ChanSFormer: A Channel Agnostic Vision Transformer for Multi-Channel Cell Painting Images

Jingwei Zhang
Stony Brook University
jingwezhang@cs.stonybrook.edu

Srinivasan Sivanandan
Insitro
srinivasan@insitro.com

Dimitris Samaras
Stony Brook University
samaras@cs.stonybrook.edu

Abstract

High-content multi-channel imaging, from Cell Painting assays to remote sensing, is central to modern scientific pipelines. Because these channels are highly heterogeneous and experimental configurations frequently evolve, a practical vision backbone must be fundamentally channel-agnostic. Current channel-adaptive vision transformers rely on global self-attention with learnable channel embeddings, which can be computed only for previously known channels. Attempts to overcome this limitation of new channels either discard the embeddings, losing channel identity, or process each channel independently, thereby eliminating critical cross-channel interactions. To address this, we introduce **ChanSFormer**, an efficient Vision Transformer that replaces rigid embeddings with disentangled spatial-channel attention and individual channel CLS tokens. This architecture natively preserves both distinct channel identities and cross-channel information flow. Furthermore, the individual channel class token design unlocks more informative feature representations for self-supervised learning and enables representation-based channel sampling while remaining channel-agnostic. We evaluate ChanSFormer on two biology multi-channel datasets, CHAMMI, JUMP-CP and a satellite dataset, So2Sat. Experimental results show that ChanSFormer outperforms state-of-the-art methods by 3.93%–4.27% on JUMP-CP in classification accuracy. It also demonstrates exceptional cross-dataset transferability under a self-supervised setting, outperforming previous methods by up to 2.72% in accuracy in the same classification tasks. Furthermore, the disentangled attention reduces the quadratic complexity of channel $\times$ spatial sequence length of the global attention baseline, thus improving throughput by 55%–313%. Code is available at https://github.com/cvlab-stonybrook/ChanSFormer.

1 Introduction

High-content imaging using the Cell Painting assay is a cornerstone of modern drug discovery [1–3]. This technique generates images with multiple channels, typically five or more, where each channel uses a specific fluorescent dye to reveal a distinct cellular component such as the nucleus, mitochondria, or cytoskeleton. Because every channel carries a different layer of biological information, the resulting data is highly heterogeneous across channels [4]. Compounding this, research pipelines often evolve through experiments that use varying numbers of channels or introduce novel fluorescent markers. A practical vision backbone for this setting must therefore be flexible and channel-agnostic, i.e., invariant to the number and ordering of input channels.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Vision Transformers (ViTs) [5] help analyze these complex images, but they still have their limitations. Given the heterogeneous nature of the channels, standard ViTs on cell images tokenize each channel independently [6–9] and then employ a single, global self-attention mechanism that processes the concatenated tokens from all channels. To distinguish between tokens originating from different channels within this shared computational space, a learnable channel embedding is added to each token [6, 8]. Although this solves the channel-identity problem, it introduces a critical limitation: the model becomes channel-specific, hard-coded to the initial channel configuration and unable to adapt to experimental setups with potential new channels.

Some methods [9, 10] try to address this channel-specificity by modifying the transformer architecture. However, they either drop the channel embedding [9], so that global self-attention can no longer distinguish tokens from different channels, or embed each channel independently [10], so that the backbone never models cross-channel interactions.

Another limitation of the global-attention template is computational. Concatenating tokens from all channels dramatically increases the input sequence length: a 224×224 image with 8 channels yields 8 times more tokens than a natural RGB image at the same resolution. This exacerbates the quadratic complexity of self-attention and slows both training and inference [6].

To address these limitations, we introduce **ChanSFormer**, a Vision Transformer designed from the outset to be fundamentally channel-agnostic. Our main contributions are:

- **Disentangled channel-spatial attention.** We replace monolithic global attention with separate channel attention and spatial attention. Channel attention natively preserves cross-channel interactions without requiring hard-coded channel embeddings.
- **Individual channel CLS tokens.** We introduce individual channel CLS tokens that dynamically carry distinct channel information by copying a single learnable CLS token to each channel before the first layer and then allowing the tokens to evolve in later layers through the disentangled attention. This individual token design also unlocks a more informative representation for self-supervised learning.
- **Higher throughput.** The disentangled attention reduces the quadratic complexity of channel $\times$ spatial sequence length in the global attention to the quadratic complexity of channel length plus the quadratic complexity of spatial length for faster training and inference.

Extensive experiments in both supervised and self-supervised (DINOv2 [11]) regimes validate our approach on two biology multi-channel datasets, CHAMMI, JUMP-CP and a satellite dataset, So2Sat. ChanSFormer achieves comparable or superior accuracy compared to the state-of-the-art (SOTA) methods [8, 6] in the supervised classification, outperforming them by 3.93%–4.27% on JUMP-CP in classification accuracy. Moreover, it demonstrates exceptional cross-dataset transferability under a self-supervised protocol, outperforming SOTA methods by up to 2.72% in accuracy in the same classification tasks. Finally, the disentangled attention accelerates throughput by 55%–313% compared to the global-attention baseline.

2 Related work

Channel-adaptive models for multi-channel imaging. Multi-channel microscopy and remote-sensing data have channels that differ in physical meaning across experiments, so a model that commits to a fixed channel ordering at input time is of limited use. The CHAMMI benchmark [12] framed this problem for cell imaging and introduced ConvNeXt or ViT backed channel-adaptive baselines, DepthwiseViT, TemplateViT [13, 14], and HyperViT [15], that each replace only the first layer to accept variable channel counts while sharing the rest of the backbone. Vision Transformer approaches keep the backbone unified throughout the depth: ChannelViT [6] tokenizes each channel with a shared linear projection, adds a learnable *channel embedding* to every token, and runs global self-attention over the concatenated sequence. It also introduces Hierarchical Channel Sampling (HCS) that samples channels uniformly at random during training, to improve its channel robustness. ChAda-ViT [7] follows the same global-attention template while padding inputs to a fixed channel count; DiChaViT [8] extends this design with a diverse channel sampling strategy and auxiliary losses that push the learned channel embeddings apart. These ViT-based methods share a single architectural choice: a global self-attention that relies on the channel embedding to distinguish tokens from different channels. That choice ties the trained weights to a specific channel configuration.

Self-supervised pretraining for multichannel imaging. Self-supervised pretraining for natural images has converged on joint-embedding frameworks such as DINO [16], iBOT [17], and DINOv2 [11], and on masked-image modelling as in MAE [18]. Extending these frameworks to multichannel imaging has followed several directions. Cell-DINO [19] applies DINOv2 to cellular microscopy with channel-aware augmentations but treats the channel stack as fixed. ChA-MAEViT [20] pairs masked autoencoding with a channel-aware multi-channel ViT, targeting cross-channel learning through the MAE objective. CA-MAE [9] also uses masked autoencoding but removes the channel embedding, giving an architecturally flexible ViT that accepts any channel subset at inference. BoC [10] pretrains a microscopy foundation model at scale with DINOv2 by running the backbone on each channel independently. IC-ViT [21] takes a two-stage route that uses masked-image pretraining on single channels followed by multi-channel finetuning, so cross-channel interactions are learned only at finetuning time.

To achieve channel agnosticism, prior methods adopt two opposing extremes. CA-MAE [9] removes channel embeddings while retaining global attention, severely compromising the model's ability to distinguish distinct channels. Conversely, BoC [10] processes each channel entirely independently, preserving channel identity but eliminating critical cross-channel information flow during feature extraction. ChanSFormer bridges this gap: by employing decomposed channel-spatial attention alongside individual channel CLS tokens, our architecture natively preserves both continuous cross-channel interactions and distinct channel identities without relying on fixed channel embeddings. Our decomposition is related to factorized attention in video [22] and multi-view [23] models, which encode each frame or view with shared transformer blocks and interleave attention across frames or views. Unlike frames, which are ordered in time, or views, which are related by camera geometry, microscopy channels are pixel-aligned but unordered; ChanSFormer therefore encodes no channel ordering, making the encoder permutation-equivariant along the channel axis (Appendix I).

3 Method

We present ChanSFormer, a Vision Transformer [5] designed to be channel-agnostic for multichannel imagery while remaining efficient and accurate. The model operates on an input $X \in \mathbb{R}^{C\times H\times W}$, where H and W are the spatial dimensions and C is the number of channels.

3.1 Overall architecture

Figure 1 (a) gives an overview, each of the C input channels is independently processed by a shared patch-embedding layer $\Phi : \mathbb{R}^{P\times P} \to \mathbb{R}^D$ with patch size P and embedding dimension D, producing $N = (H/P)(W/P)$ spatial tokens per channel. A spatial positional embedding in $\mathbb{R}^{N\times D}$ is added, which is *shared* across channels. Individual channel class (CLS) tokens (Section 3.2) are prepended to each channel's tokens, producing the encoder input $z^{(0)} \in \mathbb{R}^{C\times(N+1)\times D}$.

