Accelerating the Training of Video Super-resolution Models

Lijian Lin, Xintao Wang[*], Zhongang Qi, Ying Shan
ARC Lab, Tencent PCG
ljlin@stu.xmu.edu.cn, xintao.alpha@gmail.com, {zhongangqi, yingsshan}@tencent.com

Abstract

Despite that convolution neural networks (CNN) have recently demonstrated high-quality reconstruction for video super-resolution (VSR), efficiently training competitive VSR models remains a challenging problem. It usually takes an order of magnitude more time than training their counterpart image models, leading to long research cycles. Existing VSR methods typically train models with fixed spatial and temporal sizes from beginning to end. The fixed sizes are usually set to large values for good performance, resulting to slow training. However, is such a rigid training strategy necessary for VSR? In this work, we show that it is possible to gradually train video models from small to large spatial/temporal sizes, *i.e.*, in an easy-to-hard manner. In particular, the whole training is divided into several stages and the earlier stage has smaller training spatial shape. Inside each stage, the temporal size also varies from short to long while the spatial size remains unchanged. Training is accelerated by such a multigrid training strategy, as most of computation is performed on smaller spatial and shorter temporal shapes. For further acceleration with GPU parallelization, we also investigate the large minibatch training without the loss in accuracy. Extensive experiments demonstrate that our method is capable of largely speeding up training (up to 6.2× speedup in wall-clock training time) without performance drop for various VSR models.

Introduction

Video super resolution (VSR) (Kappeler et al. 2016; Liu et al. 2020; Chan et al. 2021b; Liang et al. 2022; Wang et al. 2021, 2019; Tian et al. 2020) aims to recover a high-resolution (HR) video from a low-resolution (LR) input, which has gained increasing attention in computer vision community. However, training VSR models is much slower than training image SR models due to the additional temporal dimension. The slow training leads to long research cycles, which impedes the development of VSR models.

Existing VSR models (Wang et al. 2019; Liu et al. 2021; Zhu et al. 2019; Yan, Lin, and Tan 2019; Chan et al. 2021a) are typically trained with fixed spatial and temporal sizes. The sizes are usually set to large values to achieve good performance. Larger sizes require the models to process more spatial and temporal information, which is time-consuming.

[*]Corresponding author.

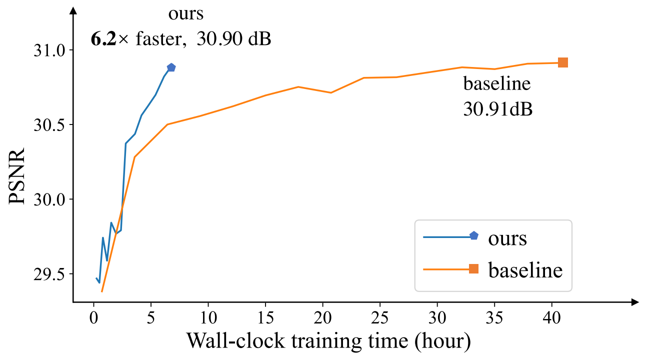


Figure 1: Wall-clock training time speedup and performance comparisons on REDS4 with the BasicVSR-M model. Our method significantly accelerates training (*i.e.,* 6.2×) while maintaining baseline accuracy (30.90 *vs*. 30.91).

Whereas, training on small sizes is relatively easier and faster, but less accurate. It is a natural idea to gradually train VSR models from small to large spatial/temporal sizes, *i.e.*, in an easy-to-hard manner. Specifically, in the early stage of training, the VSR models can be trained with small spatial and temporal sizes, which are relatively easier to learn. When the models perform well on small sizes, we then gradually enlarge the spatial and temporal shapes, making the models focus on reconstructing finer details. Such a learning strategy imitates the way we learn new skills, starting from learning the simple tasks, and then gradually learning the complex and challenging ones. In such a way, the training time is largely reduced, as most of computation is performed on small spatial and short temporal shapes.

Directly applying the above easy-to-hard training strategy to VSR models leads to inferior performance. The reasons are two-fold. Firstly, the spatial and temporal dimensions in videos are highly correlated. Changing the spatial size will affect the learning on the temporal dimension, and altering temporal size affects that of spatial. Thus, simultaneously varying the spatial and temporal sizes from small to large is not optimal. Secondly, the learning rate in VSR models usually starts at a large value and then gradually decays to a small one. With such a learning rate scheduler, the learning rate is relatively small when the spatial and temporal sizes are switched to large ones, hindering the learning ability of the models.

In this paper, we propose a simple yet effective multigrid training strategy that learns to reconstruct in an easy-to-hard manner. The strategy adopts a hierarchical design for altering the spatial and temporal sizes with two cycles. Specifically, we first employ a spatial cycle that varies the spatial size from small to large. Then, inside each spatial stage with fixed spatial size, we further employ a temporal cycle that moves the temporal size from short to long. In order to fit the different degrees of task difficulty when switching spatial and temporal sizes, we introduce a dynamic learning rate scheduler, where the learning rate is re-started with a large value when the spatial or temporal size changes. Large learning rate enhances the exploration ability of the VSR models in transferring from easy tasks (small spatial and short temporal sizes) to harder ones (large spatial and long temporal sizes). Experiments demonstrate that our multigrid training strategy in this easy-to-hard manner achieves significant speedup in wall-clock training time without losing accuracy.

In order to further accelerate the training of VSR, we resort to making full use of the GPU parallelism by large minibatch training. It has been widely explored in high-level vision tasks to accelerate training without accuracy loss (Krizhevsky 2014; Goyal et al. 2017; Chen et al. 2016). However, large minibatch training is still under investigation in VSR. In this paper, we revisit the training of VSR and study how larger minibatch sizes affect the training of VSR. Similar to (Goyal et al. 2017), we apply a linear scaling rule to adjust the learning rate according to minibatch sizes. In addition, it is necessary to have a warmup phase that trains the network with a small learning rate early in training. As a result, each training iteration can process more samples, leading to faster training (see Figure 1).

