S4ND: Modeling Images and Videos as Multidimensional Signals Using State Spaces

Eric Nguyen*†, Karan Goel*‡, Albert Gu*‡,
Gordon W Downs‡, Preey Shah‡, Tri Dao‡, Stephen A. Baccus§, Christopher Ré‡
†Department of BioEngineering, Stanford University
‡Department of Computer Science, Stanford University
§Department of Neurobiology, Stanford University
{etnguyen,albertgu,gwdowns,preey,trid,baccus}@stanford.edu
{kgoel,chrismre}@cs.stanford.edu

Abstract

Visual data such as images and videos are typically modeled as discretizations of inherently continuous, multidimensional signals. Existing continuous-signal models attempt to exploit this fact by modeling the underlying signals of visual (e.g., image) data directly. However, these models have not yet been able to achieve competitive performance on practical vision tasks such as large-scale image and video classification. Building on a recent line of work on deep state space models (SSMs), we propose S4ND, a new multidimensional SSM layer that extends the continuous-signal modeling ability of SSMs to multidimensional data including images and videos. We show that S4ND can model large-scale visual data in 1D, 2D, and 3D as continuous multidimensional signals and demonstrates strong performance by simply swapping Conv2D and self-attention layers with S4ND layers in existing state-of-the-art models. On ImageNet-1k, S4ND exceeds the performance of a Vision Transformer baseline by 1.5% when training with a 1D sequence of patches, and matches ConvNeXt when modeling images in 2D. For videos, S4ND improves on an inflated 3D ConvNeXt in activity classification on HMDB-51 by 4%. S4ND implicitly learns global, continuous convolutional kernels that are resolution invariant by construction, providing an inductive bias that enables generalization across multiple resolutions. By developing a simple bandlimiting modification to S4 to overcome aliasing, S4ND achieves strong zero-shot (unseen at training time) resolution performance, outperforming a baseline Conv2D by 40% on CIFAR-10 when trained on 8×8 and tested on 32×32 images. When trained with progressive resizing, S4ND comes within $\sim 1\%$ of a high-resolution model while training 22% faster.

1 Introduction

Modeling visual data such as images and videos is a canonical problem in deep learning. In the last few years, many modern deep learning backbones that achieve strong performance on benchmarks like ImageNet [53] have been proposed. These backbones are diverse, and include 1D sequence models such as the Vision Transformer (ViT) [13], which treats images as sequences of patches, and 2D and 3D models that use local convolutions over images and videos (ConvNets) [37, 25, 56, 58, 60, 42, 24, 32, 49, 64, 15].

A commonality among modern vision models capable of achieving state-of-the-art (SotA) performance is that they treat visual data as discrete pixels rather than continuous-signals. However, images

*Equal contribution.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

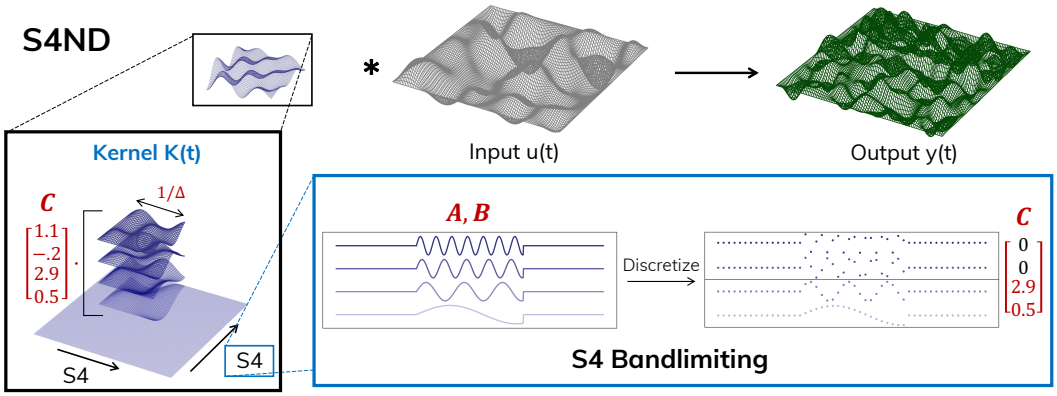


Figure 1: (**S4ND**.) (Parameters in *red*.) (*Top*) S4ND can be viewed as a depthwise convolution that maps a multidimensional input (*black*) to output (*green*) through a continuous convolution kernel (*blue*). (*Bottom Left*) The kernel can be interpreted as a linear combination (controlled by $\boldsymbol{C}$) of basis functions (controlled by $\boldsymbol{A}, \boldsymbol{B}$) with flexible width (controlled by step size Δ). For structured $\boldsymbol{C}$, the kernel can further factored as a low-rank tensor product of 1D kernels, and can be interpreted as independent S4 transformations on each dimension. (*Bottom Right*) Choosing $\boldsymbol{A}, \boldsymbol{B}$ appropriately yields Fourier basis functions with controllable frequencies. To avoid aliasing in the final discrete kernels, the coefficients of $\boldsymbol{C}$ corresponding to high frequencies can simply be masked out.

and videos are discretizations of multidimensional and naturally continuous signals, sampled at a fixed rate in the spatial and temporal dimensions. Ideally, we would want approaches that are capable of recognizing this distinction between data and signal, and directly model the underlying continuous-signals. This would give them capabilities like the ability to adapt the model to data sampled at different resolutions.

A natural approach to building such models is to parameterize and learn continuous convolutional kernels, which can then be sampled differently for data at different resolutions [16, 50, 54, 20, 21]. Among these, deep state space models (SSM) [20], in particular S4 [21], have achieved SotA results in modeling sequence data derived from continuous-signals, such as audio [17]. However, a key limitation of SSMs is that they were developed for 1D signals, and cannot directly be applied to visual data derived from multidimensional "ND" signals. Given that 1D SSMs outperform other continuous modeling solutions for sequence data [21], and have had preliminary success on image [21] and video classification [31], we hypothesize that they may be well suited to modeling visual data when appropriately generalized to the setting of multidimensional signals.

Our main contribution is S4ND, a new deep learning layer that extends S4 to multidimensional signals. The key idea is to turn the standard SSM (a 1D ODE) into a multidimensional PDE governed by an independent SSM per dimension. By adding additional structure to this ND SSM, we show that it is equivalent to an ND continuous convolution that can be factored into a separate 1D SSM convolution per dimension. This results in a model that is efficient and easy to implement, using the standard 1D S4 layer as a black box. Furthermore, it can be controlled by S4's parameterization, allowing it to model both long-range dependencies, or finite windows with a learnable window size that generalize conventional local convolutions [22].

We show that S4ND can be used as a drop-in replacement in strong modern vision architectures while matching or improving performance in 1D, 2D, and 3D. With minimal change to the training procedure, replacing the self-attention in ViT with S4-1D improves top-1 accuracy by 1.5%, and replacing the convolution layers in a 2D ConvNeXt backbone [42] with S4-2D preserves its performance on ImageNet-1k [10]. Simply inflating (temporally) this pretrained S4-2D-ConvNeXt backbone to 3D improves video activity classification results on HMDB-51 [38] by 4 points over the pretrained ConvNeXt baseline. Notably, we use S4ND as global kernels that span the entire input shape, which enable it to have global context (both spatially and temporally) in every layer of a network.

Additionally, we propose a low-pass bandlimiting modification to S4 that encourages the learned convolutional kernels to be smooth. While S4ND can be used at any resolution, performance suffers when moving between resolutions due to aliasing artifacts in the kernel, an issue also noted by prior

work on continuous models [50]. While S4 was capable of transferring between different resolutions on audio data [21], visual data presents a greater challenge due to the scale-invariant properties of images in space and time [52], as sampled images with more distant objects are more likely to contain power at frequencies above the Nyquist cutoff frequency. Motivated by this, we propose a simple criteria that masks out frequencies in the S4ND kernel that lie above the Nyquist cutoff frequency.

The continuous-signal modeling capabilities of S4ND open the door to new training recipes, such as the ability to train and test at different resolutions. On the standard CIFAR-10 [36] and Celeb-A [43] datasets, S4ND degrades by as little as 1.3% when upsampling from low- to high-resolution data (e.g. $128 \times 128 \rightarrow 160 \times 160$), and can be used to facilitate progressive resizing to speed up training by 22% with $\sim 1\%$ drop in final accuracy compared to training at the high resolution alone. We also validate that our new bandlimiting method is critical to these capabilities, with ablations showing absolute performance degradation of up to $20\%+$ without it.

