LEARNING STRIDES IN CONVOLUTIONAL NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Convolutional neural networks typically contain several downsampling operators, such as strided convolutions or pooling layers, that progressively reduce the resolution of intermediate representations. This provides some shift-invariance while reducing the computational complexity of the whole architecture. A critical hyperparameter of such layers is their stride: the integer factor of downsampling. As strides are not differentiable, finding the best configuration either requires cross-validation or discrete optimization (e.g. architecture search), which rapidly become prohibitive as the search space grows exponentially with the number of downsampling layers. Hence, exploring this search space by gradient descent would allow finding better configurations at a lower computational cost. In this work, we introduce DiffStride, the first downsampling layer with learnable strides. Our layer learns the size of a cropping mask in the Fourier domain, that effectively performs resizing in a differentiable way. Our experiments on audio and image classification show the generality of our solution, which can be used as a drop-in replacement to standard downsampling layers and outperform them. In particular, we show that introducing our layer into a ResNet18 architecture allows keeping a consistent high performance on CIFAR10, CIFAR100 and ImageNet even when training starts from poor random stride configurations. Moreover, formulating strides as learnable variables allows us to introduce a regularization term that controls the computational complexity of the architecture, and we show on the ImageNet classification task how it allows trading off accuracy for efficiency.

1 INTRODUCTION

Convolutional neural networks (CNNs) (Fukushima, 1980; LeCun et al., 1989) have been the most widely used neural architectures across a wide range of tasks, including image classification (Krizhevsky et al., 2012; He et al., 2016a; Huang et al., 2017), audio pattern recognition (Kong et al., 2020), text classification (Conneau et al., 2017), machine translation (Gehring et al., 2017) and speech recognition (Amodei et al., 2016; Sercu et al., 2016; Zeghidour et al., 2018). Convolution layers, which are the building block of CNNs, project input features to a higher-level representation while preserving their resolution. When composed with non-linearities and normalization layers, this allows for learning rich mappings at a constant resolution, e.g. autogressive image synthesis (van den Oord et al., 2016). However, many tasks infer high-level, low-resolution, information (identity of a speaker (Muckenhirn et al., 2018), presence of a face (Chopra et al., 2005)) by integrating over low-level, high-resolution measurements (waveform, pixels). This integration requires extracting the right features, discarding irrelevant information over several downsampling steps. To that end, pooling layers and strided convolutions aggressively reduce the resolution of their inputs, providing several benefits. First, they act as a bottleneck that forces features to focus on information relevant to the task at hand. Second, pooling layers such as lowpass filters (Zhang, 2019) improve shift-invariance. Third, a reduced resolution implies a reduced number of floating-point operations and a higher receptive field in the subsequent layers.

Pooling layers can usually be decomposed into two basic steps: (1) computing local statistics densely over the whole input (2) sub-sampling these statistics by an integer striding factor. Past work has mostly focused on improving (1), proposing better alternatives to max and average pooling that avoid aliasing (Zhang, 2019; Fonseca et al., 2021), preserve the important local details (Saeedan et al., 2018), or adapt to the training data Gulcehre et al. (2014); Lee et al. (2016). Observing

that integer strides reduce resolution too quickly (e.g. a $(2, 2)$ striding reduces the output size by 75%), Graham (2014) proposed fractional max-pooling, that allows for decimal strides, allowing for integration of more downsampling layers into a network. Similarly, Rippel et al. (2015) introduce spectral pooling which, by cropping its inputs in the Fourier domain, performs downsampling with fractional (i.e. rational) strides while emphasizing lower frequencies.

While fractional strides give more flexibility in designing downsampling layers, they increase the size of an already gigantic search space. Indeed, as strides are hyperparameters, finding the best combination requires cross-validation or architecture search (Zoph & Le, 2017; Baker et al., 2017; Tan et al., 2019), which rapidly become infeasible as the number of configurations grows exponentially with the number of downsampling layers. This led Zoph & Le (2017) not to search for strides in most of their experiments. Talebi & Milanfar (2021) and Jin et al. (2021) proposed a neural network that learns a resizing function for natural images, but of which the scaling factor (i.e. the stride) can only be searched by cross-validation. Thus, the nature of strides as hyperparameters — rather than trainable parameters — hinders the discovery of convolutional architectures, even using powerful frameworks such as architecture search, and learning strides by backpropagation would unlock exploration.

In this work, we introduce DiffStride, the first downsampling layer that learns its strides jointly with the rest of the network. To do so, we combine two previously proposed methods. Inspired by Rippel et al. (2015), we cast downsampling in the spatial domain as cropping in the frequency domain. However, and unlike Rippel et al. (2015), rather than cropping with a fixed bounding box controlled by a striding hyperparameter, DiffStride learns the size of its cropping box by backpropagation. To do so, we propose a 2D version of an attention window with learnable size proposed by Sukhbaatar et al. (2019) for language modeling. On five audio classification tasks, using DiffStride as a drop-in replacement to strided convolutions improves performance while providing interpretability on the optimal per-task receptive field. By integrating DiffStride into a ResNet18 (He et al., 2016a), we show on CIFAR (Krizhevsky et al., 2009) and ImageNet (Deng et al., 2009) that even when initializing strides randomly, our model converges to the best performance obtained with the properly cross-validated strides of (He et al., 2016a). Moreover, casting strides as learnable parameters allows us to propose a regularization that directly minimizes computation and memory usage.