In summary, we make the following contributions:

- We propose a multigrid training strategy for efficient VSR training. This strategy trains the VSR models in an easy-to-hard manner by varying the spatial and temporal sizes from small to large.
- Large minibatch training is investigated in VSR to effectively accelerate the training of VSR models.
- Extensive experiments on various VSR models demonstrate the effectiveness and generalization of multigrid training and large minibatch training. Especially, our method is capable of achieving up to 6.2× speedup in wall-clock training time while maintaining accuracy for recently VSR models.

Related Works

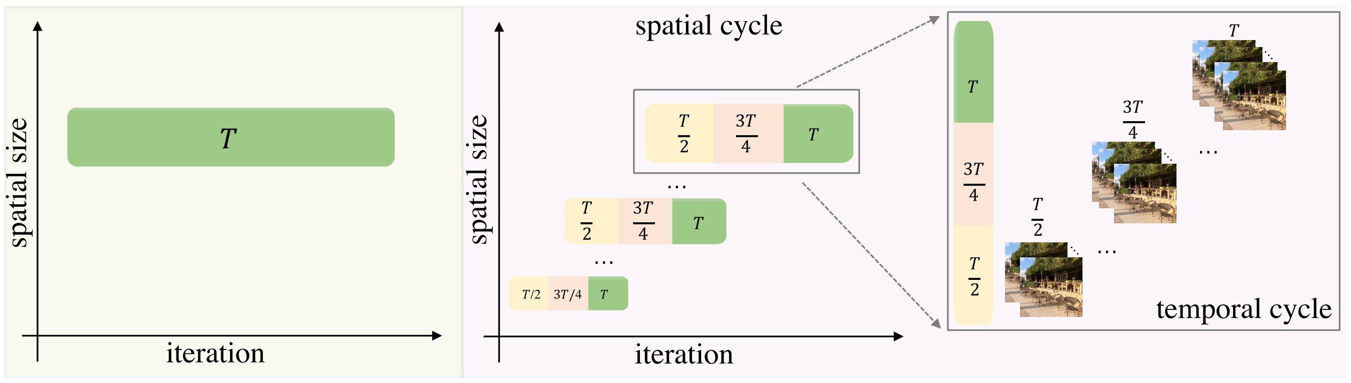
Video Super Resolution. Existing VSR methods can be roughly classified into two types: sliding-window-based methods (Wang et al. 2019; Caballero et al. 2017; Tao et al. 2017; Xue et al. 2019a; Tian et al. 2020; Jo et al. 2018), and recurrent-based methods (Chan et al. 2021a; Isobe et al. 2020a,b; Cao et al. 2021). Sliding-window methods tend to restore a single frame using several neighboring frames within a temporal window. Several sliding-window methods (Caballero et al. 2017; Tao et al. 2017; Xue et al. 2019a) adopt optical flow between frames to guide the spatial warping for temporal alignment. EDVR (Wang et al. 2019) further designs a pyramid alignment module to perform alignment in a coarse-to-fine manner and a fusion module to fuse the features of different frames.

As one of the representative recurrent-based methods, RSDN (Isobe et al. 2020a) proposes a structure-detail block and a hidden state adaptation module to exploit previous frames to super-resolve the LR frame. BasicVSR (Chan et al. 2021a) adopts a bidirectional recurrent design to propagate the information in videos and employs a simple flow-based alignment to align the features, achieving state-of-the-art performance. Despite their promising performance, the long training time hinders the development of VSR models. In this paper, we aim at accelerating the training of VSR models without a performance drop. We evaluate the effectiveness of the proposed training strategy on both the sliding-window-based (*i.e.,* EDVR) and the recurrent-based (*i.e.,* BasicVSR) VSR methods.

Curriculum Learning. Curriculum learning is a training strategy that trains machine learning models from easy to hard, which imitates the learning order in human curricula. Researchers have exploit its powers in increasing the convergence speed and improving the performance over various tasks, *e.g.*, object detection (Chen and Gupta 2015; Li et al. 2017; Sangineto et al. 2018) and neural machine translation (Wang, Chen, and Zhu 2021; Wang, Caswell, and Chelba 2019; Tay et al. 2019). Among the works in curriculum learning, Bengio *et al.* (Bengio et al. 2009) trains machine learning models by gradually increasing the complexity of training samples. The work in (Karras et al. 2018) proposes to gradually increase the model complexity by adding new layers during training, which both decreases the training time and achieves better performance. Note that, our easy-to-hard training strategy can be treated as a type of curriculum learning, which has not been investigated in VSR.

Efficient training. Recently has witnessed great success in accelerating training in high-level vision tasks (Wu et al. 2020; Goyal et al. 2017; Huang et al. 2019; Mostafa and Wang 2019; Jeong, Park, and Ha 2018; Wu et al. 2019; Qin et al. 2018) (*e.g.,* image classification, object detection). The work in (Goyal et al. 2017) presents a linear scaling rule that speeds up training by using large minibatches. Wu *et al.* (Wu et al. 2020) propose to accelerate the training of video action recognition models with variable minibatch shapes, which achieves a significant speedup in wall-clock training time. The work in (You et al. 2020) designs a layer-wise training framework for graph convolution networks (Kipf and Welling 2016) that disentangles the feature aggregation and feature transformation during training, leading to a great reduction of time and memory consumption.

Despite the success of the above-mentioned works, how to accelerate the training of VSR models has still barely been investigated. This paper revisits the training of VSR models and presents two effective techniques to speed up VSR training while maintaining accuracy.



