Data Overfitting for On-Device Super-Resolution with Dynamic Algorithm and Compiler Co-Design

Gen Li[1], Zhihao Shu[2], Jie Ji[1], Minghai Qin, Fatemeh Afghah[1], Wei Niu[2], and Xiaolong Ma[1]

[1] Clemson University
[2] University of Georgia
{gen,xiaolom}@clemson.edu

Abstract. Deep neural networks (DNNs) are frequently employed in a variety of computer vision applications. Nowadays, an emerging trend in the current video distribution system is to take advantage of DNN's overfitting properties to perform video resolution upscaling. By splitting videos into chunks and applying a super-resolution (SR) model to overfit each chunk, this scheme of SR models plus video chunks is able to replace traditional video transmission to enhance video quality and transmission efficiency. However, many models and chunks are needed to guarantee high performance, which leads to tremendous overhead on model switching and memory footprints at the user end. To resolve such problems, we propose a Dynamic Deep neural network assisted by a Content-Aware data processing pipeline to reduce the model number down to one (Dy-DCA), which helps promote performance while conserving computational resources. Additionally, to achieve real acceleration on the user end, we designed a framework that optimizes dynamic features (e.g., dynamic shapes, sizes, and control flow) in Dy-DCA to enable a series of compilation optimizations, including fused code generation, static execution planning, etc. By employing such techniques, our method achieves better PSNR and real-time performance (33 FPS) on an off-the-shelf mobile phone. Meanwhile, assisted by our compilation optimization, we achieve a 1.7× speedup while saving up to 1.61× memory consumption. Code available in https://github.com/coulsonlee/Dy-DCA-ECCV2024.

Keywords: Super-resolution · Dynamic neural network · Compilation optimizations

1 Introduction

With the rapid advancement of artificial intelligence, Deep Neural Networks (DNNs) have emerged as a cornerstone technology in various computer vision tasks, revolutionizing the field of image processing [15,22,24,34,61]. Among this, Video Super-Resolution (VSR) [4,10,30,48,51] has garnered increasing attention in recent years. Among the various approaches explored in the realm of VSR, a

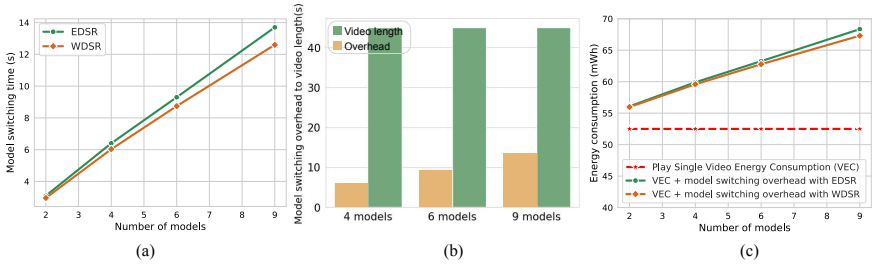

Fig. 1: Model switching overhead on currently widely used backbones in video data overfitting. Figure (a) show the switching time in EDSR [37] and WDSR [63]. Figure (b) demonstrates the comparison of video length and switching overhead. Figure (c) shows the total energy consumption brining by model switching.

rising trend is focused on utilizing Super Resolution (SR) models to upscale the resolution of low-resolution (LR) videos instead of directly transmitting high-resolution (HR) videos [38, 60]. This emerging representative aims to address the challenge of high bandwidth consumption between servers and clients, which often occurs when directly transmitting HR videos.

In the context of video transmission using neural networks, one prevalent approach is the utilization of conventional VSR models [55,56], which are designed to cater to all types of videos. However, to achieve optimal performance, these models typically demand larger parameter sizes, rendering their deployment on mobile devices impractical [23,54,65]. Additionally, there is no assurance that a single model can consistently yield optimal results for all videos. To tackle these challenges, researchers have turned their attention to leveraging the overfitting property of DNNs. Instead of pursuing a one-size-fits-all approach, a novel strategy has emerged to utilize a dedicated model to overfit the whole video [59,60]. To enhance the quality of super-resolved video and reduce model size, raw videos are often split into segments based on time or spatial information [33,35,38], allowing each model to focus on a smaller segment. However, during the HR video recovery process, the frequent loading and unloading of numerous models can lead to significant overhead at the user end [65]. As shown in Figure 1, the overhead of switching model takes a large portion of total video length and brings more than 50% additional energy consumption, which is impossible to achieve real-time performance and system efficiency.

Therefore, it is essential to find a video transmission framework capable of meeting the requirements for SR video quality and the resource demands of prevalent edge devices. Several key points are not fully discussed in previous works. (i) The number of models and corresponding model size should be minimized under certain SR video quality requirements. (ii) A corresponding algorithm-compiler-hardware optimization framework that can ensure a real-time system and reasonable on-device resource usage. To fulfill the above expectations, this paper proposes *Dy-DCA*, as shown in figure 2, which consists of a dynamic deep neural network and a fine-grained data preprocessing methodology

to achieve better performance while minimizing the model switching overhead on the hardware side. Meanwhile, a compiler-level optimization framework for accelerating dynamic DNNs is designed to achieve real-time inference and save memory consumption.

(i) **To minimize models while maintaining high PSNR.** The prior art [33] splits video frames into evenly small patches and regroups them into different chunks according to texture complexity. Although this helps reduce the required model number and increases PSNR, these chunks & model pairs may still lead to I/O and model switching overhead at the user end [6, 65]. Thus, in *Dy-DCA*, we propose a fine-grained data processing method that splits the frames into patches with uneven sizes (e.g., use large patches for monotonous backgrounds and smaller patches for detailed foregrounds), then overfits these data with a designed dynamic neural network, bringing the total transmitted model down to one. The uneven splitting minimizes the total number of patches, thereby reducing both server training effort and user-end I/O overhead. The dynamic neural network has a dynamic routing node and itself follows a tree structure to handle patches of different texture complexity.

