Brain Reconstruction by Self Supervised Semantic Stitching of Non-overlapping 3D Microscopic Image (Manuscript)

Bokai Xu, Chaoyu Yang, Fang Xu*, Pengcheng Zhou*
Brain Cognition and Brain Disease Institute,
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences

Abstract

We proposed a deep learning pipeline for stitching two non-overlapping brain slices or any other kind of 3D microscopic images (especially with offset, or slicing loss, not consecutive) without any human annotation, so it could generalize to any kind of stitching tasks. The key idea is to train a foundation vision model using unsupervised paradigm on a large and thorough randomly cropped dataset, i.e. Masked Autoencoder, to obtain a model which has the capability to inpaint 3D brain slice. During inference, extrapolate two brain slice for a proper amount of layers to construct overlapping areas, then we reformulate this problem into 3D stitching problem with overlapping area. Specifically, we explore the pretraining of 3D masked autoencoder with different data preprocessing method, and the influence of patch size and loss function on reconstruction accuracy. Besides this, we also explored other methods for stitching without overlapping area, such as learning to optimize, and siamese network. To effectively evaluate the performance of different methods, we proposed several benchmark methods to quantitatively gauge different methods, including inverse-problem method and self-consistency method. We also use human based evaluation methods, by checking stitching results manually on different scales. We conduct ablation study of each stitching method by using raw pixel based module to replace the main module of our methods. All codes will be made open sourced at `https://github.com/bokesyo/semantic_stitching_3d`.

1 Introduction

VISOR2 [13] fast brain imaging could enable us to obtain the image of whole brain in an extremely short period of time, however, this technique, along with various biological imaging technique, require samples to be sliced into non-overlapping 3D slices. Slicing could lead to rigid deformation (including rotation and translation) and non-rigid deformation. This shortcoming could lead to several problems when we analyze the neuronal circuits and construct whole brain connectomics later. The challenge is to stitch those non-overlapping brain slices, with possible slicing loss, into a semantic-consecutive reconstructed brain. Usually without off-the-shelf human annotated supervised dataset for a specific domain, like brain imaging for mouse, it is hard to realize this. In this paper, we introduce an entirely self supervised pipeline based on deep learning to semantically stitch non-overlapping microscopic 3D images. Due to its unsupervised nature, it could generalize to any other domains without extra human annotation.

*Corresponding authors.

2 Related Work

2.1 Non-overlapping Stitching

Stitching with overlapping area is a typical problem which could be handled by a variety of tractable approaches, however, in brain imaging case, there is no overlapping, and there is possible slicing loss. An optimization based method [13] is to flatten the surface of brain slices and assume there is no slicing loss. Then, a deformation field is applied and an optimization process is performed to make sure the similarity (mutual information) of two surfaces is maximized. However, this process may not work well when there is a gap between two brain slices, and the optimization process may lead to some local minima. The stitching result, may not be semantic reasonable as well. To make further improvement, one could use deep learning techniques to handle semantic reasonable stitching.

To handle this problem, one could use a optical flow estimation method [10] by learning to optimize. However, this needs synthesized data using computer graphic technique, which is not plausible on microscopic imaging scenario. Others [8] curated an annotated dataset on their specific dataset, which labels neuron fibers endpoints manually, then train a model from scratch by using such dataset. At inference time, the model predicts keypoints for input brain slice images as an indicator to stitch. This pipeline works for specific domain, but are lack of generalizability. Human annotation is commonly lacked in our domain, and human annotation, sometimes biased, prefer certain structures in the image. To make sure the pipeline generalize to any other image domain rather than specific domain, we may not rely on human annotation and should turn to unsupervised learning.

With such vision, we may need to use self-supervised pipelines to obtain a deep vision model to learn high-level semantic representations without supervision from our brain dataset.

2.2 Sliding Window

A hinder for applying deep learning models on brain slices is, the sizes of brain slice image are not fixed. So, we need a method to process brain image that could generalize to varying sizes. Another difficulty is, the brain slice image is commonly large, no such a model could handle $32 \times 2000 \times 2000$ pixels in one go as well. Based on the above two considerations, we fixed input window size to be $32 \times 32 \times 32$, and deploy *sliding window* method to process the whole brain slice.

2.3 Self-supervised Pretraining

Masked Autoencoder [4] is a paradigm to train foundation models. It uses pixel reconstruction task to train an encoder and decoder transformer. Upon training finished, the encoder-decoder could perform pixel reconstruction or extrapolation, and its enocder could be used to produce semantic representations for an input image. Masked Autoencoders are scalable self supervised learners, so we postulate that with the scale of model parameters increases, the capabilities (semantic understanding and pixel extrapolation) of the model could also increases predictably as we scale up. Context Autoencoder [2], another self supervised paradigm, made some modifications to Masked Autoencoder, thus making semantic representations produced by encoder be solely semantic. The reason is that in Masked Autoencoder, the semantic representations produced by encoder could also includes some intriguing information. Both paradigms produce foundation models with different number of trainable parameters (scales). However, those models could not be directly used on biological imaging scenario, as they are designated to 2D natural image processing, rather than 3D microscopic image processing, so we train such models from scratch in this work.

3 Quantitative Evaluation Benchmark

There is no existing evaluation benchmark. So, we construct two distinct methods to quantitatively evaluate the performance.

3.1 Inverse Problem Benchmark

Due to a lack of labelled dataset to gauge model's capability to perform stitching, it is natural to construct dataset to do that. We have plenty of intact brain slice image already (because each brain slice is intact, the intra-slice stitching has been done), so we just divide it into two part, lower and upper slice. Then, we apply a random rigid transform to the lower slice, then restore the slice using our pipelines. By calculating the restoration error, we could quantitatively evaluate models' performance. For inverse problem benchmark, there is a hyperparameter to be considered, the slicing loss. Slicing loss simulated the loss during slicing, and has a unit of pixels. For example, if offset = 0, there is no slicing loss. If offset is 1, the slicing loss is 4um. If offset is 2, the slicing loss is 8um, so and so forth.

