Branchy parallelized U-net structure Branching, finetune, reparameterized, pruning paralleled branched Pruning reparameterized super-resolution model

Anonymous CVPR submission

Paper ID *****

Abstract

Single image super-resolution (SISR) is a computer vision technique that uses deep convolution neural networks to reconstruct high-resolution images from low-resolution images. It has been used in many applications, but due to power constraints on mobile devices, efficient image super-resolution (EISR) is becoming increasingly important and researchers are designing lightweight networks through methods like network pruning, parallelism, and knowledge distillation. Efficient network design is key to achieving better performance in EISR, and many designs adopt the U-Net architecture. By deconstructing its branching structure, the main structure of the U-Net can be transformed into mutually independent branches, increasing parallelism and reducing the number of layers executed serially in the network to improve inference speed within the GPU's capacity. This paper proposed Parallel RFDN (PRFDN) based on the pre-trained RFDN, which is a typical EISR model with a backbone similar to U-Net. Our method disentangles the sequentially computed trunks in RFDN into branches and performs re-parametrization to make these branches inference in parallel on single devices. After that, we further perform pruning on the model and finetune it to achieve higher performance.

1. Introduction

Single image super-resolution (SISR) is a computer vision technique to reconstruct high-resolution images from low-resolution images. Deep convolution neural networks have shown impressive performance in this area and many deep learning-based methods have been proposed to address this ill-posed problem. This technique has been used in many applications (e.g. video gaming [4] and video streaming [6, 12, 18]) to achieve displays with high resolution and high refresh rates and some of the related applications have been deployed on lots of mobile devices [13, 17]. However, mobile phones have a power-constrained design with a dedicated power supply, which makes efficient image super-resolution (EISR) a growing interest.

Efficient network design is key to achieving better performance in EISR. To make the network more efficient for resource-limited devices, researchers are designing generic lightweight networks with reduced parameters through methods like network pruning [2, 8, 11], parallelism [3, 7, 11, 15] and knowledge distillation [5, 10, 16, 19]. However, when the model size becomes sufficiently small, these methods fail to further increase the frame rate of super-resolution. This is because reducing the computational scale of each layer of the model beyond the point where the GPU is fully loaded does not improve the inference speed of the model, if the length of serially executed layers cannot be reduced. Therefore, lots of EISR designs [1, 9, 13] involve careful consideration of network architecture, design, and utilization of limited features to generate more representative features for super-resolution, which can use fewer layers to achieve a good super-resolution quality.

Moreover, recent years have witnessed the wide use of U-Net [14] in the design of deep neural networks. Many designs of the EISR model also adopted the U-Net architecture, and the RFDN [10] is a representative example, which won the first prize in the AIM 2020 competition. Through the observation of the U-Net structure, we found its main structure can be transformed into mutually independent branches by deconstructing its branching structure, thereby increasing parallelism. Furthermore, multiple branches can be executed in parallel on the same GPU by reparameterization. Within the GPU's capacity, this method can reduce the number of layers that are executed serially in the network, therefore improving the inference speed.

In this paper, we proposed Parallel RFDN (PRFDN) based on the pre-trained RFDN. Our method disentangles

the sequentially computed trunks in RFDN into branches and performs re-parametrization to make these branches inference in parallel on single devices. After that, we further perform pruning on the model and finetune it to achieve higher performance.

1.1. Paper ID

Make sure that the Paper ID from the submission system is visible in the version submitted for review (replacing the "*****" you see in this document). If you are using the LaTeX template, **make sure to update paper ID in the appropriate place in the tex file**.

1.2. Methods

We proposed Parallel RFDN (PRFDN) as is shown in figure 1. Our method consists of four stages to transform a pre-trained RFDN [10] into PRFDN.

1.3. Branching

To accelerate the inference, we first consider reducing the data dependency in the model to achieve higher parallelism. According to our observation, the main structure of U-Net can be transformed into mutually independent branches to increase parallelism. Our method disentangles the sequentially computed trunks of RFDN into branches. As is shown in figure 1b, after the branching, the major part of the model will consist of four independent branches that can calculate in parallel. To improve the accuracy, we also design a small SR block (SRFDB) based on the RFDB of RFDN and add them before the input of each branch. This type of block is utilized to fit the output of cascaded RFDBs, such that the input to the disentangled RFDB block closely resembles the input received in the original RFDN. This ensures model accuracy at a reduced cost.