(ii) **To ensure real-time performance on device.** Although dynamic DNN resolves the system overhead caused by model switching, the dynamic input shapes, and control flow in the model poses many challenges for the compiler-hardware-level optimizations (e.g., loop fusion [71], execution order planning [2], etc.). Due to very conservative assumptions and/or expensive analyses at runtime, current approaches [1, 26] face difficulties in achieving practical on-device efficiency. In this paper, we *finish the last piece of our design* for the system by proposing a nuanced approach that optimizes DNN dynamic features, which allows us to *close the loop* (algorithm, software, hardware) and achieve on-device intelligence. The foundation of our approach is an in-depth study of operators that form the basis for modern DNNs. These operators are classified into several groups on the basis of how the output shapes relate to the input shapes and values. Under this classification, we introduce a data-flow analysis framework dedicated to inferring the shapes and dimensions of intermediate tensors. Subsequently, the outcomes of the analysis are used for enabling a number of compilation optimizations, which include operator fusion and fused code generation, static execution planning, and runtime memory allocation.

We summarize our contributions as follows:

- With the proposed context-aware data pipeline paired with dynamic DNN, patches with appropriate content and dimensions are directed to the suitable processing path, thus enhancing PSNR while reducing model shifting overhead.
- Given the existence of dynamic features, we introduce a data-flow analysis framework based on the classified DNN operators, which enables us to infer the shapes and dimensions of intermediate tensors. This design helps reduce the inference latency at the edge while reducing memory consumption.
- Based on the data-flow analysis framework, we implement a series of compiler-level optimizations (e.g. fused code generation, static execution planning,

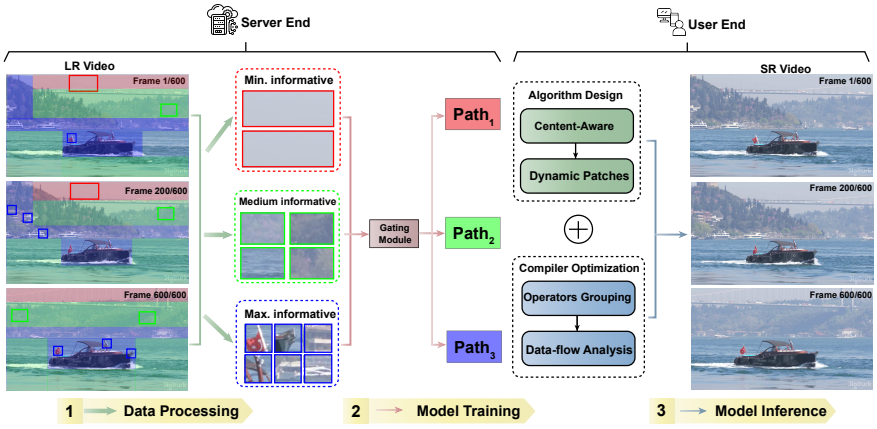

Fig. 2: Overview of the proposed framework *Dy-DCA*. We split video frames into different shapes, and all patches will be distributed at a learnable gating module, then overfitted by a dynamic SR model. The dynamic SR model and LR patches will be delivered to users for video super-resolution. The on-device inference is accelerated by our designed compiler optimization framework.

etc.) to solve the challenges brought by dynamic shape and routing, achieving 1.7× overall speedup across various dynamic features.

2 Algorithm and hardware co-design

2.1 Motivations

The main promotions of previous arts concentrated on elevating the PSNR and expediting the training process [33, 35, 38, 60]. An important aspect that hasn't received adequate attention is whether these improvements can actually be implemented on the devices used by end-users while maintaining the required quality. These methods often require multiple models for high performance and are sensitive to model number [35, 38, 60]. Also, based on the result in Figure 1, numerous models will bring tremendous overhead, which may have a severe impact on the user end. Thus, our goal is to minimize the total number of models while maintaining good performance.

We investigate the patterns of these methods and find that there are two video segmentation methods used in their framework. [60] first operates on the temporal axis (e.g., every 5 seconds of content as a segment). As there may be consistent and repeated content in the whole video, different models may learn the same content at different segments, which is not effective. Also, a single frame covers information with large differences in texture complexity (e.g., foreground and background) [3, 16, 31, 53, 64], potentially making overfitting more difficult.

[33] splits video frames into evenly small patches (around 50×50) and regroups them into different chunks according to texture complexity. Although

this method helps reduce the number of models, the small patches may carry similar information, which introduces meaningless computation and I/O overheads when training or inferring them. In order to reduce the need for multiple models, a better segmentation pattern and model structure need to be proposed.

2.2 Algorithm level optimization for hardware friendliness

To address the model switching overhead mentioned above, in this section, we propose a scalable dynamic deep neural network paired with a fine-grained data processing method that reduces the number of models down to one while maintaining good performance and a reasonable model size.

As shown in Figure 2, for an input video, we first split those frames into different shape patches. These patches contain various levels of texture complexity. (e.g., the minimum informative patches are concentrated in the background section, while the maximum ones are on the foreground object.) To achieve this, we propose a coarse- to fine-grained data processing pipeline to dynamically produce patches of different texture complexity. In our framework, frames will first be split into large patches and evaluated by a general SR model to get the corresponding PSNR value for each patch. Guided by the PSNR values, we can roughly determine the texture complexity of each patch [16,33,35,53,64]. Specifically, for the patches where the PSNR value is greater than a certain threshold, as they contain less information, we will not further split those patches. For those lower ones, we will split them into smaller patches and follow the previous evaluation & split step. This forms an iterative processing pipeline to provide patches with different shapes and texture complexity. In this manner, patches with similar complexity features will almost all have the same shape. In *Dy-DCA*, each execution path is able to learn a similar distribution, which boosts the super-resolution performance. Also, the total number of patches in a video drops a lot compared to the method [33], thus reducing the I/O pressure on the user end.

To overfit the above patches with different shapes, we propose a dynamic deep neural network with different paths and a routing node. Specifically, with multiple paths for various resolutions and a learnable routing node to distribute input patches to corresponding paths, our proposed *Dy-DCA* resolves the system overhead caused by model switching. Meanwhile, taking advantage of modular SR neural network design [37, 63, 67], we delicately design the structure of each path to maintain better performance and a reasonable model size for the resource-constrained devices.