To use inverse problem benchmark, it is required to choose a slicing gap level (offset). Notably offset = 0 is a not appropriate, because the intra-slice stitching is a case where offset = 0.

We present the benchmark result on different methods. We noted that, as offset > 0, the performance of siamese network method has no significance improvement than trivial method.

3.2 Self-consistency Benchmark

Even without human annotation, we could make a hypothesis that a trustful stitching result will always converge to a single meaningful result given different initial states, while a less trustful stitching result will output divergent results given different initial states. Based on such a hypothesis, we could design a self consistency benchmark to gauge different methods without any human annotation. We use sliding window method to produce different initial states, and use a multi-scale stitching pipeline described above, then for each cell in the finest scale, we could obtain multiple possible results. Then we could calculate the variance of results of that cell, and average all the variances of cells, to produce an average variance. By doing so, we could evaluate the stitching results.

4 Foundation Model Pretraining

4.1 Patch Embedding

The ViT [3] based masked autoencoder [4], process input image by patches, because transformer [11] is desgined for a group of tokens, to adapt transformer to image modality, patch embedding should be introduced. This paradigm, states that *an image is worth 16 by 16 words*. For example, each p_d, p_h, p_w pixels are packed into a $p_d \times p_h \times p_w$ vector, where $p_{d,h,w}$ are patch sizes, followed by a linear projection from $p_d \times p_h \times p_w$ to d_{enc}, where d_{enc} is the hidden dimension of transformer [11]. We inherit this in our work. Additionally, we replace 2D patch embedding layer to 3D counterpart.

4.2 Model Input Window Size

The training dataset is randomly cropped from a large raw 4um-resolution dataset of our microscopic imaging consisting of 100+ whole mouse brains, each mouse brain has around 40 brain slices of 300um. The masked autoencoder model, could not handle entire brain slice because the non-downsampled slice is too large, and the size is not constant value. Thus, we fixed the input window size of Masked Autoencoder by $32 \times 32 \times 32$. This size, is a result of two considerations:

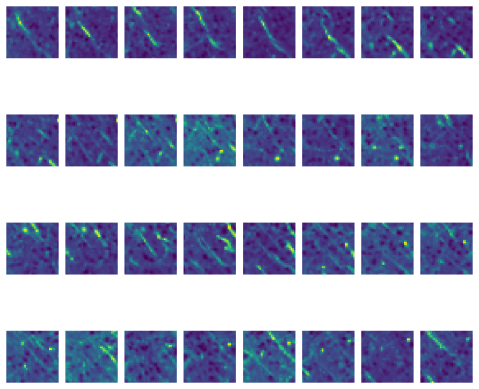
1. **Perception of Structrues:** Ideally, the receptive field should be as large as possible. A too restricted field, such as $8 \times 8 \times 8$, limits the model to local structures, making it difficult to recognize broader patterns.
2. **Computation Cost:** A significantly larger receptive field greatly increases computation costs. An increase by a factor of 2 in the edge length of a 3D tensor leads to an 8-fold increase in the number of tokens since the model is 3D. The attention operation requires $O(N^2)$ memory and compute, where N is the number of tokens. Hence, doubling the receptive field leads to a 64-fold increase in memory and computation. There is alternative choice like Swin Transformer [7], which replace global attention with shifted window attention, reducing memory and compute cost to $O(N)$. Due to the limit of compute resources, we did not use a large receptive field.

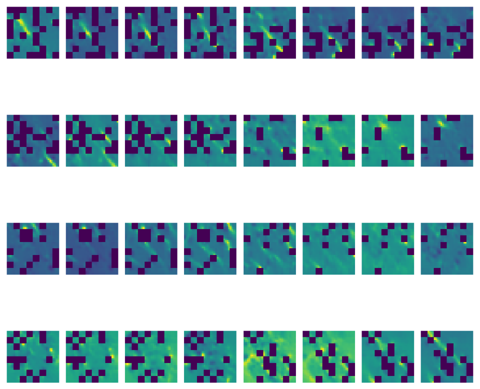
4.3 Patch Size

The patch size, is proven to be crucial for successful pretraining of our masked autoencoder. We explored patch size of $8 \times 8 \times 8$ and $4 \times 4 \times 4$. We manually checked the training process by performing pixel reconstruction results from model checkpoints on another evaluation dataset to avoid memorizing effect. We empirically found that in $8 \times 8 \times 8$ setting, the reconstruction for large and low frequency structures is good, but when it comes to thin and subtle structruers like neural fibers, the performance is not satisfying, the reconstructed tiny sturcutures could not be recognized clearly. We hypothesized that it is due to the training objective, and we use Mean Average Loss, SSIM Loss [12] instead of standard Mean Square Loss, and did not observe any improvement. Furthermore, with patch size replaced by $4 \times 4 \times 4$, which introduces $8\times$ more tokens, the reconstruction for all structures have a significant improvement than $8 \times 8 \times 8$ setting, especially for tiny structures like neural fibers. Finally, we fixed patch size to be $4 \times 4 \times 4$ for further experiments. This is explainable because when patch size is $4 \times 4 \times 4$, one token in transformer could only be responsible for 64 pixels, but this number is 512 pixels otherwise.

4.4 Training Data Cleaning

During pretraining, we observed training failure and huge frustration of loss curve. We postulate that it was due to the merge of noise data points. After filtering the noisy image by using entropy as an indicator (we keep entropy greater than a specific value), the training failure is eliminated. The noisy images are liquids that captured during imaging, and they does not consist any information, which could be treated as pure noise.


