Appendix for
"EfficientTrain: Exploring Generalized Curriculum Learning for Training Visual Backbones"

A. Implementation Details

A.1. Training Models on ImageNet-1K

Dataset. We use the data provided by ILSVRC2012[1] [17]. The dataset includes 1.2 million images for training and 50,000 images for validation, both of which are categorized in 1,000 classes.

Training. Our approach is developed on top of a state-of-the-art training pipeline of deep networks, which incorporates a holistic combination of various model regularization & data augmentation techniques, and is widely applied to train recently proposed models [19, 20, 4, 21, 18]. Our training settings generally follow from [18], while we modify the configurations of weight decay, stochastic depth and exponential moving average (EMA) according to the recommendation in the original papers of different models (*i.e.*, ConvNeXt [18], DeiT [19], PVT [20], Swin Transformer [4] and CSWin Transformer [21])[2]. The detailed hyper-parameters are summarized in Table 14.

The baselines presented in Table 6 directly use the training configurations in Table 14. Based on Table 14, our proposed EfficientTrain curriculum performs low-frequency cropping and modifies the value of m in RandAug during training, as introduced in Table 5. The results in Tables 8 and 9 adopt a varying number of training epochs on top of Table 6.

In addition, the low-frequency cropping operation in EfficientTrain leads to a varying input size during training. Notably, visual backbones can naturally process different sizes of inputs with no or minimal modifications. Specifically, once the input size varies, ResNets and ConvNeXts do not need any change, while vision Transformers (*i.e.*, DeiT, PVT, Swin and CSWin) only need to resize their position bias correspondingly, as suggested in their papers. Our method starts the training with small-size inputs and the reduced computational cost. The input size is switched midway in the training process, where we resize the position bias for ViTs (do nothing for ConvNets). Finally, the learning ends up with full-size inputs, as used at test time. As a consequence, the overall computational/time cost to obtain the final trained models is effectively saved.

Inference. Following [19, 20, 4, 21, 18], we use a crop ratio of 0.875 and 1.0 for the inference input size of 224^2 and 384^2, respectively.

Training Config	Values / Setups
Input size	224^2
Weight init.	Truncated normal (0.2)
Optimizer	AdamW
Optimizer hyper-parameters	β_1, β_2=0.9, 0.999
Initial learning rate	4e-3
Learning rate schedule	Cosine annealing
Weight decay	0.05
Batch size	4,096
Training epochs	300
Warmup epochs	20
Warmup schedule	linear
RandAug [86]	(9, 0.5)
Mixup [98]	0.8
Cutmix [99]	1.0
Random erasing [100]	0.25
Label smoothing [101]	0.1
Stochastic depth [102]	Following the values in original papers [18, 19, 20, 4, 21].
Layer scale [103]	1e-6 (ConvNeXt [18]) / None (others)
Gradient clip	5.0 (DeiT [19], PVT [20] and Swin [4]) / None (others)
Exp. mov. avg. (EMA) [104]	0.9999 (ConvNeXt [18] and CSWin [21]) / None (others)
Auto. mix. prec. (AMP) [105]	Inactivated (ConvNeXt [18]) / Activated (others)

Table 14: **Basic training hyper-parameters for the models in Table 6.**

[1] https://image-net.org/index.php

[2] The training of ResNet [1] follows the recipe provided in [18].

A.2. ImageNet-22K Pre-training

Dataset and pre-processing. In our experiments, the officially released processed version of ImageNet-22K[3] [17, 106] is used. The original ImageNet-22K dataset is pre-processed by resizing the images (to reduce the dataset's memory footprint from 1.3TB to $\sim$250GB) and removing a small number of samples. The processed dataset consists of $\sim$13M images in $\sim$19K classes. Note that this pre-processing procedure is officially recommended and accomplished by the official website.

Pre-training. We pre-train CSWin-Base/Large and ConvNeXt-Base/Large on ImageNet-22K. The pre-training process basically follows the training configurations of ImageNet-1K (*i.e.*, Table 14), except for the differences presented in the following. The number of training epochs is set to 120 with a 5-epoch linear warm-up. For all the four models, the maximum value of the increasing stochastic depth regularization [102] is set to 0.1 [18, 21]. Following [21], the initial learning rate for CSWin-Base/Large is set to 2e-3, while the weight-decay coefficient for CSWin-Base/Large is set to 0.05/0.1. Following [18], we do not leverage the exponential moving average (EMA) mechanism. To ensure a fair comparison, we report the results of our implementation for both baselines and EfficientTrain, where they adopt exactly the same training settings (apart from the configurations modified by EfficientTrain itself).