2 Related Work

Image Classification. There is a long line of work in image classification, with much of the 2010s dominated by ConvNet backbones [37, 25, 56, 58, 60]. Recently, Transformer backbones, such as ViT [13], have achieved SotA performance on images using self-attention over a sequence of 1D patches [40, 39, 62, 71, 12]. Their scaling behavior in both model and dataset training size is believed to give them an inherent advantage over ConvNets [13], even with minimal inductive bias. Liu et al. [42] introduce ConvNeXt, which modernizes the standard ResNet architecture [25] using modern training techniques, matching the performance of Transformers on image classification. We select a backbone in the 1D and 2D settings, ViT and ConvNeXt, to convert into continuous-signal models by replacing the multi-headed self-attention layers in ViT and the standard Conv2D layers in ConvNeXt with S4ND layers, maintaining their top-1 accuracy on large-scale image classification.

S4 & Video Classification. To handle the long-range dependancies inherent in videos, [31] used 1D S4 for video classification on the Long-form Video Understanding dataset [67]. They first applied a Transformer to each frame to obtain a sequence of patch embeddings for each video frame independently, followed by a standard 1D S4 to model across the concatenated sequence of patches. This is akin to previous methods that learned spatial and temporal information separately [33], for example using ConvNets on single frames, followed by an LSTM [27] to aggregate temporal information. In contrast, modern video architectures such as 3D ConvNets and Transformers [24, 32, 49, 64, 15, 35, 41, 67, 2, 1] show stronger results when learning spatiotemporal features simultaneously, which the generalization of S4ND into multidimensions now enables us to do.

Continuous-signal Models. Visual data are discretizations of naturally continuous signals that possess extensive structure in the joint distribution of spatial frequencies, including the properties of scale and translation invariance. For example, an object in an image generates correlations between lower and higher frequencies that arises in part from phase alignment at edges [47]. As an object changes distances in the image, these correlations remain the same but the frequencies shift. This relationship can potentially be learned from a coarsely sampled image and then applied at higher frequency at higher resolution.

A number of continuous-signal models have been proposed for the visual domain to learn these inductive biases, and have led to additional desirable properties and capabilities. A classic example of continuous-signal driven processing is the fast Fourier transform, which is routinely used for filtering and data consistency in computational and medical imaging [11]. NeRF represents a static scene as a continuous function, allowing them to render scenes smoothly from multiple viewpoints [45]. CKConv [51] learns a continuous representation to create kernels of arbitrary size for several data types including images, with additional benefits such as the ability to handle irregularly sampled data. FlexConv [50] extends this work with a learned kernel size, and show that images can be trained at low resolution and tested at high resolution if the aliasing problem is addressed. S4 [21] increased abilities to model long-range dependancies using continuous kernels, allowing SSMs to achieve SotA on sequential CIFAR [36]. However, these methods including 1D S4 have been applied to relatively low dimensional data, e.g., time series, and small image datasets. S4ND is the first continuous-signal model applied to high dimensional visual data with the ability to maintain SotA performance on large-scale image and video classification.

Progressive Resizing. Training times for large-scale image classification can be quite long, a trend that is exacerbated by the emergence of foundation models [4]. A number of strategies have emerged for reducing overall training time. Fix-Res [61] trains entirely at a lower resolution, and then fine-tunes at the higher test resolution to speed up training in a two-stage process. Mix-and-Match [28] randomly samples low and high resolutions during training in an interleaved manner. An effective method to reduce training time on images is to utilize progressive resizing. This involves training at a lower resolution and gradually upsampling in stages. For instance, fastai [14] utilized progressive resizing to train an ImageNet in under 4 hours. EfficientNetV2 [60] coupled resizing with a progressively regularization schedule, increasing the regularization as well to maintain accuracy. In EfficientNetV2 and other described approaches, the models eventually train on the final test resolution. As a continuous-signal model, we demonstrate that S4ND is naturally suited to progressive resizing, while being able to generalize to *unseen* resolutions at test time.

3 Preliminaries

State space models. S4 investigated state space models, which are linear time-invariant systems that map signals $u(t) \mapsto y(t)$ and can be represented either as a linear ODE (equation (1)) or convolution (equation (2)). Its parameters are $\boldsymbol{A} \in \mathbb{C}^{N \times N}$ and $\boldsymbol{B}, \boldsymbol{C} \in \mathbb{C}^N$ for a state size N.

$$\begin{aligned} x'(t) &= \boldsymbol{A}x(t) + \boldsymbol{B}u(t) \\ y(t) &= \boldsymbol{C}x(t) \end{aligned} \tag{1}$$

$$\begin{aligned} K(t) &= \boldsymbol{C}e^{t\boldsymbol{A}}\boldsymbol{B} \\ y(t) &= (K * u)(t) \end{aligned} \tag{2}$$

Basis functions. For the clearest intuition, we think of the convolution kernel as a linear combination (controlled by $\boldsymbol{C}$) of **basis kernels** $K_n(t)$ (controlled by $\boldsymbol{A}, \boldsymbol{B}$)

$$K(t) = \sum_{k=0}^{N-1} \boldsymbol{C}_k K_k(t) \qquad K_n(t) = (e^{t\boldsymbol{A}}\boldsymbol{B})_n \tag{3}$$

Discretization. The SSM (1) is defined over a continuous-time axis and produces continuous-time convolution kernels (2)(3). Given a discrete input sequence $u_0, u_1, \ldots$ sampled uniformly from an underlying signal $u(t)$ at a step size Δ (i.e. $u_k = u(k\Delta)$), the kernel can be sampled to match the rate of the input. Note that instead of directly sampling the kernel, standard discretization rules should be applied to minimize the error from the discrete to the continuous-time kernel [21]. For inputs given at different resolutions, the model can then simply change its Δ value to compute the kernel at different resolutions.

We note that the step size Δ does not have to be exactly equal to a "true sampling rate" of the underlying signal, but only the relative rate matters. Concretely, the discrete-time kernel depends only on the *product* $\Delta\boldsymbol{A}$ and $\Delta\boldsymbol{B}$, and S4 learns separate parameters $\Delta, \boldsymbol{A}, \boldsymbol{B}$.

S4. S4 is a special SSM with prescribed $(\boldsymbol{A}, \boldsymbol{B})$ matrices that define well-behaved basis functions, and an algorithm that allows the convolution kernel to be computed efficiently. Variants of S4 exist that define different basis functions, such as simple diagonal SSMs [23], or one that defines **truncated Fourier functions** $K_n(t) = \sin(2\pi nt)\mathbb{I}([0,1])$ [22] (Fig. 1). These versions of S4 have easy-to-interpret basis functions that will allow us to control the frequencies in the kernel (Section 4.2).

4 Method

We describe the proposed S4ND model for the 2D case only, for ease of notation and presentation. The results extend readily to general dimensions; full statements and proofs for the general case are in Appendix A. Section 4.1 describes the multidimensional S4ND layer, and Section 4.2 describes our simple modification to restrict frequencies in the kernels. Fig. 1 illustrates the complete S4ND layer.

4.1 S4ND

We begin by generalizing the (linear time-invariant) SSM (1) to higher dimensions. Notationally, we denote the individual time axes with superscripts in parentheses. Let $u = u(t^{(1)}, t^{(2)})$

and $y = y(t^{(1)}, t^{(2)})$ be the input and output which are signals $\mathbb{R}^2 \to \mathbb{C}$, and $x = (x^{(1)}(t^{(1)}, t^{(2)}), x^{(2)}(t^{(1)}, t^{(2)})) \in \mathbb{C}^{N^{(1)} \times N^{(2)}}$ be the SSM state of dimension $N^{(1)} \times N^{(2)}$, where $x^{(\tau)} : \mathbb{R}^2 \to \mathbb{C}^{N^{(\tau)}}$.

Definition 1 (Multidimensional SSM)**.** *Given parameters $\boldsymbol{A}^{(\tau)} \in \mathbb{C}^{N^{(\tau)} \times N^{(\tau)}}$, $\boldsymbol{B}^{(\tau)} \in \mathbb{C}^{N^{(\tau)} \times 1}$, $\boldsymbol{C} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}$, the 2D SSM is the map $u \mapsto y$ defined by the linear PDE with initial condition $x(0,0) = 0$:*

$$\begin{aligned}
\frac{\partial}{\partial t^{(1)}} x(t^{(1)}, t^{(2)}) &= (\boldsymbol{A}^{(1)} x^{(1)}(t^{(1)}, t^{(2)}), x^{(2)}(t^{(1)}, t^{(2)})) + \boldsymbol{B}^{(1)} u(t^{(1)}, t^{(2)}) \\
\frac{\partial}{\partial t^{(2)}} x(t^{(1)}, t^{(2)}) &= (x^{(1)}(t^{(1)}, t^{(2)}), \boldsymbol{A}^{(2)} x^{(2)}(t^{(1)}, t^{(2)})) + \boldsymbol{B}^{(2)} u(t^{(1)}, t^{(2)}) \\
y(t^{(1)}, t^{(2)}) &= \langle \boldsymbol{C}, x(t^{(1)}, t^{(2)}) \rangle
\end{aligned} \tag{4}$$

Note that Definition 1 differs from the usual notion of multidimensional SSM, which is simply a map from $u(t) \in \mathbb{C}^n \mapsto y(t) \in \mathbb{C}^m$ for higher-dimensional $n, m > 1$ but still with 1 time axis. However, Definition 1 is a map from $u(t_1, t_2) \in \mathbb{C}^1 \mapsto y(t_1, t_2) \in \mathbb{C}^1$ for *scalar* input/outputs but over *multiple* time axes. When thinking of the input $u(t^{(1)}, t^{(2)})$ as a function over a 2D grid, Definition 1 can be thought of as a simple linear PDE that just runs a standard 1D SSM over each axis independently.