This encoder input $z^{(0)}$ is then fed into a stack of L ChanSFormer encoders, each of which decomposes the global self-attention of a standard ViT encoder [5] into a channel and a spatial attention (Section 3.3). The output of the stack is $z^{(L)} \in \mathbb{R}^{C\times(N+1)\times D}$; we extract its C channel CLS tokens $\{\mathrm{CLS}_c^{(L)}\}_{c=1}^{C}$ as the per-channel representation. A dedicated channel sampling mechanism (Section 3.4) varies the number and identity of input channels during supervised training.

Two output heads consume $\{\mathrm{CLS}_c^{(L)}\}$ depending on the training objective.

Supervised aggregation. As shown in Figure 1(b), for supervised classification tasks, we aggregate the channel CLS tokens into a single image token z_{img} via a query-based attention pooling:

$$z_{\text{img}} = \mathrm{softmax}\Big(q\,K^{\text{pool}}\big(\mathrm{CLS}^{(L)}\big)^{\top}/\sqrt{D}\Big)\,V^{\text{pool}}\big(\mathrm{CLS}^{(L)}\big), \tag{1}$$

where $q \in \mathbb{R}^D$ is a single learnable query, K^{pool} and V^{pool} are learnable linear projections from $\mathbb{R}^D$ to $\mathbb{R}^D$, and $\mathrm{CLS}^{(L)} \in \mathbb{R}^{C\times D}$ stacks the channel CLS tokens produced by the last encoder. A single query keeps the pool cost linear in C.

Self-supervised adaptation. As shown in Figure 1(c), for DINOv2 [11] self supervised pretraining, we do not use an aggregator but feed all the channel CLS tokens directly to the student and teacher heads, treating $\{\mathrm{CLS}_c^{(L)}\}_{c=1}^{C}$ as C independent views of the input. The remaining patch tokens are

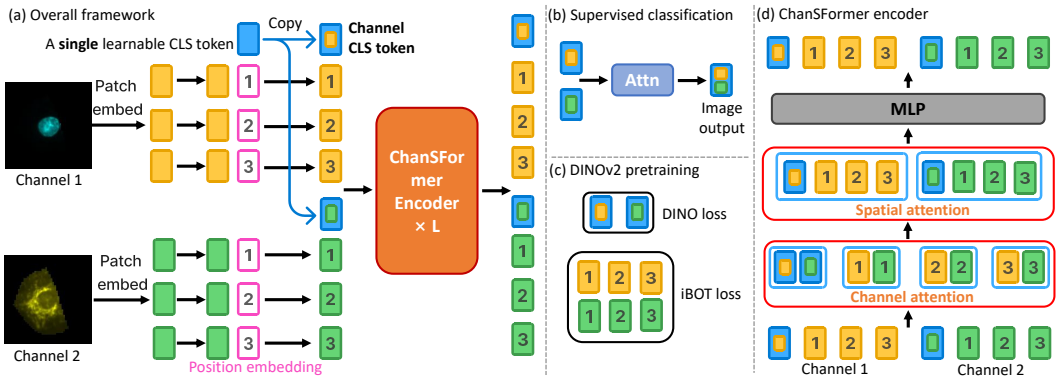


Figure 1: **(a) Overall framework of ChanSFormer**. Each channel of an input image is independently projected by a shared patch embedding; a shared spatial positional embedding is added, and individual channel class (CLS) tokens, copied from a single learnable CLS token, are concatenated. The resulting tokens pass through L ChanSFormer encoders. **(b) Supervised classification**. The channel CLS tokens are aggregated via attention into an image-level representation. **(c) Self-supervised DINOv2 pretraining**. The channel CLS tokens are treated as distinct samples to generate the DINO loss [11], while the remaining patch tokens generate the iBOT [17] loss. **(d) Architecture of the ChanSFormer encoder**. A single global attention block is replaced by channel attention (across channels, per spatial position) followed by spatial attention (across spatial positions, per channel).

used to generate the iBOT [17] loss. This setup is closely related to BoC [10], where each channel is independently embedded, except that our ChanSFormer encoder still allows cross-channel information exchange inside the backbone. Using C channel CLS tokens rather than a single aggregated token also yields more supervision per image for the self-distillation objective, increasing the effective batch size and the diversity within the batch.

3.2 Individual channel class tokens

The first important design of ChanSFormer is the individual channel class token. In standard ViT, a class token is used to aggregate information across spatial positions. For multi-channel images, ideally each channel should have its own learnable class token. However, to keep the model channel-agnostic, we instantiate a single learnable CLS token $\mathrm{CLS} \in \mathbb{R}^D$ and *copy* it into C channels at the input, so that $\mathrm{CLS}_c^{(0)} = \mathrm{CLS}$ for all c. The tokens are initially identical, but because each CLS_c interacts only with its own channel's spatial tokens during spatial attention (Section 3.3), they evolve through the stack and come to represent features of one channel each.

This departs from the convention in TimeSformer [22], which pools or averages a single global CLS across time frames. Averaging is appropriate when slices carry highly correlated information (e.g., frames of a video or RGB channels) but is lossy when each slice is semantically distinct, as in multichannel fluorescence microscopy where different channels image different cellular components. Keeping one CLS per channel preserves the channel-specific information for the downstream heads described in Section 3.1.

Apart from its primary role as a channel-agnostic aggregator, the channel CLS token has two other important uses. First, in self-supervised learning, as mentioned in Section 3.1, it serves as an independent image view for each channel, effectively increasing the batch size and within-batch diversity. Second, in supervised learning, it allows us to sample channels according to the relationships among these channel CLS tokens, which we detail in Section 3.4 to drive channel sampling. This sampling role is analogous to the learnable channel embedding used in channel-specific ViTs [8].

3.3 Disentangled channel-spatial attention

As shown in Figure 1 (d), each ChanSFormer encoder ℓ replaces the multi-head self-attention block in a standard ViT [5] with two attentions applied sequentially: first a *channel attention*, then a *spatial*

attention. The rest of the block (MLP, layer normalization, residual structure) remain unchanged. Let $z^{(\ell-1)} \in \mathbb{R}^{C\times(N+1)\times D}$ be the input.

Channel attention. At each spatial position $s \in \{1, \dots, N+1\}$, we attend across the C channel tokens. Let $z_s^{(\ell-1)} \in \mathbb{R}^{C\times D}$ denote the tokens at position s, the channel-attention is

$$z'_s = z_s^{(\ell-1)} + \mathrm{CAttn}\Big(\mathcal{LN}\big(z_s^{(\ell-1)}\big)\Big), \qquad \mathrm{CAttn}(x) = \mathrm{softmax}\left(\frac{Q'(x)\,K'(x)^\top}{\sqrt{D}}\right) V'(x), \quad (2)$$

with Q', K', V' learnable linear projections and $\mathcal{LN}$ denoting layer normalization. Information crosses the channel axis but not the spatial axis at this step.

Spatial attention. Then, for each channel $c \in \{1, \dots, C\}$, we attend across the $(N+1)$ spatial tokens of that channel. Let $z'_c \in \mathbb{R}^{(N+1)\times D}$ denote the tokens for channel c, the spatial-attention is

$$z''_c = z'_c + \mathrm{SAttn}\big(\mathcal{LN}\big(z'_c\big)\big), \qquad \mathrm{SAttn}(x) = \mathrm{softmax}\left(\frac{Q(x)\,K(x)^\top}{\sqrt{D}}\right) V(x), \quad (3)$$

with Q, K, V learnable linear projections from $\mathbb{R}^D$ to $\mathbb{R}^D$ shared across all channels. Compared with a global self-attention over CN tokens, one ChanSFormer encoder costs $O(CN^2 + NC^2)$ instead of $O((CN)^2)$. The saving is pronounced when N is large (small patch size) or C is moderate, the regime typical of multichannel microscopy.

Hardware-aware channel attention optimization. Channel attention operates in a regime that breaks the assumptions of standard attention implementations. Modern attention algorithms, including Memory Efficient Attention [24], assume a high *hardware occupancy* originating from long input sequence length, such as N in spatial attention or CN in global attention. However, channel attention operates over a sequence length of C (typically < 16), which is too short to saturate NVIDIA TensorCores for matrix multiplication. Meanwhile, existing algorithms require the sequence length dimension at a fixed location. Thus, channel attention must conduct excessive tensor permutations to meet this requirement. As a result, when applied trivially to channel attention, existing algorithms will suffer from degraded throughput due to ***low hardware occupancy*** and ***excessive memory operations***.

To resolve these problems, we fuse the entire channel attention computation into a single Triton kernel and batch multiple short instances together, so that the memory overheads are amortized across hundreds of attention operations rather than paid per instance. Furthermore, our attention kernel adapts its internal computation to the channel size C. At small sizes that are unable to saturate TensorCores, we fall back to CUDA cores for a higher throughput (see Table 4); at larger sizes, we engage the TensorCores for matrix multiplication, as in regular attention algorithms. These novel design choices place our kernel in a complementary regime to FlashAttention [25, 26]: where FlashAttention optimizes a moderate number of long sequences, ours optimizes an excessive number of short sequences.