Fine-tuning. We evaluate the ImageNet-22K pre-trained models by fine-tuning them and reporting the corresponding accuracy on ImageNet-1K. The fine-tuning process of ConvNeXt-Base/Large follows their original paper [18]. The fine-tuning of CSWin-Base/Large adopts the same setups as ConvNeXt-Base/Large. We empirically observe that this setting achieves a better performance than the original fine-tuning pipeline of CSWin-Base/Large in [21].

A.3. Object Detection and Segmentation on COCO

Our implementation of RetinaNet [96] follows from [107]. Our implementation of Cascade Mask-RCNN [97] is the same as [18].

A.4. Experiments in Section 4

In particular, the experimental results provided in Section 4 are based on the training settings listed in Table 14 as well, expect for the specified modifications (*e.g.*, with the low-passed filtered inputs). The computing of CKA feature similarity follows [108].

[3]https://image-net.org/data/imagenet21k_resized.tar.gz

B. Additional Results

B.1. Wall-time Training Cost

The detailed wall-time training cost for the models presented in Table 6 of the paper is reported in Table 15. The numbers of GPU-hours are benchmarked on NVIDIA 3090 GPUs. The batch size for each GPU and the total number of GPUs are configured conditioned on different models, under the principle of saturating all the computational cores of GPUs.

	Model	Input Size (inference)	#Param.	#FLOPs	Top-1 Accuracy (300 epochs)		Wall-time Training Cost (in GPU-hours)		Speedup
					Baseline	**EfficientTrain**	Baseline	**EfficientTrain**	
ConvNets	ResNet-50 [1]	224^2	26M	4.1G	78.8%	**79.4%**	205.2	**142.2**	**1.44×**
	ConvNeXt-Tiny [18]	224^2	29M	4.5G	82.1%	**82.2%**	379.5	**254.1**	**1.49×**
	ConvNeXt-Small [18]	224^2	50M	8.7G	83.1%	**83.2%**	673.5	**449.9**	**1.50×**
	ConvNeXt-Base [18]	224^2	89M	15.4G	**83.8%**	83.7%	997.0	**671.6**	**1.48×**
Isotropic ViTs	DeiT-Tiny [19]	224^2	5M	1.3G	72.5%	**73.3%**	60.5	**38.9**	**1.55×**
	DeiT-Small [19]	224^2	22M	4.6G	80.3%	**80.4%**	137.7	**90.9**	**1.51×**
Multi-stage ViTs	PVT-Tiny [20]	224^2	13M	1.9G	75.5%	**75.5%**	99.0	**66.8**	**1.48×**
	PVT-Small [20]	224^2	25M	3.8G	79.9%	**80.4%**	201.1	**129.2**	**1.56×**
	PVT-Medium [20]	224^2	44M	6.7G	81.8%	**81.8%**	310.9	**208.3**	**1.49×**
	PVT-Large [20]	224^2	61M	9.8G	82.3%	**82.3%**	515.9	**337.3**	**1.53×**
	Swin-Tiny [4]	224^2	28M	4.5G	81.3%	**81.4%**	232.3	**155.5**	**1.49×**
	Swin-Small [4]	224^2	50M	8.7G	83.1%	**83.2%**	360.3	**239.4**	**1.50×**
	Swin-Base [4]	224^2	88M	15.4G	83.4%	**83.6%**	494.5	**329.9**	**1.50×**
	CSWin-Tiny [21]	224^2	23M	4.3G	82.7%	**82.8%**	290.1	**187.5**	**1.55×**
	CSWin-Small [21]	224^2	35M	6.9G	83.4%	**83.6%**	438.5	**291.0**	**1.51×**
	CSWin-Base [21]	224^2	78M	15.0G	84.3%	**84.3%**	823.7	**528.2**	**1.56×**

Table 15: **Accuracy v.s. wall-time training cost for the deep networks trained on ImageNet-1K** (*i.e.*, corresponding to the models presented in Table 6 of the paper).

