Super-Resolution Streaming Media System Based on Deep Learning

Dongxiao Lin, *1900271053* Siqi Luo, *1900271029,* and Jingming He, *1900271030*

Abstract—

With the rapid development of Internet video streaming, the quality of the video we transmit has been increasing. When bandwidth resources are limited, it is generally necessary to reduce the video quality to ensure fluency, which has a negative impact on user experience quality (QoE). Besides, the current video transmission infrastructure does not fully utilize the computing power of the client. In this paper, we implement a video transmission framework that uses the computing resources of the client to perform super-resolution processing of low-quality video through deep neural networks (DNNs) to maximize the user's QoE. Moreover, we propose a fast and lightweight super-resolution model and carry it in the system we designed. Through comparison experiments with the baseline, we can reduce the amount of model parameters and processing time while ensuring the quality of the output super-resolution video.

Index Terms—Deep Learning, Super-Resolution, Streaming Media System.

1 Introduction

With the rapid development of Internet video streaming, as well as the advancement of camera equipment and video processing tools, the number and quality of the videos we produce and transmit are increasing. However, due to unstable network transmission or limited bandwidth resources, it is necessary to reduce the video quality to ensure the fluency, which will result in a decrease in user experience quality (QoE). Traditional compression and decompression algorithms can bypass bandwidth limitations, but unstructured data such as video may not accurately transmit video content after compression. Besides, with the development of hardware devices (such as graphics processors), the computing power of the client continues to improve. The current video transmission infrastructure has not fully utilized the computing power of the client, resulting in a waste of computing resources. In this paper, we implement a video transmission framework that uses the computing resources of the client to perform super-resolution processing of low-quality video through deep neural networks (DNNs) to obtain the high-resolution video in a limited bandwidth environment.

Super-resolution based on deep learning has experienced tremendous progress in recent years [1], [4], [2], [5]. Compared with traditional algorithms, deep learning models can retain more textures and details in videos when learning the mapping from low-quality videos to high-quality videos. However, the work in recent years has focused on improving the quality of the output image, resulting in higher complexity, and the computing power of the GPU on the client is much lower than that of the professional graphics computing server, so using these models will cause lag in video playback. Unlike the focus of previous work, we propose a fast, lightweight super-resolution model, and carried it in the system we designed. The network is based on the fire module proposed by [3], and the sub-pixel convolutional of ESPCN [5]. The comparison experiment with the baseline proves that we can reduce the amount of model parameters and processing time while ensuring the quality of the output super-resolution video. Moreover, our model can adapt to clients with different computing capabilities by adjusting hyperparameters.

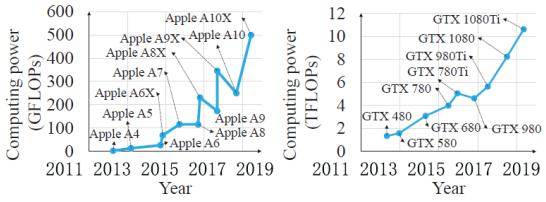
2 Background

Part of the traditional approaches to improve video streaming quality is to work hard on bandwidth. Finding video quality that fits the current state of the network can make more efficient use of existing bandwidth. However, the upper limit of video quality improvement using this method is limited to network bandwidth.

The other part uses various compression and decompression algorithms, bypassing bandwidth limitation, and providing better services. Nevertheless, traditional compression and decompression algorithms also have some limitations. For instance, as for unstructured data such as video, it is not necessarily able to accurately transmit video content after compression to a certain extent.

super-resolution refers to the reconstruction of the corresponding high-resolution image from the low-resolution image observed. Super-resolution has important application value in surveillance, medical images, or other aspects where raw high-quality images/videos are not available.

The traditional super-resolution method is based on interpolation. The common interpolation algorithms include nearest-neighbor interpolation, bilinear interpolation, and bicubic interpolation. The traditional algorithm only uses the information of the low-resolution image itself, and the pixel points of each position are interpolated based on the



(a) Mobile (b) Desktop

Fig. 1: Growth of GPU's processing power

information around the pixel, so the reconstructed image is generally not of good quality.

3 Motivation and Goal

There are two major defects in the current video streaming transmission methods: One is that the computing power of the client is not fully utilized. Figure 1 shows the exponential growth of GPU computing power overtime on mobile devices as well as desktops. The latest mobile devices even have dedicated hardware for neural processing. However, the current video transmission infrastructure does not make full use of the computing power of the client. Another disadvantage is that video encoding capabilities are currently limited. The codec can only encode and compress within a group of pictures (GOP), and GOP is usually unable to capture the redundancy within a large time scale with the order of magnitude of seconds of online video.