(a) Original window


(b) Reconstructed masked patches, with unmasked patches removed

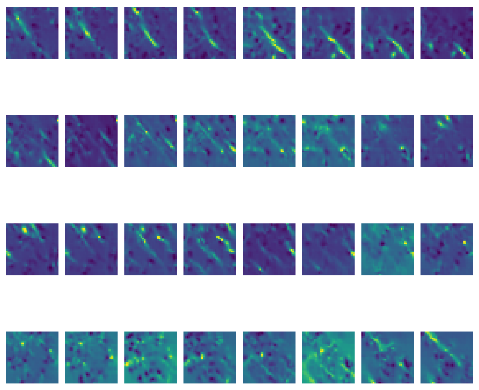

(c) Reconstructed window

Figure 1: Pixel Reconstruction of Pretrained Masked Autoencoder 3D

5 Semantic Stitching with Foundation Model

Once we have trained our foundation Masked Autoencoder, we created two distinct sub-pipelines based on the foundation model, based on different capabilities of the pretrained model.

(i) The encoder of pretrained MAE could be used to construct a discriminator, accomplishing stitching by predicting whether two slices are consecutive – **Discriminative Method**.

(ii) The encoder-decoder of pretrained MAE could be used to extrapolate brain slice, so as to reformulate the non-overlapping stitching problem into a tractable overlapping stitching problem – **Generative Method**.

We mainly discuss Discriminative Method: Siamese Networks in this work. We leave Generative Method to Appendix.

5.1 Intuition

Semantic representation ($32 \times 32 \times 32 \rightarrow 8 \times 8 \times 8 \times d_{enc}$) derived from raw input by pretrained encoder encodes high level semantic (or symbolic) information about both local information and global information in each d_{enc} token. We **hypothesized** that there is a surface semantic representation (SSR) space, which is different from raw semantic representation (RSR) space, focus on the common information of consecutive brain slices. So, by mapping two non-overlapping brain slices closely in SSR space, it is possible to build a discriminator for non-overlapping stitching.

We have a premise that raw pixels, are not that continuous between two non-overlapping slices. In contract, the SSR captures the contiguous symbolic representation of brain tissue **across** slices, allowing for alignment despite the absence of overlapping regions.

For instance, if the upper slice has the upper half of a cell, the lower slice has the lower half of a cell. However, there are slicing loss, the pixels of lower and upper slices could not map well. But the symbolic representation by pretrained encoder is consecutive: for upper slice, the semantic representation is like 'upper half of a neural cell'; for lower slice, the semantic representation is like 'lower half of a neural cell'. After the mapping from RSR to SSR, the upper slice will have a semantic representation of 'cell', and the lower slice will have a semantic representatio of 'cell' as well. In this case, it is easy to map these two areas together. If we have multiple pairs of such structures, we could have a good stitching result.

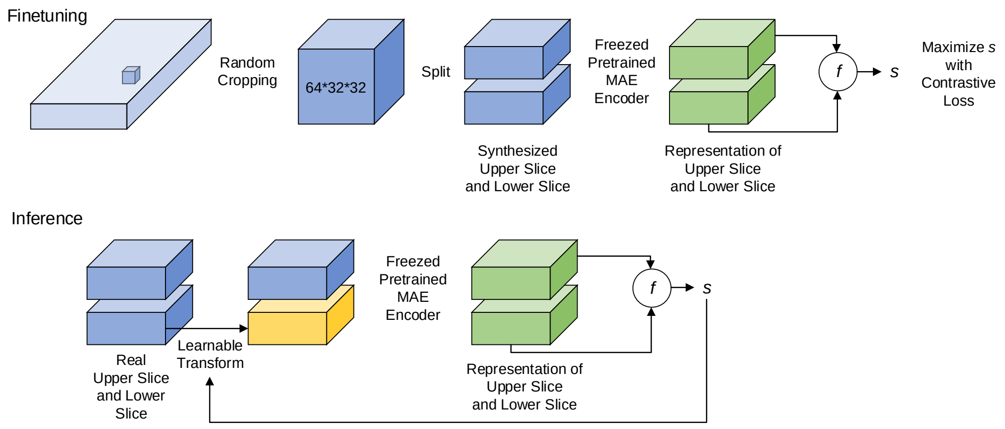

Finetuning
Random Cropping
64*32*32
Split
Freezed Pretrained MAE Encoder
f
s
Maximize s with Contrastive Loss
Synthesized Upper Slice and Lower Slice
Representation of Upper Slice and Lower Slice
Inference
Freezed Pretrained MAE Encoder
f
s
Real Upper Slice and Lower Slice
Learnable Transform
Representation of Upper Slice and Lower Slice

Figure 2: Training and Inference pipelines for Siamese Network Method

5.2 Implementation

Consider a pair windows from upper and lower slice, we call the upper window $X_u \in [B, C, D, H, W]$ and lower window $X_l \in [B, C, D, H, W]$. For upper slice window, we care about its bottom, so we readout the last layer of depth dimension (D). For lower slice window, we care about its top, so we readout the first layer of depth dimension (D).

$$\text{SSR}_u = \text{Linear}_u(\text{Encoder}(X_u)[:, :, -1, :, :]) \in [B, H/4, W/4, d] \tag{1}$$

$$\text{SSR}_l = \text{Linear}_l(\text{Encoder}(X_l)[:,:,0,:,:]) \in [B, H/4, W/4, d] \tag{2}$$

where Encoder is the freezed pretrained MAE encoder $[B, C, D, H, W] \to [B, D/4, H/4, W/4, d]$; Linear_u and Linear_l are trainable siamese network with single linear layer maps $[d] \to [d]$.

5.3 Training Objective

Now our goal is to maximize the similarity of paired SSR and minimize the similarity of unpaired SSR. To realize this, we use element-wise cosine similarity. In practice, we employ a contrastive loss function to refine the Siamese Network, encouraging it to minimize the distance between SSRs of continuous slices while maximizing the distance between non-continuous slices. This differential learning approach effectively teaches the model to recognize and align slices based on their shared semantic content.