Analogous to equation (2), the 2D SSM can also be viewed as a multidimensional convolution.

Theorem 1. (4) *is a time-invariant system that is equivalent to a 2D convolution $y = K * u$ by the kernel*

$$K(t^{(1)}, t^{(2)}) = \langle \boldsymbol{C}, (e^{t^{(1)} \boldsymbol{A^{(1)}}} \boldsymbol{B^{(1)}}) \otimes (e^{t^{(2)} \boldsymbol{A^{(2)}}} \boldsymbol{B^{(2)}}) \rangle \tag{5}$$

This kernel is a linear combination of the $N^{(1)} \times N^{(2)}$ basis kernels $\{K^{(1)}_{n^{(1)}}(t^{(1)}) \otimes K^{(1)}_{n^{(2)}}(t^{(2)}) : n^{(1)} \in [N^{(1)}], n^{(2)} \in [N^{(2)}]\}$ where $K^{(\tau)}$ are the standard 1D SSM kernels (3) *for each axis .*

However, a limitation of this general form is that the number of basis functions $N^{(1)} \times N^{(2)} \times \dots$ grows exponentially in the dimension, increasing the parameter count (of $\boldsymbol{C}$) and overall computation dramatically. This can be mitigated by factoring $\boldsymbol{C}$ as a low-rank tensor.

Corollary 4.1. *Suppose that $\boldsymbol{C} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}$ is a low-rank tensor $\boldsymbol{C} = \sum_{i=1}^{r} \boldsymbol{C_i^{(1)}} \otimes \boldsymbol{C_i^{(2)}}$ where each $\boldsymbol{C_i^{(\tau)}} \in \mathbb{C}^{N^{(\tau)}}$. Then the kernel* (5) *also factors as a tensor product of 1D kernels*

$$K(t^{(1)}, t^{(2)}) = \sum_{i=1}^{r} K_i^{(1)}(t^{(1)}) \otimes K_i^{(2)}(t^{(2)}) := \sum_{i=1}^{r} (\boldsymbol{C_i^{(1)}} e^{t^{(2)} \boldsymbol{A^{(1)}}} \boldsymbol{B^{(1)}}) \otimes (\boldsymbol{C_i^{(2)}} e^{t^{(2)} \boldsymbol{A^{(2)}}} \boldsymbol{B^{(2)}})$$

In our experiments, we choose $\boldsymbol{C}$ as a rank-1 tensor, but the rank can be freely adjusted to tradeoff parameters and computation for expressivity. Using the equivalence between (1) and (2), Corollary 4.1 also has the simple interpretion as defining an independent 1D SSM along each axis of the multidimensional input.

4.2 Resolution Change and Bandlimiting

SSMs in 1D have shown strong performance in the audio domain, and can nearly preserve full accuracy when tested zero-shot on inputs sampled at very different frequencies [21]. This capability relies simply on scaling Δ by the relative change in frequencies (i.e., if the input resolution is doubled, halve the SSM's Δ parameter). However, sampling rates in the spatial domain are often much lower than temporally, leading to potential aliasing when changing resolutions. A standard technique to avoid aliasing is to apply a low-pass filter to remove frequencies above the Nyquist cutoff frequency.

For example, when $\boldsymbol{A}$ is diagonal with n-th element $\boldsymbol{a}_n$, each basis function has simple form $K_n(t) = e^{t\boldsymbol{a}_n} \boldsymbol{B}_n$. Note that the frequencies are mainly controlled by the imaginary part of $\boldsymbol{a}_n$. We propose the following simple method: for any n such that $\boldsymbol{a}_n \cdot \Delta < \frac{1}{2}\alpha$, mask out the corresponding coefficient of the linear combination $\boldsymbol{C}_n$ (equation (3)). Here α is a hyperparameter that controls the cutoff; theoretically, $\alpha = 1.0$ corresponds to the Nyquist cutoff if the basis functions are pure sinusoids. However, due to the decay $e^{\Re(\boldsymbol{a}_n)}$ arising from the real part as well as approximations arising from using finite-state SSMs, α often has to be set lower empirically.

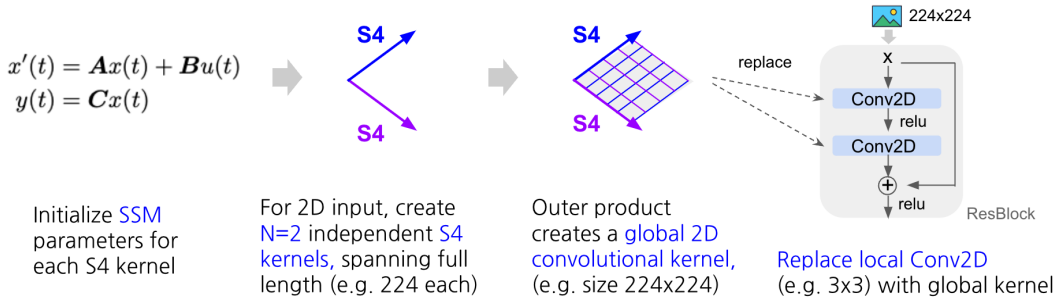


Figure 2: (**Flowchart of S4ND for images: 2D example**.) S4ND can process images as 2D inputs by initializing an SSM per spatial dimension x and y of the input. Two independent S4 kernels are then instantiated that span the entire input lengths of each dimension (e.g., 224 as shown above). Computing an outer product of the two 1D kernels produces a global convolutional kernel (e.g., 224x224). This global kernel can replace standard local Conv2D layers where ever they are used, such as ResNet or ConvNeXt blocks. A similar procedure can be done in 3D (with 3 S4 kernels) to create 3D global kernels for videos.

5 Experiments

We evaluate S4ND on large-scale image classification in Section 5.1 in the 1D and 2D settings, followed by activity classification in videos in Section 5.2 in the 3D setting, where using S4ND as a drop-in replacement for standard deep learning layers matches or improves performance in all settings. In Section 5.3, we performed controlled ablations to highlight the benefits of S4ND as a continuous-signal model in images.

5.1 S4ND in 1D & 2D: Large-scale Image Classification

First, we show that S4ND is a drop-in replacement for existing visual modeling layers such as 1D self-attention and 2D local convolutions, with no degradation in top-1 performance when used in modern backbones such as ViT [13] and ConvNeXt [42] on ImageNet-1k [10].

Baselines and Methodology. We consider large-scale image classification on the ImageNet-1k dataset, which consists of 1000 classes and 1.3M images. We start with two strong baselines: ViT-B (base, 88M parameters) for processing images in the 1D setting and ConvNeXt-T (tiny, 28.4M) in the 2D setting. (We omit the postfix "B" and "T" for brevity). More recent works using Transformers on images have surpassed ViT, but we focus on the original ViT model to highlight specifically the drop-in capability and performance difference in self-attention vs. S4ND layers. We first swap the self-attention layers in ViT with S4ND layers, and call this model S4ND-ViT. Notably, we simplify ViT by removing the positional encodings, as S4ND does not require injecting this inductive bias. Similarly, we swap the local Conv2D layers in the ConvNeXt blocks with S4ND layers, which we call S4ND-ConvNeXt, a model with global context at each layer. Both S4ND variants result in similar parameter counts compared to their baseline models.

Table 1: (**Performance on image classification.**) Top-1 test accuracy benchmarks for images in the 1D and 2D settings. ConvNeXt-M, for "micro", is a reduced model size for Celeb-A, while "-ISO" is an isotropic S4ND backbone [21].