2 Methods

In this section, we first present background on spatial and spectral pooling, and propose DiffStride for learning strides of downsampling layers. For simplicity, we focus on 2D CNNs since they are generic enough to be used for image (LeCun et al., 1989; Krizhevsky et al., 2012; He et al., 2016a) and audio (Amodei et al., 2016; Kong et al., 2020) processing (taking time-frequency representation inputs in the latter case). However, these methods are easily applicable to the 1D (e.g. time-series) and 3D (e.g. video) cases.

2.1 Notations

Let $x \in \mathbb{R}^{H \times W}$ denote a two-dimensional real tensor and its Discrete Fourier Transform (DFT) $y = \mathcal{F}(x) \in \mathbb{C}^{H \times W}$ is obtained through the decomposition on a fixed set of basis filters (Lyons, 2004):

$$\mathcal{F}(x)_{mn} = \frac{1}{\sqrt{HW}} \sum_{h=0}^{H-1} \sum_{w=0}^{W-1} x_{hw} e^{-2\pi i \left(\frac{mh}{H} + \frac{nw}{W}\right)}, \forall m \in \{0, \dots, H-1\}, \forall n \in \{0, \dots, W-1\}. \tag{1}$$

The DFT transformation is linear and its inverse is given by its conjugate $\mathcal{F}(.)^{-1} = \mathcal{F}(.)^*$. The Fourier transform of a signal in the real domain $x \in \mathbb{R}^{H \times W}$ is said to be *conjugate symmetric* (Hermitian-symmetry), we can compute only half the positive frequencies for the width dimension and omit the negative frequencies ($y_{mn} = y^*_{(H-m)(W-n)}$). In addition, the DFT is differentiable with regard to its inputs, and since it is a linear operator, the gradient is its conjugate linear, i.e. the inverse DFT. More, formally, if we consider $\mathcal{L} : \mathbb{C}^{H \times W} \longrightarrow: \mathbb{R}$ as a loss taking as input the Fourier representation y, we can compute the gradient of $\mathcal{L}$ with regard to x, by using the inverse DFT:

$$x \in \mathbb{R}^{H \times W}, y = \mathcal{F}(x), \frac{\partial \mathcal{L}}{\partial x} = \mathcal{F}^*(\frac{\partial L}{\partial y}) = \mathcal{F}^{-1}(\frac{\partial L}{\partial y}). \tag{2}$$

We have the similar property for the inverse Fourier transform.

We denote by L the total number of convolution layers in a CNN architecture and each layer is indexed by l. The $\circ$ symbol represents the element-wise product between two tensors, $\otimes$ the outer product between two vectors. S represents the stride parameters, and sg is the stop gradient operator (Bengio et al., 2013; Yin et al., 2019), defined has the identity function during forward pass and with zero partial derivatives.

2.2 Downsampling in convolutional neural networks

An elementary mechanism for downsampling representations in a CNN is strided convolutions which jointly convolve inputs and finite impulse response filters and decimate the output. Alternatively, one can disentangle both operations by first applying a non-strided convolution followed by a pooling operation that computes local statistics (e.g. using an average, max (Boureau et al., 2010)) before decimation. In both settings, downsampling does not benefit from the global structure of its inputs and can discard important information (Hinton, 2014; Saeedan et al., 2018). Moreover, and as observed by Graham (2014), the integer nature of strides only allow for drastic reductions in resolution: a 2D-convolution with strides $S = (2, 2)$ striding reduces the of its inputs by 75%. Furthermore, the number of strides combinations grows exponentially with the number of downsampling layers, such that the space of hyperparameters one can explore by cross-validation or architecture search is very limited. This limitation is likely to translate into lower performance, as Section 3.2 shows that the choice of strides of a ResNet18 architecture can account for an absolute difference of $> 18\%$ in accuracy on CIFAR-100.

2.3 Spectral pooling

Energy of natural signals is typically not uniformly distributed in the frequency domain, with signals such as sounds (Singh & Theunissen, 2003), images (Ruderman, 1994) and surfaces (Kuroki et al., 2018) concentrating most of the information in the lower frequencies. Rippel et al. (2015) build on this observation to introduce *spectral pooling* which alleviates the loss of information of spatial pooling, while enabling fractional downsizing factors.

We consider an input $x \in \mathbb{R}^{H \times W}$ and strides $S = (S_h, S_w) \in [1, H) \times [1, W)$. First, the DFT is computed $y = \mathcal{F}(x) \in \mathbb{C}^{H \times W}$ and for simplicity we assume that the center of this matrix is the DC component — the zero frequency. Then, a bounding box of size $\lfloor \frac{H}{S_h} \rfloor \times \lfloor \frac{W}{S_w} \rfloor$ crops this matrix around its center to produce $\tilde{y} \in \mathbb{C}^{\lfloor \frac{H}{S_h} \rfloor \times \lfloor \frac{W}{S_w} \rfloor}$. Finally, this output is brought back to the spatial domain with an inverse DFT: $\tilde{x} = \mathcal{F}^{-1}(\tilde{y}) \in \mathbb{R}^{\lfloor \frac{H}{S_h} \rfloor \times \lfloor \frac{W}{S_w} \rfloor}$. In practice, x is typically a multi-channel input (i.e. $x \in \mathbb{R}^{H \times W \times C}$) and the same cropping is applied channel-wise. Moreover, since x is real-valued and thanks to Hermitian symmetry (see Section 2.1 for more details), only the positive half of the DFT coefficients are computed, which allows saving computation and memory while ensuring that the output $\tilde{x}$ remains real-valued.

