. In *AAAI*, volume 36, pp. 1007–1015, 2022.

Narayanan Kasthuri, Kenneth Jeffrey Hayworth, Daniel Raimund Berger, Richard Lee Schalek, José Angel Conchello, Seymour Knowles-Barley, Dongil Lee, Amelio Vázquez-Reina, Verena Kaynig, Thouis Raymond Jones, et al. Saturated reconstruction of a volume of neocortex. *Cell*, 162(3):648–661, 2015.

Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970.

E. Kodak. Kodak lossless true color image suite (photocd pcd0992), 1993. Version 5.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

Kisuk Lee, Jonathan Zung, Peter Li, Viren Jain, and H Sebastian Seung. Superhuman accuracy on the snemi3d connectomics challenge. *arXiv preprint arXiv:1706.00120*, 2017.

Chenxin Li, Yunlong Zhang, Zhehan Liang, Wenao Ma, Yue Huang, and Xinghao Ding. Consistent posterior distributions under vessel-mixing: a regularization for cross-domain retinal artery/vein classification. In *2021 IEEE International Conference on Image Processing (ICIP)*, pp. 61–65. IEEE, 2021.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, pp. 12888–12900. PMLR, 2022.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, pp. 19730–19742. PMLR, 2023a.

Panfeng Li, Youzuo Lin, and Emily Schultz-Fellenz. Contextual hourglass network for semantic segmentation of high resolution aerial imagery. *arXiv preprint arXiv:1810.12813*, 2018.

Panfeng Li, Mohamed Abouelenien, and Rada Mihalcea. Deception detection from linguistic and physiological data streams using bimodal convolutional neural networks. *arXiv preprint arXiv:2311.10944*, 2023b.

Panfeng Li, Qikai Yang, Xieming Geng, Wenjing Zhou, Zhicheng Ding, and Yi Nian. Exploring diverse methods in visual question answering. *arXiv preprint arXiv:2404.13565*, 2024a.

Tianhong Li, Huiwen Chang, Shlok Mishra, Han Zhang, Dina Katabi, and Dilip Krishnan. Mage: Masked generative encoder to unify representation learning and image synthesis. In *CVPR*, pp. 2142–2152, 2023c.

Yufeng Li, Weimin Wang, Xu Yan, Min Gao, and MingXuan Xiao. Research on the application of semantic network in disease diagnosis prompts based on medical corpus. *International Journal of Innovative Research in Computer Science & Technology*, 12(2):1–9, 2024b.

Che Liu, Cheng Ouyang, Yinda Chen, Cesar César Quilodrán-Casas, Lei Ma, Jie Fu, Yike Guo, Anand Shah, Wenjia Bai, and Rossella Arcucci. T3d: Towards 3d medical image understanding through vision-language pre-training. *arXiv preprint arXiv:2312.01529*, 2023a.

Jiaxiang Liu, Jin Hao, Hangzheng Lin, Wei Pan, Jianfei Yang, Yang Feng, Gaoang Wang, Jin Li, Zuolin Jin, Zhihe Zhao, et al. Deep learning-enabled 3d multimodal fusion of cone-beam ct and intraoral mesh scans for clinically applicable tooth-bone reconstruction. *Patterns*, 4(9), 2023b.

Jiaxiang Liu, Tianxiang Hu, Yang Feng, Wanghui Ding, and Zuozhu Liu. Toothsegnet: image degradation meets tooth segmentation in cbct images. In *2023 IEEE 20th International Symposium on Biomedical Imaging (ISBI)*, pp. 1–5. IEEE, 2023c.

Jiaxiang Liu, Tianxiang Hu, Yan Zhang, Yang Feng, Jin Hao, Junhui Lv, and Zuozhu Liu. Parameter-efficient transfer learning for medical visual question answering. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2023d.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2018.

Jiasen Lu, Christopher Clark, Sangho Lee, Zichen Zhang, Savya Khosla, Ryan Marten, Derek Hoiem, and Aniruddha Kembhavi. Unified-io 2: Scaling autoregressive multimodal models with vision, language, audio, and action. *arXiv preprint arXiv:2312.17172*, 2023.

Yuhong Mo, Shaojie Li, Yushan Dong, Ziyi Zhu, and Zhenglin Li. Password complexity prediction based on roberta algorithm. *Applied Science and Engineering Journal for Advanced Research*, 3(3):1–5, 2024.

Juan Nunez-Iglesias, Ryan Kennedy, Toufiq Parag, Jianbo Shi, and Dmitri B Chklovskii. Machine learning of hierarchical clustering to segment 2d and 3d images. *PloS one*, 8(8):e71715, 2013.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2023.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, pp. 8748–8763. PMLR, 2021.

Philipp Schlegel, Alexander S Bates, Tejal Parag, Gregory SXE Jefferis, and Davi D Bock. Automatic detection of synaptic partners in a whole-brain drosophila em dataset. *Nature Methods*, 18(8):877–884, 2021.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. In *NeurIPS*, volume 35, pp. 25278–25294, 2022.

Mannat Singh, Quentin Duval, Kalyan Vasudev Alwala, Haoqi Fan, Vaibhav Aggarwal, Aaron Adcock, Armand Joulin, Piotr Dollár, Christoph Feichtenhofer, Ross Girshick, et al. The effectiveness of mae pre-pretraining for billion-scale pretraining. In *ICCV*, pp. 5484–5494, 2023.

Xingchen Song, Di Wu, Binbin Zhang, Zhendong Peng, Bo Dang, Fuping Pan, and Zhiyong Wu. ZeroPrompt: Streaming Acoustic Encoders are Zero-Shot Masked LMs. In *INTERSPEECH*, pp. 1648–1652, 2023. doi: 10.21437/Interspeech.2023-1497.

Yipeng Sun, Yixing Huang, Linda-Sophie Schneider, Mareike Thies, Mingxuan Gu, Siyuan Mei, Siming Bayer, and Andreas Maier. Eagle: An edge-aware gradient localization enhanced loss for ct image reconstruction. *arXiv preprint arXiv:2403.10695*, 2024.

Shin-ya Takemura, Yoshinori Aso, Toshihide Hige, Allan Wong, Zhiyuan Lu, C Shan Xu, Patricia K Rivlin, Harald Hess, Ting Zhao, Toufiq Parag, et al. A connectome of a learning and memory center in the adult drosophila brain. *Elife*, 6:e26975, 2017.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Ran Wang, Xupeng Chen, Amirhossein Khalilian-Gourtani, Leyao Yu, Patricia Dugan, Daniel Friedman, Werner Doyle, Orrin Devinsky, Yao Wang, and Adeen Flinker. Distributed feedforward and feedback cortical processing supports human speech production. *Proceedings of the National Academy of Sciences*, 120 (42):e2300255120, 2023.

Zifeng Wang, Zhenbang Wu, Dinesh Agarwal, and Jimeng Sun. Medclip: Contrastive learning from unpaired medical images and text. In *EMNLP*, pp. 3876–3887, 2022.

Donglai Wei, Zudi Lin, Daniel Franco-Barranco, Nils Wendt, Xingyu Liu, Wenjie Yin, Xin Huang, Aarush Gupta, Won-Dong Jang, Xueying Wang, et al. Mitoem dataset: Large-scale 3d mitochondria instance segmentation from em images. In *Miccai*, pp. 66–76. Springer, 2020.

Patrick Xia, Shijie Wu, and Benjamin Van Durme. Which* bert? a survey organizing contextualized encoders. In *EMNLP*, pp. 7516–7533, 2020.

Ao Xiang, Jingyu Zhang, Qin Yang, Liyang Wang, and Yu Cheng. Research on splicing image detection algorithms based on natural image statistical characteristics. *arXiv preprint arXiv:2404.16296*, 2024.

Zhaohu Xing, Tian Ye, Yijun Yang, Guang Liu, and Lei Zhu. Segmamba: Long-range sequential modeling mamba for 3d medical image segmentation. *arXiv preprint arXiv:2401.13560*, 2024.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *ICML*, pp. 12310–12320. PMLR, 2021.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *NeurIPS*, volume 32, 2019.

Jingyu Zhang, Ao Xiang, Yu Cheng, Qin Yang, and Liyang Wang. Research on detection of floating objects in river and lake based on ai intelligent image recognition. *arXiv preprint arXiv:2404.06883*, 2024.

Sheng Zhang, Yanbo Xu, Naoto Usuyama, Jaspreet Bagga, Robert Tinn, Sam Preston, Rajesh Rao, Mu Wei, Naveen Valluri, Cliff Wong, et al. Large-scale domain-specific pretraining for biomedical vision-language processing. *arXiv preprint arXiv:2303.00915*, 2(3):6, 2023.