3.4 Representation based channel sampling

Randomly sampling the input channel set during training is an effective regularizer for channel-adaptive ViTs [6, 8], improving models' robustness. DiChaViT [8] demonstrated that guiding this sampling using the pairwise distances among learned channel embeddings outperforms uniform random sampling (HCS [6]). However, this strategy tightly couples the sampler to the channel embedding, the very component we removed to make ChanSFormer channel-agnostic.

We therefore need a sampling signal that does not depend on a fixed, input-time lookup. The channel CLS tokens (Section 3.2) naturally fill this role: as they process data, they accumulate channel-specific representations. To establish a stable, dataset-level metric of channel relationships, we maintain an exponential moving average (EMA) matrix $S \in \mathbb{R}^{C\times C}$ computed from the channel CLS tokens at a designated backbone layer $\ell^\star$:

$$S \leftarrow m\,S + (1-m)\,\mathrm{sim}\Big(\mathrm{CLS}^{(\ell^\star)}\Big), \quad (4)$$

where $\mathrm{sim}(\cdot)$ denotes the pairwise cosine similarity function, and m is a momentum close to 1 so that S reflects stable relations across many batches. The update happens inside the forward method and is decoupled from any downstream loss.

During training, we draw a subset size $C' \sim \mathrm{Uniform}\{1, \ldots, C\}$, and use the relation matrix S, scaled by a temperature T, to probabilistically guide the selection of the C' channels, biasing the sampler away from purely uniform draws. This allows us to retain the sampling benefits of representation-based subset sampling while remaining channel-agnostic. Channel sampling is disabled at inference.

4 Experiments

We evaluate ChanSFormer in supervised classification and self-supervised pretraining regimes on two multi-channel cell benchmarks, CHAMMI, JUMP-CP and a multi-channel satellite benchmark, So2Sat. Because our decomposed spatial-channel attention introduces additional weights in channel attention, we ensure a fair comparison against all small-scale baselines by reducing our backbone's embedding dimension from standard 384 to 320 and the number of attention heads to 5. This adjustment keeps ChanSFormer's total parameter count slightly below the baseline models.

4.1 Datasets and metrics

CHAMMI [12] is a multichannel microscopy benchmark comprising roughly 220K images drawn from three sources: the Allen/WTC-11 cell dataset (3 channels), the Human Protein Atlas (HPA, 4 channels), and Cell Painting (CP, 5 channels). To evaluate domain generalization, we follow the established protocol [8] by extracting feature representations and using a 1-Nearest Neighbour classifier to compute the macro-average F1-score for each task. Following [8], we report the *averaged score* of the metric from the WTC and HPA subsets.

JUMP-CP [27] features single-cell images subjected to diverse chemical and genetic perturbations. Our evaluation isolates the compound perturbation plate BR00116991 [28], encompassing 161 classes (160 specific treatments plus one control). Each image contains 8 channels in total: 5 capturing fluorescence and 3 capturing brightfield information. Following [6, 8], we use the official train/val/test split and report the top-1 classification accuracy both on the full 8-channel input and on the 5-channel fluorescence-only subset.

So2Sat [29] consists of 32×32 satellite image patches categorized into 17 distinct local climate zones. The data spans 18 channels per patch, combining 8 Sentinel-1 synthetic aperture radar (SAR) channels and 10 Sentinel-2 multispectral channels. Adopting the city-split partition, our setup uses 352K images for training, 24K for validation and 24K for testing. We report the top-1 classification accuracy under two settings [8]: *Full* (all 18 channels) and *Sen-1* (the 8 Sentinel-1 radar channels).

4.2 Implementation details

For supervised tasks, we follow the training recipe of DiChaViT [8]. For self-supervised pretraining, we utilize DINOv2 [11] combined with the iBOT [17] objective, where we do not use KoLeo regularizer and we use a low weight (0.6) on iBOT loss to stabilize training on small-sized models. We use PyTorch and train all models from scratch. Model training is conducted on distinct hardware setups based on dataset scale: JUMP-CP on $4\times$ RTX PRO 6000 GPUs, So2Sat on $4\times$ A100 GPUs, and CHAMMI on a single Quadro RTX 8000 GPU. Throughput is evaluated on a Quadro RTX 8000. Detailed hyper-parameters are listed in the Appendix.

4.3 Results

4.3.1 Supervised tasks

We first evaluate ChanSFormer on supervised classification tasks across the three datasets. We compare our approach against a comprehensive suite of channel-adaptive baselines: HyperViT [15], DepthwiseViT [12], TemplateViT [13, 14], ChAda-ViT [7], ChannelViT [6], and DiChaViT [8]. As detailed in Table 1, ChanSFormer establishes a new SOTA on both JUMP-CP and So2Sat. On JUMP-CP, our model surpasses DiChaViT by 3.93% to 4.27%. It demonstrates similarly consistent improvements on So2Sat (1.40% to 1.99%). Notably, these performance gains are achieved using a more compact backbone ($20.1\mathrm{M}$ parameters compared to the $21.5\mathrm{M}$ of recent ViT baselines), proving our architectural efficiency. On the CHAMMI benchmark, we perform marginally behind DiChaViT by 0.85%. This is expected as we omit its embedding-dependent Channel Diversification

Table 1: **Supervised classification on three multichannel benchmarks.** For JUMP-CP and So2Sat, *Full* evaluates on all channels, *Flu.* evaluates on the fluorescence channels for JUMP-CP and *Sen-1* evaluates on the Sentinel-1 channels for So2Sat. The highest number is bold, the second highest is underlined. #Params is the number of parameters. The reported numbers are the mean of 3 runs.

		CHAMMI	JUMP-CP		So2Sat	
Method	#Params	Avg	Full	Flu.	Full	Sen-1
HyperViT [15]	24.7M	54.54	47.07	42.43	60.73	41.88
DepthwiseViT [12]	22.6M	60.94	49.86	44.98	60.41	43.41
TemplateViT [13, 14]	21.5M	57.02	52.48	43.85	55.86	37.28
ChAda-ViT [7]	21.5M	63.88	65.03	42.15	56.98	12.38
ChannelViT [6]	21.5M	64.90	67.51	56.49	62.75	47.70
DiChaViT [8]	21.5M	**69.68**	<u>69.19</u>	<u>57.98</u>	<u>63.36</u>	<u>47.76</u>
ChanSFormer	20.1M	<u>68.83</u>	**73.46**	**61.91**	**64.76**	**49.75**

Loss, without which DiChaViT's score drops to 68.07% [8], falling below ChanSFormer's 68.83%. This minor performance trade-off is well worth the architectural flexibility gained for seamlessly handling varying channel configurations.

4.3.2 Self-supervised tasks

Besides its superior performance in supervised tasks, we evaluate ChanSFormer's representations in a self-supervised regime. We compare DINOv2 pretrained our approach against two baselines: BoC [10], which achieves channel-agnosticism by processing channels independently, and ChannelViT [6], which models cross-channel interactions but relies on channel embeddings. So2Sat is excluded from the pretraining because of its small resolution. To allow dynamic adaptation to varying number of channels, we randomly sample 2 channels for each image during training. As shown in Table 2, ChanSFormer outperforms both baselines in nearly all settings. For in-domain linear probing, our model surpasses BoC by 1.32% on CHAMMI and 2.91% on JUMP-CP. The advantage largely persists during cross-dataset transfer, where ChanSFormer leads BoC by up to 2.72%. On the other hand, ChannelViT suffers a performance degradation during cross-dataset transfer, lagging behind ChanSFormer by up to 16.88%. This collapse occurs because ChannelViT's learned representations are tied to its pretraining channel configuration, forcing it to zero-pad the embeddings for unseen channels in the target dataset. This gap is further exacerbated by a train-eval mismatch, as we sample only two channels in training but use full channels in evaluation. These phenomena demonstrate ChanSFormer's strong generalization to novel channel sets because of its channel-agnostic design.

Table 2: **Self-supervised linear-probe transfer** across multichannel datasets. Models are pretrained with DINOv2 [11] on the dataset listed in the *Pretrain* column, then evaluated by linear probe on the target dataset. *Flu.* and *Sen-1* represent the fluorescence and Sentinel-1 channels respectively. For ChannelViT, the channel embeddings for unseen channels are zero-padded. † marks evaluations involving channels absent from SSL pretraining.