Note that the cosine similarity is

$$s(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}||||\mathbf{b}||} \tag{3}$$

For a minibatch of training data, we have

$$L_{\text{positive}} = 1 - \sum_i s(\text{SSR}_u[i], \text{SSR}_l[i]) \tag{4}$$

Training with only positive similarity loss is not enough, as all the representations converge to a homogeneous token. We add a contrastive loss, measuring the similarity of non-continuous upper and lower slices, i.e., *inter slice contrastive loss*, formulated by

$$L_{\text{inter}} = \sum_{(i,j), j \neq i} s(\text{SSR}_u[i], \text{SSR}_l[j]) \tag{5}$$

We explored the above loss function, and found that the mapping partially works. It could successfully detect where two images are roughly consecutive. However, when the upper window and lower window are almostly consecutive, but not seamlessly fit, the similarity score are already high, in other words, this suggests that tokens in an input window are homogeneous.

To handle this problem, we need to add another contrastive loss, i.e., *intra slice contrastive loss*, formulated by

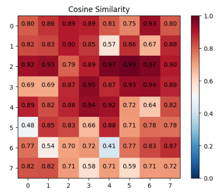
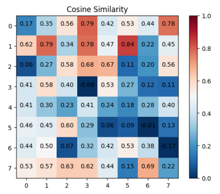
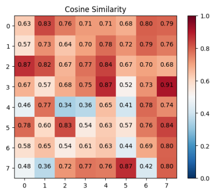
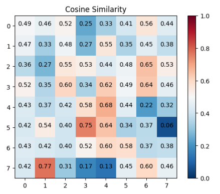
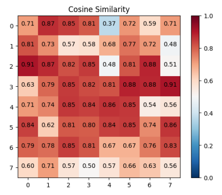
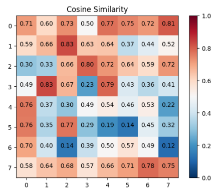
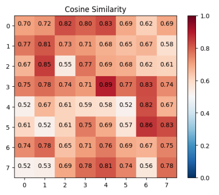
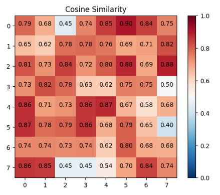
$$L_{\text{intra}} = \sum_i \sum_{(j,k)} \sum_{(m,n) \neq (j,k)} s(\text{SSR}_u[i,j,k], \text{SSR}_l[i,m,n]) \tag{6}$$

We conduct a set of comparative experiment to show the effectiveness of our proposed siamese network method.

5.4 Validation

We validate the improvement of semantic stitching over trivial method (pixel-level stitching). To realize this, we first compare the finetuned pretrained model with $L_{\text{positive}} + L_{\text{inter}}$ (semantic) and non-pretrained model (pixel) on consecutive (positive brain slice pairs) and non-consecutive (negative brain slice pairs) brain slices. Next, we add 5% erosion on the top of lower slice, and conducted the validation again. The method is, input two raw brain slices separately, then get the two output from the model, after that we calculate the cosine similarity of the pair of brain slices, then plot the heatmap. A higher value means in this area the two brain slices are considered very similar. Results are shown in Tab. 1.

Table 1: Validation of Siamese Networks

	Positive pairs	**Negative pairs**	**Positive pairs with erosion**	**Negative pairs with erosion**
Semantic model				
Pixel model				

From the ablation study we noted that the finetuned pixel model could discriminate the positive and negative pairs if these pairs have no erosion. However, with erosion, the model failed to discriminate, the underlying reason could be, the pixel model only learn to attend to the last few layer of upper slice and the first few layers of lower slice, thus with erosion, the model failed. In contrast, the semantic model did not fail with erosion, because the semantic representation is more robust and consecutive.

5.5 Multiscale Stitching

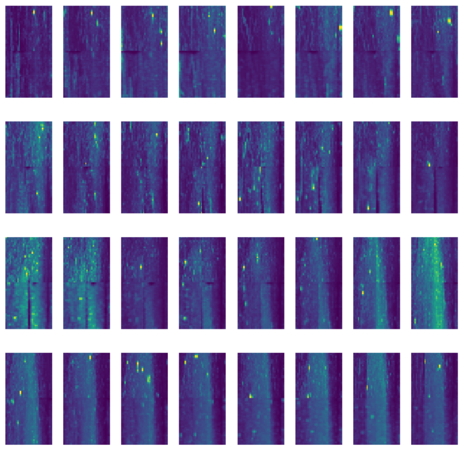
We investigated the effectiveness of dicriminative method on different scales. Empirically, we found that on a coarse scale. For example, the perceptive field is $1024 \times 1024 \times 32$, then downsampled to model input size $32 \times 32 \times 32$, the performance is good in both semantic stitching and pixel-level stitching (trivial method). With the scale become finer, the semantic stitching method has a better performance compared with pixel-level stitching.

For implementation, we used the simplest implementation: first implement stitching on $1024 \times 1024 \times 32$ scale, save the deformantion parameters for next scale. For the next scale, we divide the $1024 \times 1024 \times 32$ area into 4 child nodes, each has a size of $512 \times 512 \times 32$. The stitching of child node will inherit the deformation parameters from parent node. The deformation field is applied on the downsampled whole brain slice, so the complex increases for finer scale. This could be optimized by cropping the most relavant area for stitching, so the complexity could be maintained nearly constant as the scale comes to the finest one, denoted as C. Assume the total number of scales to be 6 (1024, 512, 256, 128, 64, 32), the amount of computation needed is $\sum_{i=1}^{N} 4^i C$.