In view of the shortage of traditional video streaming technologies and interpolation-based super-resolution methods, we put forward a new framework. It uses low-resolution video as transmission, which greatly reduces bandwidth constraints, and then uses the deep neural network(DNN) model embedded in the Fire module on the client-side to improve the resolution of the video to achieve high resolution, thus making full use of the client's computing resources. In a video, many elements are repeated in a relatively long time, which is conducive to train the model of converting low-resolution elements into high-resolution elements.

4 Related Work

Dynamic Adaptive Streaming over HTTP(DASH) is a kind of adaptive bit-rate streaming technology. Its main idea is to cut media files into small video segments, each of which is encoded into several different bit-rates to meet network requirements of different clients. DASH is primarily limited by bandwidth, and when network resources fluctuate, the user's quality of experience(QoE) drops noticeably. Therefore, in this paper, we are always transmitting low-resolution video on the server-side, which reduces the bandwidth restriction.

As we know, the traditional interpolation-based super-resolution methods can not get a good quality of the high-resolution image. With the rapid development of deep learning technology, the image super-resolution method based on deep learning achieves the optimal performance and effect on many testing tasks. The more representative super-resolution models based on deep learning include super SRCNN [1], FSRCNN [2], ESPCN [5], VDSR [4], etc. However, these models are in high complexity, and the training time is correspondingly longer. The super-resolution based on deep learning algorithms can be improved in three directions:

- optimizing image quality.
- accelerating the processing speed of the model and improve its real-time performance.
- reducing the parameters of the model.

Most of the previous models focus on the first point. Since our super-resolution model is used in network streaming systems, therefore, we pay more attention to the latter two points.

Under the premise of ensuring basic image quality, the goal of our model is to accelerate the speed of mapping from low resolution to super-resolution to avoid the occurrence of lag during video playback. Besides, since the trained model needs to be passed from the server to the client, the number of model parameters should be minimized to avoid excessive bandwidth. Read-time performance is pursed in video streaming media transmission. Therefore, we build a lightweight super-resolution model based on deep learning.

5 Super Resolution Based on Deep Learning

5.1 Overall Process

Figure 2 shows the overall architecture of the super-resolution model based on deep learning. If the original input is an image, it is processed directly. Otherwise, if the input is video, the video will be split into frames and processed frame by frame.

YUV is a color-coding method. "Y" means luminance, which is the grayscale value. "U" and "V" mean chrominance, which describes image color and saturation and is used to specify the color of pixels. Studies have shown that the human eye is more sensitive to low-frequency signals than to high-frequency ones. In other words, the human eye is more sensitive to changes in brightness than to changes in color. So for humans, the Y channel is more critical than the UV channel.

Therefore, when a deep learning model is used to process image or frame, a low-resolution image is first converted from RGB space into YUV space. Then, depending on the sensitivity of human eyes to brightness changes, the complex super-resolution deep DNN model is only applied to the Y channel, while the remaining UV channel is treated with simple bicubic interpolation. Finally, the DNN output merges the interpolated UV channel to output the final color image. It is not hard to see that the most critical and time-consuming part is in the DNN model.

5.2 The Fire Module

In order to realize real-time video transmission, we design a lightweight DNN model. It is achieved by reducing the parameters required for model training. Inspired by [3], we use two strategies for reducing parameters and eventually integrate them into a Fire module:

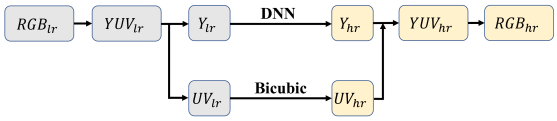


Fig. 2: The overall process of performing super resolution on a frame of the video.

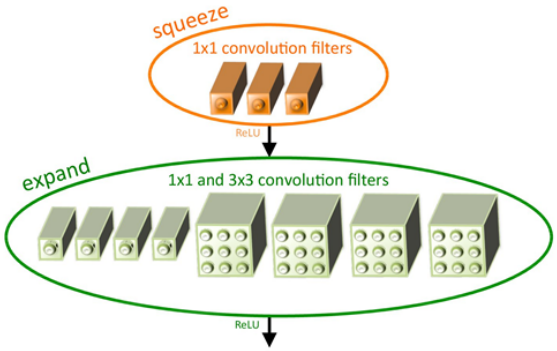


Fig. 3: The structure of fire module

1) reducing 3×3 convolution with 1×1 convolution. Through this step, the number of parameters of a convolution operation is reduced by 9 times.
2) reducing the number of input channels of 3×3 convolution layer. A convolution layer of a 3×3 convolution kernel, in which the total number of parameters is $3 \times 3 \times M \times N$ (where M and N are the numbers of channels of input Feature Map and output Feature Map respectively).