2.3 Compiler level optimization to better support algorithm

In Section 2.2, we utilize a dynamic neural network to resolve the system overhead caused by model switching. However, our approach introduces dynamic input in the form of differently shaped patches, which are subsequently distributed across various paths, a process known as routing. This introduces uncertainty into model execution, impeding the compiler's ability to achieve high efficiency.

Thus, a compiler-level optimization method should be proposed to resolve the challenges brought by dynamic input and routing.

DNN Operator Classification. In dynamic DNNs, each tensor can be categorized into an input tensor (I) and an output tensor (O) at operator (T^l), where l is the operator index. We denote the shape of an input tensor as $I(S)$ and its corresponding value as $I(V)$. Similarly, for an output tensor, we have $O(S)$ and $O(V)$ to represent shape and value, respectively.

Our key observation is that dynamism (routing, input) brings uncertainty to optimization. For instance, loop fusion [71] relies on knowledge of the identical index space between two loops, typically corresponding to the dimensions of respective input tensors. Likewise, planning execution order [2] to minimize memory usage or organizing memory allocation [44] is hindered in the absence of static knowledge regarding tensor sizes.

Thus, in order to help predict intermediate tensor shape and value, we first group DNN operators into four types: Input Shape Determined Output ($I(S) \rightarrow O(S,V)$), Input Shape Determined Output Shape ($I(S) \rightarrow O(S)$), Input Shape & Value Determined Output Shape ($I(S,V) \rightarrow O(S)$), and Execution Determined Output ($Exec \rightarrow O(S,V)$).

(i) Input Shape Determined Output: The output tensor shapes are determined by the input tensor shape, but the input values do not impact the output. The representative operators include `Shape` and `Eyelike`.

(ii) Input Shape Determined Output Shape: The output shapes are dependent on the input shapes, much like in the preceding category. The output values, however, depend on all the input values. Examples include `Conv`, `Add`, and `Pooling`. The significance of this category, as compared to the next set of categories, is that if the input shape of this operator is known, compiler optimizations (e.g., operator fusion, execution/memory optimizations) are enabled.

(iii) Input Shape & Value Determined Output Shape: The output values are dependent on the input shapes and all the input values, just like the previous category. The distinction is that only a portion of the input data is used by the output shapes. Examples include `Extend` and `Range`.

(iv) Execution Determined Output: Similar to the previous two categories, the output values rely on the input shapes and all the input values. Examples include `Nonzero` and `If`.

Data-flow analysis. In this section, we present a novel data-flow analysis methodology to infer the intermediate result tensor shape based on the DNN operator classification discussed above. Such an analysis enables a number of optimizations, including dynamic DNN operator fusion, execution path planning, etc.

Our main finding is that, for many operators and operator combinations (such as an input shape determined output operator and an input shape determined output shape operator), it is possible to infer the shape of the intermediate result tensor to some extent even without knowing the input tensor shape.

Table 1: DNN Operators Classification. These operators are from ONNX (Open Neural Network Exchange) [43].

Operator Class	Example of Operators	Representatives
I(S) → O(S, V)	Shape, ConstantOfShape, Eyelike	Shape
I(S) → O(S)	Add, AveragePool, Cast, Concat, Conv, Elementwise w/ broadcast, Gather, MatMul, MaxPool, Reduce, Relu, Round, Sigmoid, Softmax	Conv, MatMul
I(S,V) → O(S)	Expand, GroupNormalization, MaxUnpool, Onehot, Range, Reshape, Resize, Slice, TopK, Upsample	Reshape, Range
Exec → O(S, V)	If, Loop, NMS, Nonzero, <Switch, Combine>	If, Loop

Data-flow domain: To infer the shape and value of the intermediate tensor, we first define the value domain of tensor shape (S) and tensor value (V) for subsequent optimization algorithms. As shown in Figure 3, the lattice includes known constants, symbolic constants, and operator-inferred constant. The lattice also includes undefined and not-a-constant (nac) at the top and bottom, respectively.

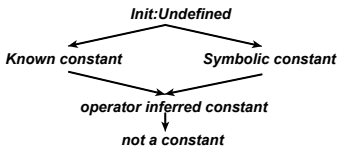

Fig. 3: The lattice of the data-flow domain.

Transfer function: The transfer function is used to transfer the Shape and Value from the input tensor to the output tensor based on the operator type. In our proposed dynamic neural network, there are two kinds of transfer functions: `Update` and `Merge`. `Update` transfers from the input tensor to the output tensor for an individual operator. `Merge` operates on branch control flow and merges (output) tensors from multiple possible execution paths.

Algorithm for data-flow analysis: The computation graph (G) of a dynamic neural network can be viewed as a Directed Acyclic Graph (DGA). ① We first sort the operators in G in depth-first order and initialize the output shape and value map of each operator as undefined. ② If the operator is a control-flow node (like Combine or Switch), it needs to call the Merge function to merge analysis results from multiple control-flow paths. ③ For non-control-flow operators, we send them to forward transfer functions. These functions are based on dynamism classification of DNN operators in Table 1. For example, if the operator in the case "Input Shape Determined Output Shape" like Conv and Add, the function will return the predicted shape. ④ Then, for the case of "Input Shape Determined Output", if the value domain of operator shape is not undefined or nac, we then set a symbolic value to the output tensor value to facilitate subsequent analysis. The algorithm will iteratively process ②-④ until there are no updates on the shape and value of the output tensor of an operator.

Operator fusion. A common challenge encountered in dynamic DNNs is the absence of knowledge regarding the tensor shapes of two operators. In such

cases, the DNN compiler faces limitations, as it is either unable to perform fusion between these operators or has to generate a multitude of code versions, each accommodating a possible combination of shapes for the two operators. In reality, when we are dealing with the merging of more than two operators, the need to generate separate code for all the potential combinations becomes quite extensive.

This problem can be resolved by our suggested data-flow analysis, which makes use of (potentially symbolic) shape information. Fusion can be enabled and/or made simpler by information like the fact that the two operators have tensors of the same shape, even if the precise dimensions are not known until runtime.