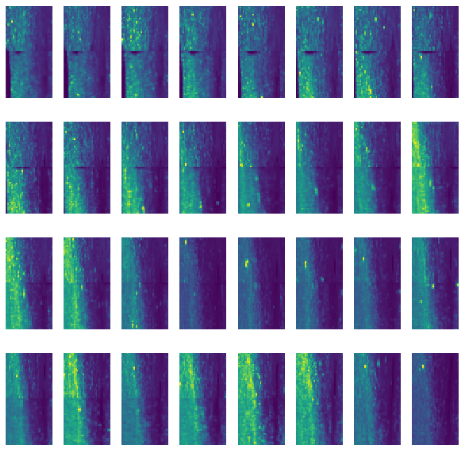
Assume that our algorithm always reduce the error from 10% (unit: 2^N) to 5% (unit: 2^N) on the most coarse scale, we have that, in the next scale, the initial error will be less than 10% (unit: 2^{N-1}) (because we downsample $2\times$ between two consecutive scales). If the assumption holds for each scale, the error on the finest scale will be less than 10% (unit: 2^0). To further validate this assumption on our model, we conducted quatitative experiment using Inverse Problem Method.

We present example from multiscale semantic stitching on real data in Fig.3. First we conducted semantic stitching on 1024×1024 scale, by querying our finetuned siamese networks for similarity. The querying method includes both grid search ans gradient descent. The grid search is useful when the initial deform is huge, so the gradient descent does not work, that is because the `grid_sampling` function in PyTorch only consider the nearest 4 pixels for deformation, which will lead no gradient if the initial deformation is very large.

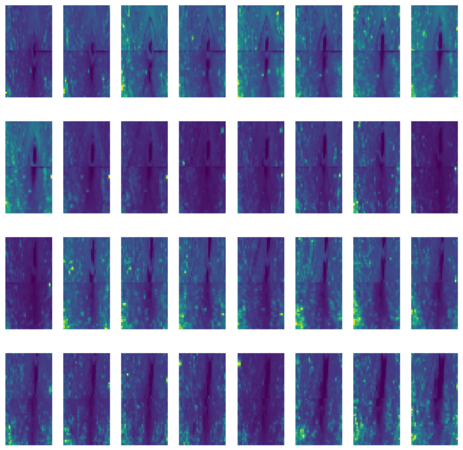
From the result (Fig.3) of multiscale semantic stitching, we could find that the fine scale could be stitched well, while the pixel stitching failed at 256×256 scale.



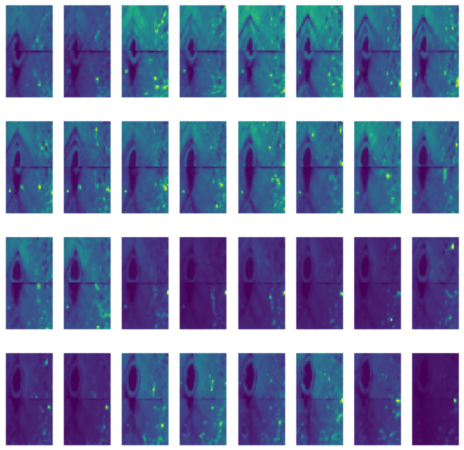
(a) semantic stitching result on 512 $\times$ 512 scale



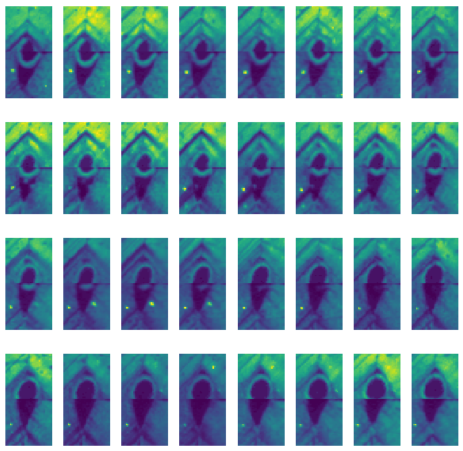
(b) semantic stitching result on 256 $\times$ 256 scale, initiated from 3a



(c) semantic stitching result on 128 $\times$ 128 scale, initiated from 3b

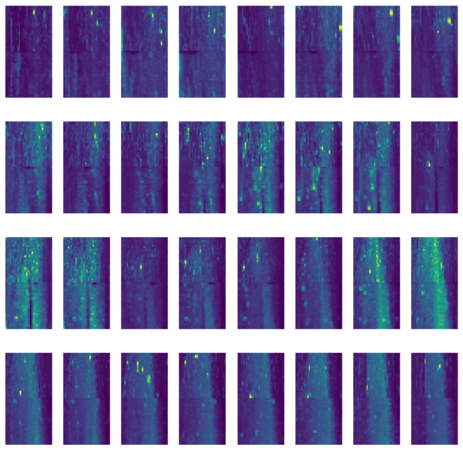


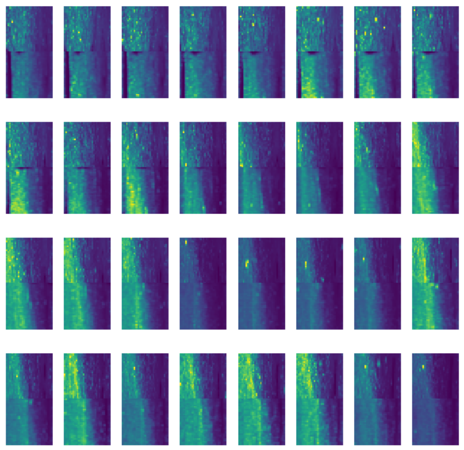
(d) semantic stitching result on 64 $\times$ 64 scale, initiated from 3c