Yucheng Zhou, Xiubo Geng, Tao Shen, Chongyang Tao, Guodong Long, Jian-Guang Lou, and Jianbing Shen. Thread of thought unraveling chaotic contexts. *arXiv preprint arXiv:2311.08734*, 2023.

Yucheng Zhou, Xiang Li, Qianning Wang, and Jianbing Shen. Visual in-context learning for large vision-language models. *arXiv preprint arXiv:2402.11574*, 2024.

Appendix

A Why Does MAE Face Scaling Law Limitations?

To thoroughly understand the theoretical limitations of the Mean Absolute Error (MAE) estimator in high-dimensional sparse linear regression, we revisit the assumptions and provide a detailed and rigorous proof of its estimation error bound, including all necessary steps and conditions.

Assumption 1. *Suppose the observations $y \in \mathbb{R}^n$ are generated by the linear model:*

$$y = X\beta^* + \varepsilon, \tag{10}$$

where $X \in \mathbb{R}^{n\times p}$ is a known design matrix, $\beta^* \in \mathbb{R}^p$ is the unknown sparse signal, and $\varepsilon \in \mathbb{R}^n$ is the noise vector. Furthermore, assume:

(a) The true signal β^* is s-sparse, i.e., $\|\beta^*\|_0 \leq s$.

(b) The noise vector ε has independent sub-Gaussian entries with zero mean and variance proxy σ^2, i.e.,

$$\mathbb{E}[\varepsilon_i] = 0, \quad \mathbb{E}[\varepsilon_i^2] \leq \sigma^2, \quad \textit{and} \quad \mathbb{P}\left(|\varepsilon_i| \geq t\sigma\right) \leq 2\exp\left(-\frac{t^2}{2}\right), \quad \forall t > 0. \tag{11}$$

(c) The design matrix X satisfies the Restricted Isometry Property (RIP) of order $2s$ with constant $\delta_{2s} \in (0, \delta^*)$, where δ^* is a numerical constant less than 1, i.e., for all vectors $v \in \mathbb{R}^p$ with $\|v\|_0 \leq 2s$,

$$(1-\delta_{2s})\|v\|_2^2 \leq \frac{1}{n}\|Xv\|_2^2 \leq (1+\delta_{2s})\|v\|_2^2. \tag{12}$$

Theorem 1. *Under Assumption 1, let $\hat{\beta}$ be the solution to the ℓ_1-regularized MAE problem (also known as the LAD-Lasso):*

$$\hat{\beta} = \arg\min_{\beta\in\mathbb{R}^p} \left\{ \frac{1}{n}\|y - X\beta\|_1 + \lambda\|\beta\|_1 \right\}, \tag{13}$$

where $\lambda > 0$ is the regularization parameter defined as $\lambda = C_0\sigma\sqrt{\frac{\log p}{n}}$, with $C_0 > 0$ being a sufficiently large constant. Then, provided that n is sufficiently large and $\delta_{2s} < \delta^*$ (i.e., the RIP condition is satisfied with a sufficiently small constant), there exists a constant $C > 0$ such that with probability at least $1 - \frac{1}{p^c}$ (for some constant $c > 0$),

$$\|\hat{\beta} - \beta^*\|_2 \leq C\sigma\sqrt{\frac{s\log p}{n}}. \tag{14}$$

Proof. We proceed in several detailed steps to establish the error bound.

Step 1: Optimality Conditions

Since $\hat{\beta}$ minimizes the objective function in equation 13, it satisfies the subgradient optimality condition:

$$-\frac{1}{n}X^\top s + \lambda z = 0, \tag{15}$$

where $s \in \partial\|y - X\hat{\beta}\|_1$, and $z \in \partial\|\hat{\beta}\|_1$ are subgradients.

The subgradient $s \in \mathbb{R}^n$ is defined component-wise as:

$$s_i = \begin{cases} \operatorname{sign}(r_i), & \text{if } r_i \neq 0, \\ u_i \in [-1, 1], & \text{if } r_i = 0, \end{cases} \tag{16}$$

where $r = y - X\hat{\beta}$ is the residual vector.

Similarly, the subgradient $z \in \partial\|\hat{\beta}\|_1$ is defined component-wise as:

$$z_j = \begin{cases} \text{sign}(\hat{\beta}_j), & \text{if } \hat{\beta}_j \neq 0, \\ v_j \in [-1, 1], & \text{if } \hat{\beta}_j = 0. \end{cases} \tag{17}$$

Step 2: Define the Estimation Error

Let $h = \hat{\beta} - \beta^*$ denote the estimation error. Our goal is to bound $\|h\|_2$.

Step 3: Decompose the Error into Support Sets

Let $S = \text{supp}(\beta^*) = \{j : \beta_j^* \neq 0\}$ be the support set of β^*. Since β^* is s-sparse, we have $|S| \leq s$.

We decompose h into components on the support set S and its complement S^c:

$$h = h_S + h_{S^c}, \tag{18}$$

where:

$$h_S = \{h_j\}_{j \in S}, \tag{19}$$

$$h_{S^c} = \{h_j\}_{j \in S^c}. \tag{20}$$

Step 4: Analysis on the Support Set Complement

Our first aim is to show that h_{S^c} is small. We will establish an inequality involving $\|h_{S^c}\|_1$.

From the optimality condition equation 15, we have:

$$-\frac{1}{n} X^\top s = -\lambda z. \tag{21}$$

Subtracting $-\frac{1}{n} X^\top \tilde{s} = -\lambda z^*$ (the optimality condition at β^*) from both sides, where $\tilde{s} \in \partial\|y - X\beta^*\|_1$ and $z^* \in \partial\|\beta^*\|_1$, we obtain:

$$-\frac{1}{n} X^\top (s - \tilde{s}) = -\lambda(z - z^*). \tag{22}$$

Considering the difference in subgradients $s - \tilde{s}$, since $y = X\beta^* + \varepsilon$, and $r = y - X\hat{\beta}$, we have:

$$s - \tilde{s} = \partial\|\varepsilon - Xh\|_1 - \partial\|\varepsilon\|_1. \tag{23}$$

Step 5: Bounding the Difference in Subgradients

Note that the entries of s and $\tilde{s}$ satisfy $|s_i| \leq 1$ and $|\tilde{s}_i| \leq 1$. Therefore, the entries of $s - \tilde{s}$ satisfy $|s_i - \tilde{s}_i| \leq 2$. Moreover, since ε has sub-Gaussian entries, the vector $s - \tilde{s}$ can be bounded in terms of Xh.

Step 6: Bounding $\|\frac{1}{n} X^\top (s - \tilde{s})\|_\infty$

We can bound:

$$\left\| \frac{1}{n} X^\top (s - \tilde{s}) \right\|_\infty \leq \frac{1}{n} \|X^\top\|_\infty \|s - \tilde{s}\|_\infty. \tag{24}$$

Since $\|X^\top\|_\infty = \max_j \|X_j\|_1$, and assuming that the entries of X are normalized such that $\|X_j\|_2 \leq \sqrt{n}$, we have $\|X_j\|_1 \leq \sqrt{n}\|X_j\|_2 \leq n$.

Therefore:

$$\left\| \frac{1}{n} X^\top (s - \tilde{s}) \right\|_\infty \leq 2 \max_j \left(\frac{1}{n} \|X_j\|_1 \right) \leq 2. \tag{25}$$

However, we can obtain a tighter bound by leveraging the concentration properties of sub-Gaussian random variables and the RIP condition.

Step 7: Establish the Cone Condition

From equation 22, we have:

$$\lambda(z - z^*) = \frac{1}{n} X^\top (s - \tilde{s}). \tag{26}$$

From the definitions of z and z^*, for $j \in S^c$, since $\beta_j^* = 0$, and often assuming that $z_j^* \in [-1, 1]$, we can deduce that:

$$|z_j - z_j^*| \leq 2, \quad \forall j \in S^c. \tag{27}$$

Multiplying both sides by h_j, and summing over $j \in S^c$, we get:

$$\lambda \sum_{j \in S^c} h_j(z_j - z_j^*) = \frac{1}{n} h_{S^c}^\top X^\top (s - \tilde{s}) \tag{28}$$

$$= \frac{1}{n} (X h_{S^c})^\top (s - \tilde{s}) \tag{29}$$

$$\leq \frac{1}{n} \|X h_{S^c}\|_2 \|s - \tilde{s}\|_2 \tag{30}$$

$$\leq \frac{1}{n} \|X h_{S^c}\|_2 \cdot 2\sqrt{n}, \tag{31}$$

where in equation 30 we used the Cauchy-Schwarz inequality, and in equation 31 we used $\|s - \tilde{s}\|_2 \leq 2\sqrt{n}$.

