Downsampling 등의 크기를 결정하는 것은 큰 hyperparam이다.

하지만 이는 NAS로도 찾기엔 시간이 오래 걸리고, 훈련이 불가능하다.

따라서 stride를 훈련 가능한, diffstride 제시.

* Introduction.

Pooling 이나 stride conv는 관련 정보의 집중하게 하고, shift-invariant와 high receptive field의 연관이 있다.

이 pooling layer는 local 계산(1) 후 subsampling(2)으로 진행된다.

보통 (1)을 개선하기 위해 aliasing을 없애거나 하지 않는 방법을 제시한다.

Stride 2가 output을 1/4로 너무 급격히 줄이기 때문에 보통 stride나 maxpooling도 제시된다.

Fractional stride는 layer의 flexible을 주지만, search space를 증가시킨다.

적절한 stride를 찾기 힘들 않다.

따라서 본문에서는 stride를 학습한다.

Diffstride는 spatial domain의 downsampling을 frequency domain의 crop처럼 cast한다.

↳ backpropagation을 통해 split. → 2D attention window

* Methods.

2D CNN을 basic으로 하지만, 다른 곳에도 적용 가능.

- Notations.

	Periodic	Aperiodic
Continuous	Discrete aperiodic *Fourier series*	Continuous aperiodic *Fourier integral*
Discrete	Discrete periodic *Discrete FT*	Continuous periodic *Discrete-time FT*

↳ time domain에서 continuous를 처리하기 위해 sampling을 하여 discrete하게 만들고,

DTFT로 frequency domain으로 변환한다.

하지만 DTFT의 결과가 continuous 하기 때문에 컴퓨터에서 사용할 수 없다.

따라서 DTFT의 frequency sampling 형식인 DFT를 사용

↳ time domain에서도 periodic

$x \in R^{H \times W}$일 때, DFT는 $y = \mathcal{F}(x) \in C^{H \times W}$를 basis filter로 분해할 수 있다

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

↳ linear이고, inverse는 conjugate. $\mathcal{F}(\cdot)^{-1} = \mathcal{F}(\cdot)^*$

↳ FT에서 real-value signal x는 conjugate symmetric이고,

width의 절반의 positive half frequency로 x를 recon하고,

negative frequency를 omit한다. $y_{mn} = y^*_{(H-m) \bmod H, (W-n) \bmod W}$

↳ DFT, DFT^{-1}는 input에 대해 differentiable하며, DFT'는 conjugate linear operator

↳ loss $\mathcal{L}: C^{H \times W} \to R$ 에 대해, DFT결과 y가 input인 DFT^{-1}로 gradient 계산 가능

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

이후로 L은 CNN에서 층 conv layer 수, $\circ$은 element-wise곱, $\otimes$는 outer product, $\lfloor \cdot \rfloor$은 floor operator

- Downsampling in CNN

CNN에서 downsampling을 위한 기본 mechanism은

1. input과 finite impulse filter의 stride conv
   ↳ 통신 이론 개념을 끌어 다루며 conv의 kernel을 이렇게 표현 한다는 것도

2. Non-strided conv 후 pooling

위의 두 downsampling은 모두 global structure를 보지 못함.

그리고 $S=2$는 resolution을 너무 급격히 줄임

또한 조합 경우의 수가 너무 많아서 NAS가 힘듦.

- Spectral pooling

보통 signal 들은 고르게 분포된 것이 아니고 low에 많이 몰려 있다.

↳ 이를 이용해 spectral pooling은 spatial 정보 손실을 줄임

↳ aliasing 없이 low-frequency 보존

DFT가 $y = \mathcal{F}(x)$고, $\xrightarrow{\text{simplicity}}$ y가 frequency 이기 때문에 center는 DC로 가정.

그리고 high-frequency 말고, center 주변을 $\lfloor \frac{H}{s_h} \rfloor \times \lfloor \frac{W}{s_w} \rfloor$ 만큼 crop $= \tilde{y}$

$\tilde{y}$를 inverse DFT로 $\tilde{x}$ recon

↳ $\tilde{x} = \mathcal{F}^{-1}(\tilde{y})$

↳ real-value 일 때는, positive half로만 계산 가능 → memory, computation saving

Spectral pooling은 int stride가 필요한 일반 downsampling과 달리,

output만 int면 되기 때문에 fine-grained downsizing이 가능하다.

그리고 low-frequency를 남기기 때문에 큰 aliasing 방지.

↳ 하지만 역시 S는 미분이 안됨.

- DiffStride

Spectrum pooling과 비슷하게, fourier domain에서 crop

Crop할 window W의 크기를 학습.

W는 input shape, smoothness factor R, stride에 의해 parameterized.

└ 1D masking의 outer product로 설계

└ adaptive self-attention.