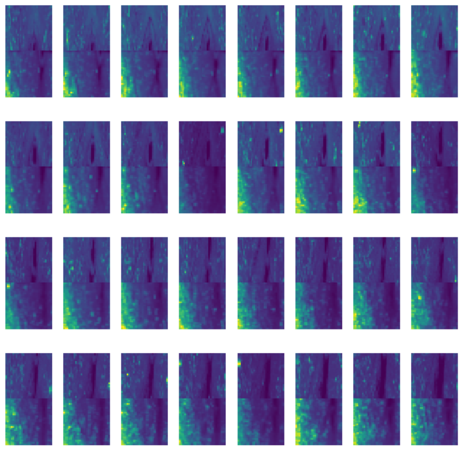


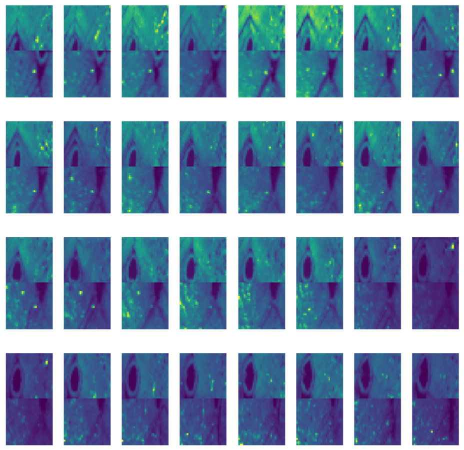
(e) semantic stitching result on 32 $\times$ 32 scale, initiated from 3d

Figure 3: Semantic stitching results on real data on multiple scales


(a) pixel stitching result on 512×512 scale


(b) pixel stitching result on 256×256 scale, initiated from 4a


(c) pixel stitching result on 128×128 scale, initiated from 4b


(d) pixel stitching result on 64×64 scale, initiated from 4c

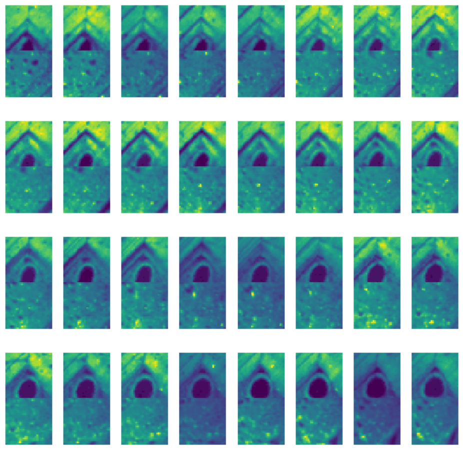

(e) pixel stitching result on 32×32 scale, initiated from 4d

Figure 4: Pixel stitching results on real data on multiple scales

5.6 Quantitative Evaluation

We found that different scales have different features, and the properties of the features have an vital influence of stitching performance. For example, the 1024×1024 scale and 512×512 scale, are easier for semantic stitching, however, for 32×32 scale, which is the finest, demonstrated strange behavior. The feature seems not to support our assumption about SSR (surface semantic representation) because the feature is not that continuous between two brain slices. For example, a structure could suddenly vanish when it just crossed the slicing position, in this case, we could never find the common semantic representations.

Here we present the benchmarking result with Inverse Problem Method. We compare 5 different models.

1. `trivial` is a trivial model, which directly patchify $4 \times 4 \times 4$ pixels as a feature token, the same with pretrained MAE Encoder.

2. `ft` is our main model, pretrained with MAE and finetuned with our proposed siamese network method.

3. `trivial_trainable` is also trivial, but it inherites from `trivial`, and was finetuned with the same loss function as the `ft` model.

4. `2d` is another trivial model, it only consider the last layer of upper slice and first layer of lower slice, then compute cosine similarity as loss function.

5. `pixel_ncc` is another trivial model, which only consider the last layer of upper slice and first layer of lower slice, then compute the similarity loss by using NCC loss.

We only consider rigid deformation, that is, translation and rotation. For translation, we consider t_x and t_y, for rotation we consider θ.

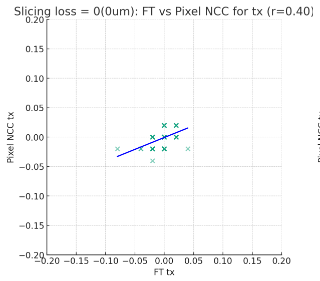
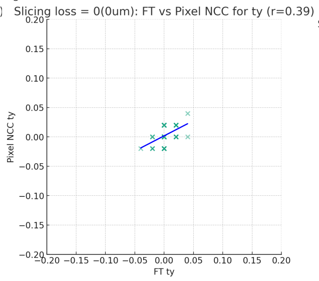
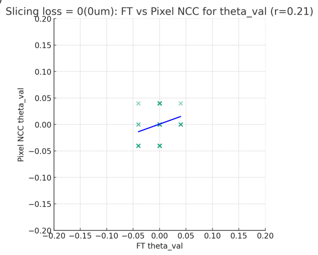
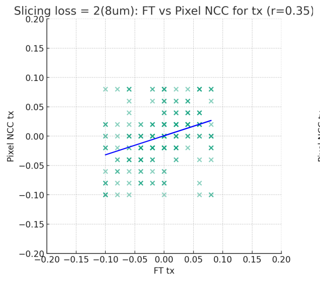
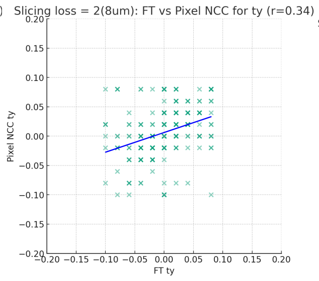
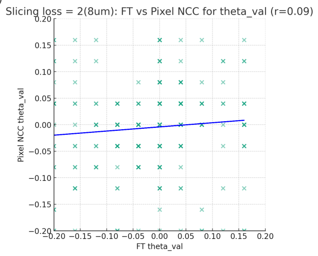
Table 2: Quantitative Evaluation of Siamese Network

Model	Slicing loss	Average error in t_x	Average error in t_y	Average error in θ
trivial	0	0.067	0.067	0.049
	2	0.064	0.061	0.052
trivial_trainable	0	0.028	0.028	0.049
	2	0.050	0.048	0.098
2d	0	0.006	0.005	0.018
	2	0.030	0.030	0.032
pixel_ncc	0	0.005	0.004	0.005
	2	0.029	0.028	0.051
ft	0	0.004	0.002	0.002
	2	0.033	0.028	0.060

5.7 Error Analysis

From the qualitative evaluation result, we found that `ft` might learned a function similar to trivial method `pixel_ncc`. To validate this conjecture, for a set of stitching task synthesized with Inverse Problem Method, we apply `ft` and `pixel_ncc` respectively, thus we have their errors, denoted by $(\epsilon_{\text{ft}}, \epsilon_{\text{pixel\_ncc}})_i$. We plot the scatter plot of $(\epsilon_{\text{ft}}, \epsilon_{\text{pixel\_ncc}})_i$, shown in Tab.3.