1.4. Training

Since we only change the data flow but not the structure of RFDB, the pre-trained RFDN parameters can still be loaded into the major part of our branch model (only except for those SRFDBs). To benefit from the pre-training, we load the pre-trained RFDN parameters into our branch model before training our branch model. In this step, we first trained only SRFDBs for 100 epochs (on LSDIR dataset and DIV2K dataset) with freeze parameters in RFDBs, and then trained the whole model for 100 epochs.

1.5. Re-parametrization

Without much data dependency, branches in our model can be computed in parallel. However, a single GPU cannot compute two or more different models in parallel. To address this issue and achieve parallel computing of multiple branches on a single GPU, we should merge the four branches into a single branch. As is shown in figure 1b, three of these four branches have exactly the same structure with only different parameters, and the only difference between the remaining branch and the other branches is the absence of an RFDBS. Therefore, we should 1) merge those four convolution layers at the beginning of the branches into one, 2) merge those three RFDBS into one, and 3) merge those four RFDBs into one.

In our method, we create a bigger RFDB and a bigger SRFDB with bigger convolution layers and move the weight and bias from the RFDBs and SRFDBs into them. More specifically, we merge the convolution layers according to the following derivation.

Assuming $Y = Conv(X, K)$ represents a convolution layer, where $K \in \mathbb{R}^{c_o \times c \times h_k \times w_k}$ is the convolution kernel; $X \in \mathbb{R}^{c \times h \times w}$ is the input of convolution layer and $Y \in \mathbb{R}^{c_o \times h' \times w'}$ is the output; c, h, w are the channel, height, and width of the input respectively; c_o, h', w' are the channel, height, and width of the output respectively; h_k and w_k are the height and width of the kernel respectively. If we want to merge n convolution layers $Y_i = Conv(X_i, K_i)$ into a big convolution layer $Conv_b$, this $Conv_b$ can be represented as:

$$
\begin{aligned}
Y &= Conv_b([X_1, \dots, X_n], K_1, \dots, K_n) \\
&= [Y_1, \dots, Y_n] \\
&= \sum_{i=1}^{n} [\mathbf{0}, \dots, Y_i, \dots, \mathbf{0}] \\
&= \sum_{i=1}^{n} [\mathbf{0}, \dots, Conv(X_i, K_i), \dots, \mathbf{0}] \\
&= Conv([X_1, \dots, X_n], \begin{bmatrix} K_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & K_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & K_n \end{bmatrix}) \\
&= Conv(X_{concat}, K_{diag})
\end{aligned}
\tag{1}
$$

Therefore, when computing the merged convolution layer $Conv_b$, all the input $X_i, i \in [1, n]$ would be concatenated into X_{concat} and all the kernel $K_i, i \in [1, n]$ would be re-parameterized into K_{diag} (in first two dimensions) before performing a normal convolution operation.

Inspired by Torch-Pruning [2], we implemented our re-parametrization algorithm as a convolution constructor function:

```
def merge_conv(convs, c, c_o, k, **kwargs):
  n = len(convs)
  conv = nn.Conv2d(c*n, c_o*n, k, **kwargs)
  for i, j in product(range(n), range(n)):
    if j == i:
      conv.weight[j*c_o:(j+1)*c_o,
                  i*c:(i+1)*c,
                  ...] = \
```

```
      convs[i].weight
  else:
    conv.weight[j*c_o:(j+1)*c_o,
                i*c:(i+1)*c,
                ...] = \
      torch.zeros(convs[i].weight.shape)
for i in range(n):
    conv.bias[i*c_o:(i+1)*c_o] = \
      convs[i].bias
return conv
```

As is shown in figure 1c, we merge and re-parametrize the four convolution layer, three SRFDBs and four RFDBs into a bigger convolution layer, SRFDB and RFDB respectively. After re-parameterization, we get a bigger model that is equivalent to that of the branches of the branch model computed in parallel on a single GPU.