Unlike spatial pooling that requires **integer strides**, spectral pooling only requires **integer output dimensions**, which allows for much more fine-grained downsizing. Moreover, spectral pooling aggregates statistics over the entire input and acts as a low-pass filter, only keeping the lower frequencies i.e. the most relevant information in general. However, and similarly to spatial pooling, spectral pooling is differentiable with respect to its inputs but not with respect to its strides. Thus, one still needs to pass S as hyperparameters for each downsampling layer, with a search space that is even bigger than with spatial pooling, since strides are not constrained to integer values anymore.

2.4 DiffStride

To address the aforementioned limitation of spatial and spectral pooling, we propose DiffStride, a new downsampling layer that adds to the benefits of spectral pooling the ability to learn its strides by backpropagation.

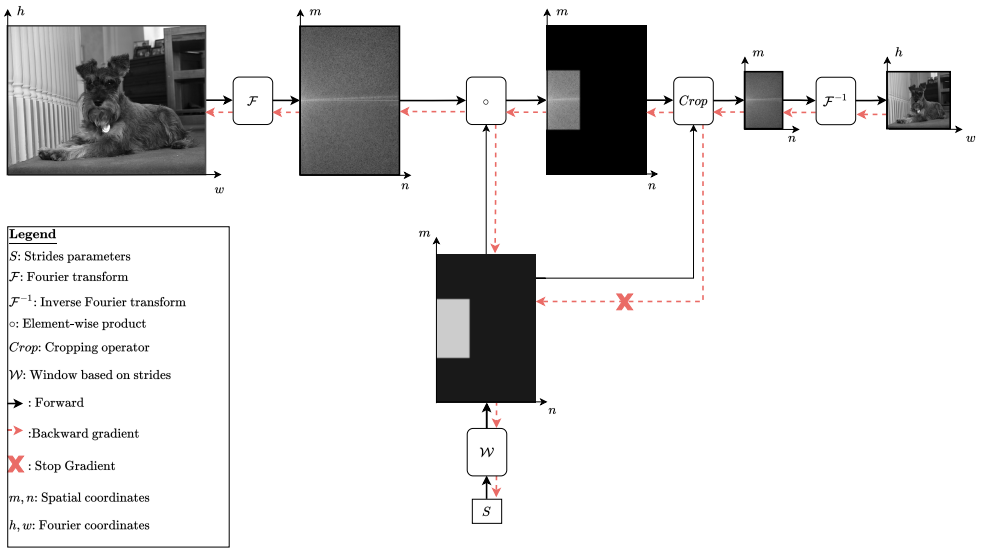


Figure 1: DiffStride forward and backward pass, using a single-channel image for clarity. We only compute the positive half of DFT coefficients due to conjugate symmetry.

To downsample $x \in \mathbb{R}^{H \times W}$, DiffStride performs cropping in the Fourier domain similarly to spectral pooling. However, instead of using a fixed bounding box DiffStride learns its size by backpropagation. This learnable box $\mathcal{W}$ is parametrized by its height and width, both controlled by the strides and the size of the input. We design $\mathcal{W}$ as the outer product between two 1D masking functions, vertical and horizontal. These 1D masks are directly derived from the adaptive span attention introduced by (Sukhbaatar et al., 2019) to learn the attention span of self-attention models for natural language processing.The vertical mask is defined as follows:

$$\text{mask}_{(S_h,H,R)}(m) = \min\left[\max\left[\frac{1}{R}(R + \frac{H}{S_h} - h), 0\right], 1\right], \tag{3}$$

where $S = (S_h, S_w)$ are the strides and R an hyperparameter that controls the smoothness of the mask. The horizontal mask $\text{mask}_{(S_w,W,R)}$ can be defined in the same fashion to construct the 2D mask $\mathcal{W}$:

$$\mathcal{W}(S_h, S_w, H, W, R) = \text{mask}_{(S_h,H,R)} \otimes \text{mask}_{(S_w,W,R)} \tag{4}$$

We use $\mathcal{W}$ in two ways: (1) we apply it to the Fourier representation of the inputs via an element-wise product, which performs low-pass filtering (2) we crop the Fourier coefficients where the mask is zero (i.e. the output has dimensions $\lfloor \frac{H}{S_h} + R \rfloor \times \lfloor \frac{W}{S_w} + R \rfloor$).

The first step is differentiable with respect to strides S, however the cropping operation is not. Therefore, we apply a stop gradient operator (Bengio et al., 2013) to the mask before cropping. This way, gradients can flow to the strides through the **differentiable low-pass filtering** operation, but not through the **non-differentiable cropping**. Finally, the cropped tensor is transformed back into the spatial domain using an inverse DFT. All these steps are summarized by Algorithm 1 and illustrated on a single channel image in the Figure 1.

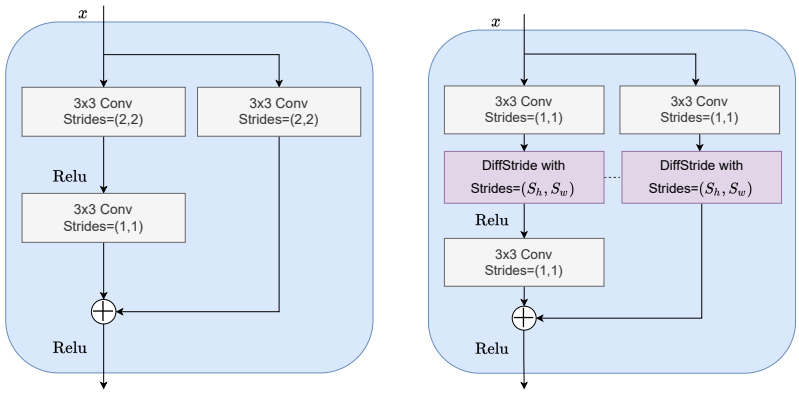
When x is a multi-channel input (i.e. $x \in \mathbb{R}^{H \times W \times C}$), we learn the same strides S for all channels, which is required in standard CNNs as otherwise each channel would have different spatial dimensions. In spatial and spectral pooling, strides are typically tied along the spatial axes (i.e. $S = (2, 2)$ or $(3, 3)$), which we can also do in DiffStride by sharing a single learnable stride on both dimensions. However, our experiments in Section 3 show that learning specific strides for the vertical and horizontal axis is beneficial, not only when processing time-frequency representations of