B.2. On the Continuous Selection of B

Notably, the basic formulation behind EfficientTrain considers a continuous function of B, *i.e.*, f : epoch$\rightarrow B$. Theoretically, we can obtain an optimal curriculum if we directly solve for a strictly continuous f. However, directly solving for a continuous function is computationally intractable. To achieve a reasonable trade-off between the computational cost and the effectiveness of the solution, we adopt an approximation approach, *i.e.*, approximating the continuous function of B with a staircase function. Specifically, we divide the training process into N stages and solve for a value of B for each stage, where we set $N = 5$ and obtained the EfficientTrain curriculum.

Importantly, such approximation works reasonably well. As shown in our paper, its solution (EfficientTrain) considerably improves the training efficiency of deep networks, and exhibits superior generalizability across different backbone architectures and various training settings. Besides, as shown in Table 16, further approaching solving for a continuous function of B (*e.g.*, $N = 10$) only yields limited gains.

	Baseline	$N=3$	$N=4$	$N=5$ (**EfficientTrain**)	$N=10$
Top-1 Accuracy	81.3%	81.3%	**81.4%**	**81.4%**	81.3%
Training Speedup	1.00×	1.34×	1.46×	1.55×	**1.63×**

Table 16: **Effects when we approach solving for a continuous function:** f : epoch$\rightarrow B$ **(by increasing N in Alg. 1).** Swin-T on ImageNet-1K.

C. Proof of Proposition 1

In this section, we theoretically demonstrate the difference between two transformations, namely low-frequency cropping and image down-sampling. In specific, we will show that from the perspective of signal processing, the former perfectly preserves the lower-frequency signals within a square region in the frequency domain and discards the rest, while the image obtained from pixel-space down-sampling contains the signals mixed from both lower- and higher- frequencies.

C.1. Preliminaries

An image can be seen as a high-dimensional data point $\boldsymbol{X} \in \mathbb{R}^{C_0 \times H_0 \times W_0}$, where C_0, H_0, W_0 represent the number of channels, height and width. Since each channel's signals are regarded as independent, for the sake of simplicity, we can focus on a single-channel image with even edge length $\boldsymbol{X} \in \mathbb{R}^{2H \times 2W}$. Denote the 2D discrete Fourier transform as $\mathcal{F}(\cdot)$. Without loss of generality, we assume that the coordinate ranges are $\{-H, -H+1, \ldots, H-1\}$ and $\{-W, -W+1, \ldots, W-1\}$. The value of the pixel at the position $[u, v]$ in the frequency map $\boldsymbol{F} = \mathcal{F}(\boldsymbol{X})$ is computed by

$$\boldsymbol{F}[u,v] = \sum_{x=-H}^{H-1} \sum_{y=-W}^{W-1} \boldsymbol{X}[x,y] \cdot \exp\left(-j2\pi\left(\frac{ux}{2H} + \frac{vy}{2W}\right)\right).$$

Similarly, the inverse 2D discrete Fourier transform $\boldsymbol{X} = \mathcal{F}^{-1}(\boldsymbol{F})$ is defined by

$$\boldsymbol{X}[x,y] = \frac{1}{4HW} \sum_{u=-H}^{H-1} \sum_{v=-W}^{W-1} \boldsymbol{F}[u,v] \cdot \exp\left(j2\pi\left(\frac{ux}{2H} + \frac{vy}{2W}\right)\right).$$

Denote the low-frequency cropping operation parametrized by the output size $(2H', 2W')$ as $\mathcal{C}_{H',W'}(\cdot)$, which gives outputs by simple cropping:

$$\mathcal{C}_{H',W'}(\boldsymbol{F})[u,v] = \frac{H'W'}{HW} \cdot \boldsymbol{F}[u,v].$$

Note that here $u \in \{-H, -H+1, \ldots, H-1\}, v \in \{-W, -W+1, \ldots, W-1\}$, and this operation simply copies the central area of $\boldsymbol{F}$ with a scaling ratio. The scaling ratio $\frac{H'W'}{HW}$ is a natural term from the change of total energy in the pixels, since the number of pixels shrinks by the ratio of $\frac{H'W'}{HW}$.