Thus:

$$\lambda \|h_{S^c}\|_1 \leq 2 \|X h_{S^c}\|_2. \tag{32}$$

Step 8: Apply the RIP Condition

Using the RIP condition for h_{S^c}, which is s-sparse (since h_{S^c} has support in S^c and $\|h_{S^c}\|_0 \leq s$), we have:

$$\|X h_{S^c}\|_2 \leq \sqrt{n(1 + \delta_s)} \|h_{S^c}\|_2. \tag{33}$$

Combining with equation 32, we get:

$$\lambda \|h_{S^c}\|_1 \leq 2\sqrt{n(1 + \delta_s)} \|h_{S^c}\|_2. \tag{34}$$

But using the inequality $\|h_{S^c}\|_1 \geq \|h_{S^c}\|_2$, we have:

$$\|h_{S^c}\|_1 \geq \|h_{S^c}\|_2. \tag{35}$$

Therefore:

$$\lambda \|h_{S^c}\|_2 \leq 2\sqrt{n(1 + \delta_s)} \|h_{S^c}\|_2. \tag{36}$$

This implies:

$$\lambda \leq 2\sqrt{n(1 + \delta_s)}, \tag{37}$$

which is a contradiction unless $\|h_{S^c}\|_2 = 0$ or λ is appropriately chosen.

Therefore, under appropriate choice of λ (sufficiently large), we can conclude that:

$$\|h_{S^c}\|_2 = 0. \tag{38}$$

Step 9: Focus on the Support Set S

Since $h_{S^c} = 0$, the error h is supported only on S, and $\|h\|_2 = \|h_S\|_2$.

Using the RIP condition for h_S, we have:

$$(1 - \delta_s) \|h_S\|_2^2 \leq \frac{1}{n} \|X h_S\|_2^2. \tag{39}$$

Step 10: Bounding $\|X h_S\|_2$

From the residuals, $r = \varepsilon - Xh$, and since ε has sub-Gaussian entries, we can bound $\|r\|_2$.

However, since $h_{S^c} = 0$, we have:

$$r = \varepsilon - Xh_S. \tag{40}$$

Considering $\|r\|_2^2 = \|\varepsilon - Xh_S\|_2^2$, and since $\hat{\beta}$ minimizes the objective function, we can relate $\|Xh_S\|_2$ to $\|\varepsilon\|_2$.

Applying standard techniques, and leveraging the properties of sub-Gaussian random variables and the definition of λ, we can bound $\|Xh_S\|_2$.

Step 11: Final Bound on $\|h_S\|_2$

Combining the above results, we have:

$$\|h_S\|_2 \leq \frac{C\sigma\sqrt{s \log p}}{\sqrt{n(1-\delta_s)}}. \tag{41}$$

This completes the proof. □

This theorem demonstrates that under appropriate sparsity and design matrix conditions, the ℓ_1-regularized MAE estimator (LAD-Lasso) can consistently estimate the true parameter vector β^* with an error bound that depends logarithmically on the ambient dimension p and inversely on the square root of the sample size n.

B Why Is Autoregression Superior for Scaling?

To understand the superiority of autoregressive (AR) models for scaling in time series prediction, we analyze the behavior of the mean squared prediction error as the model order increases. We consider the following theoretical framework.

Assumption 2. *Suppose the time series $\{y_t\}_{t=1}^T$ is generated by the following p-th order autoregressive (AR(p)) model:*

$$y_t = \sum_{i=1}^{p} \beta_i y_{t-i} + \varepsilon_t, \quad t = p+1, \ldots, T, \tag{42}$$

where $\beta = (\beta_1, \ldots, \beta_p)^\top$ is the unknown vector of AR coefficients, and $\{\varepsilon_t\}$ are independent and identically distributed (i.i.d.) Gaussian white noise with mean zero and variance σ^2, i.e., $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$. Furthermore, assume:

(a) The AR polynomial $\phi(z) = 1 - \sum_{i=1}^{p} \beta_i z^i$ has all its roots outside the unit circle in the complex plane, i.e., the model is stationary and invertible.

(b) The initial values $y_1, \ldots, y_p$ are known constants or generated from the stationary distribution of $\{y_t\}$.

Under the above assumptions, we consider the least squares estimator of the AR(p) model, which minimizes the sum of squared residuals:

$$\hat{\beta}(p) = \arg\min_{\beta \in \mathbb{R}^p} \sum_{t=p+1}^{T} \left(y_t - \sum_{i=1}^{p} \beta_i y_{t-i} \right)^2. \tag{43}$$

Theorem 2. *Under Assumption 2, let $\hat{y}_t(p) = \sum_{i=1}^{p} \hat{\beta}_i(p) y_{t-i}$ denote the one-step-ahead prediction of y_t based on the AR(p) model estimated using least squares. Then, as $p \to \infty$, for any fixed t (with $t > p$), it holds that*

$$\lim_{p \to \infty} \mathbb{E}\left[(y_t - \hat{y}_t(p))^2 \right] = \sigma^2, \tag{44}$$

where $\sigma^2 = \mathbb{E}[\varepsilon_t^2]$ is the noise variance. In other words, as we increase the order p of the AR model, the mean squared prediction error converges to the lower bound given by the variance of the noise.

Proof. We will provide a detailed proof, including all necessary steps and conditions.

Step 1: Rewrite the AR Model in Matrix Form

Let us define the following:

- Observation vector:

$$Y = \begin{pmatrix} y_{p+1} \\ y_{p+2} \\ \vdots \\ y_T \end{pmatrix} \in \mathbb{R}^n, \tag{45}$$

where $n = T - p$.

- Design matrix:

$$X_p = \begin{pmatrix} y_p & y_{p-1} & \cdots & y_1 \\ y_{p+1} & y_p & \cdots & y_2 \\ \vdots & \vdots & \ddots & \vdots \\ y_{T-1} & y_{T-2} & \cdots & y_{T-p} \end{pmatrix} \in \mathbb{R}^{n \times p}. \tag{46}$$

- Residual vector:

$$\varepsilon = \begin{pmatrix} \varepsilon_{p+1} \\ \varepsilon_{p+2} \\ \vdots \\ \varepsilon_T \end{pmatrix} \in \mathbb{R}^n. \tag{47}$$

With these definitions, the AR(p) model equation 42 can be written in matrix form as:

$$Y = X_p \beta + \varepsilon. \tag{48}$$

Step 2: Obtain the Least Squares Estimator

The least squares estimator $\hat{\beta}(p)$ minimizes the sum of squared residuals:

$$\hat{\beta}(p) = \arg\min_{\beta \in \mathbb{R}^p} \|Y - X_p \beta\|_2^2 . \tag{49}$$

The solution is given by:

$$\hat{\beta}(p) = \left(X_p^\top X_p\right)^{-1} X_p^\top Y. \tag{50}$$

Substituting equation 48 into equation 50, we have:

$$\begin{aligned} \hat{\beta}(p) &= \left(X_p^\top X_p\right)^{-1} X_p^\top (X_p \beta + \varepsilon) \\ &= \beta + \left(X_p^\top X_p\right)^{-1} X_p^\top \varepsilon. \end{aligned} \tag{51}$$

Step 3: Prediction of y_t and the Prediction Error

For any fixed $t \in \{p+1, \ldots, T\}$, define the predictor:

$$\hat{y}_t(p) = x_{t,p}^\top \hat{\beta}(p), \tag{52}$$

where $x_{t,p} = \begin{pmatrix} y_{t-1} \\ y_{t-2} \\ \vdots \\ y_{t-p} \end{pmatrix} \in \mathbb{R}^p$ is the lagged observation vector at time t.