Aiming to keep a small number of parameters in CNN, not only the number of 3×3filers(see # 1 above) should be reduced, but also the number of input channels in the 3×3 convolution layer should be reduced(see # 2 above). We combine these two strategies into the Fire module. The structure of the Fire module is shown in Figure 3. The Fire module consists of the Squeeze layer and the Expand layer. The Squeeze layer is composed of a group of continuous 1×1 convolution kernels. And the Expand layer is composed of a mixture of several continuous 1×1 filters and 3×3filters.

The structure of the Fire module is shown in Figure 4. The Fire module consists of the Squeeze layer and the Expand layer. The Squeeze layer aims to reduce the number of input channels. It consists of C/θ convolution kernels of 1×1 size, where θ is an adjustable parameter used to control the number of channels that Squeeze outputs. The expand layer is a combination of several 1×1 and 3×3 convolution filter. Finally, the convolution results of the two different filters are merged as output.

Figure 4 shows the Fire module in more detail. It assumes that the pixel of the Feature Map is $H \times W$, and the number of channels is C. In order to reduce the number of input channels. Squeeze uses 1×1 convolution kernel to compress the input channels. Next, the Expand layer takes the output of Squeeze as input. It uses several 1×1 filters and 3×3 filters for convolution. Finally, combine the convolution results of the two and obtain a Feature Map of $H * W * C'$ size as the output of the Fire module.

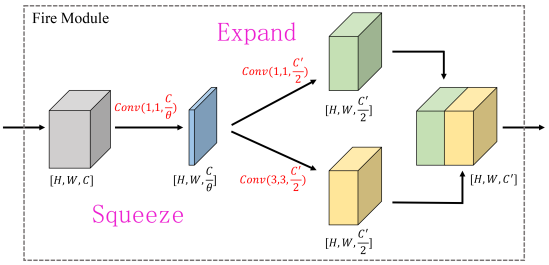


Fig. 4: The fire module in our super resolution network.

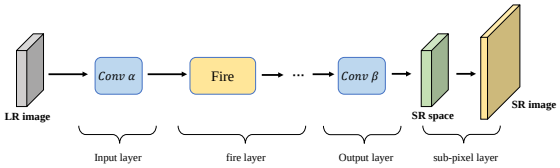


Fig. 5: The overall super resolution network architecture.

5.3 Network Architecture

The Fire module greatly reduces the number of parameters without changing the quality of images, which improves the training speed of the model. We embed the Fire module into the DNN model as the architecture of this paper. Figure 5 shows the DNN model with the Fire module embedded.

The input of the model is a low-resolution image or video frame. We use traditional convolution as the input layer (Convα) and output layer (Convβ) of the network. The feature map after Convα is the input of the fire layer. And in the fire layer, multiple Fire modules can be stacked. After the output layer, we use a sub-pixel convolution proposed by [5] to aggregates the feature maps from low-resolution space and builds the high-resolution image in a single step.

6 Streaming Media System Design

We use the B/S architecture in our streaming media super-resolution system. Figure 6 shows the overall of the system design.

Backend service. We use Webpack to pack all resources to separate files. So resources can be hosted by any static service, such as Nginx, Apache. For video resources, every mp4 file is encoded into a fragmented mp4 format. Therefore, it would reduce the cost of switching video quality.

Controller. The core logic can be divided into two parts, processor and web worker. In processor, it implements a resource loader, user interface handler and video processor.

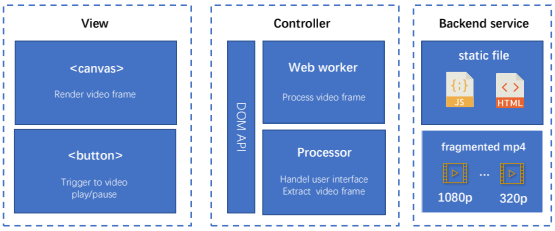


Fig. 6: The overall streaming media system design.

Fig. 7: The communication process between processor and web worker.

Fig. 8: A Screenshot of our streaming media system user interface.