(a) **Baseline** (b) **Multigrid**

Figure 2: Training process of the proposed multigrid training strategy *vs*. baseline training. (a) Baseline training typically adopts a constant and large spatial/temporal size during the whole training process. (b) Multigrid training first varies the spatial size from small to large as the training processes. Then, for each stage with fixed spatial size, the temporal size moves from small to large as well. Yellow, pink, and green indicate temporal sizes $\frac{T}{2}$, $\frac{3T}{4}$, and T, respectively.

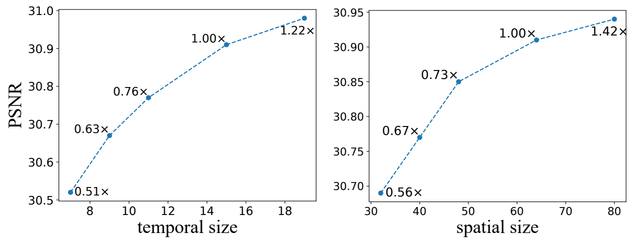
Method

In this section, we first present our multigrid training strategy. Then, we introduce how to train VSR models with large minibatches.

Multigrid Training

Despite the success of image SR methods, directly applying image models to videos leads to inferior performance, as they process each frame separately and thus ignore the rich information among frames. A common practice to improve the accuracy of VSR is to train the SR methods with multiple frames (*i.e.,* large temporal size). However, as the number of frames grows, training becomes slower (Figure 3(a)), as the models need to process more temporal information in one forward. Similarly, larger spatial size yields better VSR performance but long training time (Figure 3(b)). It is natural to raise the question: *is it necessary to keep the spatial and temporal sizes large and fixed during the whole training process?* In this paper, we show that the answer is *No*. An intuitive idea is to first train the VSR models at small spatial and temporal sizes, and then gradually switch to larger ones, *i.e.,* in an easy-to-hard manner. Specifically, in the early stage of training, the network is trained with small spatial and temporal sizes, which is relatively easier and faster. However, it suffers from limited information contained in small sizes, leading to inferior performance. One can improve the performance by increasing the spatial and temporal sizes, due to larger sizes make the network focus on fusing more information and reconstructing finer details. In such a way, most of the training iterations are conducted with smaller spatial and shorter temporal sizes, leading to faster training.

Multigrid Training Strategy. Motivated by the above discussions, we propose a multigrid VSR training strategy that varies the spatial and temporal sizes from small to large throughout training. Figure 2 illustrates the overview of our multigrid training strategy. This strategy adopts a hierarchical design with two cycles for altering spatial and temporal sizes. Specifically, the whole training is divided into several stages and the earlier stage has smaller training spatial shapes. Inside each stage, the temporal size also varies from short to long while spatial size remains unchanged. Next, we will introduce the details of the proposed spatial cycle and temporal cycle.



(a) different temporal sizes (b) different spatial sizes

Figure 3: PSNR performance *vs*. different temporal size (a), and different spatial size (b). '$\cdot\times$' indicates the relative wall-clock time speedup compared to baseline ($1.00\times$). The results are obtained with the BasicVSR-M model.

Spatial Cycle. For the spatial cycle, there exists large design space for 1) different spatial sizes, and 2) duration of each spatial stage. Intuitively, the training will be faster if the spatial size starts at smaller values. However, training with a very small spatial size (*e.g.,* 16×16) leads to a large accuracy drop. The reason might be the unsatisfying optical flow estimation due to the insufficient information produced by such small patches. Therefore, the spatial sizes in the spatial cycle should not be too small. Moreover, in order to achieve baseline accuracy, we set the spatial size in the last spatial stage to the default size ($H \times W$) used in the baseline. For simplicity, we equally divide the whole training process into s spatial stages, each trained with a fixed spatial size.

Temporal Cycle. Similarly, the challenge of designing a temporal cycle lies in two aspects: 1) different temporal sizes, and 2) duration of each temporal stage. As larger temporal sizes yield longer training time, a natural desire is to start training with a smaller temporal size. However, the performance of VSR drops a lot with a very small temporal size (*e.g.,* 3), as too few adjacent frames could not provide enough complementary information. Therefore, we start the temporal cycle with a temporal size not less than 6 and gradually enlarge it until reaches the original temporal size T in the baseline. We equally divide each temporal cycle into f temporal stages.

Moreover, our experiments suggest that directly increasing the spatial and temporal sizes at the same time leads to sub-optimal results. Thus, rather than changing them synchronously, we adopt a hierarchical design with two cycles for altering spatial and temporal sizes. In particular, for each spatial stage, the temporal sizes will also be varied through a complete temporal cycle, leading to a total of $p = s \times f$ spatial-temporal stages in the whole training process.

Dynamic Learning Rate Scheduler. Simply applying the multigrid strategy to VSR training causes an accuracy drop. The devil is the learning rate scheduler. Typically, the learning rate in VSR training is initialized with a relatively large value and then decayed as training progresses. If we apply the multigrid training into a baseline VSR network using the original learning rate scheduler, the learning rate in training larger spatial and temporal sizes will be smaller. The small learning rate hinders the exploration ability of the network when spatial and temporal sizes are switched to larger ones.