Static execution planning. The computational graph of dynamic DNN supports a variety of orderings for the execution of operators. The ordering decision affects the peak memory usage, which further affects the effectiveness of the cache and the execution latency. However, it has been proven that finding an optimal path in the computational graph is a NP-hard problem [2]. Therefore, it is hard to find an optimal plan for modern large DNNs with more than hundreds of operators.

In our data-flow analysis, the general assumption is that a method based on graph partitioning is appropriate because a globally optimal solution is impractical. It turns out that the analysis results can direct the segmentation of the graph, as well as the choice of solution within each sub-graph. The main reason lies in the value domain (undefined, known constant, symbolic constants, operator-inferred constants, and not a constant) we defined before, which facilitates the generation of an optimal execution plan. For example, in a sub-graph G_sub, if all tensors in G_sub are known, the optimal execution plan for G_sub can be obtained statistically by an exhaustive search. If there exist known constant, symbolic constants, and operator-inferred constants, it is still possible to generate a close optimal execution under certain requirements like memory usage and latency limitation. For those operators that have a not constant output tensor shape, it provides an opportunity to partition the original graph into sub-graphs that can be independently analyzed.

3 Experimental results

In this section, we evaluate Dy-DCA on multiple videos to show its superiority. Also, to demonstrate the effectiveness of our proposed algorithm and compiler co-design approach, we deploy Dy-DCA on an off-the-shelf mobile phone. The detailed information of dataset and implementation can be found in Section 3.1. We compare our method with current multimodel methods [33, 35, 60], and the main results are shown in Section 3.2. In Section 3.3, we deploy our model on edge and achieve real-time inference speed, which shows the advantages of our proposed compiler optimization. In Section 3.4, we compare our method with SRVC [27], both of which employ a single model to accommodate all videos.

Table 2: Comparison results of Dy-DCA with different data overfitting methods.

Model	Data Scale	UVG-Beauty ×2	UVG-Beauty ×3	UVG-Beauty ×4	UVG-Bosphorus ×2	UVG-Bosphorus ×3	UVG-Bosphorus ×4	UVG-HoneyBee ×2	UVG-HoneyBee ×3	UVG-HoneyBee ×4
EDSR	NAS 60	45.74	42.71	40.52	44.99	41.47	38.04	44.54	41.51	40.23
	EMT 35	46.84	43.79	41.80	46.12	42.33	39.14	45.21	42.29	39.38
	STDO 33	47.02	44.12	42.03	46.64	42.72	39.71	45.82	42.74	42.15
	Ours	**47.21**	**44.42**	**42.35**	**46.87**	**43.06**	**40.25**	**46.02**	**43.24**	**42.74**
WDSR	NAS 60	46.23	43.24	41.10	45.45	42.01	38.92	44.89	41.97	41.08
	EMT 35	47.04	44.17	42.40	46.42	42.86	39.92	45.87	43.01	42.12
	STDO 33	47.50	44.73	42.88	46.98	43.06	40.22	46.10	43.29	42.74
	Ours	**47.64**	**44.85**	**43.10**	**47.10**	**43.31**	**40.29**	**46.33**	**43.52**	**43.02**
		game-45s ×2	**game-45s** ×3	**game-45s** ×4	**inter-45s** ×2	**inter-45s** ×3	**inter-45s** ×4	**vlog-45s** ×2	**vlog-45s** ×3	**vlog-45s** ×4
EDSR	NAS 60	43.22	36.72	34.32	43.31	35.80	32.67	48.48	44.12	42.12
	EMT 35	44.04	37.89	35.27	43.89	36.21	33.07	48.86	44.71	42.80
	STDO33	45.65	39.93	37.24	44.52	38.28	35.51	49.84	45.47	43.07
	Ours	**45.72**	**40.17**	**37.60**	**44.74**	**38.53**	**35.77**	**49.89**	**45.61**	**43.19**
WDSR	NAS 60	43.70	37.25	34.93	43.41	36.05	33.11	48.52	44.75	42.80
	EMT 60	44.47	38.14	35.72	43.92	36.73	33.47	49.07	44.72	42.87
	STDO33	45.71	40.33	37.76	44.54	38.72	36.03	49.76	45.95	43.99
	Ours	**45.86**	**40.51**	**37.98**	**44.76**	**39.02**	**36.44**	**49.84**	**46.13**	**44.12**

3.1 Experiment settings

To evaluate the overfitting performance of our framework, we we adopt the UVG [40] and VSD4K [38] datasets. UVG dataset contains 16 test video sequences and each video has multiple resolutions and bitrates to choose from. In the VSD4K video dataset, there are 6 video categories, and each of the categories contains various video lengths. We set the resolution for HR videos to 1080p and the bitrate at 10bit, and LR videos are generated by bi-cubic interpolation to match different scaling factors.

We utilize EDSR [37], and WDSR [63] as our backbone. Both of them use a modular design, which provides flexibility in our design of dynamic neural networks. During training, we split frames into different sizes of patches. The PSNR thresholds we set for splitting are 40 and 30. Specifically, patches with a PSNR greater than 40 will not be split. The next round larger than 30 will not be split. [41] Regarding the hyperparameter configuration of training the SR models, we follow the setting of [37,38,63]. We adopt the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.009$, $\epsilon = 10^{-8}$ and we use L1 loss as a loss function.

3.2 Evaluation on VSD4K and UVG datasets

In this part, we sample three video categories from VSD4K and UVG and tested them on 45-second videos (VSD4K) and 20-second videos (UVG), respectively. Our results are shown in Table 2. The visual results are shown in Figure 4. We compare with the state-of-the-art DNN-based SR video overfitting methods, such as NAS [60] that splits a video into multiple chunks in advance and overfit each of each chunk with independent SR model, EMT [35] utilizes same training pattern