1.6. Pruning

The re-parametrized model is big. To reduce the resource consumption and further accelerate the inference, we applied selective channel-wise auto-pruning (Torch-Pruning [2]) on our re-parametrized model, as is shown in figure 1d.

Specifically, as channel-wise pruning would modify the output channel number of certain layers (especially convolution layers), channel indices of channel-wise split and concatenate operations should be adjusted accordingly. However, determining which channels are pruned during auto-pruning is difficult to implement in code. In our method, we propose to replace all channel-wise split and concatenate operations with equivalent 1x1 convolutions. This allows auto-pruning to directly prune these operations and eliminates the need to adjust their channel indices separately.

After re-parametrization, we used Torch-Pruning to prune the model. In our implementation, we prune out 90% channels from our model in 16 steps, and between each pruning step, we finetune the model by a 10-epoch training.
FAQ

Q: Are acknowledgements OK?
A: No. Leave them for the final copy.

Q: How do I cite my results reported in open challenges?
A: To conform with the double-blind review policy, you can report results of other challenge participants together with your results in your paper. For your results, however, you should not identify yourself and should not mention your participation in the challenge. Instead present your results referring to the method proposed in your paper and draw conclusions based on the experimental comparison to other results.

Method	Frobnability
Theirs	Frumpy
Yours	Frobbly
Ours	Makes one's heart Frob

Table 1. Results. Ours is better.

1.7. References

List and number all bibliographical references in 9-point Times, single-spaced, at the end of your paper. When referenced in the text, enclose the citation number in square brackets, for example [**?**]. Where appropriate, include page numbers and the name(s) of editors of referenced books. When you cite multiple papers at once, please make sure that you cite them in numerical order like this [**?**, **?**, **?**, **?**, **?**]. If you use the template as advised, this will be taken care of automatically.

2. Final copy

You must include your signed IEEE copyright release form when you submit your finished paper. We MUST have this form before your paper can be published in the proceedings.

Please direct any questions to the production editor in charge of these proceedings at the IEEE Computer Society Press: https://www.computer.org/about/contact.

References

[1] Zongcai Du, Ding Liu, Jie Liu, Jie Tang, Gangshan Wu, and Lean Fu. Fast and Memory-Efficient Network Towards Efficient Image Super-Resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 853–862, 2022. 1

[2] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. *The Thirty-Fourth IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 1, 2, 3

[3] 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:1704.04861 [cs]*, Apr. 2017. 1

[4] Tianxing Jin, Songtao He, and Yunxin Liu. Towards accurate gpu power modeling for smartphones. In *Proceedings of the 2nd Workshop on Mobile Gaming*, MobiGames '15, page 7–11, New York, NY, USA, 2015. Association for Computing Machinery. 1

[5] Mehrdad Khani, Pouya Hamadanian, Arash Nasr-Esfahany, and Mohammad Alizadeh. Real-Time Video Inference on Edge Devices via Adaptive Model Streaming. In *2021*