In this paper, we propose a dynamic learning rate scheduler, which adjusts the learning rate to fit the different degrees of task difficulty when switching spatial/temporal sizes. Specifically, the scheduler re-starts the learning rate with large values when the spatial or temporal size changes. Following previous practice (Wang et al. 2019; Chan et al. 2021a), we apply the cosine annealing strategy for better convergence. In the multigrid training, the learning rate η_t at iteration t is formulated as follows:

$$\eta_t = \begin{cases} \cos(\frac{t-\sum_{j=1}^{s(t)-1} P_j}{I_{total}}) \times \eta, & 0 < s(t) \leq p-1 \\ \cos(\frac{t-\sum_{j=1}^{p-1} P_j}{P_p}) \times \eta, & s(t) = p-1 \end{cases}, \tag{1}$$

where, η indicates the initial learning rate used in baseline. P_j represents the number of training iterations for spatial-temporal stage j, and I_{total} is the total training iterations, satisfying: $\sum_{j=1}^{p} P_j = I_{total}$. $s(t) \in \{1, 2, ..., p\}$ indicates the iteration t belongs to the $s(t)$ spatial-temporal stage, *i.e.*, when $0 \leq t < P_1, s(t) = 1$. Since the total iterations I_{total} is always larger than $t - \sum_{j}^{s(t)-1} P_j$, the learning rate η_t will never decay to zero for $P_1, P_2, ..., P_{p-1}$, which avoids wasting training iterations with too small learning rates.

Large Minibatch Training

Recently, researchers have made significant developments in accelerating training by increasing minibatch sizes in high-level vision tasks (Goyal et al. 2017; Krizhevsky 2014) (*e.g.* image classification, object detection). Increasing minibatch sizes enables a network to process more samples in parallel, thus they can train the same number of epochs faster. However, how to train VSR networks faster by using larger minibatch sizes while maintaining accuracy has barely been investigated. In this paper, we investigate the large minibatch training for VSR. Similar to the practice developed in high-level tasks (Goyal et al. 2017), we conclude two important rules for accelerating training with large minibatches. 1) Linearly scale the learning rate when the minibatch size changes. 2) Warmup the network with a smaller learning rate at the beginning.

Next, we will review the training of VSR networks to discuss why the above-mentioned rules are effective. Considering a typical VSR training with minibatch using a loss $L(w)$:

$$L(w) = \frac{1}{n} \sum_{x \in X} l(x, w), \tag{2}$$

where X is a minibatch and $n = |X|$ indicates the number of samples in X (*i.e.,* minibatch size). w is the weights of a VSR network. $l(\cdot, \cdot)$ is the loss between the output of the network and ground truth.

We analyze the differences between training m iterations with m small minibatches X_{0-m}, and training a single iteration with one large minibatch $\boldsymbol{\mathcal{X}}$. Each of the minibatch X_{0-m} holds n samples. $\boldsymbol{\mathcal{X}}$ consists of those small minibatches X_{0-m}, which means $|\boldsymbol{\mathcal{X}}| = mn$. According to Eq. 2, after m iteration of training using m minibatches X_{0-m}, the weights are updated as follows:

$$w_{t+m} = w_t - \eta \frac{1}{n} \sum_{i}^{m} \sum_{x \in X_i} \nabla l(x, w_{t+i}), \tag{3}$$

where η is the learning rate and t indicates the training iteration index. ∇l is the gradient according to loss $l(\cdot, \cdot)$. Similarly, when executing a single iteration with minibatch $\boldsymbol{\mathcal{X}}$ of size mn, the weights will be:

$$w'_{t+1} = w_t - \eta' \frac{1}{mn} \sum_{x \in \boldsymbol{\mathcal{X}}} \nabla l(x, w_t). \tag{4}$$

Note that if we assume for $i < m$, $w_i \approx w_{i+m}$, then:

$$\sum_{i}^{m} \sum_{x \in X_i} \nabla l(x, w_{t+i}) \approx \sum_{x \in \boldsymbol{\mathcal{X}}} \nabla l(x, w_t). \tag{5}$$

Thus, when the learning rate is $\eta' = m\eta$, we will have $w_{t+m} \approx w'_{t+1}$. This indicates that we can train the network with larger minibatch sizes and fewer iterations to approximate the baseline training with linear scaled learning rate.

Besides, this linear scaling rule relies on the assumption that $w_i \approx w_{i+m}$. This assumption might fail when the weights change rapidly. Since the rapid changing of weights usually occurs in the early stages of training, we apply a warmup strategy that gradually increases the learning rate from a small value to a large one to alleviate this issue.

Experiments

Implementation Details

Spatial Cycle. We equally divide the training process into $s = 2$ spatial stages. The spatial sizes in these two stages are set to $max(32 \times 32, \frac{H}{2} \times \frac{W}{2})$ and $H \times W$, respectively. Training samples with different spatial sizes are generated by randomly cropping the original frames.

model	id	minibatch	multi	speedup	PSNR					SSIM				
					average	clip_00	clip_11	clip_15	clip_20	average	clip_00	clip_11	clip_15	clip_20
BasicVSR-M	①	16	-	-	30.91	28.12	31.78	33.50	30.23	0.8824	0.8337	0.8885	0.9148	0.8927
Large-Batch	②	64	-	3.9×	30.93	28.13	31.82	33.53	30.24	0.8824	0.8344	0.8885	0.9137	0.8930
Multi-S	③	64	S	4.9×	30.89	28.11	31.77	33.47	30.23	0.8822	0.8337	0.8881	0.9146	0.8926
Multi-T	④	64	T	5.0×	30.93	28.13	31.83	33.56	30.22	0.8830	0.8342	0.8889	0.9161	0.8928
ours	⑤	64	S&T	**6.2**×	30.90	28.10	31.72	33.58	30.20	0.8820	0.8328	0.8871	0.9160	0.8921
BasicVSR*	⑤	32	-	-	31.42	28.40	32.47	30.63	30.63	0.8909	0.8434	0.8979	0.9224	0.9000
BasicVSR (our impl.)	⑥	32	-	-	31.58	28.48	32.74	34.33	30.78	0.8934	0.8462	0.9010	0.9239	0.9026
Large-Batch	⑦	64	-	1.9×	31.62	28.51	32.77	34.44	30.78	0.8943	0.8467	0.9017	0.9259	0.9029
Multi-S	⑧	64	S	2.4×	31.56	28.47	32.73	34.29	30.75	0.8932	0.8460	0.9008	0.9236	0.9023
Multi-T	⑨	64	T	2.5×	31.61	28.50	32.76	34.41	30.76	0.8941	0.8467	0.9014	0.9257	0.9026
ours	⑩	64	S&T	**3.1**×	31.54	28.46	32.65	34.32	30.72	0.8925	0.8447	0.8998	0.9237	0.9016