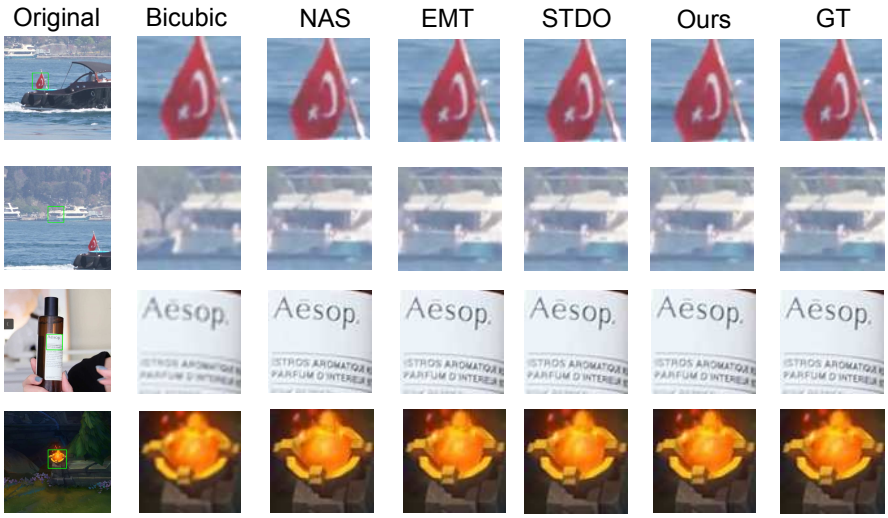

Fig. 4: Super-resolution quality comparison on Dy-DCA and baseline methods.

while taking advantage of meta learning to accelerate training, and STDO [33] that uses texture information to a divided video chunk and overfits with multiple models. As we can see, we achieve a higher PSNR across these different videos.

3.3 Deployment on Mobile Devices

The on-device evaluations are conducted using a OnePlus 11 mobile phone, equipped with a Snapdragon 8 Gen 2 processor [46]. This processor is built on one Cortex-X3 based prime core, four Cortex A-715 and A-710 based performance core, three Cortex A-510 based efficient core, and paired with a Qualcomm Adreno 740 GPU, delivering superior performance while maintaining efficient power consumption. We tested our model on Alibaba MNN [25] with standard configuration, which utilizes 4 threads and employs FP16 precision to optimize computational efficiency. The inference process is executed 20 times, with the first 5 iterations serving as a warm-up phase to ensure system stability and to negate the effects of start-up transients on the experimental outcomes. The results are then averaged to account for any variations, and measured the performance under the designated conditions. The main points of our assessment are to methodically analyze model shift overhead, run-time memory usage, power consumption, and image loading overhead.

Memory and latency result. In Table 3, the memory consumption and end-to-end execution latency are tested by MNN. As *Dy-DCA* has different input patches shape and these patches will be sent to different paths, we randomly chose 50 frames to inference then averaged them. Since NAS, EMT and STDO utilize static DNN, they do not support dynamic input shapes and routing,

thus we do not show the results of these methods on our proposed optimization framework. Overall, with our designed compiler optimization framework, we achieve 1.7× speedup and 1.61× memory reduction compared with MNN. The average inference speed on mobile GPU is 30ms, which achieves real-time requirement [66].

Table 3: Memory consumption and end-to-end execution latency comparison among MNN and our proposed data-flow analysis framework. S stands for the number of parameter of a 16-4 WDSR [63] model. We test on the 45 seconds video. The "/" format is mobile CPU/GPU.

Method	Model	MNN(MB)		Ours(MB)		MNN(ms)		Ours(ms)	
	Size	Min	Max	Min	Max	Min	Max	Min	Max
NAS[60]	9S	79/415	79/415	-	-	457/271	476/275	-	-
EMT [35]	9S	79/415	79/415	-	-	457/271	476/275	-	-
STDO [33]	4S	72/175	72/175	-	-	334/158	342/162	-	-
Ours	S	66/170	69/184	47/103	53/116	66/48	82/54	52/28	59/32
Our's framework speedup		1.45× - 1.61×		1		1.30× - 1.70×		1	

Overhead analysis. In Table 4, As our proposed *Dy-DCA* reduces the model down to one and provides a fine-grained video frame splitting module, we analyze the overhead in terms of model switching and I/O. As STDO also splits video frames into patches, compared with it, we achieve 4× saving on model switching overhead and 7× saving on I/O.

Table 4: The comparison of model switching and I/O overhead between STDO and our method.

Method	Model Number	switching Overhead	Patches Number	Average I/O Overhead	PSNR
STDO [33]	4	4×	1.2×10e5	6×	47.50
Ours	1	1×	2.16×10e4	1×	47.64

3.4 More discussion on one-size-fits-all method

The method named SRVC proposed by [27] can adaptively fit different content with incremental model changes in a single model, which mitigates the model switching overhead. However, when the video content becomes complicated, the super-resolution quality drops dramatically. In the following Table 5, the inter-45s and game-45s have richer information. Compared with our method, the performance of SRVC drops a lot when using WDSR as the backbone with a scaling

factor of ×2. Especially for gmae-45s, a 31.03 dB is not acceptable for a high-quality video requirement. Furthermore, our method is based on algorithm and compiler co-design, which is more favorable to practical usage. The operators and data flow can be fully supported by mobile devices.

Table 5: The comparison of SRVC and our method on different types of videos.

Method	**vlog-45s**	**inter-45s**	**game-45s**	**Average**
SRVC [27]	48.71	39.26	31.03	39.68
Dy-DCA (Ours)	49.84	44.76	45.86	46.82

4 Ablation Study

Different number of paths. We evaluate different number of paths in the network structure of Dy-DCA. We compare the PSNR and the latency tested on mobile GPU for different dynamic paths of Dy-DCA. As shown in the following Table 6, more paths bring higher PSNR by introducing fine-grained data patch groups with more trainable parameters, which sacrifices latency at the user end.

Table 6: The PSNR and inference speed (Mobile GPU) on different number of paths.

Method	Path	PSNR	Ours(ms)	
(Backbone)	Number	Value	Min	Max
Dy-DCA(WDSR)	2	45.86	28	32
Dy-DCA(WDSR)	3	46.33	33	37
Dy-DCA(WDSR)	4	46.51	37	42

Long video. We also test different video lengths to show our design is capable of handling longer and more complex video contents. We select a 2-minute game video in VSD4K [38] and combine the game-45s video and the vlog-45s video together into a 90s-long video (combine-90s). We conduct experiments using WDSR as a backbone with a scaling factor of 4. These two videos are longer and more complex when compared with the videos we show in Table 2 in our paper. As we can see in the following Table 7, the 2-min game video is still maintaining an acceptable PSNR, and the PSNR of combine-90s is close to the average value of overfitting two videos (game-45s and vlog-45s). The results show that our proposed framework can work well even in longer videos with complex contents.