Also, denote the down-sampling operation parametrized by the ratio $r \in (0,1]$ as $\mathcal{D}_r(\cdot)$. For simplicity, we first consider the case where $r = \frac{1}{k}$ for an integer $k \in \mathbb{N}^+$, and then extend our conclusions to the general cases where $r \in (0,1]$. In real applications, there are many different down-sampling strategies using different interpolation methods, *i.e.*, nearest, bilinear, bicubic, etc. When k is an integer, these operations can be modeled as *using a constant convolution kernel to aggregate the neighborhood pixels*. Denote this kernel's parameter as $\boldsymbol{w}_{s,t}$ where $s, t \in \{0, 1, \ldots, k-1\}$ and $\sum_{s=0}^{k-1} \sum_{t=0}^{k-1} \boldsymbol{w}_{s,t} = \frac{1}{k^2}$. Then the down-sampling operation can be represented as

$$\mathcal{D}_{1/k}(\boldsymbol{X})[x', y'] = \sum_{s=0}^{k-1} \sum_{t=0}^{k-1} \boldsymbol{w}_{s,t} \cdot \boldsymbol{X}[kx' + s, ky' + t].$$

C.2. Propositions

Now we are ready to demonstrate the difference between the two operations and prove our claims. We start by considering shrinking the image size by k and k is an integer. Here the low-frequency region of an image $\boldsymbol{X} \in \mathbb{R}^{H \times W}$ refers to the signals within range $[-H/k, H/k - 1] \times [-W/k, W/k - 1]$ in $\mathcal{F}(\boldsymbol{X})$, while the rest is named as the high-frequency region.

Proposition 1.1. *Suppose that the original image is $\boldsymbol{X}$, and that the image generated from the low-frequency cropping operation is $\boldsymbol{X}_c = \mathcal{F}^{-1} \circ \mathcal{C}_{H/k,W/k} \circ \mathcal{F}(\boldsymbol{X}), k \in \mathbb{N}^+$. We have that all the signals in the spectral map of $\boldsymbol{X}_c$ is only from the low frequency region of $\boldsymbol{X}$, while we can always recover $\mathcal{C}_{H/k,W/k} \circ \mathcal{F}(\boldsymbol{X})$ from $\boldsymbol{X}_c$.*

Proof. The proof of this proposition is simple and straightforward. Take Fourier transform on both sides of the above transformation equation, we get

$$\mathcal{F}(\boldsymbol{X}_c) = \mathcal{C}_{H/k,W/k} \circ \mathcal{F}(\boldsymbol{X}).$$

Denote the spectral of $\boldsymbol{X}_c$ as $\boldsymbol{F}_c = \mathcal{F}(\boldsymbol{X}_c)$ and similarly $\boldsymbol{F} = \mathcal{F}(\boldsymbol{X})$. According to our definition of the cropping operation, we know that

$$\boldsymbol{F}_c[u,v] = \frac{H/k \cdot W/k}{HW} \cdot \boldsymbol{F}[u,v] = \frac{1}{k^2} \cdot \boldsymbol{F}[u,v].$$

Hence, the spectral information of $\boldsymbol{X}_c$ simply copies $\boldsymbol{X}$'s low frequency parts and conducts a uniform scaling by dividing k^2.

Proposition 1.2. *Suppose that the original image is $\boldsymbol{X}$, and that the image generated from the down-sampling operation is $\boldsymbol{X}_\mathrm{d} = \mathcal{D}_{1/k}(\boldsymbol{X}), k \in \mathbb{N}^+$. We have that the signals in the spectral map of $\boldsymbol{X}_\mathrm{d}$ have a non-zero dependency on the high frequency region of $\boldsymbol{X}$.*

Proof. Taking Fourier transform on both sides, we have

$$\mathcal{F}(\boldsymbol{X}_\mathrm{d}) = \mathcal{F}(\mathcal{D}_{1/k}(\boldsymbol{X})).$$