		CHAMMI	JUMP-CP		So2Sat	
Pretrain	Method	Avg	Full	Flu.	Full	Sen-1
CHAMMI	BoC [10]	48.92	39.36†	36.42†	67.50†	**46.46**†
	ChannelViT [6]	44.29	23.94†	24.71†	66.65†	44.54†
	ChanSFormer	**50.24**	**40.82**†	**39.14**†	**68.08**†	46.23†
JUMP-CP **Flu.**	BoC [10]	40.85†	46.00†	42.24	65.47†	45.28†
	ChannelViT [6]	32.88†	30.72†	32.97	65.00†	44.47†
	ChanSFormer	**42.86**†	**46.13**†	**45.15**	**66.33**†	**45.34**†

4.3.3 Computational efficiency

Model-level throughput. Table 3 demonstrates the throughput improvement yielded by ChanSFormer's overall architecture. By disentangling the $O((CN)^2)$ global attention into sequential

Table 3: **Model-level throughput (images/sec).** Higher is better. Benchmarked on 8 channel 224×224 inputs. ChanSFormer speeds up both training and inference over global attention. Naive is naive attention as in Table 4.

Mode	Model	Patch 16	Patch 8
Inference	ChannelViT	72.2	7.7
	ChanSFormer (naive)	106.7	22.0
	ChanSFormer	**112.0**	**22.9**
Training	ChannelViT	16.3	1.5
	ChanSFormer (naive)	30.8	5.8
	ChanSFormer	**33.9**	**6.2**

Table 4: **Speed and GPU memory for channel attention.** Benchmarked on input tensors of shape $(B, C, N+1, D) = (64, 8, 197, 384)$. Lower is better. Our optimized attention reduces execution time and GPU memory footprint compared to standard implementations.

	Inference		Training	
Method	Speed	Memory	Speed	Memory
Naive attention	7.1 ms	1029 MB	26.3 ms	1117 MB
Mem-efficient	6.0 ms	797 MB	18.6 ms	2283 MB
Ours	**4.9** ms	**634** MB	**12.1** ms	**877** MB

channel and spatial attention with $O(CN^2 + NC^2)$ complexity, ChanSFormer alleviates the primary computational bottleneck. This reduction alone, measured with naive channel attention, accelerates inference by 48%–186% and training by 89%–286% over ChannelViT. Our optimized kernel, which amortizes the fixed overheads of short channel sequences, adds a further 4%–10% over the naive implementation. Together, they increase the inference throughput of the full model by 55% at patch size 16 and 197% at patch size 8, and the training throughput by 108% to 313%. The performance advantage widens drastically as the spatial sequence length N grows, matching the predicted scaling for multichannel processing.

Hardware-aware channel attention. Table 4 isolates the performance of our optimized channel attention kernel with an input of 8 channels ($C = 8$). Standard hardware-aware implementations, such as memory-efficient attention [24], are heavily optimized for long sequences. They struggle with our extremely short channel sequence regime, which inflates their training workspace memory to $2283\,\mathrm{MB}$ due to the fixed overheads discussed in Section 3.3. Our optimized kernel sidesteps these issues by fusing operations and adapting to small tile sizes. It reduces the running time during the full training step by 35% to 54%, while simultaneously reducing the memory footprint by 20% to 62%.

4.3.4 Robustness to missing and novel channels

Following [6, 8], we further evaluate the model's robustness by testing its supervised classification performance on partial channels of the JUMP-CP dataset. We systematically drop channels and report the average accuracy across all $\binom{8}{k}$ combinations of k available channels. As shown in Table 5, ChanSFormer outperforms the baselines whenever at least two channels are available, leading DiChaViT by 0.20% to 3.70% with 2–7 channels, highlighting the robustness of our channel-agnostic design to missing channels. This setting removes subsets of channels that were observed during training, and therefore measures robustness to missing channels rather than generalization to channels with unseen semantics.

Table 5: **Test accuracy under partial channels of JUMP-CP.** Each column reports the mean ± std over all $\binom{8}{k}$ combinations of k available channels out of 8 [6, 8]. ChanSFormer achieves comparable or better performance in all 8 cases.

	Number of channels for evaluation (k)							
Method	8	7	6	5	4	3	2	1
ChannelViT [6]	67.51	$60.36_{\pm 9.1}$	$52.74_{\pm 12.2}$	$44.89_{\pm 13.2}$	$36.88_{\pm 12.3}$	$29.36_{\pm 9.3}$	$23.70_{\pm 5.0}$	$20.78_{\pm 1.6}$
DiChaViT [8]	69.19	$61.91_{\pm 9.3}$	$54.49_{\pm 12.4}$	$46.35_{\pm 13.4}$	$38.00_{\pm 12.4}$	$30.09_{\pm 9.3}$	$23.97_{\pm 4.9}$	$20.90_{\pm 1.6}$
ChanSFormer	**73.46**	$\mathbf{65.61}_{\pm 10.0}$	$\mathbf{57.32}_{\pm 12.8}$	$\mathbf{48.52}_{\pm 13.7}$	$\mathbf{39.50}_{\pm 12.5}$	$\mathbf{30.92}_{\pm 9.2}$	$\mathbf{24.17}_{\pm 4.6}$	$20.65_{\pm 1.3}$

We next evaluate the generalization of ChanSFormer to novel channels in the self-supervised setting. Unlike the partial-channel setting above, which only removes channels observed during training, the cross-dataset transfer in Table 2, following CA-MAE [9] and BoC [10], already evaluates transfer to channels whose semantics are absent from pretraining (marked by †). In addition, we conduct an experiment on CHAMMI in which either HPA or Allen is excluded from SSL pretraining, using the

Table 6: **Leave-one-source-out self-supervised transfer.** Models are pretrained with DINOv2 [11] on CHAMMI with one source held out, then evaluated by linear probe (JUMP-CP) or the official CHAMMI protocol. All evaluations in this table involve channels absent from SSL pretraining. Bold denotes the best result under each pretraining source.

		CHAMMI	JUMP-CP	
Pretrain	Method	Avg	Full	Flu.
CHAMMI: Allen + CP (HPA held out)	BoC [10]	35.72	19.63	19.65
	ChannelViT [6]	33.92	19.54	19.53
	ChanSFormer	**46.00**	**38.38**	**37.25**
CHAMMI: HPA + CP (Allen held out)	BoC [10]	**46.76**	37.75	35.67
	ChannelViT [6]	40.93	19.62	19.62
	ChanSFormer	44.38	**38.58**	**37.58**

Table 7: **Ablation on channel embeddings and cross-channel interactions.** Channel embeddings are not critical for ChanSFormer compared with ChannelViT with global attention, and cross-channel interactions prove essential. *Flu.* and *Sen-1* represents the fluorescence and Sentinel-1 channels respectively.

	CHAMMI	JUMP-CP		So2Sat	
Method	Avg	Full	Flu.	Full	Sen-1
ChanSFormer	68.83	73.46	61.91	64.76	49.75
w. channel embedding	69.03	73.78	62.36	64.78	49.55
ChannelViT [6]	64.90	67.51	56.49	62.75	47.70
w/o channel embedding (CA-MAE [9])	50.37	61.93	51.56	62.58	46.73
Spatial-only (BoC [10] + CLS channel attn)	61.73	41.56	39.10	63.41	47.70

same pretraining recipe as in Section 4.3.2. Every evaluation in this setting therefore involves channels that are absent from pretraining. As shown in Table 6, ChanSFormer outperforms both baselines in five of the six settings and remains far ahead of ChannelViT, whose zero-padded embeddings for unseen channels leave it near 19.5% on JUMP-CP. Thus, the advantage of ChanSFormer extends beyond robustness to missing channels to the novel-channel transfer regime that motivates this work.

4.3.5 Ablation studies

Ablation on channel embeddings. Table 7 shows the reliance of different architectures to fixed channel embeddings. Adding channel embeddings to ChanSFormer yields only marginal improvements. In stark contrast, removing them from ChannelViT causes a catastrophic performance degradation (e.g., a 14.53 drop on CHAMMI). Note that this embedding-free ChannelViT represents the architectural backbone of CA-MAE [9], an existing channel-agnostic method. These results demonstrate that our decomposed attention and individual CLS tokens natively preserve channel identities, rendering hard-coded embeddings obsolete.

Ablation on cross-channel interactions. We further ablate the necessity of modeling cross-channel correlations. The "Spatial-only" baseline in Table 7 processes spatial patches independently, effectively adapting the channel-agnostic approach of BoC [10] to supervised tasks by utilizing the same query-based attention pooling described in Section 3.1. The severe performance degradation observed across all datasets confirms that late-stage feature aggregation is insufficient for complex multi-channel data. These results demonstrate that cross-channel correlations are vital for robust modeling of multi-channel imagery.

Ablation on channel sampling and CLS tokens. The first three rows of Table 8 sequentially ablate our architectural components in the supervised setting. First, replacing our representation-based sampling with purely uniform sampling (HCS [6]) causes a consistent performance drop (2.66% on CHAMMI, 0.65% on JUMP-CP Full, and 0.13% on JUMP-CP Flu.). Next, building upon this HCS baseline, we replace the individual channel CLS tokens with a single CLS token [22]. In this TimeSformer-style baseline, the CLS token is copied during spatial attention and subsequently

Table 8: **Ablation of ChanSFormer's design choices.** The *Setting* column distinguishes between supervised classification and DINOv2 linear probes. For the DINOv2 ablation, models are pretrained on either CHAMMI or JUMP-CP (indicated as PT) to compare feature granularity. *Flu.* represents the fluorescence channels.