Table 7: The PSNR results on different video lengths.

video	game-45s	vlog-45s	game-2min	combine-90s
PSNR	45.86	49.84	44.72	47.34

5 Related works

5.1 DNN-based image and video super-resolution

For Single Image Super-Resolution (SISR) tasks, SRCNN [13] is the pioneer of applying DNN to image super-resolution. Then, followed by FSRCNN [14] and ESPCN [49], both of them make progress in efficiency and performance. After this, with the development of deep neural networks, more and more network backbones are used for SISR tasks. For example, VDSR [29] uses the VGG [50] network as the backbone and adds residual learning to further improve the effectiveness. Similarly, [32, 37, 63] proposed a SR network using ResNet [22] as a backbone. With the emergence of channel attention mechanisms networks represented by SENet [45], various applications of attention mechanisms poured into the area of image super resolution [11, 42, 67, 68]. Observing the remarkable performance of the transformer architecture in computer vision, as exemplified by [15] in their work on image processing, an increasing number of researchers are now employing various vision transformer models for image super-resolution tasks. [7, 36, 39]. The Video Super-Resolution (VSR) methods mainly drive from SISR. Some of the works described above that were primarily created for SISR, including EDSR [37] and WDSR [63], all have results on VSR. Several of the recent VSR works perform alignment to calculate optical flow by DNNs in order to estimate the motions between images [4, 30, 48, 51]. However, accurate optical flow may not be easy to compute for videos with occlusion and large motions. Another method to perform alignment is called deformable convolution methods, which is first proposed by [10]. The Deformable convolution (DConv) [10] was first used to deal with geometric transformation in vision tasks because the sampling operation of CNN is fixed. TDAN [52] applies DConv to align the input frames at the feature level, which avoids the two-stage process used in previous optical flow based methods. Other works like DNLN [55] and D3Dnet [62] also apply Dconv in their model to achieve a better alignment performance. However, these models incorporate with DConv may suffer from high computation complexity and difficulty for convergence. To increase the robustness of alignment and account for the visual informativeness of each frame, EDVR [56] uses their proposed Pyramid, Cascading and Deformable convolutions (PCD) alignment module and the temporal-spatial attention (TSA) fusion module.

5.2 Development of Dynamic DNN

The development of dynamic DNNs is extensive, including recurrent neural networks (RNNs) and their derivatives, along with instance-specific dynamic mod-

els, spatial-oriented dynamic networks, and time-oriented dynamic models [21]. Our motivation for utilizing DNNs in super-resolution stems from their adaptability and the comprehensive range of dynamic capabilities they offer, allowing for enhanced performance in reconstructing high-resolution details from low-resolution inputs. This survey [21] highlights numerous studies pertinent to dynamic inference, illustrating the establishment of model aggregations through either cascaded or concurrent configurations and the selective activation of models based on input conditions. Additionally, this advancement underscores the significance of deploying Spiking Neural Networks (SNNs), which conduct data-driven inference through the propagation of pulse signals [18]. This review also underscores several pivotal publications in these domains, such as Dynamic Network Surgery [20], Spatially Adaptive Computation Time (SACT) [17], Dynamic Conditional Networks (DCNs) [69], Dynamic Filter Networks [70], Dynamic Convolutional Neural Networks (DCNNs) [9], Dynamic Routing Between Capsules [47], Dynamic Skip Connections (DSCs) [19], and Dynamic Time Warping Networks (DTWNs) [5]. The discussion concludes by reflecting on the unresolved challenges in this field and suggesting intriguing paths for future research. These include the development of theories for dynamic networks, the crafting of efficient decision-making, and diversified application exploration in various disciplines.

5.3 Content-Aware DNN

It is not possible to develop a DNN model that can efficiently handle all web video. To ensure reliability and performance, [60] suggests that the video delivery system take into account employing DNN models to overfit each video chunk. Several livestreaming and video streaming applications [8,12,28,57,58] make use of overfitting property to guarantee great client performance. [28] proposes a live video ingest framework, which adds an online learning module to the original NAS [60] framework to further ensure quality. NEMO [58] selects key frames to apply super-resolution. This greatly reduces the amount of computation on the client sides. CaFM [38] splits a long video into several time-based chunks and design a handcrafted layer along with a joint training technique to reduce the number of SR models and improve performance. EMT [35] proposes to leverage meta-tuning and challenge patches sampling technique to further reduce model size and computation cost. STDO [33] takes spatial information as well as temporal information into account to further enhance model performance.

6 Conclusion

In this paper, we introduce a content-aware dynamic DNN to overfit videos. This design reduces the required model number down to one, thus reducing the model switching overhead at the user end. In order to resolve the challenges brought By using dynamic input patches and routing in dynamic DNN, we propose a data-flow analysis framework to predict the shape and value of intermediate tensor. Subsequently, the outcomes of the analysis are used to enable a number of compilation optimizations, which achieve real-time performance on the edge.