Algorithm 1: DiffStride layer

Inputs : The input tensor $x \in \mathbb{R}^{H \times W}$, the tensor shapes (H, W)
the strides $S = (S_h, S_w) \in [1, H) \times [1, W)$,
the smoothness factor to build the low-pass filter R.

Output: Tensor $\tilde{x} \in \mathbb{R}^{\lfloor \frac{H}{S_h} + R \rfloor \times \lfloor \frac{W}{S_w} + R \rfloor}$

1. $y \longleftarrow \mathcal{F}(x)$ ▷ Fourier representation of the input.
2. $\text{mask} \longleftarrow \mathcal{W}(S_h, S_w, H, W, R)$ ▷ Construction of the mask. See equations 3 and 4
3. $\tilde{y} \longleftarrow y \circ \text{mask}$ ▷ Application of the mask as a low-pass filter.
4. $\tilde{y} \longleftarrow sg(Crop(y, \text{mask}))$ ▷ Crop the tensor with the mask and stop gradients.
5. $\tilde{x} \longleftarrow \mathcal{F}^{-1}(\tilde{y})$ ▷ Return to the spatial domain for the output.



(a) Residual block with strided convolutions

(b) Residual block with a shared DiffStride

Figure 2: Comparison side by side of the shortcut blocks in classic ResNet architectures with strided convolutions, and with DiffStride that learns the strides of the block.

audio, but also — more counter-intuitively— when classifying natural images. While in this section we focus on the 2D case, using a single 1D mask allows deriving DiffStride in 1D CNNS, while performing the outer product between three 1D masks allows applying DiffStride to 3D inputs.

2.4.1 Residual Block with DiffStride

Unlike previous systems that only fed the outputs of the lth layer to the $(l+1)^{th}$ (Krizhevsky et al., 2012), ResNets (He et al., 2016a;b) introduced skip-connections that operate in parallel to the main branch. ResNets stack two types of blocks: (1) identity blocks that maintain the input channel dimension and spatial resolution and (2) shortcut blocks that increase the output channel dimension while applying reducing the spatial resolution with a strided convolution (see Figure 2a).

We integrate DiffStride into these shortcut blocks by replacing strided convolutions by convolutions without strides followed by DiffStride. The summation of the outputs of the main branch and the shortcut one require identical spatial dimensions. To ensure this, we share the learnables strides parameters between the two instances of DiffStride layers in the main and residual branches. Figure 2b) illustrates this new shortcut block.

2.4.2 Regularizing computation and memory with DiffStride

Designing architectures with limited time (number of operations) and space (memory) complexity is necessary for practical usage, in particular on limited hardware. Since measures of complexity such as the number of floating-point operations (FLOPs) are not differentiable with respect to the parameters of a model, searching for efficient architectures is typically done via high-level explo-

ration (e.g. introducing separable convolutions (Howard et al., 2017)), architecture search (Howard et al., 2019; Tan & Le, 2019) or using continuous relaxations of complexity (Paria et al., 2020).

A standard 2D convolution with a square kernel of size k^2 and C' output channels has a computational cost of $k^2 \times C \times C' \times H \times W$ when operating on $x \in \mathbb{R}^{H \times W \times C}$. Its memory usage— in terms of the number of activations to store— is $C' \times H \times W$. Considering a fixed number of channels and kernel size, both the computational complexity and memory usage of a convolution layer are thus linear functions of its input size $H \times W$. This illustrates our argument made in Section 1 that downsampling does not only improve performance by discarding irrelevant information, but also reduces the complexity of the upper layers. More importantly, in the context of DiffStride the input size $H^l \times W^l$ of layer l is determined as follows: $H^l \times W^l = \lfloor \frac{H^{l-1}}{S_h^{l-1}} + R \rfloor \times \lfloor \frac{W^{l-1}}{S_w^{l-1}} + R \rfloor$, which is differentiable with respect to the strides at the previous layer S^{l-1}. Furthermore, it also depends on spatial dimensions at the previous layer $H^{l-1} \times W^{l-1}$, which themselves are a function of S^{l-2}. By recurrence, the computational cost and memory usage are proportional to $\sum_{l=1}^{l=L} \prod_{i=1}^{l} \frac{1}{S_h^i \times S_w^i}$. Since in the context of DiffStride the kernel size and number of channels remain constant during training, we can thus directly regularize our model towards time and space efficiency by adding the following regularizer to our training loss:

$$\lambda J((S^l)_{l=1}^{l=L}) = \lambda \sum_{l=1}^{l=L} \prod_{i=1}^{l} \frac{1}{S_h^i \times S_w^i}, \tag{5}$$

where λ is the regularization weight. In Section 3.2, we show that training on ImageNet with different values for λ allows us to trade-off accuracy for efficiency in a smooth fashion.