└ conjugate symmetric으로 horizontal axis의 positive frequency만 고려, 0에 대해 vertical mirroring

$$\text{mask}^{h}_{(S_h,H,R)}(m) = \min\left[\max\left[\frac{1}{R}(R+\frac{H}{2S_h}-|\frac{H}{2}-m|),0\right],1\right], m\in[0,H] \quad (3)$$

$$\text{mask}^{w}_{(S_w,W,R)}(n) = \min\left[\max\left[\frac{1}{R}(R+\frac{W}{2S_w}+1-n),0\right],1\right], n\in[0,\frac{W}{2}+1] \quad (4)$$

$$\mathcal{W}(S_h,S_w,H,W,R) = \text{mask}^{h}_{(S_h,H,R)} \otimes \text{mask}^{w}_{(S_w,W,R)}$$

W는 fourier representation의 crop window로 쓰인다. $\lfloor\frac{H}{S_h}+2\times R\rfloor \times \lfloor\frac{W}{S_w}+2\times R\rfloor$

└ like low pass filter

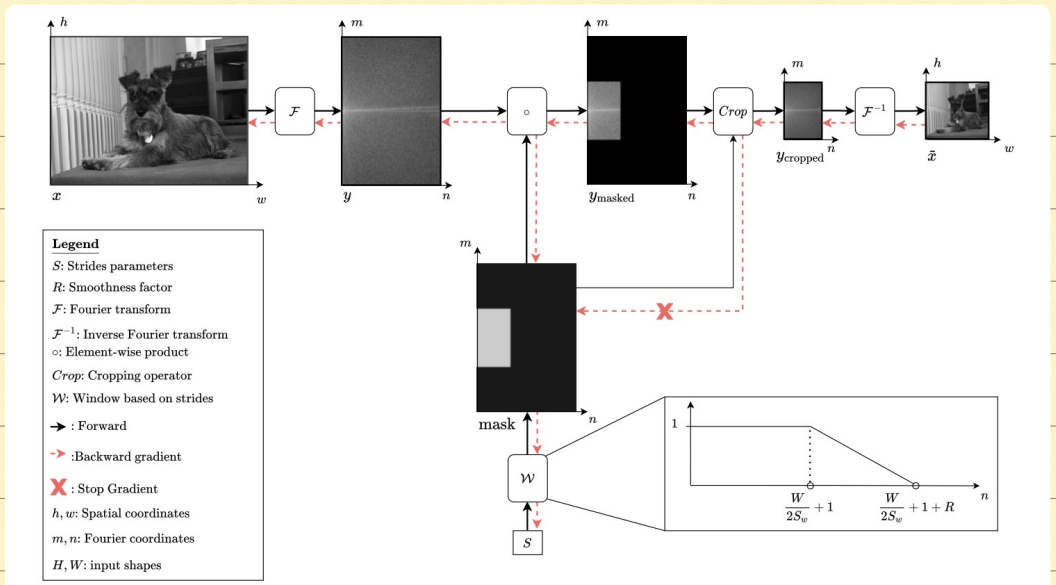


Figure 1: DiffStride forward and backward pass, using a single-channel image. We only compute the positive half of DFT coefficients along the horizontal axis due to conjugate symmetry. The zoomed frame shows the horizontal mask $\text{mask}^{w}_{(S_w,W,R)}(n)$. Here $S=(S_h,S_w)=(2.6,3.1)$.

Algorithm 1: DiffStride layer

Inputs : Input $x \in \mathbb{R}^{H \times W}$, strides $S = (S_h, S_w) \in [1, H) \times [1, W)$, smoothness factor R.

Output: Downsampled output $\tilde{x} \in \mathbb{R}^{\lfloor \frac{H}{S_h} + 2 \times R \rfloor \times \lfloor \frac{W}{S_w} + 2 \times R \rfloor}$

$y \longleftarrow \mathcal{F}(x)$ ▷ Project input to the Fourier domain.

$\text{mask} \longleftarrow \mathcal{W}(S_h, S_w, H, W, R)$ ▷ Construct the mask. See Equation 5.

$y_{\text{masked}} \longleftarrow y \circ \text{mask}$ ▷ Apply the mask as a low-pass filter.

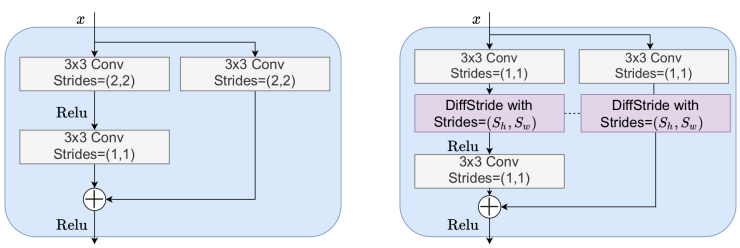
$y_{\text{cropped}} \longleftarrow Crop(y_{\text{masked}}, sg(\text{mask}))$ ▷ Crop the tensor with the mask after stopping gradients.

$\tilde{x} \longleftarrow \mathcal{F}^{-1}(y_{\text{cropped}})$ ▷ Return to the spatial domain.

3D에 대해서도 1D를 3개써서 동일하게 학습가능

- Residual Block with DiffStride

Residual의 평균 stride를 공유해서 학습



(a) Residual block with a strided convolution.

(b) Residual block with a shared DiffStride layer.

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.