Table 1: Quantitative comparison on REDS4 with BasicVSR. We report the wall-clock speedup relative to baseline training (②-⑤ *vs.*①, and ⑦-⑩ *vs.*⑥, respectively). * means the results are collect from the original paper. Best performance is highlighted with bold. 'S' and 'T' denote spatial and temporal, respectively.

model	id	minibatch	multi	speedup	PSNR					SSIM				
					average	clip_00	clip_11	clip_15	clip_20	average	clip_00	clip_11	clip_15	clip_20
EDVR-M*	①	32	-	-	30.46	27.70	31.19	33.41	29.53	0.8684	0.8134	0.8716	0.9123	0.8762
EDVR-M (our impl.)	②	32	-	-	30.45	27.70	31.19	33.40	29.52	0.8687	0.8141	0.8718	0.9125	0.8763
Large-Batch	③	64	-	1.9×	30.46	27.70	31.21	33.39	29.54	0.8689	0.8140	0.8723	0.9126	0.8768
ours	④	64	S	**2.3**×	30.44	27.68	31.16	33.41	29.51	0.8685	0.8137	0.8715	0.9128	0.8762

Table 2: Quantitative comparison on REDS4 with EDVR-M. We report the wall-clock speedup relative to baseline training (③-④ *vs.*②). * means the results are collect from the original paper. Best performance is highlighted with bold. 'S' and 'T' denote spatial and temporal, respectively.

Temporal Cycle. For each spatial stage in the spatial cycle, we further equally divide it into $f = 3$ temporal stage. The temporal sizes (*i.e.,* number of consecutive frames fed into VSR models) in these three temporal stages are set in an increasing way: $max(6, \frac{T}{2})$, $\frac{3T}{4}$ and T. These three temporal sizes cover an intuitive range and work well in practice. By doing so, there will be $p = 2 \times 3 = 6$ spatial-temporal stages in total during the whole training process.

Learning Rate Scheduler. Our dynamic learning rate scheduler consists of p periods, which are synchronized with the above-mentioned p spatial-temporal stages. For each period, the learning rate begins at a large value (the initial learning rate used in baseline) and then decays following the cosine annealing (Loshchilov and Hutter 2016).

Datasets and Evaluation Metrics. We conduct our experiments on the REDS (Nah et al. 2019) and Vimeo-90K (Xue et al. 2019b) datasets, which are widely-used and challenging datasets for VSR. REDS contains 300 video clips with a total of $300,000$ frames. Following (Chan et al. 2021a; Wang et al. 2019), we adopt REDS4 as our test set, and use the left as training set. Vimeo-90K contains $64,612$ training, and $7,824$ testing 7-frame video sequences. All the datasets are commonly used in VSR and licensed for research purposes. The performance is measured in terms of PSNR and SSIM. As we use remote data access, the unstable data loading highly affects the measure of wall-clock training time. Thus, we report the wall-clock training time without the data loading time.

Training and Inference Details. We adopt two VSR models to evaluate the effectiveness of the proposed method: BasicVSR-M (Chan et al. 2021a), BasicVSR (Chan et al. 2021a), and EDVR-M (Wang et al. 2019) (M denotes the medium size). For BasicVSR-M and BasicVSR, the spatial sizes used in the spatial cycle are: 32×32 and 64×64 on both REDS and Vimeo-90K. The temporal sizes in the temporal cycle are $\{7, 11, 15\}$ and $\{6, 10, 14\}$ on REDS and Vimeo-90K, respectively. Note that EDVR-M adopts a sliding window design, where the temporal size is determined by its model architecture, and usually cannot be changed for both training and testing. Thus, we only apply the spatial cycle to it. The spatial sizes for EDVR-M are 32×32 and 64×64 on both REDS and Vimeo-90K. The learning rate of training on 32×32 spatial size begins at $2e - 4$, as too large learning rate may cause severe performance drop for EDVR-M. In addition, we train these models with linear learning rate warmup for the first $5,000$ iterations. The training and analyses are performed with PyTorch on NVIDIA V100 GPUs in an internal cluster.

Experiments on REDS

Results on BasicVSR-M and BasicVSR. The quantitative results obtained by BasicVSR-M and BasicVSR are summarized in Table 1. Applying multigrid training and large minibatch training to BasicVSR-M and BasicVSR achieves significant speedup (*i.e.,* $6.2\times$, $3.1\times$ for BasicVSR-M and BasicVSR, respectively) without losing accuracy. Specifically, 1) the training becomes $3.9\times$ and $1.9\times$ faster when training BasicVSR-M and BasicVSR with $4\times$ and $2\times$ larger minibatches. The speedup can be attributed to that large minibatch sizes enable better GPU parallelization. 2) The