Table 3: Error Analysis

	t_x **error scatter plot**	t_y **error scatter plot**	θ **error scatter plot**
0um slicing loss			
8um slicing loss			

We notice that although the error of `pixel_ncc` has a positive correlation with `ft`, we have more to say about. If the conjecture is true, all the points should be located on the line $y = x$, however, it does not. So, we could only conclude that the behavior of `ft` is similar to `pixel_ncc` in some cases, but `ft` is not a wrapped version of `pixel_ncc`.

5.7.1 Scaling

As the model parameters scale up and with more compute, the downstream performance will increase as well, which is known as scaling law [5]. We believe that with more training data and more parameters of our foundation model, the performance will continue to improve. We leave the scaling effect for future work.

References

[1] David Capel. Image mosaicing. In *Image Mosaicing and super-resolution*, pages 47–79. Springer, 2004.

[2] Xiaokang Chen, Mingyu Ding, Xiaodi Wang, Ying Xin, Shentong Mo, Yunhao Wang, Shumin Han, Ping Luo, Gang Zeng, and Jingdong Wang. Context autoencoder for self-supervised representation learning. *International Journal of Computer Vision*, pages 1–16, 2023.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[4] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[5] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[8] Juhyuk Park, Ji Wang, Webster Guan, Lars A Gjesteby, Dylan Pollack, Lee Kamentsky, Nicholas B Evans, Jeff Stirman, Xinyi Gu, Chuanxi Zhao, et al. Integrated platform for multi-scale molecular imaging and phenotyping of the human brain. *bioRxiv*, pages 2022–03, 2022.

[9] Yair Poleg and Shmuel Peleg. Alignment and mosaicing of non-overlapping images. In *2012 IEEE International Conference on Computational Photography (ICCP)*, pages 1–8. IEEE, 2012.

[10] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[12] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[13] Fang Xu, Yan Shen, Lufeng Ding, Chao-Yu Yang, Heng Tan, Hao Wang, Qingyuan Zhu, Rui Xu, Fengyi Wu, Yanyang Xiao, et al. High-throughput mapping of a whole rhesus monkey brain at micrometer resolution. *Nature biotechnology*, 39(12):1521–1528, 2021.

A Generative Method: Image Mosaicing

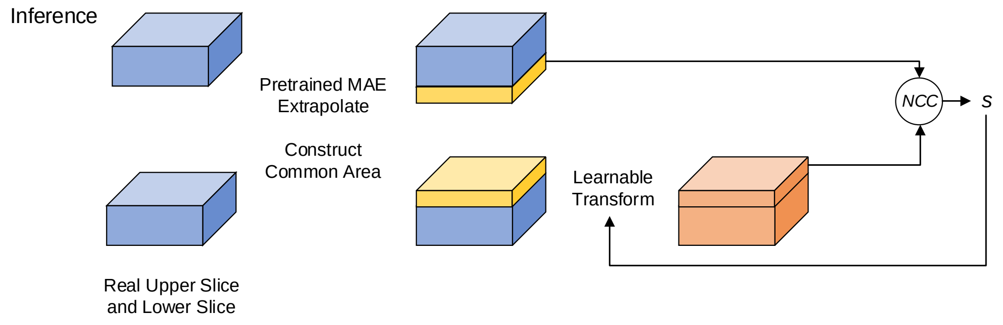


Figure 5: Inference pipeline for Image Mosaicing Method

A.1 Intuition

Image mosaicing [1], in its typical setting, is to stitch several 2D overlapping images to produce a panorama image. Besides its traditional cases, there are several attempts to perform image mosaicing on non-overlapping image [9]. It requires an extrapolation step. This step could be implemented by different methods, based on difference inductive bias of researchers. The original work uses an extrapolation method that is not learnable and demonstrated good results. In our experiments, we use extrapolation based on our pretrained masked autoencoder, here both encoder and decoder are utilized to do image completion.

The completion method is simple: for upper slice, an extrapolation is performed downward, and N extra layer will be completed by masked autoencoder, while for lower slice, an extrapolation is performed upward, N extra layer will be completed. The generalizability of masked autoencoder could to some extent play a vital role here: during training, random masking 75% of the input tokens by only encode 25% unmasked tokens, and during inference here, we encode all lower or upper slice, and add mask token for the N layer to be completed, then encoder will complete the masked token. We finally readout the masked tokens. An extra normalization is required in our experiment setting.

A.2 Implementation

The Image Mosaicing Stitching consists of two steps.

(i) For the extrapolation process, we set a parameter $N = 4k$, where $k \in \mathbb{N}^+$, indicating how many layers should the pretrained MAE extrapolate. Here 4 means the patch size is $4 \times 4 \times 4$, as the model process image by patches. For example, $N = 4$ indicates that we will program the MAE to predict one layer of patches and convert them into pixels, yielding $4 \times 32 \times 32$ predicted pixels (indicated as yellow blocks in the pipeline).

(ii) For the optimization process, we aim to find an optimal transform that could make upper and lower slice continuous. The objective is some loss function on the overlapping area of both extrapolated image. In our basic setting, we use a rigid transform with four parameters, including translation parameters t_x, t_y, t_z, and rotation parameters θ. Gradient Descent and Grid Search are both used in our implementation. For gradient descent, we use Adam [6] optimizer here.