Model	Dataset	Params	Acc
ViT-B	ImageNet	88.0M	78.9
S4ND-ViT-B	ImageNet	88.8M	**80.4**
ConvNeXt-T	ImageNet	28.4M	82.1
S4ND-ConvNeXt-T	ImageNet	30.0M	**82.2**
Conv2D-ISO	CIFAR-10	2.2M	93.7
S4ND-ISO	CIFAR-10	5.3M	**94.1**
ConvNeXt-M	Celeb-A	9.2M	91.0
S4ND-ConvNeXt-M	Celeb-A	9.6M	**91.3**

Training. For all ImageNet models, we train from scratch with no outside data and adopt the training procedure from [62, 69], which uses the AdamW optimizer [44] for 300 epochs, cosine decay learning rate, weight decay 0.05, and aggressive data augmentations including RandAugment [7], Mixup [72], and AugMix [26]. We add RepeatAug [29] for ConvNeXt and S4ND-ConvNeXt. The

initial learning rate for ViT (and S4ND-ViT) is 0.001, while for ConvNeXt (and S4ND-ConvNeXt) it is 0.004. See Appendix B.1 for additional training procedure details.

Results. Table 1 shows top-1 accuracy results for each model on ImageNet. After reproducing the baselines, S4ND-ViT was able to moderately boost performance by +1.5% over ViT, while S4ND-ConvNeXt matched the original ConvNeXt's performance. This indicates that S4ND is a strong primitive that can replace self-attention and standard 2D convolutions in practical image settings with large-scale data.

5.2 S4ND in 3D: Video Classification

Next, we demonstrate the flexible capabilities of S4ND in settings involving pretraining and even higher-dimensional signals. We use the activity recognition dataset HMDB-51 [38] which involves classifying videos in 51 activity classes.

Baselines and Methodology. Prior work demonstrated that 2D CNNs (e.g. pretrained on ImageNet) can be adapted to 3D models by 2D to 3D kernel *inflation* (I3D [5]), in which the 2D kernels are repeated temporally N times and normalized by $1/N$. Our baseline, which we call ConvNeXt-I3D, uses the 2D ConvNeXt pretrained on ImageNet (Section 5.1) with I3D inflation. Notably, utilizing S4ND in 3D enables global context *temporally* as well. We additionally test the more modern spatial-temporal separated 3D convolution used by S3D [68] and R(2+1)D [65], which factor the 3D convolution kernel as the outer product of a 2D (spatial) by 1D (temporal) kernel. Because of its flexible factored form (Section 4.1), S4ND automatically has these inflation capabilities. We inflate the pretrained S4ND-ConvNeXt simply by loading the pretrained 2D model weights for the spatial dimensions, and initializing the temporal kernel parameters $\boldsymbol{A}^{(3)}, \boldsymbol{B}^{(3)}, \boldsymbol{C}^{(3)}$ from scratch. We note that this model is essentially identical to the baseline ConvNeXt-S3D except that each component of the factored kernels use standard 1D S4 layers instead of 1D local convolutions. Finally, by varying the initialization of these parameters, we can investigate additional factors affecting model training; in particular, we also run an ablation on the kernel timescales Δ.

Training. Our training procedure is minimal, using only RGB frames (no optical flow). We sample clips of 2 seconds with 30 total frames at 224×224, followed by applying RandAugment; we performed a small sweep of the RandAugment magnitude for each model. All models are trained with learning rate 0.0001 and weight decay 0.2. Additional details are included in Appendix B.2.

Table 2: (**HMDB-51 Activity Recognition with ImageNet-pretrained models.**) (*Left*) Top-1 accuracy with 2D to 3SD kernel inflation. (*Right*) Ablation of initial temporal kernel lengths, controlled by S4's Δ parameter.

	PARAMS	FLOW	RGB
Inception-I3D	25.0M	61.9	49.8
ConvNeXt-I3D	28.5M	-	58.1
ConvNeXt-S3D	27.9M	-	58.6
S4ND-ConvNeXt-3D	31.4M	-	**62.1**

INIT. LENGTH	ACC
20.0	53.74
4.0	58.33
2.0	60.30
1.0	62.07

Results. Results are presented in Table 2. Our baselines are much stronger than prior work in this setting, 8% top-1 accuracy higher than the original I3D model in the RGB frames only setting, and confirming that separable kernels (S3D) perform at least as well as standard inflation ($+0.53\%$). S4ND-ConvNeXt-3D improves over the baseline ConvNeXt-I3D by $+4.0\%$ with no difference in models other than using a temporal S4 kernel. This even exceeds the performance of I3D when trained on optical flow.

Finally, we show how S4ND's parameters can control for factors such as the kernel length (Table 2). Note that our temporal kernels $K^3(t^3)$ are always full length (30 frames in this case), while standard convolution kernels are shorter temporally and require setting the width of each layer manually as a hyperparameter [68]. S4 layers have a parameter Δ that can be interpreted such that $\frac{1}{\Delta}$ is the expected length of the kernel. By simply adjusting this hyperparameter, S4ND can be essentially initialized with length-1 temporal kernels that can automatically learn to cover the whole temporal length if needed. We hypothesize that this contributes to S4ND's improved performance over baselines.

Table 3: (**Settings for continuous capabilities experiments.**) Datasets and resolutions used for continuous capabilities experiments, as well as the model backbones used are summarized.

Dataset	Classes	Resolution			Backbone
		base	mid	low	
CIFAR-10	10	32×32	16×16 ($2\times$)	8×8 ($4\times$)	Isotropic
Celeb-A	40 multilabel	160×160	128×128 ($1.25\times$)	64×64 ($2.50\times$)	ConvNeXt

Table 4: **Zero-Shot Resolution Change.** Results for models trained on one resolution (one of low / mid / base), and zero-shot tested on another. Results are averaged over 2 random seeds.

Resolution		CIFAR-10			Celeb-A	
Train	Test	S4ND	Conv2D	FlexNet-16	S4ND	Conv2D
base	base	$\mathbf{93.10 \pm 0.22}$	91.9 ± 0.2	92.2 ± 0.1	$\mathbf{91.75 \pm 0.00}$	91.44 ± 0.03
mid	mid	$\mathbf{88.80 \pm 0.12}$	87.2 ± 0.1	86.5 ± 2.0	$\mathbf{91.63 \pm 0.04}$	91.09 ± 0.08
mid	base	$\mathbf{88.77 \pm 0.03}$	73.1 ± 0.3	82.7 ± 2.0	$\mathbf{90.14 \pm 0.38}$	80.52 ± 0.08
low	low	$\mathbf{78.17 \pm 0.13}$	76.0 ± 0.2	-	$\mathbf{90.95 \pm 0.02}$	90.37 ± 0.04
low	mid	$\mathbf{78.86 \pm 0.22}$	57.4 ± 0.3	-	$\mathbf{84.44 \pm 1.04}$	80.45 ± 0.11
low	base	$\mathbf{73.71 \pm 0.47}$	33.1 ± 1.3	-	$\mathbf{84.73 \pm 0.54}$	80.59 ± 0.14

5.3 Continuous-signal Capabilities for Images

Images are often collected at varied resolutions, even with the same hardware, so it is desirable that models generalize to data sampled at different resolutions. We show that S4ND inherits this capability as a continuous-signal model, with strong zero-shot performance when changing resolutions, and the ability to train with progressively resized multi-resolution data. We perform an ablation to show our proposed bandlimiting modification is critical to achieving strong performance when changing resolutions.

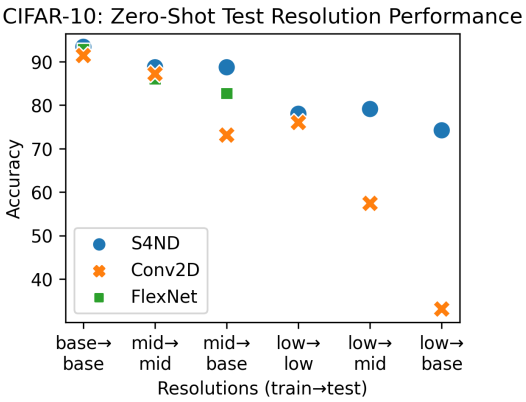


Figure 3: (**CIFAR-10 zero-shot comparison.**) When trained and tested on the same resolutions, all models have similar performance (with S4ND slightly better). But, when trained and tested on different resolutions (the zero-shot setting), S4ND significantly outperforms Conv2D and FlexNet.

Setup. We focus on image classification on 2D benchmark datasets: a dataset with low-resolution images (CIFAR-10) and one with higher-resolution images (Celeb-A). For each dataset, we specify a base image resolution (base), and two lower resolutions (mid, low), summarized in Table 3. To highlight that S4ND's continuous capabilities are independent of backbone, we experiment with 2 different 2D backbones: an isotropic, fixed-width model backbone on CIFAR-10 [36] and a small ConvNeXt backbone on Celeb-A [43]. For each backbone, we compare S4ND's performance to Conv2D layers as a standard, widely used baseline. Additional details can be found in Appendix B.3.

We first verify that S4ND achieves comparable test classification performance with baseline Conv2D models. Table 4 and Fig. 3 show that we exceed the performance of Conv2D on both tasks, with S4ND improving over Conv2D models by 0.4% on CIFAR-10 and 0.3% on Celeb-A.