- Regularizing Computation and Memory Cost with DiffStride.

memory 소모도 linear이기 때문에 효율적

kernel의 크기나 channel 수는 일정하게 유지되기 때문에 reg 가능.

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

* Experiments.

- Audio classification.

Setting	Single-task			Multi-task		
Task	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	69.8 $\pm$ 0.4	70.4 $\pm$ 0.4	**73.0** $\pm$ 0.8
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
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	85.0 $\pm$ 9.3	86.0 $\pm$ 9.2	**88.3** $\pm$ 8.7	83.5 $\pm$ 10.0	83.5 $\pm$ 9.6	**85.0** $\pm$ 8.9

Table 1: Test accuracy (% $\pm$ sd over 3 runs) for audio classification in the single (one model per task) and multi-task (one model for all tasks) settings.

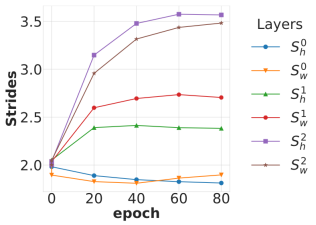
	Learned Strides		Equivalent cut-off frequencies	
	Time	Frequency	Time (Hz)	Frequency (Cyc/Mel)
Acoustic scenes	1.89 $\pm$ 0.05	1.99 $\pm$ 0.03	26.25 $\pm$ 0.63	0.2448 $\pm$ 0.009
Birdsong detection	1.91 $\pm$ 0.02	1.96 $\pm$ 0.01	25.83 $\pm$ 0.36	0.2500 $\pm$ 0.000
Music (Instrument)	1.29 $\pm$ 0.06	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
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 (% $\pm$ sd over 3 runs) of the first layer for the single and multi-task models. The sampling rate of the input spectrogram being known (10 ms), we can convert the strides to upper cut-off frequencies (i.e. the maximum frequency kept by the lowpass-filter).

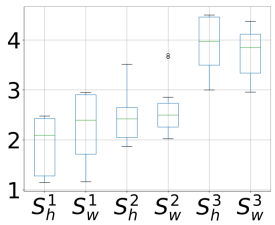
- Image classification.

	CIFAR10			CIFAR100		
Init. Strides	Strided Conv.	Spectral	DiffStride	Strided Conv.	Spectral	DiffStride
(2, 2, 2)	91.4 $\pm$ 0.2	92.4 $\pm$ 0.1	**92.5** $\pm$ 0.1	66.8 $\pm$ 0.2	**73.7** $\pm$ 0.1	73.4 $\pm$ 0.5
(2, 2, 3)	90.5 $\pm$ 0.1	92.2 $\pm$ 0.2	**92.8** $\pm$ 0.1	63.4 $\pm$ 0.5	**73.7** $\pm$ 0.2	73.5 $\pm$ 0.0
(1, 3, 1)	90.0 $\pm$ 0.4	91.1 $\pm$ 0.1	**92.4** $\pm$ 0.1	64.9 $\pm$ 0.5	70.3 $\pm$ 0.3	**73.4** $\pm$ 0.2
(3, 1, 3)	85.7 $\pm$ 0.1	90.9 $\pm$ 0.2	**92.4** $\pm$ 0.1	55.3 $\pm$ 0.8	69.4 $\pm$ 0.4	**73.7** $\pm$ 0.4
(3, 1, 2)	86.4 $\pm$ 0.1	90.9 $\pm$ 0.2	**92.3** $\pm$ 0.1	56.2 $\pm$ 0.3	69.9 $\pm$ 0.2	**73.4** $\pm$ 0.3
(3, 2, 3)	82.0 $\pm$ 0.6	89.2 $\pm$ 0.2	**92.3** $\pm$ 0.1	48.2 $\pm$ 0.2	66.6 $\pm$ 0.5	**73.6** $\pm$ 0.4
Mean accuracy	87.7 $\pm$ 3.4	91.1 $\pm$ 1.1	**92.4** $\pm$ 0.2	59.1 $\pm$ 6.7	70.6 $\pm$ 2.6	**73.5** $\pm$ 0.3

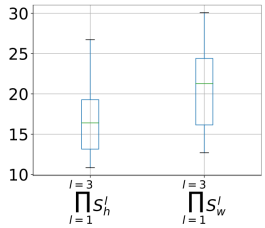
Table 3: Accuracies (% $\pm$ sd over 3 runs) on CIFAR10 and CIFAR100. First column represents strides at each shortcut block, $(2, 2, 2)$ being the configuration of (He et al., 2016a). For reference, the state-of-the-art on CIFAR10 (CIFAR100) is (Dosovitskiy et al., 2020) ((Foret et al., 2020)) with an accuracy of 99.5% (96.1%).



(a) Trajectories of the different strides $(S^l)_{l\in(1,2,3)}$ for a single run.



(b) Distribution of learned strides in ResNet-18 for the different initializations.



(c) Distribution of the final global striding factors for the different initializations.

Figure 3: Learning dynamics of DiffStride on the CIFAR10 dataset.

↳ low layer 일수록 빨리 수렴

- Limitation.

계산량이 증가함. densenet에서는 이점 x