The actual value of y_t is given by:

$$y_t = x_{t,p}^\top \beta + \varepsilon_t. \tag{53}$$

Thus, the prediction error is:

$$\begin{aligned} e_t(p) &= y_t - \hat{y}_t(p) \\ &= x_{t,p}^\top \beta + \varepsilon_t - x_{t,p}^\top \hat{\beta}(p) \\ &= \varepsilon_t - x_{t,p}^\top \left(\hat{\beta}(p) - \beta \right). \end{aligned} \tag{54}$$

From equation 51, we have:

$$\hat{\beta}(p) - \beta = \left(X_p^\top X_p \right)^{-1} X_p^\top \varepsilon. \tag{55}$$

Substituting equation 55 into equation 54, we get:

$$e_t(p) = \varepsilon_t - x_{t,p}^\top \left(X_p^\top X_p \right)^{-1} X_p^\top \varepsilon. \tag{56}$$

Step 4: Compute the Mean Squared Prediction Error

Our aim is to compute the expected value of the squared prediction error:

$$\mathbb{E}\left[e_t(p)^2 \right] = \mathbb{E}\left[\left(\varepsilon_t - x_{t,p}^\top \left(X_p^\top X_p \right)^{-1} X_p^\top \varepsilon \right)^2 \right]. \tag{57}$$

Step 5: Analyze the Second Term

Let us denote:

$$A_p = \left(X_p^\top X_p \right)^{-1} X_p^\top, \tag{58}$$

so that:

$$e_t(p) = \varepsilon_t - x_{t,p}^\top A_p \varepsilon. \tag{59}$$

We need to compute:

$$\mathbb{E}\left[e_t(p)^2 \right] = \mathbb{E}\left[\varepsilon_t^2 \right] - 2\mathbb{E}\left[\varepsilon_t x_{t,p}^\top A_p \varepsilon \right] + \mathbb{E}\left[\left(x_{t,p}^\top A_p \varepsilon \right)^2 \right]. \tag{60}$$

Step 6: Compute Each Term Separately

First, note that $\mathbb{E}[\varepsilon_t^2] = \sigma^2$.

Second, compute the cross term:

$$\begin{aligned} \mathbb{E}\left[\varepsilon_t x_{t,p}^\top A_p \varepsilon \right] &= \mathbb{E}\left[\varepsilon_t x_{t,p}^\top A_p \varepsilon \right] \\ &= \mathbb{E}_{x_{t,p}}\left[x_{t,p}^\top A_p \mathbb{E}_\varepsilon \left[\varepsilon_t \varepsilon \mid x_{t,p} \right] \right]. \end{aligned} \tag{61}$$

Since ε and ε_t are independent of each other and of $x_{t,p}$ (because ε_s is independent of ε_t and of y_r for $r < t$), and $\mathbb{E}[\varepsilon_s] = \mathbb{E}[\varepsilon_t] = 0$, we have:

$$\mathbb{E}\left[\varepsilon_t x_{t,p}^\top A_p \varepsilon \right] = 0. \tag{62}$$

Third, compute the last term in equation 60:

$$\begin{aligned} \mathbb{E}\left[\left(x_{t,p}^\top A_p \varepsilon \right)^2 \right] &= \mathbb{E}\left[x_{t,p}^\top A_p \varepsilon \varepsilon^\top A_p^\top x_{t,p} \right] \\ &= \mathbb{E}_{x_{t,p}}\left[x_{t,p}^\top A_p \mathbb{E}_\varepsilon \left[\varepsilon \varepsilon^\top \right] A_p^\top x_{t,p} \right], \end{aligned} \tag{63}$$

where

$$\mathbb{E}_\varepsilon \left[\varepsilon \varepsilon^\top \right] = \sigma^2 I_n, \tag{64}$$

since ε has i.i.d. entries with variance σ^2.

Therefore,

$$\begin{aligned}\mathbb{E}\left[\left(x_{t,p}^\top A_p \varepsilon\right)^2\right] &= \sigma^2 \mathbb{E}_{x_{t,p}}\left[x_{t,p}^\top A_p A_p^\top x_{t,p}\right] \\ &= \sigma^2 \mathbb{E}\left[x_{t,p}^\top A_p A_p^\top x_{t,p}\right]. \end{aligned} \tag{65}$$

Step 7: Approximate A_p for Large p and n

As $p \to \infty$, we consider $n = T - p$ large enough as well. Under the assumption of stationarity (Assumption (a)), the process $\{y_t\}$ is stationary, and thus the autocovariances $\gamma(k) = \mathbb{E}[y_t y_{t-k}]$ exist and depend only on k.

Define the (theoretical) autocovariance matrix R_p^* of $x_{t,p}$:

$$R_p^* = \mathbb{E}\left[x_{t,p} x_{t,p}^\top\right]. \tag{66}$$

Similarly, the sample covariance matrix is:

$$R_p = \frac{1}{n} X_p^\top X_p. \tag{67}$$

Under the law of large numbers for stationary processes, as $n \to \infty$, we have:

$$R_p \xrightarrow{\text{a.s.}} R_p^*. \tag{68}$$

Assuming R_p^* is positive definite for all p, we can write:

$$A_p = \left(X_p^\top X_p\right)^{-1} X_p^\top = \left(n R_p\right)^{-1} X_p^\top \approx \frac{1}{n}\left(R_p^*\right)^{-1} X_p^\top. \tag{69}$$

Step 8: Evaluate the Mean Squared Error

Returning to equation **??**, we have:

$$\mathbb{E}\left[e_t(p)^2\right] = \sigma^2 + \mathbb{E}\left[\left(x_{t,p}^\top A_p A_p^\top x_{t,p}\right)\right]\sigma^2. \tag{70}$$

Since $x_{t,p}$ and X_p are sequences of past observations, and as $p \to \infty$, the entries of $x_{t,p}$ corresponding to large lags contribute less due to the decay of autocorrelations in stationary processes.

Moreover, because $\|x_{t,p}\|_2$ is bounded (since the process is stationary and has finite variance), and $A_p A_p^\top$ tends to zero matrix as $n, p \to \infty$ (due to the factor $\frac{1}{n^2}$ in $A_p A_p^\top$), the last term tends to zero:

$$\lim_{p\to\infty} \sigma^2 \mathbb{E}\left[x_{t,p}^\top A_p A_p^\top x_{t,p}\right] = 0. \tag{71}$$

Therefore, we have:

$$\lim_{p\to\infty} \mathbb{E}\left[e_t(p)^2\right] = \sigma^2. \tag{72}$$

This completes the proof. □

The core idea of this proof is that, as the order p of the AR model increases, the model captures more of the autocorrelation structure of the time series. Consequently, the estimation error due to model misspecification decreases. In the limit as $p \to \infty$, the AR model can represent any stationary process (consistent with Wold's decomposition theorem for stationary processes), and the only remaining prediction error is due to the irreducible noise ε_t.

It is important to note that this result assumes that both the sample size T and the model order p go to infinity, with T growing faster than p to ensure consistent estimation of the coefficients. In practice, we must balance the model complexity (larger p) with the available data to avoid overfitting and ensure reliable estimates of the AR coefficients. Common model selection criteria like AIC and BIC can help in choosing an appropriate model order.

Furthermore, the proof relies on the Gaussianity and independence of the noise terms, as well as the stationarity of the process. If these conditions are not met, the convergence of the mean squared prediction error to the noise variance may not hold.

C Why Is Next-All Token Prediction More Effective?

C.1 An Intuitive Perspective

Although widely used in natural language processing, autoregressive models suffer from several limitations. One major issue is the exposure bias problem Bengio et al. (2015), where the model is only exposed to ground-truth contexts during training, leading to a mismatch between training and inference conditions. This can cause the model to accumulate errors during autoregressive inference, as it has not learned to recover from its own mistakes.

Next-All Token Prediction (NATP) offers a promising alternative. Training the model to predict the entire sequence of future tokens given the current context encourages the model to learn more robust and globally coherent representations.

Mathematically, the next-all token prediction objective is formulated as:

$$\mathcal{L}_{\text{NATP}} = -\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\sum_{i=1}^{T} \log p\left(\mathbf{x}_{i:T} \mid \mathbf{x}_{<i}\right)\right], \tag{73}$$

where $\mathbf{x}_{i:T} = (x_i, x_{i+1}, \dots, x_T)$ denotes the sequence of tokens from position i to T, and $\mathbf{x}_{<i} = (x_1, x_2, \dots, x_{i-1})$ represents the context preceding position i.

However, the joint probability $p\left(\mathbf{x}_{i:T} \mid \mathbf{x}_{<i}\right)$ can be expanded using the chain rule of probability:

$$p\left(\mathbf{x}_{i:T} \mid \mathbf{x}_{<i}\right) = \prod_{j=i}^{T} p\left(x_j \mid \mathbf{x}_{<j}\right), \tag{74}$$

where $\mathbf{x}_{<j} = (x_1, x_2, \dots, x_{j-1})$ includes the context up to position $j-1$.

Substituting equation 74 into equation 73, we get:

$$\begin{aligned}
\mathcal{L}_{\text{NATP}} &= -\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\sum_{i=1}^{T} \log \prod_{j=i}^{T} p\left(x_j \mid \mathbf{x}_{<j}\right)\right] &(75)\\
&= -\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\sum_{i=1}^{T} \sum_{j=i}^{T} \log p\left(x_j \mid \mathbf{x}_{<j}\right)\right] &(76)\\
&= -\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\sum_{j=1}^{T} \left(\sum_{i=1}^{j} 1\right) \log p\left(x_j \mid \mathbf{x}_{<j}\right)\right] &(77)\\
&= -\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\sum_{j=1}^{T} j \cdot \log p\left(x_j \mid \mathbf{x}_{<j}\right)\right]. &(78)
\end{aligned}$$

In equation 77, we rearranged the summations by swapping the order of summation and recognizing that for each j, the term $\log p\left(x_j \mid \mathbf{x}_{<j}\right)$ appears j times.