Setting	Method	CHAMMI	JUMP-CP Full	JUMP-CP Flu.
Supervised	**ChanSFormer**	**68.83**	**73.46**	**61.91**
	w. HCS [6] channel sampling	66.17	72.81	61.78
	w. HCS [6] w/o channel CLS token [22]	64.58	69.77	58.00
DINOv2 (PT: CHAMMI)	**ChanSFormer (channel-level)**	**50.24**	**40.82**	**39.14**
	w/ attention pooling (image-level)	48.61	38.97	38.52
DINOv2 (PT: JUMP-CP Flu.)	**ChanSFormer (channel-level)**	**42.86**	**46.13**	**45.15**
	w/ attention pooling (image-level)	41.84	45.95	45.10

averaged across channels. This modification further degrades accuracy across all datasets. Because individual channels carry highly heterogeneous information, average pooling them into a single token forcibly destroys critical modality-specific features. Therefore, maintaining independent channel tokens is essential for cell painting images.

Ablation on SSL feature granularity. We further ablate the granularity of the supervisory signals in our self-supervised learning (SSL) framework, comparing channel-independent tokens against an aggregated image-level token. As shown in the lower half of Table 8, our approach consistently achieves superior linear-probe performance, regardless of the pretraining dataset (CHAMMI or JUMP-CP). This advantage stems from the fact that different channels carry highly heterogeneous information. Retaining independent CLS tokens allows the model to utilize this heterogeneity as distinct, modality-specific supervisory signals during pretraining. This provides a richer supervisory signal per sample and increases the diversity of training signals within each batch, thus improving the performance. This comparison also separates the contribution of the backbone from that of the supervision granularity. With image-level attention pooling, ChanSFormer is trained with the same supervision granularity as ChannelViT, yet it outperforms ChannelViT (Table 2) in all six settings, mostly by a large margin, indicating that the backbone provides the main improvement. Channel-level supervision then adds a further gain in all six settings.

5 Conclusion

We presented ChanSFormer, a highly efficient, channel-agnostic Vision Transformer that replaces rigid channel embeddings with decomposed spatial-channel attention and individual channel CLS tokens. This design preserves distinct channel identities and cross-channel interactions naturally, while also unlocking superior self-supervised representation learning. Empirically, ChanSFormer achieves comparable or superior performance compared to SOTA methods across two biology multi-channel datasets and one satellite dataset. Furthermore, it demonstrates exceptional cross-dataset transferability and accelerates model throughput. By eliminating the need for channel-specific retraining, our model provides a unified, flexible backbone that paves the way for adaptable foundation models in multi-channel imaging.

Limitations. While our channel-agnostic design provides architectural flexibility, omitting fixed channel embeddings precludes the direct use of auxiliary channel-diversity losses [8], leading to slightly lower performance on CHAMMI. Adapting them is nontrivial: they are defined over static embeddings, whereas our channel CLS tokens are data-dependent representations, and directly forcing them apart may suppress meaningful cross-channel correlations. Developing a channel-agnostic diversification objective is a promising direction. Furthermore, due to resource constraints, scaling ChanSFormer to the data and model size of foundation models like BoC [10] is left to future work.

Acknowledgments and Disclosure of Funding

This work builds upon a preliminary version presented at the ImageOmics workshop at NeurIPS 2025, which was conducted while Jingwei Zhang was interning at Insitro. The subsequent extensions

and analyses at Stony Brook University were supported by USA NSF grants IIS-2123920 [D.S.], IIS-2212046 [D.S.].

Disclosure of Funding and Competing Interests: Srinivasan Sivanandan is employed by Insitro and may hold equity in the company. Insitro has filed a patent application related to the preliminary work with Jingwei Zhang and Srinivasan Sivanandan listed as inventors. The remaining author declares no competing financial interests.

References

[1] L. Von Chamier, R. F. Laine, J. Jukkala, C. Spahn, D. Krentzel, E. Nehme, M. Lerche, S. Hernández-Pérez, P. K. Mattila, E. Karinou, *et al.*, "Democratising deep learning for microscopy with zerocostdl4mic," *Nature communications*, vol. 12, no. 1, p. 2276, 2021.

[2] F. Vincent, A. Nueda, J. Lee, M. Schenone, M. Prunotto, and M. Mercola, "Phenotypic drug discovery: recent successes, lessons learned and new directions," *Nature Reviews Drug Discovery*, vol. 21, no. 12, pp. 899–914, 2022.

[3] M. Boutros, F. Heigwer, and C. Laufer, "Microscopy-based high-content screening," *Cell*, vol. 163, no. 6, pp. 1314–1325, 2015.

[4] M.-A. Bray, S. Singh, H. Han, C. T. Davis, B. Borgeson, C. Hartland, M. Kost-Alimova, S. M. Gustafsdottir, C. C. Gibson, and A. E. Carpenter, "Cell painting, a high-content image-based assay for morphological profiling using multiplexed fluorescent dyes," *Nature protocols*, vol. 11, no. 9, pp. 1757–1774, 2016.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[6] Y. Bao, S. Sivanandan, and T. Karaletsos, "Channel vision transformers: an image is worth 1 x 16 x 16 words," *arXiv preprint arXiv:2309.16108*, 2023.

[7] N. Bourriez, I. Bendidi, E. Cohen, G. Watkinson, M. Sanchez, G. Bollot, and A. Genovesio, "Chada-vit: Channel adaptive attention for joint representation learning of heterogeneous microscopy images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11556–11565, 2024.

[8] C. Pham and B. Plummer, "Enhancing feature diversity boosts channel-adaptive vision transformers," *Advances in Neural Information Processing Systems*, vol. 37, pp. 89782–89805, 2024.

[9] O. Kraus, K. Kenyon-Dean, S. Saberian, M. Fallah, P. McLean, J. Leung, V. Sharma, A. Khan, J. Balakrishnan, S. Celik, *et al.*, "Masked autoencoders for microscopy are scalable learners of cellular biology," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11757–11768, 2024.

[10] A. V. De Lorenci, S. E. Yi, T. Moutakanni, P. Bojanowski, C. Couprie, J. C. Caicedo, and W. M. A. Pernice, "Scaling channel-adaptive self-supervised learning," *Transactions on Machine Learning Research*, 2025.

[11] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, *et al.*, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[12] Z. S. Chen, C. Pham, S. Wang, M. Doron, N. Moshkov, B. Plummer, and J. C. Caicedo, "Chammi: A benchmark for channel-adaptive models in microscopy imaging," *Advances in Neural Information Processing Systems*, vol. 36, pp. 19700–19713, 2023.

[13] B. A. Plummer, N. Dryden, J. Frost, T. Hoefler, and K. Saenko, "Neural parameter allocation search," in *International Conference on Learning Representations (ICLR)*, 2022.

[14] P. Savarese and M. Maire, "Learning implicitly recurrent CNNs through parameter sharing," in *International Conference on Learning Representations (ICLR)*, 2019.

[15] D. Ha, A. Dai, and Q. Le, "Hypernetworks," in *International Conference on Learning Representations (ICLR)*, 2016.

[16] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[17] J. Zhou, C. Wei, H. Wang, W. Shen, C. Xie, A. Yuille, and T. Kong, "ibot: Image bert pre-training with online tokenizer," *arXiv preprint arXiv:2111.07832*, 2021.

[18] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

[19] T. Moutakanni, C. Couprie, S. Yi, M. Doron, Z. S. Chen, N. Moshkov, E. Gardes, M. Caron, H. Touvron, A. Joulin, *et al.*, "Cell-dino: Self-supervised image-based embeddings for cell fluorescent microscopy," *PLOS Computational Biology*, vol. 21, no. 12, p. e1013828, 2025.

[20] C. Pham, J. C. Caicedo, and B. A. Plummer, "Cha-maevit: Unifying channel-aware masked autoencoders and multi-channel vision transformers for improved cross-channel learning," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[21] W. Lian, P. Micke, J. Lindblad, and N. Sladoje, "Isolated channel vision transformers: from single-channel pretraining to multi-channel finetuning," *arXiv preprint arXiv:2503.09826*, 2025.

[22] G. Bertasius, H. Wang, and L. Torresani, "Is space-time attention all you need for video understanding?," in *Proceedings of the 38th International Conference on Machine Learning (ICML)*, vol. 139 of *Proceedings of Machine Learning Research*, pp. 813–824, 2021.

[23] Q. Miao, V. R. Golani, J. Xu, P. P. Dutta, M. Hoai, and D. Samaras, "Multi-view gaze target estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5371–5381, 2025.

[24] B. Lefaudeux, F. Massa, D. Liskovich, W. Xiong, V. Caggiano, S. Naren, M. Xu, J. Hu, M. Tintore, S. Zhang, P. Labatut, D. Haziza, L. Wehrstedt, J. Reizenstein, and G. Sizov, "xformers: A modular and hackable transformer modelling library." `https://github.com/facebookresearch/xformers`, 2022.