A.3 Evaluation

To evaluate the performance of Image Mosaicing Stitching, we use Inverse Problem Method. We do not choose offset $= 0$, because offset $= 0$ is trivial. Normally we use offset $= 2$ for evaluation. For N, we choose $N = 8$ here, as $N = 4$ produces worse result. The basic pipeline is, randomly select an intact brain slice (Fig.6) from intra-slice stitching results. After that, randomly choose an area in the brain slice, and divide it into upper and lower slices, with a slicing gap of offset > 0 . After that, extrapolate the upper slice and lower slice separately, with a window size of $32 \times 32 \times 32$

on the original scale (without downsampling). After that, fix a window of upper slice, and find the optimal transform that could make fixed upper window and lower window semantically continuous, using eithor gradient descent or grid search. The error of t_x, t_y, t_z, and θ will be averaged for benchmarking.

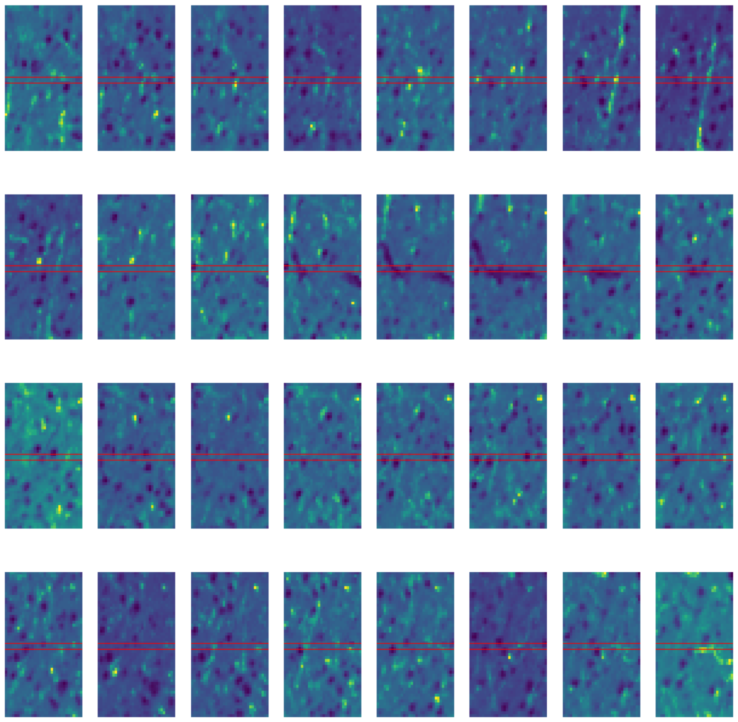
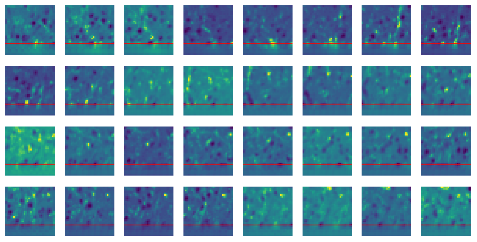
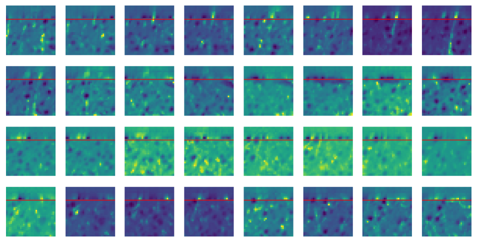


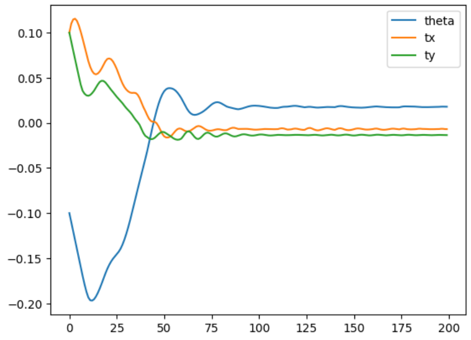
Figure 6: Randomly select a brain slice, divide it into upper and lower slice, with a gap of 3 layers (12 um) enclosed by red lines

After extrapolation (as illustrated in Fig.7a and Fig.7b), upper slice will have 8 extra layers (behind red line), and lower slice will have 8 extra layers (on top of red line).

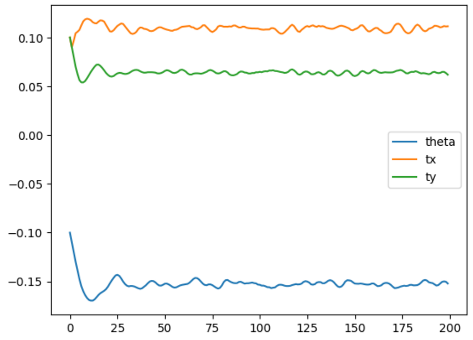
With gradient descent method, we need to add a random perturbation, here we use $t_x = -10\%$, $t_y = 10\%$, $t_z = 0$, and $\theta = -0.1$rad. We present the extrapolation result of an sample brain slice, with $N = 8$ and slicing gap of 3 layers (12 um). The semantic stitching method result (as shown in 7c) shows that all parameters converge to 0 within 100 iterations, while the trivial method could not converge to 0.


(a) Extrapolated upper slice (8 layers downward)


(b) Extrapolated lower slice (8 layers upward)



(c) Optimization process of mosaicing method with a perturbation of rigid transform ($t_x = -10\%$, $t_y = 10\%$, $\theta = -0.1\text{rad}$). All variables should converge to 0 if the method works well.



(d) Optimization process of trivial method with a perturbation of rigid transform ($t_x = -10\%$, $t_y = 10\%$, $\theta = -0.1\text{rad}$). All variables should converge to 0 if the method works well.

Figure 7: Evaluate Image Mosaicing Stitching with Inverse Problem Method