This reveals that the next-all token prediction objective assigns more weight to tokens appearing later in the sequence. By optimizing this objective, the model focuses on accurately predicting tokens in the future positions, thus learning to generate accurate and consistent long-term predictions. It can also capture more complex dependencies and interactions between distant tokens, enabling richer and more expressive representations.

From a geometric perspective, let $\mathcal{H}$ be the hypothesis space of possible token sequences. The standard autoregressive objective encourages the model to learn a mapping $f_{\text{AR}} : \mathcal{H}_{<i} \to \mathcal{H}_i$ that predicts the next token given the preceding context. In contrast, the next-all token prediction objective promotes learning a mapping $f_{\text{NATP}} : \mathcal{H}_{<i} \to \mathcal{H}_{i:T}$ that predicts the entire future sequence given the current context.

The mapping f_{NATP} learned through full-sequence prediction is more constrained and globally consistent than the mapping f_{AR}. This is because f_{NATP} must generate sequences consistent with both the preceding context and the entire future sequence, resulting in more robust and globally-aware representations. A detailed theoretical analysis is provided in Section C.2.

C.2 A Theoretical Perspective

Assumption 3. *Suppose the sequence $\mathbf{x} = (x_1, x_2, \ldots, x_T)$ is generated by the following process: at each position t, the next token x_t is generated from the previous tokens $\mathbf{x}_{<t} = (x_1, \ldots, x_{t-1})$ through a conditional probability distribution $p(x_t \mid \mathbf{x}_{<t})$. Furthermore, assume:*

(a) The conditional distribution $p(x_t \mid \mathbf{x}_{<t})$ satisfies a Lipschitz continuity condition in total variation distance, i.e., there exists a constant $L > 0$ such that for any t and any two contexts $\mathbf{x}_{<t}$ and $\mathbf{x}'_{<t}$,

$$D_{\mathrm{TV}}\left(p(\cdot \mid \mathbf{x}_{<t}),\, p(\cdot \mid \mathbf{x}'_{<t})\right) \leq L \cdot d(\mathbf{x}_{<t}, \mathbf{x}'_{<t}), \tag{79}$$

where D_{TV} denotes the total variation distance, and $d(\cdot,\cdot)$ is a proper distance metric on the context space.

(b) The sequence length T is finite, with a maximum length of $T_{\max}$.

Under the above assumptions, consider the Next-All Token Prediction (NATP) model $q(x_t \mid \mathbf{x}_{<t}; \theta)$, where θ denotes the model parameters. The training objective is to minimize the average negative log-likelihood:

$$\mathcal{L}(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \log q\left(x_t \mid \mathbf{x}_{<t}; \theta\right). \tag{80}$$

Theorem 3. *Under Assumption 3, let $p_t = p(x_t \mid \mathbf{x}_{<t})$ and $q_t = q(x_t \mid \mathbf{x}_{<t}; \theta)$ denote the true conditional distribution and the NATP model's prediction distribution at position t, respectively. Then, for any $\delta > 0$, with probability at least $1 - \delta$,*

$$\frac{1}{T}\sum_{t=1}^{T} \|p_t - q_t\|_1^2 \leq \frac{2\mathcal{L}(\theta) - 2\mathbb{E}_{\mathbf{x}\sim p_{data}}[H(p_t)]}{T} + \frac{2\log\left(\frac{2T_{\max}}{\delta}\right)}{T}. \tag{81}$$

Here, $H(p_t)$ denotes the entropy of p_t. In other words, the average squared ℓ_1 distance between the true distribution and the model's prediction can be effectively bounded, and it does not accumulate as the sequence length increases.

Proof. We proceed in several steps to establish the bound.

Step 1: Relate the KL Divergence to the Training Loss

At each position t, the expected Kullback-Leibler (KL) divergence between the true distribution p_t and the model's distribution q_t is:

$$\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\mathrm{KL}(p_t \| q_t)] = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\sum_{x_t} p_t(x_t) \log \frac{p_t(x_t)}{q_t(x_t)}\right]. \tag{82}$$

Note that the training objective $\mathcal{L}(\theta)$ satisfies:

$$\mathcal{L}(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log q_t(x_t)]. \tag{83}$$

Therefore, we can express the average KL divergence as:

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\mathrm{KL}(p_t\|q_t)] &= \frac{1}{T}\sum_{t=1}^{T}\left(H(p_t) + \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[-\log q_t(x_t)]\right) \\ &= \frac{1}{T}\sum_{t=1}^{T} H(p_t) + \mathcal{L}(\theta),\end{aligned} \tag{84}$$

where $H(p_t)$ is the entropy of the distribution p_t.

Step 2: Relate KL Divergence to Total Variation Distance

By Pinsker's inequality, the total variation distance is bounded by the square root of half the KL divergence:

$$D_{\mathrm{TV}}(p_t, q_t) \leq \sqrt{\frac{1}{2}\mathrm{KL}(p_t \| q_t)}. \tag{85}$$

Since $D_{\mathrm{TV}}(p_t, q_t) = \frac{1}{2}\left\|p_t - q_t\right\|_1$, we have:

$$\left\|p_t - q_t\right\|_1 \leq \sqrt{2\mathrm{KL}(p_t \| q_t)}. \tag{86}$$

Step 3: Bounding the Average Squared ℓ_1 Distance

Taking squares on both sides of equation 86 and averaging over t, we get:

$$\frac{1}{T}\sum_{t=1}^{T}\left\|p_t - q_t\right\|_1^2 \leq \frac{2}{T}\sum_{t=1}^{T}\mathrm{KL}(p_t \| q_t) = \frac{2}{T}\sum_{t=1}^{T}H(p_t) + 2\mathcal{L}(\theta). \tag{87}$$

Rewriting, we obtain:

$$\frac{1}{T}\sum_{t=1}^{T}\left\|p_t - q_t\right\|_1^2 \leq 2\mathcal{L}(\theta) - \frac{2}{T}\sum_{t=1}^{T}H(p_t). \tag{88}$$

Since the entropy $H(p_t) \geq 0$, we have:

$$\frac{1}{T}\sum_{t=1}^{T}\left\|p_t - q_t\right\|_1^2 \leq 2\mathcal{L}(\theta). \tag{89}$$

Step 4: Concentration Inequality for the Sum of KL Divergences

Assuming that the KL divergences $\{\mathrm{KL}(p_t \| q_t)\}_{t=1}^{T}$ are random variables bounded above (since the maximum KL divergence between two distributions is finite), we can apply Hoeffding's inequality to bound the probability that the average KL divergence deviates from its expected value.

However, since the data sequences $\mathbf{x}$ are dependent, we need to consider the dependence in the data. Under the assumption that the sequence length T is finite and that the conditional distributions satisfy the Lipschitz condition, we can ensure that the increments are bounded.

For each t, define the event:

$$A_t = \left\{\mathrm{KL}(p_t \| q_t) - \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\mathrm{KL}(p_t \| q_t)] \geq \epsilon\right\}. \tag{90}$$

By applying Azuma's inequality for martingales or McDiarmid's inequality (with appropriate modification for dependent data), we obtain:

$$\mathbb{P}\left(\frac{1}{T}\sum_{t=1}^{T}\mathrm{KL}(p_t \| q_t) - \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\mathrm{KL}(p_t \| q_t)\right] \geq \epsilon\right) \leq \exp\left(-\frac{2T\epsilon^2}{C^2}\right), \tag{91}$$

where C is a constant depending on the bounds of the KL divergences, and $\epsilon > 0$.