[25] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," *Advances in neural information processing systems*, vol. 35, pp. 16344–16359, 2022.

[26] T. Dao, "Flashattention-2: Faster attention with better parallelism and work partitioning," *arXiv preprint arXiv:2307.08691*, 2023.

[27] S. N. Chandrasekaran, B. A. Cimini, A. Goodale, L. Miller, M. Kost-Alimova, N. Jamali, J. G. Doench, B. Fritchman, A. Skepner, M. Melanson, *et al.*, "Three million images and morphological profiles of cells treated with matched chemical and genetic perturbations," *Nature Methods*, vol. 21, no. 6, pp. 1114–1121, 2024.

[28] Y. Bao and T. Karaletsos, "Contextual vision transformers for robust representation learning," *arXiv preprint arXiv:2305.19402*, 2023.

[29] X. X. Zhu, J. Hu, C. Qiu, Y. Shi, J. Kang, L. Mou, H. Bagheri, M. Haberle, Y. Hua, R. Huang, *et al.*, "So2sat lcz42: A benchmark data set for the classification of global local climate zones [software and data sets]," *IEEE Geoscience and Remote Sensing Magazine*, vol. 8, no. 3, pp. 76–89, 2020.

A Details of the channel attention kernel

We provide here the design and implementation details of the channel attention kernel. The kernel is written in Triton and supports both the forward and backward passes, with full autograd integration. Our channel size C ranges from 1 to 18 in practice, which sits well below the granularity at which mainstream attention kernels are tuned. We organize the discussion around three design choices: (i) a C-adaptive dispatch between two kernel paths, (ii) instance batching to amortize memory traffic, and (iii) a stride-aware launch interface that eliminates layout-conversion copies.

C-adaptive dispatch. NVIDIA TensorCores require the inner dimensions of a matrix multiplication to be at least 16 in order to issue a single MMA instruction; below this threshold, the operands must be padded, and the padded lanes contribute no useful work. For channel attention with $C < 16$, the score matrix $S = QK^\top \in \mathbb{R}^{C\times C}$ and the output matmul PV both fall into this regime, so a naive use of TensorCores wastes most of the issued FLOPs on padding. We therefore dispatch according to the length of channel sequences C at kernel-launch time. For short channel sequences ($C \leq 8$), we avoid using TensorCores entirely and instead use the general-purpose CUDA cores, which have no minimum-shape requirement and are not bottlenecked at this scale. For longer sequences ($C > 8$), we revert to the standard TensorCores as they are faster at this scale. The crossover between paths is determined empirically on our target hardware, an NVIDIA Quadro RTX 8000: we benchmark both paths across the practical channel-count range and select the point at which the TensorCore path becomes faster. On this GPU, the small path remains faster up through $C = 8$, after which the large path takes over. Because both the padding overhead and the TensorCore throughput depend on the GPU architecture, this threshold is hardware-dependent. On other devices, it can be selected by a lightweight device-specific microbenchmark at installation or initialization time.

Instance batching for short-sequence amortization. A single channel attention call in our model spawns hundreds to thousands of independent attention instances (one per spatial position, per head, across the batch). If each instance were launched as a separate GPU program, as in a direct port of FlashAttention [25], the per-instance computation would be far too small to amortize the fixed costs of kernel launch, register allocation, and shared memory setup. To eliminate this overhead on the small-C path, we let each program process a batch of instances together, with the batch size automatically tuned for each $(C, \text{HEAD_DIM})$ configuration. All operations—loading queries, keys, and values, the softmax reduction, and writing the result—are then executed in a batched manner, so that the launch cost is roughly divided by the batch size. When C becomes large, a single instance already provides enough parallelism to fully occupy the GPU's compute units, and batching is no longer needed; we fall back to processing one instance per program. We further prune any configuration that our cost model predicts will exceed register capacity and cause spilling.

Stride-aware launch interface. A subtle but important source of overhead in channel attention stems from layout conversion. Most attention kernels require the attention axis to occupy a fixed tensor dimension, which in our case would force an explicit permutation and a contiguous copy on every call, together with a symmetric copy when restoring the original shape. At the batch sizes we use, these copies already cost on par with the attention computation itself. We remove this overhead by making the kernel stride-aware. The launch wrapper inspects the input strides and passes the necessary stride metadata to the kernel. The kernel then uses that stride information to compute load and store offsets, handling both standard layouts and the native multidimensional layout of our queries, keys, and values with zero data movement.

Backward pass. The backward pass mirrors the forward dispatch. Following the standard FlashAttention recurrence, we save only the log-sum-exp statistics from the forward pass and recompute S, P, and O inside the backward kernel; this trades a modest amount of additional FLOPs for a substantial reduction in saved-activation memory. The three gradient tensors dQ, dK, dV are computed in a single fused kernel per path.

B Extended implementation details

Supervised Training Hyperparameters. For supervised classification tasks, we optimize all models using AdamW. For JUMP-CP and So2Sat, we apply a cosine learning rate schedule with

a 10 epoch linear warmup, a base learning rate of 5×10^{-4}, and a weight decay of 0.04. They are trained for 100 epochs using standard cross-entropy loss, with batch sizes of 256 (patch size 16) and 128 (patch size 8), respectively. For CHAMMI, we use a cosine learning rate schedule with a 3 epoch linear warmup, a learning rate of 4×10^{-5} and train for 60 epochs with a batch size of 64, a patch size of 16 and proxy loss. For the representation-based channel sampling applied across these experiments, we use an EMA momentum of $m = 0.9995$, a scaling temperature of $T = 0.01$ for CHAMMI. We use an EMA momentum of $m = 0.9995$, a scaling temperature of $T = 0.1$ for So2Sat. We use an EMA momentum of $m = 0.998$, a scaling temperature of $T = 1.0$ for JUMP-CP. Following DiChaViT [8], T is selected separately for each dataset on its validation set; its sensitivity is analyzed in Table 13.

Representation-based channel sampling. Varying the input channel set during training serves as an effective regularizer for our channel-agnostic backbone. The prior method DiChaViT [8] samples a random channel and then sample additional ones based on their dissimilarity to that initial selection. However, we observe that this heuristic often leads to an unintended outcome: selected channels may be distant from the initial anchor while remaining highly similar to each other. Our empirical analysis of this process confirms that the vast majority of channel pairs selected by this approach are indeed similar pairs. Consequently, in our representation-based sampling, we intentionally and directly sample subsets of similar channels. As demonstrated in Table 8, this strategy effectively regularizes the training process and yields superior performance. Specifically, we compute channel similarities at layer $\ell^\star = 6$. We find that representations from initial or very late layers yield slightly inferior performance, while the middle-to-late range (layers 6–9) provides the better signal for representation-based sampling. Moreover, the diagonal entries of S in Eq. (4), which represent self-similarity, are excluded from sampling. On CHAMMI, whose three sources contribute disjoint channel sets (Allen: 3 channels, HPA: 4, CP: 5), we follow the channel indexing of DiChaViT [8] and assign the channels distinct global indices, giving $C = 12$. Although S is stored as a single 12×12 buffer, it is updated and queried only within the block that corresponds to the source of the current image. S therefore effectively consists of three independent 3×3, 4×4, and 5×5 similarity matrices, with an effective C of 3, 4, or 5 depending on the source; cross-source entries are never updated or used for sampling. S is a non-trainable buffer: the channel CLS tokens are detached before the similarities are computed, so no gradient flows through S or the sampling operation. In distributed training, the batch similarity matrix is averaged across all workers before the EMA update, so that every worker maintains the same S; since S is at most 18×18, the synchronization overhead is negligible.

Self-Supervised Pretraining Hyperparameters. In our self-supervised framework based on DINOv2 [11] with iBOT [17], we adapt the default loss weighting to accommodate the presence of multiple independent channel CLS tokens. Specifically, the KoLeo regularizer weight is removed to prevent NaN and improve performance. The iBOT loss weight is reduced to 0.6 to ensure stable training. We sample only 2 channels for each image. This allows dynamic adaptation to varying numbers of channels, enlarges the batch size under a fixed memory budget, and provides a unified protocol across datasets whose images contain different numbers of channels (e.g., 3, 4, or 5 on CHAMMI). The batch size is set to 1024 for BoC and 512 for ChannelViT and ChanSFormer, so that the three models see the same number of channels per iteration. We use a schedule of 100 channel-epochs on CHAMMI and JUMP-CP. The channel-epoch refers to the number of iterations that allows the model to see the total number of channels in the dataset (e.g., assuming every sample has 4 channels, 100 channel-epochs is 200 epochs as we sample 2 channels for each sample). The base learning rate is set to 0.002 with a batch size of 1024 and the same square root scaling rule for learning rate as the vanilla DINOv2. Although each pretraining view contains only two channels, the frozen backbone is evaluated with up to 8 channels on JUMP-CP and 18 channels on So2Sat (Table 2). ChanSFormer imposes no maximum channel count, because its patch embedding, spatial positional embedding, and attention operations are all shared across the channel axis, so it operates on substantially more channels than it has observed simultaneously during pretraining.