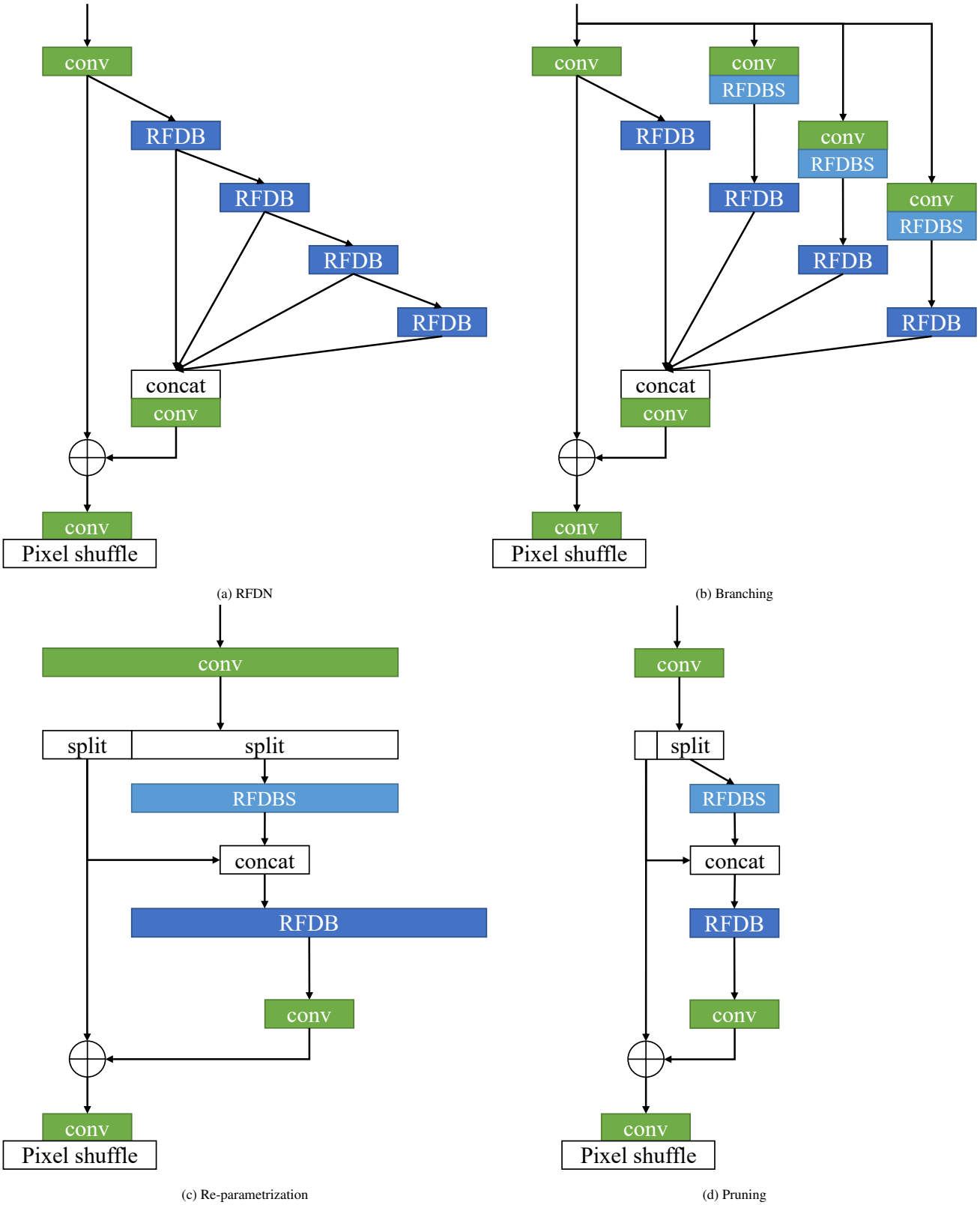

Figure 1. Transform a pre-trained RFDN into PRFDN

IEEE/CVF International Conference on Computer Vision (ICCV), pages 4552–4562, 2021. 1

[6] Jaehong Kim, Youngmok Jung, Hyunho Yeo, Juncheol Ye, and Dongsu Han. Neural-Enhanced Live Streaming: Improving Live Video Ingest via Online Learning. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 107–125, Virtual Event USA, July 2020. ACM. 1

[7] Xiangtao Kong, Hengyuan Zhao, Yu Qiao, and Chao Dong. ClassSR: A General Framework to Accelerate Super-Resolution Networks by Data Characteristic. *arXiv:2103.04039 [cs]*, Mar. 2021. 1

[8] Royson Lee, Stylianos I. Venieris, Lukasz Dudziak, Sourav Bhattacharya, and Nicholas D. Lane. MobiSR: Efficient On-Device Super-Resolution through Heterogeneous Mobile Processors. In *The 25th Annual International Conference on Mobile Computing and Networking*, MobiCom '19, pages 1–16, New York, NY, USA, Oct. 2019. Association for Computing Machinery. 1