Acknowledgments

This work is partly supported by the National Science Foundation CCF-2312616, CCF-2427875, CNS-2232048 and National Aeronautics and Space Administration (NASA) 80NSSC23K1393. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF and NASA.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: {TensorFlow}: a system for {Large-Scale} machine learning. In: 12th USENIX symposium on operating systems design and implementation (OSDI 16). pp. 265–283 (2016)
2. Ahn, B.H., Lee, J., Lin, J.M., Cheng, H.P., Hou, J., Esmaeilzadeh, H.: Ordering chaos: Memory-aware scheduling of irregularly wired neural networks for edge devices. Proceedings of Machine Learning and Systems **2**, 44–57 (2020)
3. Bengio, Y., LeCun, Y., et al.: Scaling learning algorithms towards ai. Large-scale kernel machines **34**(5), 1–41 (2007)
4. Caballero, J., Ledig, C., Aitken, A., Acosta, A., Totz, J., Wang, Z., Shi, W.: Real-time video super-resolution with spatio-temporal networks and motion compensation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4778–4787 (2017)
5. Cai, X., Xu, T., Yi, J., Huang, J., Rajasekaran, S.: Dtwnet: a dynamic time warping network. In: Advances in Neural Information Processing Systems 32 (NeurIPS 2019). NeurIPS Foundation (2019)
6. Chan, K.C., Zhou, S., Xu, X., Loy, C.C.: Investigating tradeoffs in real-world video super-resolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5962–5971 (2022)
7. Chen, H., Wang, Y., Guo, T., Xu, C., Deng, Y., Liu, Z., Ma, S., Xu, C., Xu, C., Gao, W.: Pre-trained image processing transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12299–12310 (2021)
8. Chen, J., Hu, M., Luo, Z., Wang, Z., Wu, D.: Sr360: boosting 360-degree video streaming with super-resolution. In: Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video. pp. 1–6 (2020)
9. Chen, Y., Dai, X., Liu, M., Chen, D.D., Yuan, L., Liu, Z.: Dynamic convolution: Attention over convolution kernels. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE/CVF (2020)
10. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: Proceedings of the IEEE international conference on computer vision. pp. 764–773 (2017)
11. Dai, T., Cai, J., Zhang, Y., Xia, S.T., Zhang, L.: Second-order attention network for single image super-resolution. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11065–11074 (2019)
12. Dasari, M., Bhattacharya, A., Vargas, S., Sahu, P., Balasubramanian, A., Das, S.R.: Streaming 360-degree videos using super-resolution. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications. pp. 1977–1986. IEEE (2020)

13. Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. IEEE transactions on pattern analysis and machine intelligence **38**(2), 295–307 (2015)
14. Dong, C., Loy, C.C., Tang, X.: Accelerating the super-resolution convolutional neural network. In: European conference on computer vision. pp. 391–407. Springer (2016)
15. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
16. Fan, Y., Tian, F., Qin, T., Bian, J., Liu, T.Y.: Learning what data to learn. arXiv preprint arXiv:1702.08635 (2017)
17. Figurnov, M., Collins, M.D., Zhu, Y., Zhang, L., Huang, J., Vetrov, D., Salakhutdinov, R.: Spatially adaptive computation time for residual networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017), https://arxiv.org/abs/1612.02297
18. Ghosh-Dastidar, S., Adeli, H.: Spiking neural networks (2009), https://doi.org/10.1142/S0129065709002002
19. Gui, T., Zhang, Q., Huang, X., et al.: Long short-term memory with dynamic skip connections. In: Proceedings of the AAAI Conference on Artificial Intelligence. AAAI (2018). https://doi.org/10.1609/aaai.v33i01.33016481
20. Guo, Y., Yao, A., Chen, Y.: Dynamic network surgery for efficient dnns (2016)
21. Han, Y., Huang, G., Song, S., Yang, L., Wang, H., Wang, Y.: Dynamic neural networks: A survey. IEEE (2022)
22. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
23. Hong, C., Kim, H., Baik, S., Oh, J., Lee, K.M.: Daq: Channel-wise distribution-aware quantization for deep image super-resolution networks. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2675–2684 (2022)
24. Ji, J., Li, G., Yin, L., Qin, M., Yuan, G., Guo, L., Liu, S., Ma, X.: Advancing dynamic sparse training by exploring optimization opportunities. In: Forty-first International Conference on Machine Learning (2024), https://openreview.net/forum?id=szRHR9XGrY
25. Jiang, X., Wang, H., Chen, Y., Wu, Z., Wang, L., Zou, B., Yang, Y., Cui, Z., Cai, Y., Yu, T., Lv, C., Wu, Z.: Mnn: A universal and efficient inference engine. In: MLSys (2020)
26. Jiang, X., Wang, H., Chen, Y., Wu, Z., Wang, L., Zou, B., Yang, Y., Cui, Z., Cai, Y., Yu, T., et al.: Mnn: A universal and efficient inference engine. Proceedings of Machine Learning and Systems **2**, 1–13 (2020)
27. Khani, M., Sivaraman, V., Alizadeh, M.: Efficient video compression via content-adaptive super-resolution. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4521–4530 (2021)
28. Kim, J., Jung, Y., Yeo, H., Ye, J., Han, D.: 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. pp. 107–125 (2020)

29. Kim, J., Lee, J.K., Lee, K.M.: Accurate image super-resolution using very deep convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1646–1654 (2016)
30. Kim, T.H., Sajjadi, M.S., Hirsch, M., Scholkopf, B.: Spatio-temporal transformer network for video restoration. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 106–122 (2018)
31. Kumar, M., Packer, B., Koller, D.: Self-paced learning for latent variable models. Advances in neural information processing systems **23** (2010)
32. Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al.: Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4681–4690 (2017)
33. Li, G., Ji, J., Qin, M., Niu, W., Ren, B., Afghah, F., Guo, L., Ma, X.: Towards high-quality and efficient video super-resolution via spatial-temporal data overfitting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10259–10269 (June 2023)
34. Li, G., Yin, L., Ji, J., Niu, W., Qin, M., Ren, B., Guo, L., Liu, S., Ma, X.: Neurrev: Train better sparse neural network practically via neuron revitalization. In: The Twelfth International Conference on Learning Representations (2024), `https://openreview.net/forum?id=60lNoatp7u`
35. Li, X., Liu, J., Wang, S., Lyu, C., Lu, M., Chen, Y., Yao, A., Guo, Y., Zhang, S.: Efficient meta-tuning for content-aware neural video delivery. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVIII. pp. 308–324. Springer (2022)
36. Liang, J., Cao, J., Sun, G., Zhang, K., Van Gool, L., Timofte, R.: Swinir: Image restoration using swin transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1833–1844 (2021)
37. Lim, B., Son, S., Kim, H., Nah, S., Mu Lee, K.: Enhanced deep residual networks for single image super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. pp. 136–144 (2017)
38. Liu, J., Lu, M., Chen, K., Li, X., Wang, S., Wang, Z., Wu, E., Chen, Y., Zhang, C., Wu, M.: Overfitting the data: Compact neural video delivery via content-aware feature modulation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4631–4640 (2021)
39. Mei, Y., Fan, Y., Zhou, Y.: Image super-resolution with non-local sparse attention. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3517–3526 (2021)
40. Mercat, A., Viitanen, M., Vanne, J.: Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In: Proceedings of the 11th ACM Multimedia Systems Conference. pp. 297–302 (2020)
41. Nasution, A., Efendi, S., Suwilo, S.: Image steganography in securing sound file using arithmetic coding algorithm, triple data encryption standard (3des) and modified least significant bit (mlsb). In: Journal of Physics: Conference Series. vol. 1007, p. 012010. IOP Publishing (2018)
42. Niu, B., Wen, W., Ren, W., Zhang, X., Yang, L., Wang, S., Zhang, K., Cao, X., Shen, H.: Single image super-resolution via a holistic attention network. In: European conference on computer vision. pp. 191–207. Springer (2020)
43. ONNX: Open neural network exchange. Https://www.onnx.ai
44. Pisarchyk, Y., Lee, J.: Efficient memory management for deep neural net inference. arXiv preprint arXiv:2001.03288 (2020)