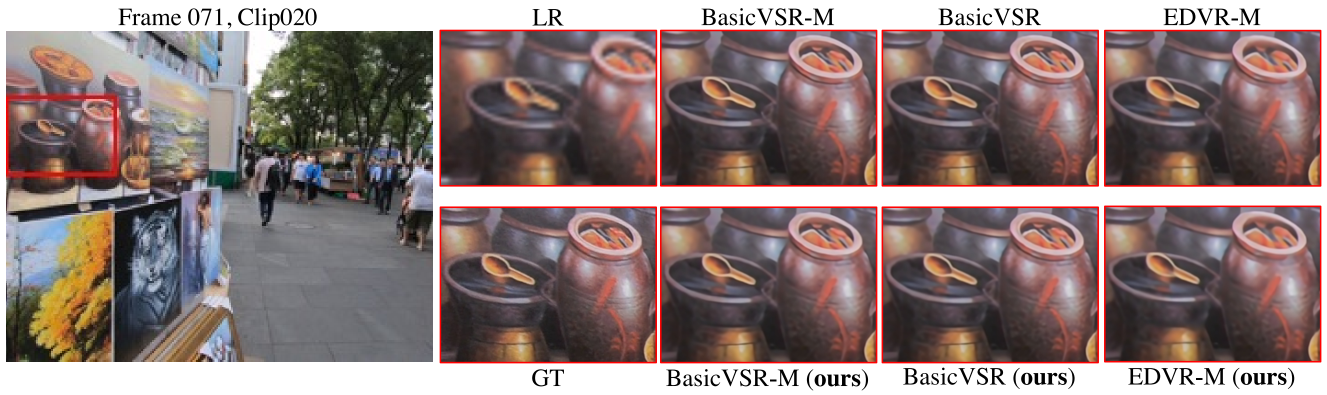


Figure 4: Qualitative results on REDS4 for 4× VSR on BasicVSR-M, BasicVSR, and EDVR-M. The methods with our multigrid training and large minibatch training achieve comparable visual results to the baselines. Zoom in for best view.

model	id	speedup	PSNR	SSIM
Baseline (BasicVSR*)	①	-	37.18	0.9450
ours	②	**3.2**×	37.32	0.9462
Baseline (EDVR-M)	③	-	35.10	0.9426
ours	④	**2.3**×	35.22	0.9437

Table 3: Quantitative comparison on Vimeo-90k. We report the wall-clock speedup relative to baseline training (② *vs.*①, and ④ *vs.*③). * means the results are collect from the original paper. Best performance is highlighted with bold.

training time can be further reduced by employing the proposed multigrid training strategy (see Table 1 ③-⑤, ⑧-⑩). Table 1 shows that both the spatial cycle and temporal cycle bring consistent speedup to BasicVSR with different model sizes. Moreover, combining them together (*i.e.,* the proposed multigrid training strategy) achieves the fastest training while maintaining accuracy. These results suggest that a VSR model can be efficiently trained in an easy-to-hard manner (*i.e.,* from small spatial/temporal sizes to larger ones), and finally reach the baseline performance.

Results on EDVR-M. Next, we apply the proposed techniques to a sliding-window-based method EDVR-M. As shown in Table 2, training EDVR-M with the multigrid training strategy and large minibatch leads to a significant 2.3× speedup. We observe that the speedup is consistent with the recurrent-based method BasicVSR. These results demonstrate that the two proposed strategies are robust and can be easily generalized to different VSR models.

Qualitative Results. We also present some qualitative results obtained by our method and baseline in Figure 4. The methods with our multigrid training and large minibatch training obtain comparable visual results to the baselines.

Experiments on Vimeo-90K

Next, we evaluate the proposed method over BasicVSR and EDVR-M on Vimeo-90K to investigate its generalization toward different VSR datasets.

Results on BasicVSR. As in Table 3 (② *vs.*①), applying the proposed multigrid training and large minibatch training to BasicVSR leads to a significant speedup (*i.e.*, 3.2×) while maintaining baseline accuracy.

Results on EDVR-M. As in Table 3 (④ *vs.*③), our strategies bring a significant speedup (*i.e.*, 2.3×) for EDVR-M without performance drop.

The speedups on the Vimeo-90K dataset are consistent with that on the REDS dataset mentioned in last subsection for those two VSR methods. These results demonstrate that the proposed multigrid training and large minibatch training are robust and can be easily generalized to both different VSR methods and different VSR datasets.

Ablation Studies and Analysis

Learning Rate Scaling. As shown in Tabel 4(a), directly increasing the minibatch size leads to a performance drop. Whereas, when conducting learning rate scaling, large minibatch training (with warmup) achieves comparable performance to baseline. This suggests that, with the linear scaled learning rate, the total gradient of large minibatch is roughly equal to that of small ones.

Warmup. As in Tabel 4(b), directly applying the linear scaling rule without warmup results in inferior performance. This is probably due to that the network changes rapidly in the early stage of training. Thus the approximation between a single step on a large minibatch, and several steps on small minibatches may fail. As Tabel 4(b) shows, with the help of the warmup phase, the performance of training with a large minibatch can achieve baseline performance.

Moreover, we train BasicVSR-M with different minibatch sizes to evaluate the robustness of large minibatch training. As shown in Tabel 4(c), increasing the minibatch sizes provides a solid and significant speedup while maintaining baseline accuracy. In addition, we observe that the speedup factors almost match the minibatch scaling factors, thanks to the GPU parallelism.