For any $u \in [-H/k, H/k-1], v \in [-W/k, W/k-1]$, according to the definition we have

$$\begin{aligned}\mathcal{F}(\boldsymbol{X}_\mathrm{d})[u,v] &= \sum_{x=-H/k}^{H/k-1} \sum_{y=-W/k}^{W/k-1} \mathcal{D}_{1/k}(\boldsymbol{X})[x,y] \cdot \exp\left(-j2\pi\left(\frac{kux}{2H} + \frac{kvy}{2W}\right)\right) \\ &= \sum_{x=-H/k}^{H/k-1} \sum_{y=-W/k}^{W/k-1} \sum_{s=0}^{k-1}\sum_{t=0}^{k-1} \boldsymbol{w}_{s,t} \cdot \boldsymbol{X}[kx+s, ky+t] \cdot \exp\left(-j2\pi\left(\frac{kux}{2H} + \frac{kvy}{2W}\right)\right), \end{aligned} \tag{*1}$$

while at the same time we have the inverse DFT for $\boldsymbol{X}$:

$$\boldsymbol{X}[x,y] = \frac{1}{2H \cdot 2W} \sum_{u'=-H}^{H-1} \sum_{v'=-W}^{W-1} \boldsymbol{F}[u',v'] \cdot \exp\left(j2\pi\left(\frac{u'x}{2H} + \frac{v'y}{2W}\right)\right). \tag{*2}$$

Plugging (*2) into (*1), it is easy to see that essentially each $\boldsymbol{F}_\mathrm{d}(u,v) = \mathcal{F}(\boldsymbol{X}_\mathrm{d})[u,v]$ is a linear combination of the original signals $\boldsymbol{F}[u',v']$. Namely, it can be represented as

$$\boldsymbol{F}_\mathrm{d}(u,v) = \sum_{u'=-H}^{H-1} \sum_{v'=-W}^{W-1} \alpha(u,v,u',v') \cdot \boldsymbol{F}(u',v').$$

Therefore, we can compute the dependency weight for any given tuple (u,v,u',v') as

$$\begin{aligned}\alpha(u,v,u',v') &= \frac{1}{4HW} \sum_{x=-H}^{H-1} \sum_{y=-W}^{W-1} \boldsymbol{w}_{x_r,y_r} \cdot \exp\left(-j2\pi\left(\frac{ux_p}{2H} + \frac{vy_p}{2W}\right)\right) \cdot \exp\left(j2\pi\left(\frac{u'x}{2H} + \frac{v'y}{2W}\right)\right) \\ &= \frac{1}{4HW} \sum_{x=-H}^{H-1} \sum_{y=-W}^{W-1} \boldsymbol{w}_{x_r,y_r} \cdot \exp\left(j2\pi\left(\frac{u'x - ux_p}{2H} + \frac{v'y - vy_p}{2W}\right)\right), \end{aligned}$$

where $x_r = x \mod k, x_p = x - x_r$, same for y_r, y_p. Further deduction shows

$$\begin{aligned}\alpha(u,v,u',v') &= \frac{1}{4HW} \sum_{x=-H}^{H-1} \sum_{y=-W}^{W-1} \boldsymbol{w}_{x_r,y_r} \cdot \exp\left(j2\pi\left(\frac{(u'-u)x_p + u'x_r}{2H} + \frac{(v'-v)y_p + v'y_r}{2W}\right)\right) \\ &= \frac{1}{4HW} \sum_{x'=-H/k}^{H/k-1} \sum_{y'=-W/k}^{W/k-1} \sum_{s=0}^{k-1}\sum_{t=0}^{k-1} \boldsymbol{w}_{s,t} \cdot \exp\left(j2\pi k\left(\frac{(u'-u)x'}{2H} + \frac{(v'-v)y'}{2W}\right)\right) \cdot \exp\left(j2\pi\left(\frac{u's}{2H} + \frac{v't}{2W}\right)\right) \\ &= \frac{1}{4HW} \sum_{x'=-H/k}^{H/k-1} \sum_{y'=-W/k}^{W/k-1} \exp\left(j2\pi k\left(\frac{(u'-u)x'}{2H} + \frac{(v'-v)y'}{2W}\right)\right) \sum_{s=0}^{k-1}\sum_{t=0}^{k-1} \boldsymbol{w}_{s,t} \cdot \exp\left(j2\pi\left(\frac{u's}{2H} + \frac{v't}{2W}\right)\right). \end{aligned}$$