45. Qin, Z., Zhang, P., Wu, F., Li, X.: Fcanet: Frequency channel attention networks. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 783–792 (2021)
46. Qualcomm: Snapdragon 8 gen 2 (2023)
47. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. In: Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS). NeurIPS Foundation (2017)
48. Sajjadi, M.S., Vemulapalli, R., Brown, M.: Frame-recurrent video super-resolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6626–6634 (2018)
49. Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1874–1883 (2016)
50. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
51. Tao, X., Gao, H., Liao, R., Wang, J., Jia, J.: Detail-revealing deep video super-resolution. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4472–4480 (2017)
52. Tian, Y., Zhang, Y., Fu, Y., Xu, C.: Tdan: Temporally-deformable alignment network for video super-resolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3360–3369 (2020)
53. Toneva, M., Sordoni, A., Combes, R.T.d., Trischler, A., Bengio, Y., Gordon, G.J.: An empirical study of example forgetting during deep neural network learning. arXiv preprint arXiv:1812.05159 (2018)
54. Wang, H., Chen, P., Zhuang, B., Shen, C.: Fully quantized image super-resolution networks. In: Proceedings of the 29th ACM International Conference on Multimedia. pp. 639–647 (2021)
55. Wang, H., Su, D., Liu, C., Jin, L., Sun, X., Peng, X.: Deformable non-local network for video super-resolution. IEEE Access **7**, 177734–177744 (2019)
56. Wang, X., Chan, K.C., Yu, K., Dong, C., Change Loy, C.: Edvr: Video restoration with enhanced deformable convolutional networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2019)
57. Xiao, X., Wang, W., Chen, T., Cao, Y., Jiang, T., Zhang, Q.: Sensor-augmented neural adaptive bitrate video streaming on uavs. IEEE Transactions on Multimedia **22**(6), 1567–1576 (2019)
58. Yeo, H., Chong, C.J., Jung, Y., Ye, J., Han, D.: Nemo: enabling neural-enhanced video streaming on commodity mobile devices. In: Proceedings of the 26th Annual International Conference on Mobile Computing and Networking. pp. 1–14 (2020)
59. Yeo, H., Do, S., Han, D.: How will deep learning change internet video delivery? In: Proceedings of the 16th ACM Workshop on Hot Topics in Networks. pp. 57–64 (2017)
60. Yeo, H., Jung, Y., Kim, J., Shin, J., Han, D.: Neural adaptive content-aware internet video delivery. In: 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18). pp. 645–661 (2018)
61. Yin, L., Li, G., Fang, M., Shen, L., Huang, T., Wang, Z., Menkovski, V., Ma, X., Pechenizkiy, M., Liu, S., et al.: Dynamic sparsity is channel-level sparsity learner. Advances in Neural Information Processing Systems **36** (2024)

62. Ying, X., Wang, L., Wang, Y., Sheng, W., An, W., Guo, Y.: Deformable 3d convolution for video super-resolution. IEEE Signal Processing Letters **27**, 1500–1504 (2020)
63. Yu, J., Fan, Y., Yang, J., Xu, N., Wang, Z., Wang, X., Huang, T.: Wide activation for efficient and accurate image super-resolution. arXiv preprint arXiv:1808.08718 (2018)
64. Yuan, G., Ma, X., Niu, W., Li, Z., Kong, Z., Liu, N., Gong, Y., Zhan, Z., He, C., Jin, Q., et al.: Mest: Accurate and fast memory-economic sparse training framework on the edge. Advances in Neural Information Processing Systems **34**, 20838–20850 (2021)
65. Zawad, S., Li, C., Yao, Z., Zheng, E., He, Y., Yan, F.: Dysr: Adaptive super-resolution via algorithm and system co-design. In: The Eleventh International Conference on Learning Representations (2022)
66. Zhan, Z., Gong, Y., Zhao, P., Yuan, G., Niu, W., Wu, Y., Zhang, T., Jayaweera, M., Kaeli, D., Ren, B., et al.: Achieving on-mobile real-time super-resolution with neural architecture and pruning search. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4821–4831 (2021)
67. Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B., Fu, Y.: Image super-resolution using very deep residual channel attention networks. In: Proceedings of the European conference on computer vision (ECCV). pp. 286–301 (2018)
68. Zhang, Y., Wei, D., Qin, C., Wang, H., Pfister, H., Fu, Y.: Context reasoning attention network for image super-resolution. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4278–4287 (2021)
69. Zhao, F., Zhao, J., Yan, S., Feng, J.: Dynamic conditional networks for few-shot learning. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 19–35. Springer (2018)
70. Zhou, J., Jampani, V., Pi, Z., Liu, Q., Yang, M.H.: Decoupled dynamic filter networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE/CVF (2021)
71. Zhu, K., Zhao, W., Zheng, Z., Guo, T., Zhao, P., Bai, J., Yang, J., Liu, X., Diao, L., Lin, W.: Disc: A dynamic shape compiler for machine learning workloads. In: Proceedings of the 1st Workshop on Machine Learning and Systems. pp. 89–95 (2021)