3 Experiments

We evaluate DiffStride layer on four supervised learning problems spanning audio and vision domains. We benchmarked the DiffStride layer by directly replacing the pooling layers in the CNNs architectures used for each task. In all our experiments, we keep the same neural network architecture and replace the striding in strided convolutions by DiffStride.

We also compare our approach wen the pooling is also performed in the Fourier domain with fixed strides, such as in (Rippel et al., 2015). We refer to this approach as *Spectral Pooling*.

3.1 Audio classification

3.1.1 Experimental setup

We use the same common architecture for the encoding of audio signal for audio classification. First, the signal, sampled at $16\,\mathrm{kHz}$, is decomposed into log-compressed mel-filterbanks, with 64 channels, calculated with a window of $25\,\mathrm{ms}$ every $10\,\mathrm{ms}$. These time-frequency representation of the audio are then fed to the describe simple CNN

The CNN12 model alternates between blocks of convolutions and pooling along time and blocks of convolutions and pooling along frequency. Finally, the output of this encoder is globally pooled with a maximum and fed to a single *head* in the case of single-task audio classification, and multiple *heads* in the case of multi-task classification. As the sounds example are of variable length, the models are trained with a random $1\,\mathrm{s}$ window excerpts.

Each model is trained with the ADAM optimizer (Kingma & Ba, 2015) and a learning rate of 10^{-4} for $1\,\mathrm{M}$ batches, with batch size 256. We trained 6 independent supervised models on 6 different classification tasks: acoustic scene classification on the TUT dataset (Heittola et al., 2018), birdsong detection (Stowell et al., 2018), musical instrumental classification and pitch estimation on the NSynth dataset (Engel et al., 2017), keyword spotting on the Google Speech Commands v2 dataset (Warden, 2018). The characteristics of the datasets used in the experiments are summarized in Table 6.

Setting	Single-task			Multi-task		
Pooling type	Strided Conv.	Spectral	DiffStride	Strided Conv.	Spectral	DiffStride
Acoustic scenes	**99.1** $\pm$ 0.2	98.6 $\pm$ 0.1	98.6 $\pm$ 0.2	**97.7** $\pm$ 0.4	**97.7** $\pm$ 0.7	**97.7** $\pm$ 0.3
Birdsong detection	78.8 $\pm$ 0.3	79.7 $\pm$ 0.3	**81.3** $\pm$ 0.1	77.3 $\pm$ 0.2	77.8 $\pm$ 0.3	**78.6** $\pm$ 0.5
Music (instrument)	72.6 $\pm$ 0.3	72.9 $\pm$ 0.5	**75.4** $\pm$ 0.0	89.4 $\pm$ 0.3	87.6 $\pm$ 0.7	**89.9** $\pm$ 0.3
Music (pitch)	91.8 $\pm$ 0.1	90.1 $\pm$ 0.0	**92.2** $\pm$ 0.1	89.4 $\pm$ 0.3	87.6 $\pm$ 0.7	**89.9** $\pm$ 0.3
Speaker Id.	**100.0** $\pm$ 0.0	**100.0** $\pm$ 0.0	**100.0** $\pm$ 0.0	**100.0** $\pm$ 0.0	**100.0** $\pm$ 0.0	**100.0** $\pm$ 0.0
Speech commands	87.3 $\pm$ 0.1	88.5 $\pm$ 0.3	**90.5** $\pm$ 0.3	83.5 $\pm$ 0.6	83.9 $\pm$ 0.4	**86.2** $\pm$ 0.8
Mean Accuracy	87.6 $\pm$ 10.3	88.3 $\pm$ 9.9	**90.1** $\pm$ 9.1	86.3 $\pm$ 11.0	86.2 $\pm$ 10.7	**87.5** $\pm$ 9.9

Table 1: Test accuracy (%) for audio classification in the single (one model per task) and multi-task (one model for all tasks) settings.

	Strides		Equivalent cut-off frequencies	
	Time	Frequency	Time (Hz)	Frequency (Cyc/Mel)
Acoustic scenes	1.89 $\pm$ 0.05	1.99 $\pm$ 0.03	26.25 $\pm$ 0.63	0.2448 $\pm$ 0.009
Birdsong detection	1.91 $\pm$ 0.02	1.96 $\pm$ 0.01	25.83 $\pm$ 0.36	0.2500 $\pm$ 0.000
Music (Instrument)	1.29 $\pm$ 0.06	2.12 $\pm$ 0.01	38.33 $\pm$ 1.57	0.2292 $\pm$ 0.009
Music (Pitch)	1.32 $\pm$ 0.10	1.61 $\pm$ 0.07	37.50 $\pm$ 2.72	0.3021 $\pm$ 0.018
Speaker Id.	1.86 $\pm$ 0.06	1.97 $\pm$ 0.03	26.67 $\pm$ 0.95	0.2448 $\pm$ 0.009
Speech commands	1.97 $\pm$ 0.00	1.95 $\pm$ 0.01	25.00 $\pm$ 0.00	0.2500 $\pm$ 0.000
Multi-task model	1.46 $\pm$ 0.01	1.79 $\pm$ 0.03	34.17 $\pm$ 0.30	0.2708 $\pm$ 0.0074

Table 2: Learned strides of the first layer for the different single audio tasks and the multi-task model. The sampling rate of the input spectrogram is known (10 ms) and we can convert the strides in the corresponding cut-off frequencies. We report the mean and standard deviation of the learned strides for three runs.