Zero-Shot Resolution Change. We train S4ND and Conv2D models at either low or mid resolution, and test them at the base resolution for each dataset. We also compare to FlexConv [50] on CIFAR-10, which is the current SotA for zero-shot resolution change. Compared to training and testing

at the base resolution, we expect that Conv2D should degrade more strongly than S4ND, since it cannot adapt its kernel appropriately to the changed resolution. Table 4 and Fig. 3 show that S4ND outperforms Conv2D on mid $\rightarrow$ base by $15+$ points and low $\rightarrow$ base by $40+$ points on CIFAR-10, and $9+$ points and $3+$ points on Celeb-A. In fact, S4ND yields better performance on low $\rightarrow$ base (a more difficult task) than Conv2D does from mid $\rightarrow$ base (an easier task), improving by 1.1% on CIFAR-10 and 3% on Celeb-A. Compared to FlexConv on CIFAR-10 mid $\rightarrow$ base, S4ND improves zero-shot performance by $5+$ points, setting a new SotA.

Progressively Resized Training. We provide an exploration of training with progressive resizing [14, 60] i.e. training in multiple stages at different resolutions. The only change we make from standard training is to reset the learning rate scheduler at the beginning of each stage (details in Appendix B.3). We compare S4ND and the Conv2D baseline with progressive resizing in Table 5.

For CIFAR-10, we train with a low $\rightarrow$ base, $80 - 20$ epoch schedule, and perform within $\sim 1\%$ of an S4ND model trained with base resolution data while speeding up training by 21.8%. We note that Conv2D attains much higher speedups as a consequence of highly optimized implementations, which we discuss in more detail in Section 6. For Celeb-A, we explore flexibly combining the benefits of both progressive resizing and zero-shot testing, training with a low $\rightarrow$ mid, $16 - 4$ epoch schedule that uses no base data. We outperform Conv2D by $7.5\%+$, and attain large speedups of $50\%+$ over training at the base resolution.

Table 5: (**Progressive resizing results.**) Validation performance for progressively resized training at base resolution, and speedup compared to training at base resolution on CIFAR-10 and Celeb-A. We use a $80 - 20$ and $16 - 4$ schedule for CIFAR-10 and Celeb-A, and also report performance training only at base resolution.

Dataset	Model	Epoch Schedule	Train Resolution	Val @ Base Res.	Speedup (Step Time)
CIFAR-10	Conv2D	-	base	91.90%	0%
	S4ND	-	base	**93.40%**	0%
	Conv2D	$80 - 20$	low$\rightarrow$ base	90.94%	51.7%
	S4ND	$80 - 20$	low$\rightarrow$ base	**92.32%**	21.8%
CelebA	Conv2D	-	base	91.44%	0%
	S4ND	-	base	**91.75%**	0%
	Conv2D	$16 - 4$	low$\rightarrow$ mid	80.89%	76.7%
	S4ND	$16 - 4$	low$\rightarrow$ mid	**88.57%**	57.3%

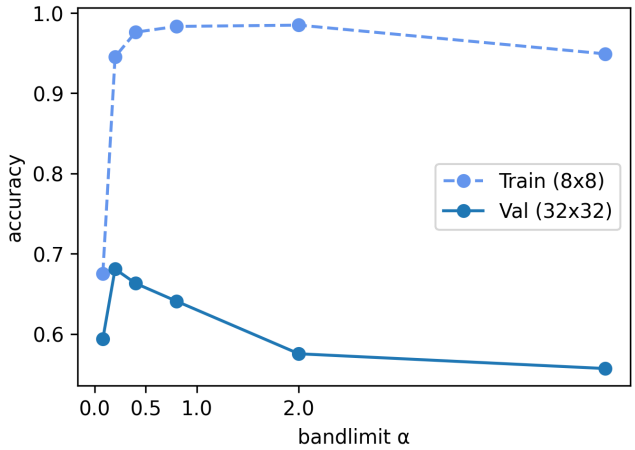
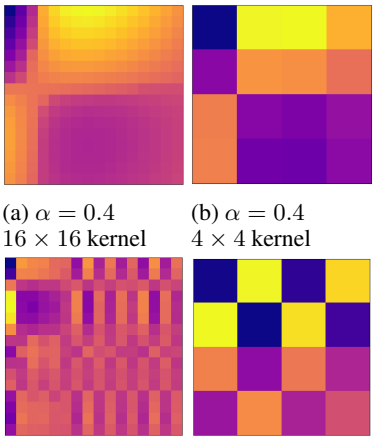


Figure 4: (**Bandlimiting ablation.**) Zero-shot performance training at 8×8 and evaluating at 32×32. Train performance is high for large enough values of $\alpha \geq 0.5$, but validation performance goes down as aliasing occurs in the kernel. Both train and validation drop for lower values of α, as it limits the expressivity of the kernel.



(a) $\alpha = 0.4$ 16×16 kernel (b) $\alpha = 0.4$ 4×4 kernel

(c) $\alpha = \infty$ 16×16 kernel (d) $\alpha = \infty$ 4×4 kernel

Figure 5: (**Effect of bandlimiting on learned kernels.**) Bandlimiting significantly increases the smoothness of the kernels when resizing resolutions.

Effect of Bandlimiting. Bandlimiting in S4ND is critical to generalization at different resolutions. We analyze the effect of the bandlimiting parameter α on CIFAR-10 performance when doing zero-

shot resolution change. We additionally vary the choice of basis function $K_n(t)$ used in the SSM. Fig. 4 shows that zero-shot performance on base degrades for larger values of α, i.e. for cutoffs that do not remove high frequencies that violate the Nyquist cutoff. As we would expect, this holds regardless of the choice of basis function. In Fig. 5, we visualize learned kernels with and without bandlimiting, showing that bandlimiting improves smoothness. Appendix B.3 includes additional experiments that analyze α on Celeb-A, and when doing progressively resized training.

6 Discussion

Summary. We introduced S4ND, a multidimensional extension of S4 that models visual data as continuous valued signals. S4ND is the first continuous model that matches SotA baselines on large-scale 1D and 2D image classification on ImageNet-1k, as well as outperforming a strong pretrained model in a 3D video classification setting. As a continuous-signal model, S4ND inherits useful properties that are absent from standard visual modeling approaches, such as zero-shot testing on unseen resolutions without a significant performance drop.

Limitations. A limitation of S4ND is its training speed in high dimensions. In the 1D image setting, S4ND-ViT has similar training speed to ViT; however, in the 2D setting, the S4ND-ConvNeXt was $2\times$ slower than the baseline ConvNeXt. We remark that vanilla local convolutions have been heavily optimized for years, and we expect that layers such as S4ND can be substantially sped up with more optimized implementations. Our core computational primitives accounting for 65% of our runtime (FFT, pointwise operations, inverse FFT) are all bottlenecked by reading from and writing to GPU memory [57]. With a more optimized implementation that fuses these operations [9] (i.e., loading the input once from GPU memory, perform all operations, then write result back to GPU memory), we expect to speed these up by 2-3$\times$. Further discussion can be found in Appendix C.

Future work. We presented a first step in using continuous-signal models in images and videos, and believe this opens the door to new capabilities and directions. For example, recent video benchmark datasets are significantly larger than the HMDB-51 dataset used in our experiments [34, 18, 8, 46, 55], and can be explored. In addition, we demonstrated capabilities in zero-shot resolution *spatially*, but even less work has been done on zero-shot testing *temporally*. This opens up an exciting new direction of work that could allow models to be agnostic to different video sampling rates as well, capable of testing on higher unseen sampling rates, or irregular (non-uniform) sampling rates. As models become larger and combine multiple modalities, S4ND has shown strong promise in being able to better model underlying continuous-signals and create new capabilities across visual data, audio, time-series (e.g., wearable sensors, health data), and beyond.

Acknowledgments

We thank Arjun Desai, Gautam Machiraju, Khaled Saab and Vishnu Sarukkai for helpful feedback on earlier drafts. This work was done with the support from the HAI-Google Cloud Credits Grant Program. We gratefully acknowledge the support of NIH under No. U54EB020405 (Mobilize), NSF under Nos. CCF1763315 (Beyond Sparsity), CCF1563078 (Volume to Velocity), and 1937301 (RTML); ONR under No. N000141712266 (Unifying Weak Supervision); ONR N00014-20-1-2480: Understanding and Applying Non-Euclidean Geometry in Machine Learning; N000142012275 (NEPTUNE); the Moore Foundation, NXP, Xilinx, LETI-CEA, Intel, IBM, Microsoft, NEC, Toshiba, TSMC, ARM, Hitachi, BASF, Accenture, Ericsson, Qualcomm, Analog Devices, the Okawa Foundation, American Family Insurance, Google Cloud, Salesforce, Total, the HAI-AWS Cloud Credits for Research program, the Stanford Data Science Initiative (SDSI), and members of the Stanford DAWN project: Facebook, Google, and VMWare. The Mobilize Center is a Biomedical Technology Resource Center, funded by the NIH National Institute of Biomedical Imaging and Bioengineering through Grant P41EB027060. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of NIH, ONR, or the U.S. Government.