Denote $\beta(u',v') = \sum_{s=0}^{k-1}\sum_{t=0}^{k-1} \boldsymbol{w}_{s,t}\cdot \exp\left(j2\pi\left(\frac{u's}{2H}+\frac{v't}{2W}\right)\right)$, which is a constant conditioned on (u',v'). Then we know

$$\begin{aligned}
\alpha(u,v,u',v') =& \frac{\beta(u',v')}{4HW}\sum_{x'=-H/k}^{H/k-1}\sum_{y=-W/k}^{W/k-1}\exp\left(j2\pi k\left(\frac{(u'-u)x'}{2H}+\frac{(v'-v)y'}{2W}\right)\right)\\
=& \frac{\beta(u',v')}{4HW}\sum_{x'=-H/k}^{H/k-1}\exp\left(j2\pi k\left(\frac{(u'-u)x'}{2H}\right)\right)\sum_{y'=-W/k}^{W/k-1}\exp\left(j2\pi k\left(\frac{(v'-v)y'}{2W}\right)\right)\\
=& \begin{cases}\frac{\beta(u',v')}{k^2}, & u'-u=a\cdot\frac{2H}{k}, v'-v=b\cdot\frac{2W}{k}, \quad a,b\in\mathbb{Z}\\ 0, & \text{otherwise}\end{cases}.
\end{aligned} \tag{*3}$$

In general, $\beta(u',v')\neq 0$ when $u'\neq c\cdot\frac{2H}{k}, v'\neq d\cdot\frac{2W}{k}, c,d\in\mathbb{Z}, cd\neq 0$. Hence, when $\frac{2H}{k}|(u'-u), \frac{2W}{k}|(v'-v)$, we have $\alpha(u,v,u',v')\neq 0$ given $uv\neq 0$, while $\alpha(u,0,u',0)\neq 0$ given $u\neq 0$, $\alpha(0,v,0,v')\neq 0$ given $v\neq 0$. Therefore, the image generated through down-sampling contains mixed information from both low frequency and high frequency, since most signals have a non-zero dependency on the global signals of the original image.

Proposition 1.3. *The conclusions of Proposition 1.1 and Proposition 1.2 still hold when $k\in\mathbb{Q}^+$ is not an integer.*

Proof. It is obvious that Proposition 1.1 can be naturally extended to $k\in\mathbb{Q}^+$. Therefore, here we focus on Proposition 1.2. First, consider up-sampling an image $\boldsymbol{X}$ by $m\in\mathbb{N}^+$ times with the nearest interpolation, namely

$$\boldsymbol{X}_{\text{up}}[mx+s,my+t]=\boldsymbol{X}[x,y],\quad s,t\in\{0,1,\dots,m-1\}.$$

Taking Fourier transform, we have

$$\begin{aligned}
\mathcal{F}(\boldsymbol{X}_{\text{up}})[u,v] =& \sum_{x=-mH}^{mH-1}\sum_{y=-mW}^{mW-1}\boldsymbol{X}_{\text{up}}[x,y]\cdot\exp\left(-j2\pi\left(\frac{ux}{2mH}+\frac{kvy}{2mW}\right)\right)\\
=& \sum_{x=-H}^{H-1}\sum_{y=-W}^{W-1}\sum_{s=0}^{m-1}\sum_{t=0}^{m-1}\boldsymbol{X}[x,y]\cdot\exp\left(-j2\pi\left(\frac{u(mx+s)}{2mH}+\frac{v(my+t)}{2mW}\right)\right).
\end{aligned} \tag{*4}$$

Similar to the proof of Proposition 1.2, by plugging (*2) into (*4), it is easy to see that $\boldsymbol{F}_{\text{up}}(u,v)=\mathcal{F}(\boldsymbol{X}_{\text{up}})[u,v]$ is a linear combination of the signals from the original image. Namely, we have

$$\boldsymbol{F}_{\text{up}}(u,v)=\sum_{u'=-H}^{H-1}\sum_{v'=-W}^{W-1}\alpha_{\text{up}}(u,v,u',v')\cdot\boldsymbol{F}(u',v').$$