3.1.2 RESULTS

We summarize the results for single-task and multi-task audio classification in the Table 1. All the experiments were performed three times and we report mean and standard deviations for the test accuracy. Overall, we observed that DiffStride improved over the classic strided convolutions in the spatial domain, and the Spectral Pooling from (Rippel et al., 2015). We observed these average improvement both for the single-task and multi-task settings. If we consider each individual task, the strided convolution only outperformed DiffStride for the Acoustic scenes classification in the single task setting. Our main hypothesis for this results concern the classes of sounds that are considered in this dataset. Indeed, the TUT acoustic scenes contain sounds that are produced by human made machine (underground metro, buses or cars) which potentially do not have the same characteristics as natural sounds. Indeed, these categories might display more inharmonious sounds with rough and strident impacts.

Table 2 shows the mean and standard deviation of the strides of the first layers that downsample the Log-compressed Mel Filterbanks in the frequency and time axis. First, the strides departed from their initialization at 2 and moved accordingly to the needs of the task and did not converge to integer values. The converted strides to cut-off frequencies showed that the strided fall in the range of required values found by behavioral studies for instance for speech intelligibility (24 Hz (Elliott & Theunissen, 2009) or by direct brain invasive neural recordings (Hullett et al., 2016; Flinker et al., 2019). Besides, the srides differed for the time and frequency axes. This confirmed the need to learn both directions, especially for domain where there is no rotation invariance of the inputs. Another notable phenomenon is the different convergence for the NSynth dataset as a function of task. For the pitch estimation, there is a greater need for faster spectral modulations. Finally, the multi-task model did not converge to the mean of strides, but actually tried to lower its values to not handicap any of the task.

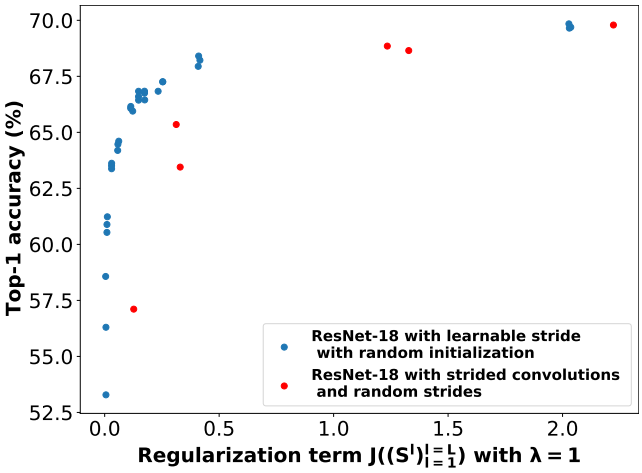


Figure 3: Top-1 accuracy (%) on the Imagenet validation set as a function of the regularization term $J((S^l)_{l=1}^{l=L})$ as defined in equation 5.

3.2 Image classification

3.2.1 Experimental setup

For image classification we use the ResNet-18 (He et al., 2016a) architecture, using either the original strided convolutions (see 2a) or DiffStride (see Figure 2b). We also compare to using spectral pooling in the same fashion as DiffStride, i.e. as an alternative to strided convolutions.

We first benchmark the three methods on the two CIFAR datasets (Krizhevsky, 2009). CIFAR10 consists of 32×32 images labeled in 10 classes with 6000 images per class. We use the official split, with 50000 images for training and 10000 images for testing. CIFAR100 uses the same set of images as CIFAR10, but with a more detailed labelling with 100 classes and 600 images per class. We also compare the ResNet-18 architectures on the ImageNet dataset (Russakovsky et al., 2015), which contains 1000 classes. The models are trained on the official training split of the Imagenet dataset (1.28 millions images for training) and we report our results on the validation set that contains 50k images. Here, we evaluate the performance of the different approaches with the top-1 and top-5 error rates. We train on all datasets with stochastic gradient descent (SGD) (Qian, 1999) with a learning rate of 0.1, a batch size of 256 and a momentum of 0.9. On CIFAR, we train models for 400 epochs dividing the learning rate 2 at half of the training and again by 2 at three-fourth of training, and using a weight decay of 5.10^{-3}. On ImageNet, we train with a weight decay of 1.10^{-3} for 90 epochs, dividing the learning rate by 10 at epochs 30, 60 and 80.

3.2.2 Results

We display the results for the experiments on CIFAR10, CIFAR100 and Imagenet in Table 3, in Table 4 and in Table 5 respectively. These results demonstrate the stability and effectiveness of DiffStride to tackle different vision classification tasks. The model with DiffStride outperforms for almost all configurations the spatial counterpart and the ResNet-18 with Spectral pooling. Especially, the ResNet-18 with DiffStride always converge to strong performance with low variance, while the other methods can be strongly affected by poor initialization.