For linear-probe evaluations on the JUMP-CP and So2Sat dataset, we use concatenated channel cls tokens from the last 4 layers and follow the same protocol in the vanilla DINOv2. For ChannelViT where we do not have the channel level cls tokens, we use the averaged-pooled token of each channel along with the single global cls token. On the CHAMMI dataset, we also concatenate channel level cls tokens but only use the features from the last layer and we follow the standard evaluation protocol as [8].

C Standard deviation of the supervised results

Here we reported the standard deviations of our method in Table 9.

Table 9: Standard deviation of ChanSFormer on 3 datasets.

Method	CHAMMI	JUMP-CP Full	JUMP-CP Flu.	So2Sat Full	So2Sat Sen-1
ChanSFormer	68.83 ± 0.51	73.46 ± 0.14	61.91 ± 0.62	64.76 ± 0.30	49.75 ± 0.33

D Subset results on the CHAMMI dataset

We detail the subset evaluation on the CHAMMI dataset, as shown in Table 10 and Table 11. In supervised learning, we surpass ChannelViT in average accuracy. We achieved a bit lower performance on the Allen and CP subsets compared with DiChaViT, which we attribute to the inability to leverage auxiliary channel-diversity losses when fixed channel embeddings are omitted. In the DINOv2 linear-probe setting, our approach demonstrates superior performance on the HPA subset and yields the highest overall accuracy across both in-domain and cross-dataset evaluations.

Table 10: Detailed breakdown of performance on CHAMMI subsets (Allen, HPA, and CP). The Avg column represents the mean of Allen and HPA. Baseline results are reported from DiChaViT [8].

Method	#Params	Allen	HPA	CP	Avg
HyperViT [15]	24.7M	45.17	63.90	26.23	54.54
DepthwiseViT [12]	22.6M	50.35	71.52	27.74	60.94
TemplateViT [13]	21.5M	49.51	64.52	25.65	57.02
ChAda-ViT [7]	21.5M	67.08	60.67	24.60	63.88
ChannelViT [6]	21.5M	67.66	62.14	27.62	64.90
DiChaViT [8]	21.5M	75.69	63.67	28.98	69.68
ChanSFormer	20.1M	69.00	68.67	27.31	68.83

Table 11: Detailed breakdown of performance on CHAMMI subsets (Allen, HPA, and CP) for DINOv2 linear-probe. The Avg column represents the mean of Allen and HPA.

Pretrain	Method	Allen	HPA	CP	Avg
CHAMMI	BoC [10]	**49.88**	47.96	**27.91**	48.92
	ChannelViT [6]	46.60	41.98	26.93	44.29
	ChanSFormer	48.96	**51.51**	25.10	**50.24**
JUMP-CP Flu.	BoC [10]	43.31	38.40	27.83	40.85
	ChannelViT [6]	32.48	33.29	**28.77**	32.88
	ChanSFormer	**44.48**	**41.25**	27.99	**42.86**

E Supervised results with ImageNet pretraining

The main paper trains all models from scratch (Table 1). Here we additionally report results when models are initialized from an ImageNet pretrained ViT-S/16, which is the more common setting on general benchmarks. We use the same training recipe as Section 4.2 with shorter schedules and lower learning rates suitable for fine-tuning: 45 epochs on CHAMMI (lr 4×10^{-5}, 3 epoch warmup, batch size 64), 50 epochs on JUMP-CP (lr 1×10^{-4}, 5 epoch warmup, batch size 256), and 20 epochs on So2Sat (lr 1×10^{-4}, 2 epoch warmup, batch size 256). We did not include ChAda-ViT as it is ChannelViT without HCS in this case. To make the spatial-attention weights compatible with the ImageNet ViT-S/16 checkpoint, ChanSFormer in this setting uses the standard ViT-S embedding dimension $D = 384$ rather than the $D = 320$ used in the main paper; this raises its parameter count to $28.9\mathrm{M}$. Table 12 summarizes the results on the full channel set of each dataset. ChanSFormer achieves the highest accuracy on all three datasets, surpassing the best baselines by 0.76% to 1.5%.

The disentangled architecture transfers well from natural-image pretraining and remains the strongest channel-adaptive ViT in this setting without relying on a learnable channel embedding.

Table 12: **Supervised classification with ImageNet ViT-S/16 pretraining.** Top-1 accuracy on the full channel set of each dataset. The highest number per column is bold and the second highest is underlined.

Method	#Params	CHAMMI	JUMP-CP	So2Sat
HyperViT [15]	24.7M	62.28	39.64	61.22
DepthwiseViT [12]	22.6M	76.52	46.97	61.55
TemplateViT [13, 14]	21.5M	70.30	35.01	60.84
ChannelViT [6]	21.5M	<u>80.68</u>	80.81	<u>68.01</u>
DiChaViT [8]	21.5M	78.83	<u>82.99</u>	67.18
ChanSFormer	28.9M	**82.18**	**84.11**	**68.77**

F Sensitivity to the sampling temperature

The temperature T controls how strongly the learned channel relations shape the sampling distribution: large values approach uniform HCS sampling, whereas small values concentrate the samples on the channel combinations favored by S. Table 13 varies T around the selected value on each dataset. On So2Sat, $T = 0.1$ and 1.0 achieve nearly identical overall performance, while $T = 0.01$ is slightly lower, mainly on Sentinel-1. On JUMP-CP, $T = 1.0$ and 10 perform similarly, while $T = 0.1$ leads to a noticeable drop. On CHAMMI, $T = 0.01$ and 0.1 are comparable, whereas $T = 0.001$ and 1.0 perform worse. Performance is therefore stable within an appropriate range for each dataset, while excessively concentrating or flattening the sampling distribution reduces it. The selected values differ across datasets partly because S has a different contrast on each: the standard deviation of its off-diagonal entries is 0.134–0.174 on the three CHAMMI sources, 0.203 on So2Sat, and 0.295 on JUMP-CP. This roughly two-fold difference does not fully account for the two orders of magnitude spanned by the selected T, since T also determines how strongly the learned relations are sharpened. Similarly, although JUMP-CP and So2Sat share a coarse two-block structure in S (Figures 2 and 3), this does not imply the same optimal T. The ten-fold difference between their selected values does not indicate sharply different optima, however: $T = 1.0$ performs well on both datasets, nearly matching $T = 0.1$ on So2Sat and giving the best result on JUMP-CP.

Table 13: **Sensitivity to the sampling temperature** T in supervised training. The selected value for each dataset is shown in bold in the T column; bold numbers denote the best result in each column.

T	So2Sat Full	Sen-1
0.01	64.85	48.35
0.1	64.76	**49.75**
1.0	**64.99**	49.46

T	JUMP-CP Full	Flu.
0.1	71.41	58.39
1.0	**73.46**	**61.91**
10	73.12	61.18

T	CHAMMI
0.001	67.34
0.01	**68.83**
0.1	68.40
1.0	66.26

G Number of channels per view in self-supervised pretraining

To examine the effect of using more channels per view, we additionally pretrain on JUMP-CP Flu. with four channels per image, reducing the image batch size from 512 to 256 so that the total number of channel views per iteration is unchanged. As shown in Table 14, four-channel pretraining consistently improves both ChannelViT and ChanSFormer over the two-channel setting, and ChanSFormer remains substantially better than ChannelViT in every evaluation setting.

Table 14: **Number of channels per view in DINOv2 pretraining** on JUMP-CP Flu., evaluated as in Table 2. The image batch size is halved for four channels to keep the number of channel views per iteration fixed.

		CHAMMI	JUMP-CP	
Method	# Channels per view	Avg	Full	Flu.
ChannelViT [6]	2	32.88	30.72	32.97
	4	34.09	31.97	35.53
ChanSFormer	2	42.86	46.13	45.15
	4	**45.04**	**48.18**	**46.49**

H Adapting ChannelViT's channel embeddings to unseen channels

In Table 2, ChannelViT handles channels that are absent from pretraining by zero-padding their channel embeddings. A stronger baseline is to learn embeddings for the unseen channels on the target dataset. We therefore train the embeddings of the unseen channels together with the linear head and keep all other pretrained weights fixed. This is considerably more expensive than linear probing: computing gradients for the embeddings requires full forward and backward passes through the backbone, whereas linear probing trains a linear head on features extracted once, and several probe configurations can share the same cached features. One adaptation run is about $20\times$ slower than a linear-probe run in our implementation, and each hyperparameter configuration requires a separate run. As shown in Table 15, adapting the channel embeddings slightly improves ChannelViT over zero-padding, but it remains below ChanSFormer in all seven unseen-channel settings. We omit in-domain settings, where the pretraining and evaluation channels are identical and no embedding needs to be adapted, and CHAMMI as a target, since its official protocol uses feature-space 1-NN classification without trainable downstream parameters.