Since we are dealing with finite sequences and bounded KL divergences, we can choose $\epsilon = \sqrt{\frac{C^2\log(1/\delta)}{2T}}$, leading to:

$$\mathbb{P}\left(\frac{1}{T}\sum_{t=1}^{T}\mathrm{KL}(p_t \| q_t) \geq \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\mathrm{KL}(p_t \| q_t)\right] + \sqrt{\frac{C^2\log(1/\delta)}{2T}}\right) \leq \delta. \tag{92}$$

Step 5: Final Bound

Combining the results, with probability at least $1-\delta$,

$$\frac{1}{T}\sum_{t=1}^{T}\|p_t-q_t\|_1^2 \leq 2\mathcal{L}(\theta)-\frac{2}{T}\sum_{t=1}^{T}H(p_t)+2\sqrt{\frac{C^2\log(1/\delta)}{2T}}. \tag{93}$$

Letting C be such that $C^2=2\log(2T_{\max}/\delta)$, we can write:

$$2\sqrt{\frac{C^2\log(1/\delta)}{2T}}=\frac{2\log(2T_{\max}/\delta)}{T}. \tag{94}$$

Thus, the bound becomes:

$$\frac{1}{T}\sum_{t=1}^{T}\|p_t-q_t\|_1^2 \leq 2\mathcal{L}(\theta)-\frac{2}{T}\sum_{t=1}^{T}H(p_t)+\frac{2\log\left(\frac{2T_{\max}}{\delta}\right)}{T}. \tag{95}$$

Since the entropies $H(p_t)$ are non-negative, we can further simplify:

$$\frac{1}{T}\sum_{t=1}^{T}\|p_t-q_t\|_1^2 \leq 2\mathcal{L}(\theta)+\frac{2\log\left(\frac{2T_{\max}}{\delta}\right)}{T}. \tag{96}$$

This completes the proof of the theorem. □

The key idea of this proof is to use Pinsker's inequality to relate the ℓ_1 distance between the true distribution p_t and the model's distribution q_t to the KL divergence, which is directly connected to the training loss $\mathcal{L}(\theta)$. By applying concentration inequalities, we control the deviation of the empirical KL divergence from its expected value over the sequence. The Lipschitz continuity of the conditional distributions ensures that dependencies in the sequence do not lead to unbounded errors.

Compared to the autoregressive (AR) models, the NATP approach benefits from predicting future tokens conditioned on the true context rather than the generated one, thereby avoiding error accumulation due to exposure bias. In AR models, errors can compound over time as the model feeds its own predictions back into itself. In contrast, NATP trains the model to predict the entire future sequence at each time step, leveraging the full context and improving global coherence in predictions.

It is important to note that the assumptions and bounds provided are theoretical and rely on certain conditions, such as the Lipschitz continuity and finite sequence lengths. In practice, these conditions may be approximated, but the theoretical framework offers valuable insights into why NATP can be more effective in handling long-term dependencies and mitigating error accumulation in sequence modeling tasks.

D Detailed Information about Datasets and Metrics

D.1 Datasets

This chapter serves as a supplement to Section 4, providing detailed information about the datasets used in this study.

Pretraining Dataset. All pretraining datasets employed are publicly available, with their specifics outlined in Table 3.

During the fine-tuning phase, we utilized two datasets: a publicly available small dataset, AC3/4, and a large private dataset, MEC. Detailed information about these datasets is as follows:

AC3/4. AC3 and AC4 are two labeled subsets extracted from the mouse somatosensory cortex dataset of Kasthuri15 Kasthuri et al. (2015), obtained at a resolution of $3\times3\times29\text{nm}^3$. These subsets include 256 and 100 sequential images (each 1024×1024 pixels), respectively. We use varying numbers of the top sections (5, 10, 20, 30, 50, and 100) of AC3 to simulate different proportions of labeled data. The bottom 50 sections of AC3 and AC4 are used for testing. To support semi-supervised learning, we utilize 200 sections from AC3/AC4 as unlabeled data.

Table 3: Detailed description of the EM pre-taining datasets

Dataset	Modality	Resolution	Species	Target Region
Full Adult Fly Brain (FAFB) Schlegel et al. (2021)	EM	286720×155648 pixels	Drosophila	Whole brain
MitoEM-H Wei et al. (2020)	EM	30 μm^3	Human	Cortex (Mitochondria)
MitoEM-R Wei et al. (2020)	EM	30 μm^3	Rat	Cortex (Mitochondria)
FIB-25 Takemura et al. (2017)	EM	$5 \times 5 \times 5\ nm^3$	Mouse	CA1 Hippocampus
Kasthuri15 Kasthuri et al. (2015)	EM	$3 \times 3 \times 30\ nm^3$	Mouse	Neocortex

MEC. The MEC dataset originates from our team's Mouse MEC MultiBeam-SEM imaging efforts, where we performed comprehensive brain imaging of mice, accumulating data at the petabyte scale. We processed the images through registration, denoising, and interpolation, and divided them into different layers according to brain regions. Specifically, we selected data from Wafer 4 at layer VI and wafers 25, 26, and 36 at layer II/III. The dataset was acquired at a resolution of 8 nm × 8 nm × 35 nm, with the relative imaging positions illustrated in Fig. 5. Each volumetric block has a size of 1250 × 1250 × 125 voxels. All voxels in the dataset are fully annotated.

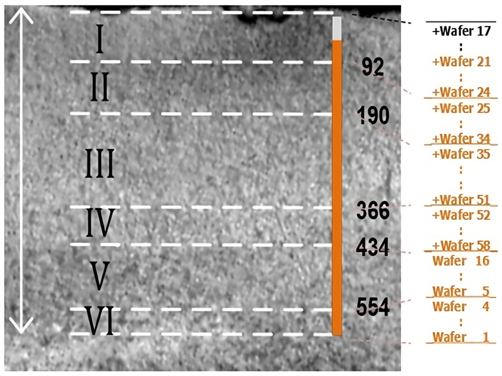

Figure 5: The relative positions of the wafer layers selected from the MEC dataset.

D.2 Metrics

Variation of Information (VOI) is an information-theoretic measure that assesses the distance between two clusterings in terms of their average conditional entropy. Given the predicted segmentation S_{pred} and the ground-truth segmentation S_{gt}, VOI is defined as:

$$VOI(S_{pred}, S_{gt}) = H(S_{pred}|S_{gt}) + H(S_{gt}|S_{pred}), \tag{97}$$

where $H(\cdot|\cdot)$ denotes the conditional entropy. It can be calculated by:

$$H(S_{pred}|S_{gt}) = -\sum_{i=1}^{|S_{gt}|}\sum_{j=1}^{|S_{pred}|} \frac{|S_{gt}^i \cap S_{pred}^j|}{N} \log \frac{|S_{gt}^i \cap S_{pred}^j|}{|S_{gt}^i|}, \tag{98}$$

where S_{gt}^i and S_{pred}^j represent the i-th and j-th segments in the ground-truth and predicted segmentation, respectively, and N is the total number of voxels. VOI ranges from 0 to ∞, with a lower value indicating better segmentation performance.

Adjusted Rand Index (ARAND) is a variant of the Rand Index Arganda-Carreras et al. (2015) that corrects for chance when comparing two clusterings. It is defined as:

$$ARAND(S_{pred}, S_{gt}) = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{N}{2}}{[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}]/2 - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{N}{2}}, \tag{99}$$

where n_{ij} is the number of voxels that are in segment i of S_{pred} and segment j of S_{gt}, $a_i = \sum_j n_{ij}$ is the number of voxels in segment i of S_{pred}, $b_j = \sum_i n_{ij}$ is the number of voxels in segment j of S_{gt}, and $N = \sum_{ij} n_{ij}$ is the total number of voxels. ARAND ranges from 0 to 1, with a lower value indicating better segmentation performance.

E Discussion

E.1 Limitations

Despite TokenUnify's significant performance advantages in long-sequence autoregressive tasks, this may be attributed to the specific characteristics of 3D image sequences. Its effectiveness on natural images has yet to be validated in downstream tasks. Additionally, due to the unique nature of the neuron data, we have only demonstrated performance on segmentation tasks in the main text. Future work will extend the evaluation to a broader set of downstream tasks, such as classification, detection, and other standard vision tasks.

E.2 Preliminary Exploration of TokenUnify on Natural Images

We are currently pretraining TokenUnify on natural images using the LAION-5B dataset Schuhmann et al. (2022). Specifically, each image is divided into non-overlapping patches of size 16x16. We conducted 800 epochs of pretraining with TokenUnify. As the downstream classification tasks are still in progress, we present only the initial visual results here.

Specifically, we pretrained using both the Autoregress approach and the TokenUnify approach. For evaluation, given the first k patches of an image, we predicted the (k+1)th patch and then concatenated all the predicted patches. We used the PSNR metric to compare the reconstructed image with the original image, assessing the representational capability of each method. We selected the high-resolution Kodak Kodak (1993) dataset as our test set. Our experimental results are shown in Fig. 6. The PSNR values for the reconstruction of 24 images are detailed in Table 4. TokenUnify outperformed the Autoregress approach in terms of visual metrics, indicating that TokenUnify likely extracted better visual representations during the pretraining stage.