[9] Yawei Li, Kai Zhang, Radu Timofte, Luc Van Gool, Fangyuan Kong, Mingxi Li, Songwei Liu, Zongcai Du, Ding Liu, Chenhui Zhou, Jingyi Chen, Qingrui Han, Zheyuan Li, Yingqi Liu, Xiangyu Chen, Haoming Cai, Yu Qiao, Chao Dong, Long Sun, Jinshan Pan, Yi Zhu, Zhikai Zong, Xiaoxiao Liu, Zheng Hui, Tao Yang, Peiran Ren, Xuansong Xie, Xian-Sheng Hua, Yanbo Wang, Xiaozhong Ji, Chuming Lin, Donghao Luo, Ying Tai, Chengjie Wang, Zhizhong Zhang, Yuan Xie, Shen Cheng, Ziwei Luo, Lei Yu, Zhihong Wen, Qi Wul, Youwei Li, Haoqiang Fan, Jian Sun, Shuaicheng Liu, Yuanfei Huang, Meiguang Jin, Hua Huang, Jing Liu, Xinjian Zhang, Yan Wang, Lingshun Long, Gen Li, Yuanfan Zhang, Zuowei Cao, Lei Sun, Panaetov Alexander, Yucong Wang, Minjie Cai, Li Wang, Lu Tian, Zheyuan Wang, Hongbing Ma, Jie Liu, Chao Chen, Yidong Cai, Jie Tang, Gangshan Wu, Weiran Wang, Shirui Huang, Honglei Lu, Huan Liu, Keyan Wang, Jun Chen, Shi Chen, Yuchun Miao, Zimo Huang, Lefei Zhang, Mustafa Ayazoğlu, Wei Xiong, Chengyi Xiong, Fei Wang, Hao Li, Ruimian Wen, Zhijing Yang, Wenbin Zou, Weixin Zheng, Tian Ye, Yuncheng Zhang, Xiangzhen Kong, Aditya Arora, Syed Waqas Zamir, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Dandan Gao, Dengwen Zhou, Qian Ning, Jingzhu Tang, Han Huang, Yufei Wang, Zhangheng Peng, Haobo Li, Wenxue Guan, Shenghua Gong, Xin Li, Jun Liu, Wanjun Wang, Dengwen Zhou, Kun Zeng, Hanjiang Lin, Xinyu Chen, and Jinsheng Fang. NTIRE 2022 Challenge on Efficient Super-Resolution: Methods and Results. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1061–1101, June 2022. 1

[10] Jie Liu, Jie Tang, and Gangshan Wu. Residual feature distillation network for lightweight image super-resolution. In *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 41–55. Springer, 2020. 1, 2

[11] Xin Liu, Yuang Li, Josh Fromm, Yuntao Wang, Ziheng Jiang, Alex Mariakakis, and Shwetak Patel. SplitSR: An End-to-End Approach to Super-Resolution on Mobile Devices. *arXiv:2101.07996 [cs]*, Jan. 2021. 1

[12] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural Adaptive Video Streaming with Pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, pages 197–210, New York, NY, USA, Aug. 2017. Association for Computing Machinery. 1

[13] Sachin Mehta, Amit Kumar, Fitsum Reda, Varun Nasery, Vikram Mulukutla, Rakesh Ranjan, and Vikas Chandra. EVRNet: Efficient Video Restoration on Edge Devices. In *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21, pages 983–992, New York, NY, USA, Oct. 2021. Association for Computing Machinery. 1

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Lecture Notes in Computer Science, pages 234–241, Cham, 2015. Springer International Publishing. 1

[15] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *arXiv:1801.04381 [cs]*, Mar. 2019. 1

[16] Chaofan Tao, Lu Hou, Wei Zhang, Lifeng Shang, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. Compression of Generative Pre-trained Language Models via Quantization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4821–4836, Dublin, Ireland, 2022. Association for Computational Linguistics. 1

[17] Hyunho Yeo, Chan Ju Chong, Youngmok Jung, Juncheol Ye, and Dongsu Han. NEMO: Enabling neural-enhanced video streaming on commodity mobile devices. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–14, London United Kingdom, Sept. 2020. ACM. 1

[18] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. Neural Adaptive Content-aware Internet Video Delivery. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 645–661, 2018. 1

[19] Yiman Zhang, Hanting Chen, Xinghao Chen, Yiping Deng, Chunjing Xu, and Yunhe Wang. Data-Free Knowledge Distillation for Image Super-Resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7852–7861, 2021. 1