4 Limitations

TODO comment on Densenet which does not rely on strides basically TODO add cifar100 densenet

The spectral pooling based-methods comes at a computational cost during training and during the hyper-parameters search. Indeed, for each reduction layer in a given neural network, the tensor needs to be go through the DFT and the inverse DFT. Even though, the Fast Fourier Transform allows to reduce the time complexity to $\mathcal{O}(M \log M)$ to obtain the DFT in one dimension, this still

Type of pooling	Strided Conv.	Spectral	DiffStride
((2, 2), (2, 2), (2, 2))	91.4 ± 0.2	92.4 ± 0.1	$\mathbf{92.5} \pm 0.1$
((2, 2), (2, 2), (3, 3))	90.5 ± 0.1	92.2 ± 0.2	$\mathbf{92.8} \pm 0.1$
((1, 1), (3, 3), (1, 1))	90.0 ± 0.4	91.1 ± 0.1	$\mathbf{92.4} \pm 0.1$
((3, 3), (1, 1), (3, 3))	85.7 ± 0.1	90.9 ± 0.2	$\mathbf{92.4} \pm 0.1$
((3, 3), (1, 1), (2, 2))	86.4 ± 0.1	90.9 ± 0.2	$\mathbf{92.3} \pm 0.1$
((3, 3), (2, 2), (3, 3))	82.0 ± 0.6	89.2 ± 0.2	$\mathbf{92.3} \pm 0.1$
Mean accuracy	87.7 ± 3.4	91.1 ± 1.1	$\mathbf{92.4} \pm 0.2$

Table 3: Test accuracy (%) for CIFAR10. The first column represents the differents initialization sets of strides that we considered. The first row represents the classic ResNet-18 architecture with the 'official' initialization.

Type of pooling	Strided Conv.	Spectral	DiffStride
((2, 2), (2, 2), (2, 2))	66.8 ± 0.2	$\mathbf{73.7} \pm 0.1$	73.4 ± 0.5
((2, 2), (2, 2), (3, 3))	63.4 ± 0.5	$\mathbf{73.7} \pm 0.2$	73.5 ± 0.0
((1, 1), (3, 3), (1, 1))	64.9 ± 0.5	70.3 ± 0.3	$\mathbf{73.4} \pm 0.2$
((3, 3), (1, 1), (3, 3))	55.3 ± 0.8	69.4 ± 0.4	$\mathbf{73.7} \pm 0.4$
((3, 3), (1, 1), (2, 2))	56.2 ± 0.3	69.9 ± 0.2	$\mathbf{73.4} \pm 0.3$
((3, 3), (2, 2), (3, 3))	48.2 ± 0.2	66.6 ± 0.5	$\mathbf{73.6} \pm 0.4$
Mean accuracy	59.1 ± 6.7	70.6 ± 2.6	$\mathbf{73.5} \pm 0.3$

Table 4: Test accuracy (%) for CIFAR100. The first column represents the differents initialization sets of strides that we considered. The first row represents the classic ResNet-18 architecture with the 'official' initialization.

represents a bottleneck in modern CNNs, where not all operations can be computed in the Fourier domain. We also found that the benefits of DiffStride can only be witnessed for architectures that are sensitive to strides values. Indeed, we found out that DenseNet architectures were not sensitive to the strides initializations. Therefore, the use of DiffStride was neither beneficial neither detrimental for the performances.

Another limitation concerns the hardward covering for DiffStride. Indeed, even though we showed that the learning of strides makes it an advantage to search for architecture parameters, the graph of computation keeps changing after each backpropagation pass. This prevents the use of XLA compilation of the graph and the use of Tensor Processor Units(TPUs) and only allows the use of Graphical Processing Units(GPUs). Finally,

5 Conclusions and Future Work

TODO

References

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pp. 173–182. PMLR, 2016.

Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=S1c2cvqee.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Type of pooling	Spatial Pooling		Spectral Pooling		DiffStride (Ours)	
	top-1	top-5	top-1	top-5	top-1	top-5
(1, 2, 2, 2)	68.65 ± 0.26	88.5 ± 0.15	69.01 ± 0.19	88.48 ± 0.02	$\mathbf{69.66} \pm 0.06$	$\mathbf{89.07} \pm 0.03$
(1, 1, 3, 1)	69.79 ± 0.15	$\mathbf{89.43} \pm 0.18$	$\mathbf{69.88} \pm 0.05$	89.15 ± 0.07	68.22 ± 0.07	88.1 ± 0.08
(1, 3, 1, 3)	68.86 ± 0.28	88.64 ± 0.15	68.63 ± 0.08	88.42 ± 0.01	$\mathbf{69.41} \pm 0.16$	$\mathbf{88.98} \pm 0.04$
(2, 2, 2, 3)	63.45 ± 0.09	85.09 ± 0.04	67.16 ± 0.17	87.25 ± 0.06	$\mathbf{69.53} \pm 0.08$	$\mathbf{89.05} \pm 0.05$
(2, 3, 1, 2)	65.35 ± 0.03	86.27 ± 0.05	66.35 ± 0.24	86.67 ± 0.15	$\mathbf{69.42} \pm 0.06$	$\mathbf{88.91} \pm 0.05$
(3, 3, 2, 3)	57.11 ± 0.11	80.42 ± 0.11	64.44 ± 0.01	85.22 ± 0.09	$\mathbf{69.43} \pm 0.11$	$\mathbf{89.03} \pm 0.02$
Mean accuracy	65.53 ± 4.49	86.39 ± 3.15	67.58 ± 1.88	87.53 ± 1.36	$\mathbf{69.28} \pm 0.50$	$\mathbf{88.85} \pm 0.35$

Table 5: Top-1 and Top-5 accuracies (%) for Imagenet. The first column represents the differents initialization sets of strides that we considered. The first row represents the classic ResNet-18 architecture with the 'official' initialization.

Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 111–118, 2010.

S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1:539–546 vol. 1, 2005.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 1107–1116, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-1104.

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*, pp. 248–255. Ieee, 2009.

Taffeta M Elliott and Frédéric E Theunissen. The modulation transfer function for speech intelligibility. *PLoS computational biology*, 5(3):e1000302, 2009.

Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. Technical report, apr 2017. URL http://arxiv.org/abs/1704.01279.