References

[1] Hassan Akbari, Liangzhe Yuan, Rui Qian, Wei-Hong Chuang, Shih-Fu Chang, Yin Cui, and Boqing Gong. Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text. *Advances in Neural Information Processing Systems*, 34, 2021.

[2] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6836–6846, 2021.

[3] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

[4] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[5] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[6] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[7] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[8] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018.

[9] Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *arXiv preprint arXiv:2205.14135*, 2022.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[11] Arjun D Desai, Beliz Gunel, Batu M Ozturkler, Harris Beg, Shreyas Vasanawala, Brian A Hargreaves, Christopher Ré, John M Pauly, and Akshay S Chaudhari. Vortex: Physics-driven data augmentations for consistency training for robust accelerated mri reconstruction. *arXiv preprint arXiv:2111.02549*, 2021.

[12] Mingyu Ding, Bin Xiao, Noel Codella, Ping Luo, Jingdong Wang, and Lu Yuan. Davit: Dual attention vision transformers. *arXiv preprint arXiv:2204.03645*, 2022.

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

[14] fastai. Training imagenet in 3 hours for 25 minutes. `https://www.fast.ai/2018/04/30/dawnbench-fastai/`, 2018.

[15] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211, 2019.

[16] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *ICML*, 2020.

[17] Karan Goel, Albert Gu, Chris Donahue, and Christopher Ré. It's raw! audio generation with state-space models. *arXiv preprint arXiv:2202.09729*, 2022.

[18] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850, 2017.

[19] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/102f0bb6efb3a6128a3c750dd16729be-Abstract.html`.

[20] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with the linear state space layer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[21] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The International Conference on Learning Representations (ICLR)*, 2022.

[22] Albert Gu, Isys Johnson, Aman Timalsina, Atri Rudra, and Christopher Ré. How to train your hippo: State space models with generalized orthogonal basis projections. *arXiv preprint arXiv:2206.12037*, 2022.

[23] Ankit Gupta. Diagonal state spaces are as effective as structured state spaces. *arXiv preprint arXiv:2203.14343*, 2022.

[24] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 3154–3160, 2017.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[26] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[28] Elad Hoffer, Berry Weinstein, Itay Hubara, Tal Ben-Nun, Torsten Hoefler, and Daniel Soudry. Mix & match: training convnets with mixed image sizes for improved accuracy, speed and scale resiliency. *arXiv preprint arXiv:1908.08986*, 2019.

[29] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8129–8138, 2020.

[30] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016.

[31] Md Mohaiminul Islam and Gedas Bertasius. Long movie clip classification with state-space video models. *arXiv preprint arXiv:2204.01692*, 2022.

[32] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1): 221–231, 2012.

[33] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[34] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[35] Dan Kondratyuk, Liangzhe Yuan, Yandong Li, Li Zhang, Mingxing Tan, Matthew Brown, and Boqing Gong. Movinets: Mobile video networks for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16020–16030, 2021.

[36] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Citeseer*, 2009.

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84 – 90, 2012.

[38] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 International conference on computer vision*, pages 2556–2563. IEEE, 2011.

[39] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. *arXiv preprint arXiv:2111.09883*, 2021.

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

[41] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021.

[42] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[43] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, pages 3730–3738. IEEE Computer Society, 2015. ISBN 978-1-4673-8391-2. URL `http://dblp.uni-trier.de/db/conf/iccv/iccv2015.html#LiuLWT15`.

[44] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[45] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020.

[46] Mathew Monfort, Alex Andonian, Bolei Zhou, Kandan Ramakrishnan, Sarah Adel Bargal, Tom Yan, Lisa Brown, Quanfu Fan, Dan Gutfreund, Carl Vondrick, et al. Moments in time dataset: one million videos for event understanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(2):502–508, 2019.

[47] Bruno A Olshausen and David J Field. Natural image statistics and efficient coding. *Network: computation in neural systems*, 7(2):333, 1996.

[48] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

[49] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *proceedings of the IEEE International Conference on Computer Vision*, pages 5533–5541, 2017.

[50] David W. Romero, Robert-Jan Bruintjes, Jakub M. Tomczak, Erik J. Bekkers, Mark Hoogendoorn, and Jan C. van Gemert. Flexconv: Continuous kernel convolutions with differentiable kernel sizes. *ArXiv*, abs/2110.08059, 2021.

[51] David W Romero, Anna Kuzina, Erik J Bekkers, Jakub M Tomczak, and Mark Hoogendoorn. Ckconv: Continuous kernel convolution for sequential data. *arXiv preprint arXiv:2102.02611*, 2021.

[52] Daniel Ruderman and William Bialek. Statistics of natural images: Scaling in the woods. *Advances in neural information processing systems*, 6, 1993.

[53] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015.

[54] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *NeurIPS*, 2017.

[55] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526. Springer, 2016.

[56] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[57] David Střelák and Jiří Filipovič. Performance analysis and autotuning setup of the cufft library. In *Proceedings of the 2nd Workshop on AutotuniNg and aDaptivity AppRoaches for Energy efficient HPC Systems*, pages 1–6, 2018.

[58] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[59] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[60] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International Conference on Machine Learning*, pages 10096–10106. PMLR, 2021.

[61] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy. *Advances in neural information processing systems*, 32, 2019.

[62] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[63] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 32–42, 2021.

[64] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[65] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[66] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[67] Chao-Yuan Wu and Philipp Krahenbuhl. Towards long-form video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1884–1894, 2021.

[68] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. *arXiv preprint arXiv:1712.04851*, 1(2):5, 2017.

[69] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 558–567, 2021.

[70] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

[71] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. *CoRR*, abs/2106.04560, 2021.

[72] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[73] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The main method can be implemented based on the released S4 repository, and we will publically release our code. All hyperparameters and experimental details are reported in the Appendix.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Yes
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Full experimental results including standard deviations can be found in the Appendix.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Reported in the Appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Theory Details

We formalize and generalize the notation of Section 4.1 and prove the results.

For the remainder of this section, we fix a dimension D (e.g. $D = 2$ for a 2-dimensional SSM). The D-dimensional SSM will be a map from a function $u : \mathbb{R}^D \to \mathbb{C}$ to $y : \mathbb{R}^D \to \mathbb{C}$.

Definition 2 (Indexing notation)**.** *Let $[d]$ denote the set $\{1, 2, \ldots, d\}$. Let $[0]$ denote the empty set $\{\}$. Given a subset $I \subseteq [D]$, let $-I$ denote its complement $[D] \setminus I$. Let $[-d] = -[d] = \{d+1, \ldots, D\}$.*

Definition 3 (Tensor products and contractions)**.** *Given $a \in \mathbb{C}^{N_1}$ and $b \in \mathbb{C}^{N_2}$, let $a \otimes b \in \mathbb{C}^{N_1 \times N_2}$ be defined as $(a \otimes b)_{n_1,n_2} = a_{n_1} b_{n_2}$.*

Given a tensor $x \in \mathbb{C}^{N_1 \times \cdots \times N_D}$ and matrix $\boldsymbol{A} \in \mathbb{C}^{N_1 \times N_1}$, define $\boldsymbol{A} \cdot^{(1)} x \in \mathbb{C}^{N_1 \times \cdots \times N_D}$ as

$$(\boldsymbol{A} \cdot^{(1)} x)_{n_1,\ldots,n_D} = \sum_m \boldsymbol{A}_{n_1 m} x_{k,n_2,\ldots,n_D}.$$

which is simply a matrix multiplication over the first dimension. Let $\cdot^{(\tau)}$ similarly denote matrix multiplication over any other axis.

It will be easier to work directly from the convolutional definition of SSM (equation (5)). We will then show equivalence to a PDE formulation (equation (4)).

Definition 4 (Multidimensional SSM)**.** *Let $N_1, \ldots, N_D$ be state sizes for each of the D dimensions. The D-dimensional SSM has parameters $\boldsymbol{A}^{(\tau)} \in \mathbb{C}^{N^{(\tau)} \times N^{(\tau)}}$, $\boldsymbol{B}^{(\tau)} \in \mathbb{C}^{N^{(\tau)}}$, and $\boldsymbol{C} \in \mathbb{C}^{N^{(1)} \times \cdots \times N^{(D)}}$ for each dimension $\tau \in [D]$.*

and is defined by the map $u \mapsto y$ given by

$$\begin{aligned} x(t) &= (u * \bigotimes_{\tau=1}^{D} e^{t^{(\tau)} \boldsymbol{A}^{(\tau)}} \boldsymbol{B}^{(\tau)})(t) \\ y(t) &= \langle \boldsymbol{C}, x(t) \rangle. \end{aligned}$$

Note that at all times $t = (t^{(1)}, \ldots, t^{(D)})$, the dimension of the state is $x(t) \in \mathbb{C}^{N_1 \times \cdots \times N_D}$.