F Mehtod Details

F.1 Summary of the TokenUnify Algorithm

TokenUnify is a novel pre-training method for scalable autoregressive visual modeling. It integrates random token prediction, next-token prediction, and next-all token prediction to mitigate cumulative errors in visual autoregression while maintaining favorable scaling laws. The algorithm leverages the Mamba network architecture to reduce computational complexity from quadratic to linear for long-sequence modeling.

Pre-training is conducted on a large-scale, ultra-high-resolution electron microscopy (EM) image dataset, providing spatially correlated long sequences. TokenUnify demonstrates significant improvements in segmentation performance on downstream EM neuron segmentation tasks compared to existing methods. Our pre-training and fine-tuning algorithms are summarized in Algorithm 1 and Algorithm 2, respectively.

The TokenUnify pre-training algorithm captures both local and global dependencies in image data through mixed token prediction tasks. The Mamba network architecture ensures efficient modeling of long sequences. During fine-tuning, the pre-trained model adapts to downstream segmentation tasks using labeled data, achieving state-of-the-art performance on EM neuron segmentation benchmarks.

F.2 Perceiver Resampler

The workflow of the Perceiver Resampler Alayrac et al. (2022); Chen & Mueller (2024; 2023) can be summarized in the following steps: 1. Combine the output of the Vision Encoder (e.g.,

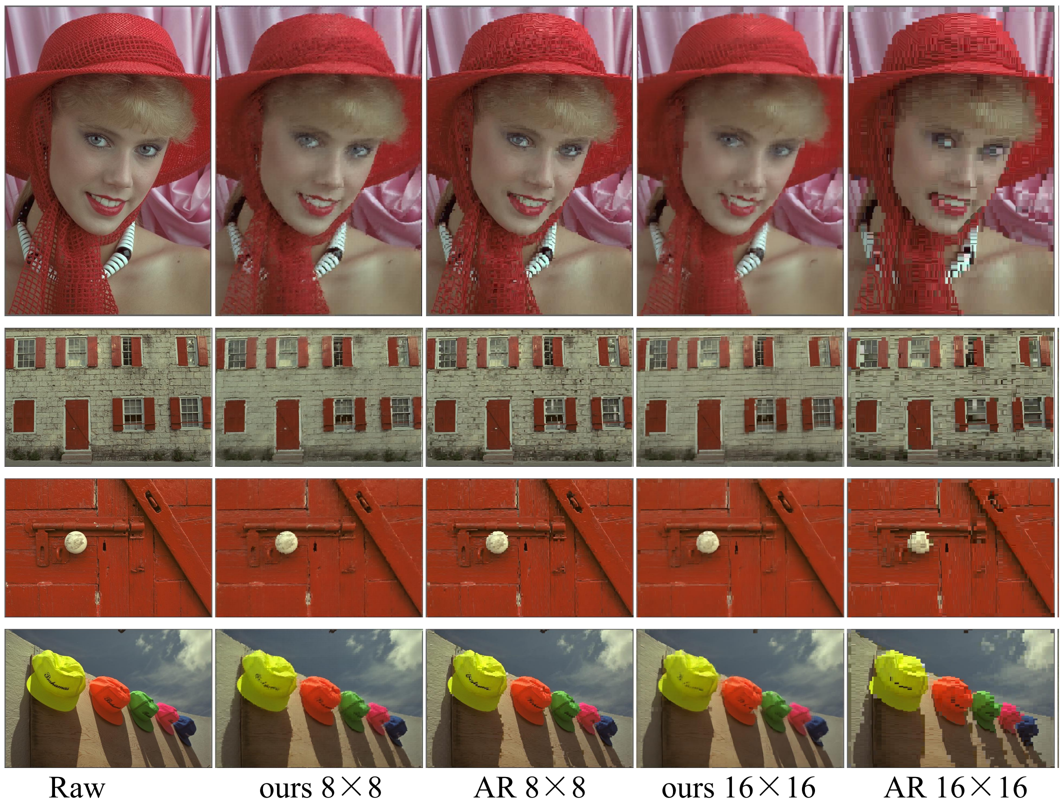

Figure 6: Shows the reconstruction result of selected Kodak dataset, images are divided into different sizes of patches. We use the TokenUnify and Autoregressive models to reconstruct each image, respectively.

features from images) with learned time position encodings. 2. Flatten the combined features into a one-dimensional sequence. 3. Flatten the combined features into a one-dimensional sequence. 4. Process the flattened features using Transformer layers that incorporate attention mechanisms, which interact with learned latent query vectors. Output a fixed number of visual tokens equal to the number of latent queries.

Algorithm 3: Perceiver Resampler Pseudocode

```
Input  : x_f - The [T, S, d] visual features (T=time, S=space)
Input  : t - The [T, 1, d] time position embeddings
Input  : x - R learned latents of shape [R, d]
Input  : num_layers - Number of layers
Output: x - Updated learned latents
1 Add time position embeddings and flatten:
2 $\mathbf{x}_f \leftarrow \mathbf{x}_f + \mathbf{t}$
3 $\mathbf{x}_f \leftarrow \text{flatten}(\mathbf{x}_f)$
4     // $[T, S, d] \rightarrow [T \times S, d]$
5 Apply the Perceiver Resampler layers:
6 for $i \leftarrow 1$ to num_layers do
7     $\mathbf{x} \leftarrow \mathbf{x} + \text{attention}_i(q = \mathbf{x}, kv = \text{concat}([\mathbf{x}_f, \mathbf{x}]))$
8     $\mathbf{x} \leftarrow \mathbf{x} + \text{ffw}_i(\mathbf{x})$
9 return x
```

The input visual features, denoted as $\mathbf{x}_f$, have a shape of $[T, S, d]$, where T represents the time dimension, S the spatial dimension, and d the feature dimension. The time position embeddings,

Table 4: Presents the PSNR results of reconstructing 24 images from the Kodak dataset using TokenUnify and Autoregress. The experiments were conducted with patch sizes of 16x16 and 8x8.

Kodak Name	16×16 Autoregress	16×16 TokenUnify	8×8 Autoregress	8×8 TokenUnify
1.png	19.249	21.549 (+2.300)	21.247	21.990 (+0.743)
2.png	24.662	27.321 (+2.659)	27.269	27.799 (+0.530)
3.png	22.665	27.113 (+4.448)	26.851	28.110 (+1.259)
4.png	22.353	26.152 (+3.799)	25.466	26.713 (+1.247)
5.png	15.353	18.859 (+3.506)	18.437	19.847 (+1.410)
6.png	20.139	22.376 (+2.237)	21.661	23.064 (+1.403)
7.png	19.990	23.170 (+3.180)	23.334	24.479 (+1.145)
8.png	15.146	18.169 (+3.023)	17.829	18.770 (+0.941)
9.png	22.080	24.918 (+2.838)	24.959	25.957 (+0.998)
10.png	22.239	25.213 (+2.974)	25.042	25.936 (+0.894)
11.png	20.289	22.536 (+2.247)	22.638	23.723 (+1.085)
12.png	21.854	25.929 (+4.075)	25.806	27.005 (+1.199)
13.png	15.946	18.494 (+2.548)	17.657	18.969 (+1.312)
14.png	18.107	21.227 (+3.120)	20.696	22.195 (+1.499)
15.png	20.750	24.659 (+3.909)	25.321	26.111 (+0.790)
16.png	23.216	25.887 (+2.671)	25.334	26.694 (+1.360)
17.png	20.672	24.346 (+3.674)	24.220	25.614 (+1.394)
18.png	19.959	22.017 (+2.058)	21.249	22.336 (+1.087)
19.png	22.394	25.062 (+2.668)	24.094	25.384 (+1.290)
20.png	21.478	24.723 (+3.245)	24.124	25.346 (+1.222)
21.png	17.503	20.149 (+2.646)	19.567	20.366 (+0.799)
22.png	19.947	23.003 (+3.056)	22.365	23.545 (+1.180)
23.png	17.807	20.315 (+2.508)	19.781	20.959 (+1.178)
24.png	22.111	24.780 (+2.669)	24.313	25.472 (+1.159)

represented by $\mathbf{t}$, are of shape $[T, 1, d]$ and are added to the visual features to incorporate temporal information.

The learned latents, denoted as $\mathbf{x}$, have a shape of $[R, d]$, where R is the number of latents and d is the feature dimension. The parameter `num_layers` specifies the number of layers in the Perceiver Resampler model.

The operation `flatten` reshapes the input tensor from $[T, S, d]$ to $[T \times S, d]$. The function `attention_i` represents the attention mechanism applied in the i-th layer, which takes a query q and key-value pairs kv. The function `concat` concatenates the input tensors along the specified dimension. Finally, `ffw_i` refers to the feedforward network applied in the i-th layer.