Adeen Flinker, Werner K Doyle, Ashesh D Mehta, Orrin Devinsky, and David Poeppel. Spectrotemporal modulation provides a unifying framework for auditory cortical asymmetries. *Nature human behaviour*, 3(4):393–405, 2019.

Eduardo Fonseca, Andrés Ferraro, and Xavier Serra. Improving sound event classification by increasing shift invariance in convolutional neural networks. *ArXiv*, abs/2107.00623, 2021.

K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pp. 1243–1252. PMLR, 2017.

Benjamin Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

Caglar Gulcehre, Kyunghyun Cho, Razvan Pascanu, and Yoshua Bengio. Learned-norm pooling for deep feedforward and recurrent neural networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 530–546. Springer, 2014.

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*, pp. 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

Toni Heittola, Annamaria Mesaros, and Tuomas Virtanen. TUT Urban Acoustic Scenes 2018, Development dataset, April 2018. URL https://doi.org/10.5281/zenodo.1228142.

Geoffrey Hinton. What's wrong with convolutional nets. MIT Brain and Cognitive Sciences-Fall Colloquium Series, Dec 2014a. URL http . . . , 2014.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *ArXiv*, abs/1704.04861, 2017.

Andrew G. Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1314–1324, 2019.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Patrick W Hullett, Liberty S Hamilton, Nima Mesgarani, Christoph E Schreiner, and Edward F Chang. Human superior temporal gyrus organization of spectrotemporal modulation tuning derived from speech stimuli. *Journal of Neuroscience*, 36(6):2014–2026, 2016.

Chen Jin, Ryutaro Tanno, Thomy Mertzanidou, Eleftheria Panagiotaki, and Daniel C. Alexander. Learning to downsample for segmentation of ultra-high resolution images, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D Plumbley. Panns: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2880–2894, 2020.

A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

Scinob Kuroki, Masataka Sawayama, and Shin'ya Nishida. Haptic texture perception on 3d-printed surfaces transcribed from visual natural textures. In *International Conference on Human Haptic Sensing and Touch Enabled Computer Applications*, pp. 102–112. Springer, 2018.

Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.

Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial intelligence and statistics*, pp. 464–472. PMLR, 2016.

Richard G. Lyons. *Understanding Digital Signal Processing (2nd Edition)*. Prentice Hall PTR, USA, 2004. ISBN 0131089897.

Hannah Muckenhirn, M. Magimai.-Doss, and S. Marcel. Towards directly modeling raw speech signal for speaker verification using cnns. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4884–4888, 2018.

Biswajit Paria, Chih-Kuan Yeh, N. Xu, B. Póczos, Pradeep Ravikumar, and I. E. Yen. Minimizing flops to learn efficient sparse representations. *ArXiv*, abs/2004.05665, 2020.

Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks : the official journal of the International Neural Network Society*, 12 1:145–151, 1999.

Oren Rippel, Jasper Snoek, and Ryan P Adams. Spectral representations for convolutional neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pp. 2449–2457, 2015.

Daniel L Ruderman. The statistics of natural images. *Network: computation in neural systems*, 5 (4):517, 1994.

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.

Faraz Saeedan, Nicolas Weber, Michael Goesele, and Stefan Roth. Detail-preserving pooling in deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9108–9116, 2018.

Tom Sercu, Christian Puhrsch, Brian Kingsbury, and Yann LeCun. Very deep multilingual convolutional neural networks for lvcsr. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4955–4959. IEEE, 2016.

Nandini C Singh and Frédéric E Theunissen. Modulation spectra of natural sounds and ethological theories of auditory processing. *The Journal of the Acoustical Society of America*, 114(6):3394–3411, 2003.

Dan Stowell, — Mike Wood, — Hanna Pamuła, Yannis Stylianou, and Hervé Glotin. Automatic acoustic detection of birds through deep learning: the first Bird Audio Detection challenge. Technical report, 2018. URL https://arxiv.org/pdf/1807.05812.pdf.

Sainbayar Sukhbaatar, Édouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive attention span in transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 331–335, 2019.

Hossein Talebi and Peyman Milanfar. Learning to resize images for computer vision tasks. *ICCV*, 2021.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2820–2828, 2019.

Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, K. Kavukcuoglu, Oriol Vinyals, and A. Graves. Conditional image generation with pixelcnn decoders. In *NIPS*, 2016.

Pete Warden. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. Technical report, 2018. URL https://arxiv.org/pdf/1804.03209.pdf.

P Yin, J Lyu, S Zhang, S Osher, YY Qi, and J Xin. Understanding straight-through estimator in training activation quantized neural nets. In *International Conference on Learning Representations*, 2019.

Neil Zeghidour, Qiantong Xu, Vitaliy Liptchinsky, Nicolas Usunier, Gabriel Synnaeve, and Ronan Collobert. Fully convolutional speech recognition. *arXiv preprint arXiv:1812.06864*, 2018.

Richard Zhang. Making convolutional networks shift-invariant again. In *ICML*, 2019.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. 2017. URL https://arxiv.org/abs/1611.01578.

A APPENDIX

Table 6: Datasets used in the experiments. Default train/test splits are always adopted.

Task	Name	Classes	Train examples	Test examples
Acoustic scenes	TUT Urban 2018	10	7,829	810
Birdsong detection	DCASE2018	2	32,129	3,561
Music (instrument)	Nsynth	11	289,205	12,678
Music (pitch)	Nsynth	128	289,205	12,678
Speaker Id.	Librispeech	251	25,740	2,799
Speech commands	Speech commands	35	84,771	10,700