Finally, it will be convenient for us to define "partial bindings" of u, x, y where one or more of the coordinates are fixed.

Definition 5 (Partial binding of u)**.** *Let $I \subseteq [D]$ and let $t^I = (t^{(I_1)}, \ldots, t^{(I_{|I|})})$ be an partial assignment of the time variables. Define $u_I(t^I)$ to be the function $\mathbb{R}^{D-|I|} \mapsto \mathbb{C}$ where the I indices are fixed to t^I.*

For example, if $D = 3$ and $I = [1]$, then

$$u_I(t^I) = u(t^{(1)}, \cdot, \cdot)$$

is a function from $\mathbb{R}^2 \to \mathbb{C}$ mapping $(t^{(2)}, t^{(3)}) \mapsto u(t^{(1)}, t^{(2)}, t^{(3)})$.

Definition 6 (Partial binding of x)**.**

$$x_I(t^I)(t^{-I}) = (u_I(t^I) * \bigotimes_{\tau \in -I} e^{t^{(\tau)} \boldsymbol{A}^{(\tau)}} \boldsymbol{B}^{(\tau)})(t^{-I})$$

Note that $x_{[0]}() : \mathbb{R}^D \mapsto \mathbb{C}^{N_1 \times \cdots \times N_D}$ coincides with the full state x.

The following more formal theorem shows the equivalence of this convolutional LTI system with a standard 1D SSM differential equation in each dimension.

Theorem 2. *Given a partial state $x_I(t^I)$ and a time variable $t^{(\tau)}$ for $\tau \notin I$, let $I' = I \cup \{\tau\}$. Then the partial derivatives satisfy*

$$\begin{aligned} \frac{\partial x_I(t^I)}{\partial t^{(\tau)}}(t^{-I}) = \boldsymbol{A}^{(\tau)} \cdot^{(\tau)} x_I(t^I)(t^{-I}) \\ + \boldsymbol{B}^{(\tau)} \otimes x_{I'}(t^{I'})(t^{-I'}) \end{aligned}$$

Proof. WLOG assume $I = [d]$, since all notions are permutation-independent. We will consider differentiating the state x_I with respect to the time variable $t^{(d+1)}$. The key fact is that differentiating a convolution $\frac{d}{dt}(f * g)$ is equivalent to differentiating one of the operands $f * (\frac{d}{dt} g)$.

$$
\begin{aligned}
\frac{\partial x_{[d]}(t^{[d]})}{\partial t^{(d+1)}}(t^{-[d]}) &= \frac{\partial}{\partial t^{(d+1)}}(u_I(t^I) * \bigotimes_{\tau \in -I} e^{t^{(\tau)}\boldsymbol{A}^{(\tau)}}\boldsymbol{B}^{(\tau)})(t^{-I}) \\
&= (u_I(t^I) * \frac{\partial}{\partial t^{(d+1)}} \bigotimes_{\tau \in -[d]} e^{t^{(\tau)}\boldsymbol{A}^{(\tau)}}\boldsymbol{B}^{(\tau)})(t^{-I}) \\
&= \left(u_I(t^I) * \left(\boldsymbol{A}^{(d+1)} e^{t^{(d+1)}\boldsymbol{A}^{(d+1)}}\boldsymbol{B}^{(d+1)} + \boldsymbol{B}^{(d+1)}\delta(t^{(d+1)}) \right) \bigotimes_{\tau \in -[d+1]} e^{t^{(\tau)}\boldsymbol{A}^{(\tau)}}\boldsymbol{B}^{(\tau)} \right)(t^{-I}) \\
&= \boldsymbol{A}^{(d+1)} \cdot^{(d+1)} \left(u_I(t^I) * \bigotimes_{\tau \in -[d]} e^{t^{(\tau)}\boldsymbol{A}^{(\tau)}}\boldsymbol{B}^{(\tau)} \right)(t^{-I}) \\
&\quad + u_{[d+1]}(t^{[d+1]})\boldsymbol{B}^{(d+1)} \bigotimes_{\tau \in -[d+1]} e^{t^{(\tau)}\boldsymbol{A}^{(\tau)}}\boldsymbol{B}^{(d+1)} \\
&= \boldsymbol{A}^{(d+1)} \cdot^{(d+1)} x_{[d]}(t^{[d]})(t^{-[d]}) \\
&\quad + \boldsymbol{B}^{(d+1)} \otimes x_{[d+1]}(t^{[d+1]})(t^{-[d+1]})
\end{aligned}
$$

□

The following corollary follows immediately from linearity of the convolution operator, allowing the order of convolution and inner product by $\boldsymbol{C}$ to be switched.

Corollary A.1. *The output y is equivalent to $y = K * u$ where*

$$K(t) = \langle \boldsymbol{C}, \bigotimes_{\tau=1}^{D} e^{t^{(\tau)}\boldsymbol{A}^{(\tau)}}\boldsymbol{B}^{(\tau)} \rangle$$

This completes the proof of Theorem 1.

Finally, Corollary 4.1 is an immediate consequence of the Kronecker mixed-product identity

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD).$$

B Experimental Details

We use PyTorch for all experiments, and build on the publicly available S4 code.

B.1 Image Classification

ImageNet training: all models were trained from scratch with no outside data using 8 Nvidia A100 GPUs. For both ViT and ConvNeXt experiments, we follow the procedure from T2T-ViT [69] and the original ConvNeXt [42], respectively, with minor adjustments. Preprocessing and dataloading was done using the TIMM [66] library. In S4ND-ViT, we turn off weight decay and remove the class token prepending of the input sequence. For the ConvNeXt models, we add RepeatAug [29], as well as reduce the batch size to 3840 for S4ND-ConvNeXt.

S4ND specific settings include a bidirectional S4 kernel followed by Goel et al [17], and a state dimension of 64 for the SSMs.

B.2 Video Classification

HMDB-51 training: all models were trained from scratch on a single Nvidia A100 GPU. We use the Pytorchvideo library for data loading and minimal preprocessing of RGB frames only (no optical

Table 6: (**Performance on image classification.**) ImageNet settings for ViT and ConvNeXt baseline models.

	ViT	ConvNeXt
image size	224^2	224^2
optimizer	AdamW	AdamW
optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.999$	$\beta_1, \beta_2 = 0.9, 0.999$
weight init	trunc. normal (std=0.02)	trunc. normal (std=0.02)
base learning rate	0.001	0.004
weight decay	0.05	0.05
dropout	None	None
batch size	4096	4096
training epochs	300	300
learning rate schedule	cosine decay	cosine decay
warmup epochs	10	20
warmup schedule	linear	linear
layer-wise lr decay [3, 6]	None	None
randaugment [7]	(9,0.5,layers=2)	(9,0.5,layers=2)
mixup [72]	0.8	0.8
cutmix [70]	1.0	1.0
repeataug [29]	None	3
random erasing [73]	0.25	0.25
label smoothing [59]	0.1	0.1
stochastic depth [30]	0.1	0.1
layer scale [63]	None	1e-6
head init scale [63]	None	None
exp.mov. avg (EMA) [48]	0.9999	None

flow). During training, we randomly sample 2 second clips from each video. At validation and test time, 2 second clips are sampled uniformly to ensure that an entire video is seen. For each ConvNeXt 3D video model (baselines and S4ND versions), we fix the hyperparameters in Table 7, and do a sweep over the RandAugment[7] settings `num_layers`$= \{1, 2\}$ and `magnitude`$= \{3, 5, 7, 9\}$.

S4ND specific settings were similar to image classification, a bidirectional S4 kernel and a state dimension of 64 for the SSMs.

Table 7: (**Performance on video classification.**) HMDB-51 settings for all 3D ConvNeXt models (baselines and S4ND models).

	ConvNeXt 3D
image size	224^2
# frames	30
clip duration (sec)	2
optimizer	AdamW
optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.999$
weight init	trunc. normal (std=0.02)
base learning rate	0.0001
weight decay	0.2
dropout (head)	0.2
batch size	64
training epochs	50
learning rate schedule	cosine decay
warmup epochs	0
stochastic depth [30]	0.2
layer scale [63]	1e-6
head init scale [63]	None
exp.mov. avg (EMA) [48]	None

B.3 Continuous Time Capabilities Experiments

We use the CIFAR-10 and Celeb-A datasets for all our experiments. Below, we include all experimental details including data processing, model training and hyperparameters, and evaluation details.