Given any (u,v,u',v'), $\alpha_{\text{up}}(u,v,u',v')$ can be computed as

$$\begin{aligned}
\alpha_{\text{up}}(u,v,u',v') =& \frac{1}{4HW}\sum_{x=-H}^{H-1}\sum_{y=-W}^{W-1}\sum_{s=0}^{m-1}\sum_{t=0}^{m-1}\exp\left(-j2\pi\left(\frac{u(mx+s)}{2mH}+\frac{v(my+t)}{2mW}\right)\right)\cdot\exp\left(j2\pi\left(\frac{u'x}{2H}+\frac{v'y}{2W}\right)\right)\\
=& \frac{1}{4HW}\sum_{x=-H}^{H-1}\sum_{y=-W}^{W-1}\exp\left(j2\pi\left(\frac{(u'-u)x}{2H}+\frac{(v'-v)y}{2W}\right)\right)\sum_{s=0}^{m-1}\sum_{t=0}^{m-1}\exp\left(-j2\pi\left(\frac{us}{2mH}+\frac{vt}{2mW}\right)\right).
\end{aligned}$$

Denote $\beta_{\text{up}}(u,v)=\sum_{s=0}^{m-1}\sum_{t=0}^{m-1}\exp\left(-j2\pi\left(\frac{us}{2mH}+\frac{vt}{2mW}\right)\right)$, which is a constant conditioned on (u,v). Then we know

$$\begin{aligned}
\alpha_{\text{up}}(u,v,u',v') =& \frac{\beta_{\text{up}}(u,v)}{4HW}\sum_{x=-H}^{H-1}\sum_{y=-W}^{W-1}\exp\left(j2\pi\left(\frac{(u'-u)x}{2H}+\frac{(v'-v)y}{2W}\right)\right)\\
=& \frac{\beta_{\text{up}}(u,v)}{4HW}\sum_{x=-H}^{H-1}\exp\left(j2\pi\left(\frac{(u'-u)x}{2H}\right)\right)\sum_{y=-W}^{W-1}\exp\left(j2\pi\left(\frac{(v'-v)y}{2W}\right)\right)\\
=& \begin{cases}\beta_{\text{up}}(u,v), & u'-u=a\cdot 2H, v'-v=b\cdot 2W, \quad a,b\in\mathbb{Z}\\ 0, & \text{otherwise}\end{cases}.
\end{aligned} \tag{*5}$$

Since $-H \leq u \leq H-1, -W \leq v \leq W-1$, we have $\beta_{\text{up}}(u,v) \neq 0$. Thus, we have $\alpha_{\text{up}}(u,v,u',v') \neq 0$ when $2H|(u'-u), 2W|(v'-v)$.

Now we return to Proposition 1.3. Suppose that the original image is $\boldsymbol{X}$, and that the image obtained through down-sampling is $\boldsymbol{X}_{\text{d}} = \mathcal{D}_{1/k}(\boldsymbol{X})$, where $k \in \mathbb{Q}^{+}$ may not be an integer. We can always find two integers m_0 and k_0 such that $\frac{k_0}{m_0} = k$. Consider first up-sampling $\boldsymbol{X}$ by m_0 times with the nearest interpolation and then performing down-sampling by k_0 times. By combining (*3) and (*5), it is easy to verify that Proposition 1.3 is true. $\square$

D. More Discussions

Potential impacts. The de-facto guarantee for the state-of-the-art performance of modern deep networks (*e.g.*, vision Transformers) incorporates an increasing model size, the large-scale training data, and a sufficiently long training procedure with delicate regularization techniques. However, the establishment of this regime comes at an intensive and unaffordable computational cost for training. Towards this direction, EfficientTrain proposes a simple, easy-to-use, but effective learning approach to reduce the training cost of visual backbones. Our work may benefit real-world applications in terms of accelerating the designing and validating of deep learning architectures or algorithms. Under environmental considerations, it will also help to reduce the carbon emission caused by training large deep learning models. For the research community, EfficientTrain may potentially motivate the researchers to focus on the generalized formulation of curriculum learning.

Limitations and future work. Currently, the EfficientTrain algorithm mainly focuses on training models with images. In the future, we will focus on extending our method to leveraging videos or texts. In addition, it would be interesting to explore whether we can extract the 'easier-to-learn' information from the lens of the spatial or temporal redundancy of vision data [109, 110, 111, 112, 113, 114, 115]. We will also focus on exploring facilitating the efficient training of deep networks by leveraging dynamic network architectures [116, 117, 118].