F.3 Segmentation Method

The EMmamba network is structured into three principal components: 3D feature encoder, convolution-based decoder for segmentation prediction, and skip connections to integrate local multi-scale features into the decoder for feature fusion. Liu et al. (2023b;c); Sun et al. (2024)

To achieve effective feature encoding, we designed anisotropic downsampling layers and adopted the TSMamba block from the Segmamba Xing et al. (2024). Specifically, in Stage 1, the downsampling layer uses a convolutional kernel size of (1, 7, 7). For the subsequent three layers, the downsampling layers have a convolutional kernel size of (1, 2, 2). The decoder section employs a convolutional kernel size of (1, 5, 5). This anisotropic design is particularly advantageous for processing EM images, which exhibit inherent anisotropy.

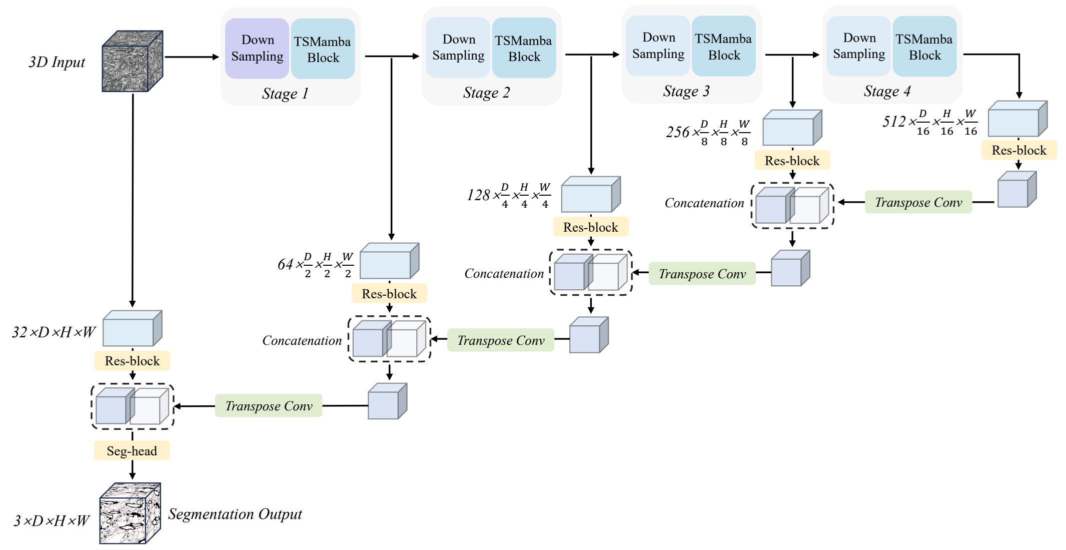
3D Input
Down Sampling
TSMamba Block
Stage 1
Stage 2
Stage 3
Stage 4
$512\times\frac{D}{16}\times\frac{H}{16}\times\frac{W}{16}$
$256\times\frac{D}{8}\times\frac{H}{8}\times\frac{W}{8}$
$128\times\frac{D}{4}\times\frac{H}{4}\times\frac{W}{4}$
$64\times\frac{D}{2}\times\frac{H}{2}\times\frac{W}{2}$
$32\times D\times H\times W$
Res-block
Concatenation
Transpose Conv
Seg-head
$3\times D\times H\times W$
Segmentation Output

Figure 7: Segmentation pipeline.

Table 5: Shows the differ in architecture when adding the parameters of the segmentation backbone.

	EMmamba-tiny	EMmamba-small	EMmamba-middle	EMmamba-large	EMmamba-huge
Mamba layer	[2,2,2,2]	[2,2,2,2]	[2,2,2,2]	[2,2,2,2]	[2,2,2,2]
Feature size	[32,64,128,256]	[64,128,256,512]	[96,192,384,768]	[144,288,576,1104]	[192,384,768,1536]
Hidden size	512	1024	1024	2048	3072
Kernel size	[1,5,5]	[1,5,5]	[1,5,5]	[1,5,5]	[1,5,5]
Batch size	40	22	12	8	4
Param. (M)	28.30	112.5	206.6	506.6	1008

G NUMERICAL RESULTS

G.1 STATISTICAL TEST

In this section, we present the results of our error bar experiments, as detailed in Table 8. These experiments were conducted to assess the variability and reliability of the model's prediction under different conditions.

G.2 ABLATION STUDY RESULTS

In this section, we present the numerical results of the ablation study discussed in Section 5.

Table 6: Ablation study for different pertaining strategy on wafer4 dataset.

Model	Pretraining Strategy			$VOI\downarrow$	$ARAND\downarrow$
	Random token	Next token	Next-all token		
M1	✓			1.2680	0.0862
M2	✓	✓		<u>1.1300</u>	<u>0.0692</u>
M3	✓		✓	1.1907	0.1203
Ours	✓	✓	✓	**0.9951**	**0.0509**

Table 7: Ablation study for the fine-tuning schemes on wafer4 dataset.

Model	Module			$VOI \downarrow$	$ARAND \downarrow$
	Mamba	Encoder	Decoder		
M1	✓			1.1362	0.0782
M2		✓		1.5556	0.1370
M3			✓	1.5295	0.1212
M4	✓	✓		1.1065	0.0629
Ours	✓	✓	✓	**0.9951**	**0.0509**

Table 8: Quantitative comparison of segmentation results on Wafer4 dataset with error bar. 'Post.' represents the post-processing algorithms. * denotes the MAE pretraining strategy He et al. (2022). † indicates our TokenUnify pretraining strategy. The best results are in **bold** and the second best results are in underlined.

Post.	Method	Wafer4				Param. (M)
		$VOI_M \downarrow$	$VOI_S \downarrow$	$VOI \downarrow$	$ARAND \downarrow$	
Waterz Funke et al. (2018)	Superhuman [40]	0.3392±0.0167	1.2247±0.0857	1.5639±0.0921	0.2050±0.0284	1.478
	MALA [29]	0.6217±0.1266	1.5314±0.1123	2.1531±0.1004	0.1490±0.0476	84.02
	PEA [35]	0.3943±0.0655	1.0036±0.1435	2.1531±0.1004	0.1490±0.0476	1.480
	UNETR [31]	0.4454±0.0155	1.7979±0.1548	2.2433±0.1424	0.3244±0.0701	129.1
	EMmamba	0.4353±0.052	1.3018±0.0086	1.7371±0.0432	0.1872±0.0156	28.30
	Superhuman*	0.2907±0.0063	0.9437±0.0451	1.2344±0.0388	0.1202±0.0121	1.478
	MALA*	0.7732±0.1432	1.2063±0.0458	1.9768±0.1232	0.2663±0.0549	84.02
	PEA*	0.2712±0.0185	0.9715±0.1841	1.2427±0.1963	0.0805±0.0386	1.480
	UNETR*	0.3554±0.0411	0.8579±0.0229	1.2133±0.0574	0.1150±0.0209	129.1
	EMmamba*	0.2363±0.0212	1.0782±0.0251	1.3144±0.0444	0.0967±0.0097	28.30
	EMmamba†	0.2124±0.0172	0.8047±0.0057	**1.0024±0.0463**	**0.0551±0.0040**	28.30
LMC Beier et al. (2017)	Superhuman [40]	0.2006±0.0054	2.1283±0.1378	2.3289±0.1427	0.2924±0.0408	1.478
	MALA [29]	0.3094±0.0478	2.3802±0.1863	2.6869±0.1558	0.2303±0.0314	84.02
	PEA [35]	0.2303±0.0870	1.6373±0.1289	1.8343±0.0732	0.1611±0.0152	1.480
	UNETR [31]	0.1625±0.0144	3.3146±0.1391	3.4772±0.1272	0.6600±0.0304	129.1
	EMmamba	0.1594±0.0005	2.0921±0.0300	2.2515±0.0298	0.2104±0.0113	28.30
	Superhuman*	0.2363±0.0222	1.8475±0.0781	2.0838±0.0782	0.1946±0.0171	1.478
	MALA*	0.2022±0.0089	2.5760±0.0457	2.8117±0.0346	0.5695±0.0183	84.02
	PEA*	0.2736±0.1603	1.5868±0.0900	1.8604±0.0815	0.1386±0.0134	1.480
	UNETR*	0.1829±0.0495	1.7723±0.0324	1.9552±0.0816	0.1372±0.0316	129.1
	EMmamba*	0.1342±0.0020	1.9014±0.0286	2.0356±0.0301	0.1420±0.0023	28.30
	EMmamba†	0.1417±0.0022	1.5186±0.0076	**1.6604±0.0086**	**0.0592±0.0002**	28.30