As noted in Table 3, we consider 3 resolutions for each dataset, a base resolution that is considered the standard resolution for that dataset, as well as two lower resolutions low and mid. Images at the lower resolutions are generated by taking an image at the base resolution, and then downsampling using the `resize` function in `torchvision`, with bilinear interpolation and antialiasing turned on.

CIFAR-10. The base resolution is chosen to be 32×32, which is the resolution at which models are generally trained on this dataset. The lower resolutions are 16×16 and 8×8. We use no data augmentation for either zero-shot resolution change or progressive resizing experiments. We train with standard cross-entropy loss on the task of 10-way classification. For reporting results, we use standard accuracy over the 10 classes.

We use a simple isotropic model backbone, identical to the one used by Gu et al. [21] for their (1D) S4 model. The only difference is that we use either the S4ND or Conv2D layer in place of the S4 block, which ensures that the model can accept a batch of 2D spatial inputs with channels. For results reported in the main body, we use a 6×256 architecture consisting of 6 layers and a model dimension of 256.

For all CIFAR-10 experiments, we train for 100 epochs, use a base learning rate of 0.01 and a weight decay of 0.03.

Celeb-A. On Celeb-A, we consider 160×160 as the base resolution, and 128×128 and 64×64 as the lower resolutions. Images on Celeb-A are generally 218×178, and we run a $\text{CenterCrop}(178) \rightarrow \text{Resize}(160)$ transform in `torchvision` to resize the image. We use no data augmentation for either zero-shot resolution change or progressive resizing experiments. We train with standard binary cross-entropy loss on the task of 40-way multilabel attribute classification on Celeb-A. For reporting results, we use the multilabel binary accuracy, i.e. the average binary accuracy across all 40 tasks.

Similar to our ImageNet experiments, we use a ConvNeXt model as the basic backbone for experiments on Celeb-A. However, use a particularly small model here, consisting of 4 stages with depths $(3, 3, 3, 3)$ and corresponding model dimensions $(64, 128, 256, 512)$. For S4ND, we simply follow the same process as ImageNet, replacing all depthwise Conv2D layers with S4ND. A minor difference is that we use a global convolution in the stem downsampling for S4ND, rather than the smaller kernel size that we use for ImageNet. We use a drop path rate of 0.1 for both S4ND and the Conv2D baseline in all experiments.

For all Celeb-A experiments, we train for 20 epochs, use a base learning rate of 0.004, and use automatic mixed precision for training.

Other fixed hyperparameters. For all experiments, we use bidirectional S4 kernels following Goel et al. [17], and use a state dimension of 64 for the S4 SSMs. We use AdamW as the optimizer.

B.3.1 Zero-Shot Resolution Change

For Table 4, we simply train on a single resolution among base, mid and low, and then directly test at all higher resolutions. For model selection at a particular test resolution, we select the best performing model for that test resolution i.e. we checkpoint the model at the epoch where it performs best at the test resolution in question. Note that we refer to "test resolutions" but report validation metrics in Table 4, as is standard practice for additional experiments and ablations.

CIFAR-10. We use a batch size of 50, and for both methods (and all resolutions), we use a cosine decay schedule for the learning rate with no restarts and a length of 100000, and a linear warmup of 500 steps.

For baseline Conv2D hyperparameters, we compare the performance of Conv2D with and without depthwise convolutions. Otherwise, all hyperparameters are fixed to the common values laid out in the previous section (and are identical to those used for S4ND).

For S4ND hyperparameters, we sweep the choice of initialization for the state space parameters A, B in the S4 model among {legs, fourier, random-linear, random-inv}[2] and sweep bandlimit (α) values among $\{0.05, 0.10, 0.20, 0.50, \infty\}$.

Celeb-A. We use a batch size of 256 when training on the lower resolutions, and 128 when training at the base resolution. For both methods (and all resolutions), we use a cosine decay schedule for the learning rate with no restarts and a length of 13000 (batch size 256) or 26000 (batch size 128), and a linear warmup of 500 steps.

For baseline ConvNeXt hyperparameters, we use a weight decay of 0.1 after an initial sweep over weight decay values $\{0.05, 0.1, 0.2, 0.5\}$.

For S4ND hyperparameters, we sweep the choice of initialization for the state space parameters A, B in the S4 model among {legs, fourier, random-linear, random-inv}, sweep bandlimit (α) values among $\{0.05, 0.10, 0.20, 0.50, \infty\}$, and use a weight decay of 1.0. The value of the weight decay was chosen based on an initial exploratory sweep over weight decay values $\{0.1, 0.2, 0.5, 1.0, 5.0\}$.

B.3.2 Progressive Resizing

For Table 5, we train with a resizing schedule that progressively increases the resolution of the data. We report all metrics on validation sets, similar to our zero-shot experiments. We only use 2 resizing stages in our experiments, as we found in preliminary experiments that $2+$ stages had little to no benefit.

All hyperparameters and training details are identical to the zero-shot resolution change experiments, except for those we describe next. When training with multiple stages, we reset the cosine learning rate scheduler at the beginning of each stage (along with the linear warmup). The length of the scheduler is varied in proportion to the length of the stage, e.g. on CIFAR-10, for a stage 50 epochs long, we change the length of the decay schedule from 100000 steps to 50000 steps (and similarly for Celeb-A). We always use a linear warmup of 500 steps for each stage. Additionally, Table 5 denotes the length (in epochs) of each stage, as well as their training resolutions.

Another important detail is that we set the bandlimit parameter carefully for each stage. We found that bandlimiting in the first stage is critical to final performance, and without bandlimiting the performance on the base resolution is substantially worse. Bandlimiting is generally not useful for a stage if it is training at the base resolution i.e. at the resolution that we will be testing at (in our experiments, this is the case only for second stage training on CIFAR-10).

On CIFAR-10, we use a bandlimit α of 0.10 or 0.20 for the first stage, depending on whether the first stage was training at 8×8 (low) or 16×16 (mid) respectively. In the second stage, we always train at the base resolution and use no bandlimiting, since as stated earlier, it has a negligible effect on performance.

For Celeb-A, we set the bandlimit parameter to 0.1 for both stages. Note that on Celeb-A, we do not train at the base resolution at all, only training at low in the first stage and mid in the second stage.

C Discussion

Runtime profiling and potential optimization. We profile the runtime of an S4ND layer on an A100 GPU, and plot the time taken by each operation in Fig. 6. We see that the majority of the time (between 65% and 80%) are taken by the FFT, pointwise operation, and inverse FFT. The higher-dimensional FFT is a very standard scientific primitive that should be substantially optimizable.

These operations are memory-bound, that is, the runtime is dominated the time to read/write to GPU memory. There is currently no library support for fusing these steps, so for each of those steps the data has to be loaded from GPU memory, arithmetic operations are performed, then the result is written back to GPU memory. One potential optimization is *kernel fusion*: the input could be loaded once, all the arithmetic operations for all three steps are performed, then the final result is written back to GPU memory. We expect that with library support for such optimization, the S4ND runtime can be reduced by 2-3$\times$.

[2] These correspond to initializations of the model that allow the kernel to be expressed as a combination of different types of basis functions. For more details, we refer the reader to Gu et al. [19, 21].

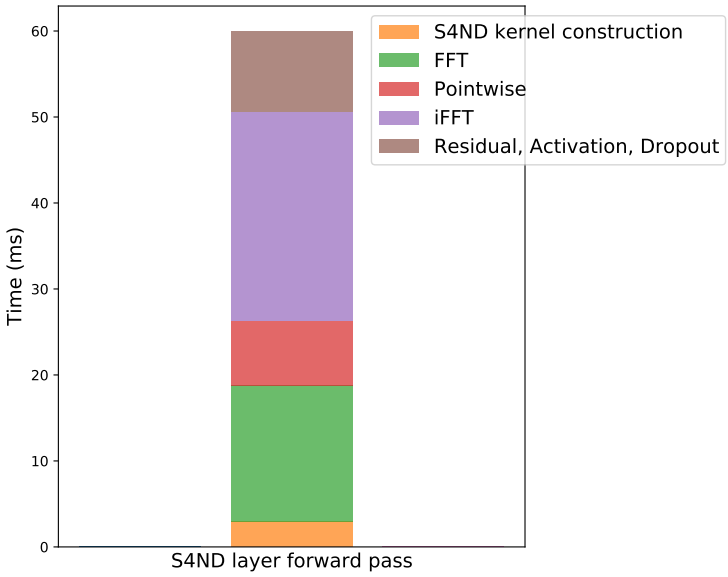


Figure 6: Timing breakdown of different steps in the S4ND forward pass, for a batch of 64 inputs, each of size 224×224, on an A100 GPU. FFT, pointwise operation, and inverse FFT take between 65% and 80% of the time.