Spatial Cycle. We evaluate the performance of the proposed spatial cycle with different combinations of dynamic spatial sizes in Tabel 5(a). All the variances of Multi-S are trained with the baseline temporal size (*i.e.,* 15) and the proposed learning rate scheduler. As shown in Table 5(a), varying spatial size from small to large always achieves the baseline

model	minibatch	lr	iteration	PSNR
baseline	16	2e-4	300k	30.91
w/o lr scaling	64	2e-4	75k	30.64
w lr scaling	64	8e-4	75k	**30.93**

(a) learning rate scaling

model	minibatch	warmup	PSNR
baseline	16	-	30.91
w/o warmup	64	-	30.81
w warmup	64	constant	30.91
	64	linear	**30.93**

(b) warmup phase

model	minibatch	speedup	PSNR
baseline	16	-	30.91
	32	2.0×	30.92
variants	48	2.9×	**30.95**
	64	**3.9**×	30.93

(c) large minibatch size *vs*. baseline

Table 4: Ablation study of Minibatch size ***vs.*** Performance. We evaluate the impact of learning rate scaling (a), warmup phase (b), and different minibatch size(c) . All the results are obtained by BasicVSR-M on REDS4. Each GPU holds 4 samples.

model	spatial size	speedup	PSNR
baseline	64	-	30.91
	32/64	4.9×	30.89
Multi-S	32/48/64	5.0×	30.89
	32/40/48/64	5.1×	30.89

(a) spatial cycle

model	temporal size	speedup	PSNR
baseline	15	-	30.91
	7/15	5.0×	30.92
Multi-T	7/11/15	5.0×	30.93
	7/9/11/15	5.2×	30.91

(b) temporal cycle

Table 5: Ablation study of spatial and temporal cycles. We evaluate the impact of different spatial cycle (a), and temporal cycle (b) designs. The sizes are presented according to their orders in training. For example, '32/64' indicates the spatial size begins at 32×32 and then is switched to 64×64. Our learning rate scheduler is used in all settings. We report the wall-clock speedup relative to baseline.

model	spatial & temporal size	speedup	PSNR
baseline	64&15	-	30.91
Multi-S&T	32&7 / 64&15	5.8×	30.81
	32&7 / 48&11 / 64&15	6.0×	30.83

(a) synchronous

model	spatial & temporal size	speedup	PSNR
baseline	64&15	-	30.91
Multi-S&T	32&7 / 32&15 / 64&7 / 64&15	6.3×	30.86
	32&7/32&11/32&15/64&7/64&11/64&15	6.2×	30.90

(b) hierarchical

Table 6: Ablation study of different combinations of spatial and temporal cycles. We combine the proposed spatial and temporal cycles in two different ways: (a) synchronous: change spatial and temporal sizes at the same time; (b) hierarchical: place the temporal cycle into each spatial stage in the spatial cycle. The sizes are presented according to their orders in training. For example, '32&7/64&15' indicates that the training beings with a spatial size of 32×32 and a temporal size of 7, and then is switched to a spatial size of 64×64 and a temporal size of 15. The proposed learning rate scheduler is used in all settings. We report the wall-clock speedup relative to baseline.

performance for different spatial size schemes. These results show the robustness of changing spatial size during training. In addition, more spatial sizes yield slightly faster training, while having slightly lower performance. In order to get a better trade-off between high performance and faster training, we adopt the '32/64' scheme as our default setting.

Temporal Cycle. The performance of different combinations of temporal sizes is summarized in Table 5(b). All the variances of Multi-T are trained with the baseline spatial size (*i.e.,* 64×64) and the proposed dynamic learning rate scheduler. Similar to the spatial cycle, training with different combinations of temporal sizes can always accelerate training while maintaining baseline accuracy. We employ the '$7/11/15$' scheme in our temporal cycle, which is a good trade-off between effectiveness and efficiency.

Mulitigrid. As shown in Table 6(a), simply combining the sizes in spatial and temporal cycles in a synchronous way (*i.e.,* change the spatial and temporal size at the same time) causes a performance drop. We conjecture that the large magnitude of information change brought by simultaneously varied spatial/temporal sizes might hinder the learning process. Table 6(b) shows that combining the spatial and temporal cycles in a hierarchical way (*i.e.,* the proposed multigrid training strategy which places the temporal cycle into each spatial stage in the spatial cycle) leads to $6.2\times$ speedup without losing accuracy. These results demonstrate the effectiveness of our multigrid design.

Conclusion

In this paper, we propose to accelerate the training of VSR methods with multigrid training and large minibatch training. Different from existing VSR methods that are trained with fixed spatial and temporal sizes, the proposed multigrid training varies the spatial and temporal sizes from small to large, *i.e.*, in an easy-to-hard manner. The training is accelerated by such a multigrid training strategy, as most of computation is performed on smaller spatial and shorter temporal shapes. Moreover, we investigate the large minibatch training without accuracy loss for further acceleration with GPU parallelization. Extensive experiments on different methods and datasets demonstrate the effectiveness and generalization of the proposed multigrid training and large minibatch training in VSR.

References

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *Proceedings of the 26th annual International Conference on Machine Learning*, 41–48.

Caballero, J.; Ledig, C.; Aitken, A.; Acosta, A.; Totz, J.; Wang, Z.; and Shi, W. 2017. Real-time video super-resolution with spatio-temporal networks and motion compensation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4778–4787.

Cao, Y.; Wang, C.; Song, C.; Tang, Y.; and Li, H. 2021. Real-Time Super-Resolution System of 4K-Video Based on Deep Learning. In *Proceedings of IEEE International Conference on Application-specific Systems, Architectures and Processors*, 69–76.

Chan, K. C.; Wang, X.; Yu, K.; Dong, C.; and Loy, C. C. 2021a. BasicVSR: The search for essential components in video super-resolution and beyond. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4947–4956.

Chan, K. C.; Wang, X.; Yu, K.; Dong, C.; and Loy, C. C. 2021b. Understanding deformable alignment in video super-resolution. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 973–981.

Chen, J.; Pan, X.; Monga, R.; Bengio, S.; and Jozefowicz, R. 2016. Revisiting distributed synchronous SGD. *arXiv preprint arXiv:1604.00981*.

Chen, X.; and Gupta, A. 2015. Webly supervised learning of convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 1431–1439.