Table 15: **Zero-padded vs. adapted channel embeddings for ChannelViT** [6] on unseen-channel transfer. Models are pretrained with DINOv2 on the *SSL source* and evaluated by linear probe on the *Target*.

SSL source	Target	ChannelViT (zero-pad)	ChannelViT (CE adapt.)	ChanSFormer
CHAMMI	JUMP-CP Full	23.94	25.54	**40.82**
	JUMP-CP Flu.	24.71	24.99	**39.14**
	So2Sat Full	66.65	66.91	**68.08**
	So2Sat Sen-1	44.54	45.14	**46.23**
JUMP-CP Flu.	JUMP-CP Full	30.72	31.10	**46.13**
	So2Sat Full	65.00	65.19	**66.33**
	So2Sat Sen-1	44.47	44.60	**45.34**

I Relation to factorized space–time attention

Our channel–spatial factorization is related to the space–time factorization of TimeSformer [22], but serves a different purpose. TimeSformer factorizes attention mainly for efficient video modeling, whereas we use the factorization to remove the dependence on fixed channel embeddings while still allowing information to flow across channels. Moreover, unlike video frames, whose temporal order is meaningful, the order of microscopy channels is arbitrary, so the encoder should be permutation-equivariant along the channel axis and the image-level prediction permutation-invariant. ChanSFormer satisfies both by construction: all operations are shared across channels and no channel ordering is encoded, so channel identity is preserved structurally through the explicit channel axis rather than through learned channel embeddings, and the attention pooling over the channel CLS tokens does not depend on their order. The two designs also differ in how the per-slice representations are aggregated. TimeSformer copies a single CLS token across frames and merges it, which suits highly correlated slices; with the channel CLS tokens replaced by such a shared token, accuracy drops by 1.59%–3.78% under the same HCS sampling (Table 8), as averaging the heterogeneous channels of microscopy images is lossy.

J Learned channel relations

Figure 2 visualizes, on JUMP-CP, the channel-similarity matrix S learned by our sampler (Section 3.4) and the channel-attention weights of the trained model. S shows a clear two-block structure. The three brightfield channels are nearly interchangeable (pairwise similarity 0.95–0.97), whereas their similarity to the fluorescence channels is substantially lower. The channel attention reflects this structure mainly through the fluorescence queries, which attend mainly to other fluorescence channels and assign each brightfield channel only about 0.07 of their attention, below the uniform level of 0.125, whereas the brightfield queries attend nearly uniformly across channels. This is qualitatively consistent with the imaging modalities: fluorescence channels capture complementary stain-specific cellular structures, while the brightfield channels provide a different transmitted-light view. It also agrees with the channel correlation structure reported by ChannelViT [6]. Although all channel CLS tokens are initialized identically, the learned S is far from uniform, indicating that they do not collapse to the same representation. On So2Sat (Figure 3), S likewise shows a two-block structure aligned with the two sensors, Sentinel-1 (SAR) and Sentinel-2 (multispectral), again consistent with ChannelViT.

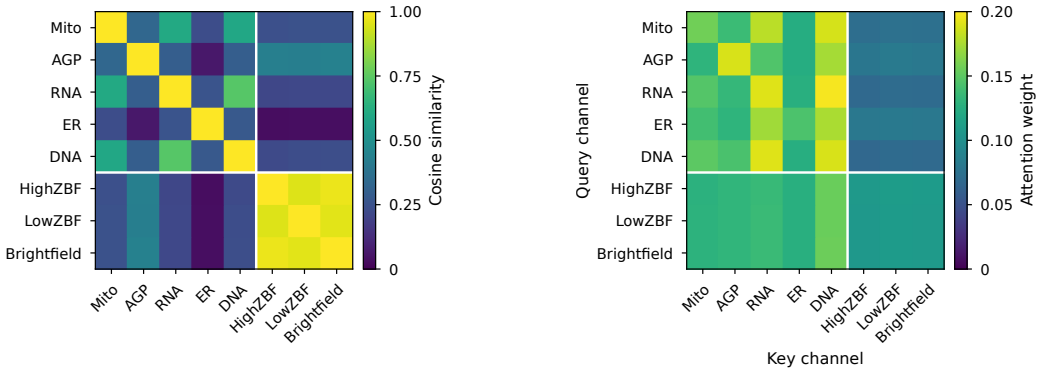


Figure 2: **Learned channel relations on JUMP-CP. Left:** the EMA channel-similarity matrix S used by our sampler. **Right:** channel-attention weights (row: query, column: key) at patch positions, averaged over all heads, all 12 layers, and the full test set with all 8 channels present; each row sums to 1 (uniform $= 0.125$). White lines separate the fluorescence (Mito, AGP, RNA, ER, DNA) and brightfield (HighZBF, LowZBF, Brightfield) channels.

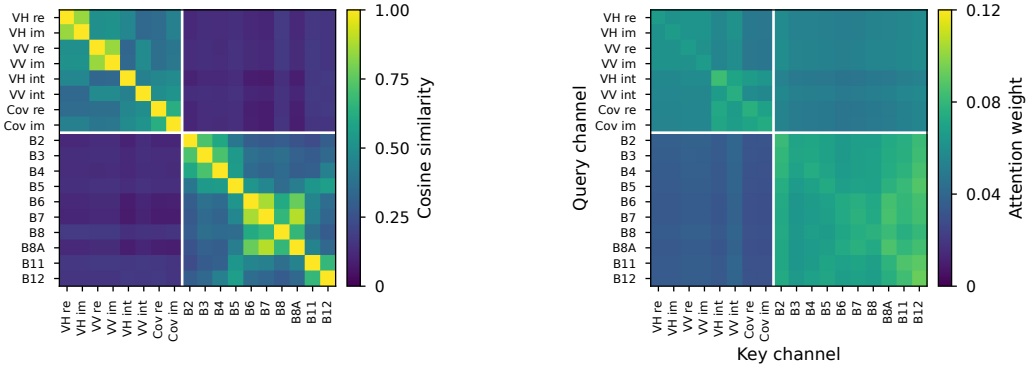


Figure 3: **Learned channel relations on So2Sat. Left:** the EMA channel-similarity matrix S used by our sampler. **Right:** channel-attention weights (row: query, column: key) at patch positions, averaged over all heads, all 12 layers, and the full test set with all 18 channels present; each row sums to 1. White lines separate the 8 Sentinel-1 channels from the 10 Sentinel-2 bands. Sentinel-1: real (re) and imaginary (im) parts of the unfiltered VH and VV channels, intensity (int) of the refined-Lee-filtered VH and VV channels, and real and imaginary parts of the refined-Lee-filtered covariance-matrix off-diagonal element (Cov).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 state our main claims (channel-agnostic backbone via disentangled spatial–channel attention and channel CLS tokens), which are supported by the results in Sections 4.3.1, 4.3.2, and 4.3.3.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A dedicated limitations paragraph appears at the end of Section 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This paper does not present any theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4.2 describes the training recipe and hardware, and Appendix B provides exhaustive hyperparameters for both supervised and self-supervised settings. The architecture is fully specified in Section 3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All three datasets (CHAMMI, JUMP-CP, So2Sat) are publicly available and properly cited. Our code is available at `https://github.com/cvlab-stonybrook/ChanSFormer`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Training and evaluation protocols are in Section 4.2; full hyperparameters (optimizer, learning rate schedule, batch size, EMA momentum, sampling temperature) are in Appendix B.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Table 9 in Appendix C reports the standard deviation of ChanSFormer over 3 runs on every supervised setting, and Table 5 reports mean $\pm$ std across all $\binom{8}{k}$ channel subsets on JUMP-CP.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 4.2 reports the GPU type per dataset. Throughput numbers in Section 4.3.3 characterize the compute footprint of training and inference.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We reviewed the NeurIPS Code of Ethics and confirm compliance. Our work uses only publicly released datasets (cell-biology microscopy and satellite imagery), introduces no personally identifiable data, and presents no foreseeable conflicts with the Code.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

Justification: Our contribution is a backbone architecture for scientific multichannel imaging (cell biology and remote sensing). It is foundational research several steps removed from any deployment with direct societal consequences, and we see no specific path to malicious use.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The proposed model is a discriminative vision backbone for scientific imaging; it is not a generative model and poses no high misuse risk.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets (CHAMMI [12], JUMP-CP [27], So2Sat [29]) and baseline implementations (ChannelViT, DiChaViT, BoC, and others) are properly cited and used under their original public-release licenses.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: This submission does not release new datasets. Our code and trained model checkpoints are released at `https://github.com/cvlab-stonybrook/ChanSFormer` with accompanying documentation.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We only use publicly available datasets, so there are no such issues.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: We only use publicly available datasets, so there are no such issues.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: We only use LLMs for writing and coding assistance, which does not affect the core methodology of the research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.