Goyal, P.; Dollár, P.; Girshick, R.; Noordhuis, P.; Wesolowski, L.; Kyrola, A.; Tulloch, A.; Jia, Y.; and He, K. 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

Huang, Y.; Cheng, Y.; Bapna, A.; Firat, O.; Chen, D.; Chen, M.; Lee, H.; Ngiam, J.; Le, Q. V.; Wu, Y.; et al. 2019. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Proceedings of Advances in Neural Information Processing Systems*, volume 32, 103–112.

Isobe, T.; Jia, X.; Gu, S.; Li, S.; Wang, S.; and Tian, Q. 2020a. Video super-resolution with recurrent structure-detail network. In *Proceedings of the European Conference on Computer Vision*, 645–660.

Isobe, T.; Li, S.; Jia, X.; Yuan, S.; Slabaugh, G.; Xu, C.; Li, Y.-L.; Wang, S.; and Tian, Q. 2020b. Video super-resolution with temporal group attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8008–8017.

Jeong, H.-J.; Park, K.-S.; and Ha, Y.-G. 2018. Image preprocessing for efficient training of YOLO deep learning networks. In *Proceedings of the International Conference on Big Data and Smart Computing*, 635–637.

Jo, Y.; Oh, S. W.; Kang, J.; and Kim, S. J. 2018. Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3224–3232.

Kappeler, A.; Yoo, S.; Dai, Q.; and Katsaggelos, A. K. 2016. Video super-resolution with convolutional neural networks. *IEEE Transactions on Computational Imaging*, 2(2): 109–122.

Karras, T.; Aila, T.; Laine, S.; and Lehtinen, J. 2018. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *International Conference on Learning Representations*.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Krizhevsky, A. 2014. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*.

Li, S.; Zhu, X.; Huang, Q.; Xu, H.; and Kuo, C.-C. J. 2017. Multiple instance curriculum learning for weakly supervised object detection. *arXiv preprint arXiv:1711.09191*.

Liang, J.; Cao, J.; Fan, Y.; Zhang, K.; Ranjan, R.; Li, Y.; Timofte, R.; and Van Gool, L. 2022. VRT: A Video Restoration Transformer. *arXiv preprint arXiv:2201.12288*.

Liu, H.; Ruan, Z.; Zhao, P.; Dong, C.; Shang, F.; Liu, Y.; and Yang, L. 2020. Video super resolution based on deep learning: A comprehensive survey. *arXiv preprint arXiv:2007.12928*.

Liu, H.; Zhao, P.; Ruan, Z.; Shang, F.; and Liu, Y. 2021. Large motion video super-resolution with dual subnet and multi-stage communicated upsampling. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 2127–2135.

Loshchilov, I.; and Hutter, F. 2016. SGDR: Stochastic Gradient Descent with Warm Restarts. *arXiv e-prints*, arXiv–1608.

Mostafa, H.; and Wang, X. 2019. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *Proceedings of the International Conference on Machine Learning*, 4646–4655.

Nah, S.; Baik, S.; Hong, S.; Moon, G.; Son, S.; Timofte, R.; and Lee, K. M. 2019. NTIRE 2019 Challenge on Video Deblurring and Super-Resolution: Dataset and Study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1996–2005.

Qin, Z.; Zhang, Z.; Li, D.; Zhang, Y.; and Peng, Y. 2018. Diagonalwise refactorization: An efficient training method for depthwise convolutions. In *Proceedings of the International Joint Conference on Neural Networks*, 1–8.

Sangineto, E.; Nabi, M.; Culibrk, D.; and Sebe, N. 2018. Self paced deep learning for weakly supervised object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(3): 712–725.

Tao, X.; Gao, H.; Liao, R.; Wang, J.; and Jia, J. 2017. Detail-revealing deep video super-resolution. In *Proceedings of the IEEE International Conference on Computer Vision*, 4472–4480.

Tay, Y.; Wang, S.; Luu, A. T.; Fu, J.; Phan, M. C.; Yuan, X.; Rao, J.; Hui, S. C.; and Zhang, A. 2019. Simple and

Effective Curriculum Pointer-Generator Networks for Reading Comprehension over Long Narratives. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Tian, Y.; Zhang, Y.; Fu, Y.; and Xu, C. 2020. Tdan: Temporally-deformable alignment network for video super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3360–3369.

Wang, W.; Caswell, I.; and Chelba, C. 2019. Dynamically Composing Domain-Data Selection with Clean-Data Selection by "Co-Curricular Learning" for Neural Machine Translation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Wang, X.; Chan, K. C.; Yu, K.; Dong, C.; and Change Loy, C. 2019. Edvr: Video restoration with enhanced deformable convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 0–0.

Wang, X.; Chen, Y.; and Zhu, W. 2021. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Wang, X.; Xie, L.; Dong, C.; and Shan, Y. 2021. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1905–1914.

Wu, C.-Y.; Girshick, R.; He, K.; Feichtenhofer, C.; and Krahenbuhl, P. 2020. A multigrid method for efficiently training video models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 153–162.

Wu, D.; Zheng, S.-J.; Bao, W.-Z.; Zhang, X.-P.; Yuan, C.-A.; and Huang, D.-S. 2019. A novel deep model with multi-loss and efficient training for person re-identification. *Neurocomputing*, 324: 69–75.

Xue, T.; Chen, B.; Wu, J.; Wei, D.; and Freeman, W. T. 2019a. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8): 1106–1125.

Xue, T.; Chen, B.; Wu, J.; Wei, D.; and Freeman, W. T. 2019b. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8): 1106–1125.

Yan, B.; Lin, C.; and Tan, W. 2019. Frame and feature-context video super-resolution. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 5597–5604.

You, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. L2-GCN: Layer-Wise and Learned Efficient Training of Graph Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Zhu, X.; Li, Z.; Zhang, X.-Y.; Li, C.; Liu, Y.; and Xue, Z. 2019. Residual invertible spatio-temporal network for video super-resolution. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 5981–5988.