- Resources loader. It handles the resources load process and fetches trained model and video to the browser side.
- User interface handler. It accepts events from the user interface and controls the render process.
- Video processor. After achieving video resources from resources loader, extract video frames using DOM APIs. Extracted video frame would be preprocessed and sent to the web worker later.

The super-resolution process is implemented in the web worker using the Tensorflow.js framework. Tensorflow.js is an implement of Tensorflow in JavaScript so that we can convert any trained Tensorflow model to browser-ready, WebGL boosted version.

This process is computationally intensive, which would block user interface render thread. The web worker is proposed to address this problem. It provides a multi-thread level API in the browser side. With the help of the web worker, we can put the calculation in the background thread. Figure 7 shows the details of the communication process between processor and web worker. A preprocessed video frame is sent to the web worker. After computation, the web worker sends the output back to the processor, and then the super-resolution video frame would be rendered.

View. We use `<canvas>` to render video frame and `<button>` to control process. Figure 8 is a Screenshot of our streaming media system user interface.

TABLE 1: Results of image super-resolution: scale=2.

Method	Fires	Sizeβ	Params(K)	AvgTime(ms)	PSNR	SSIM
Bicubic	-	-	-	0.15	29.59	0.84
ESPCN	-	-	21.28	1.26	33.67	**0.91**
Ours	2	3×3	4.36	1.13	33.65	**0.91**
Ours	1	5×5	4.80	1.16	**33.79**	**0.91**
Ours	1	3×3	**2.76**	**1.08**	33.56	**0.91**

TABLE 2: Results of image super-resolution: scale=3.

Method	Fires	Sizeβ	Params(K)	AvgTime(ms)	PSNR	SSIM
Bicubic	-	-	-	0.24	26.30	0.74
ESPCN	-	-	22.73	1.48	**30.06**	**0.83**
Ours	2	3×3	5.80	1.29	29.95	**0.83**
Ours	1	5×5	8.81	1.30	30.04	**0.83**
Ours	1	3×3	**4.20**	**1.25**	29.91	**0.83**

TABLE 3: Results of image super-resolution: scale=4.

Method	Fires	Sizeβ	Params(K)	AvgTime(ms)	PSNR	SSIM
Bicubic	-	-	-	0.26	24.19	0.67
ESPCN	-	-	24.75	1.54	27.65	**0.75**
Ours	2	3×3	6.67	1.43	27.66	**0.75**
Ours	1	5×5	14.42	1.37	**27.78**	**0.75**
Ours	1	3×3	**6.22**	**1.31**	27.61	**0.75**

7 Experiments

7.1 Implementation

The hardware used in the experiment is equipped with the NVIDIA Tesla V100 graphics card. We use a 64-bit Ubuntu 16.04 LTS operating system and TensorFlow 1.15, CUDA 9.0.

Dataset. When comparing and analyzing our proposed method with the baseline, we conduct experiments on the Timofte dataset provided in [6], which is used in many single-image super-resolution method tests. We crop the picture into N*N size image samples. The final train/test sets we chose contained 4,063/1,135 image samples, respectively. When we tested the video streaming system, the selected video data was obtained from the Internet. These are two video clips about anime scenes and real scenes.

Implementation details. In our experiment, the number of fire modules will be set to 1 or 2, the convolution kernel size of Convα is set to 5, and Convβ is set to 5 or 3. In the training phase, the initial learning rate is set to 0.0001; the maximum number of iterations is set to 100. We blur high-resolution image samples using a gaussian filter and sub-sample it by the upscaling factor to synthesize the low-resolution samples. The size of the high-resolution image sample is set to $17r \times 17r$, where r is the upscaling factor (scale).

Evaluation Metrics and Baseline. We chose the lightweight model ESPCN [5] as the baseline to verify our advantages in performance and speed. We use the peak signal to noise ratio (PSNR) and the structural similarity (SSIM) as the performance metric to evaluate our models. Besides, we tested the average processing time (AvgTime) of each image sample under the same hardware conditions and calculated the total parameters of the model.

7.2 Image Super-Resolution

We perform the super-resolution of a single image on the Timofte dataset and compare our proposed method with the baseline. We test the total params, average time, PSNR and SSIM with scale=2, 3, 4 respectively. Besides, we will perform

TABLE 4: The result of real-time super-resolution processing of video in our streaming media system.

Method	Fires	Sizeβ	Frames Per Second
ESPCN	-	-	2.647
Ours	1	3×3	4.779

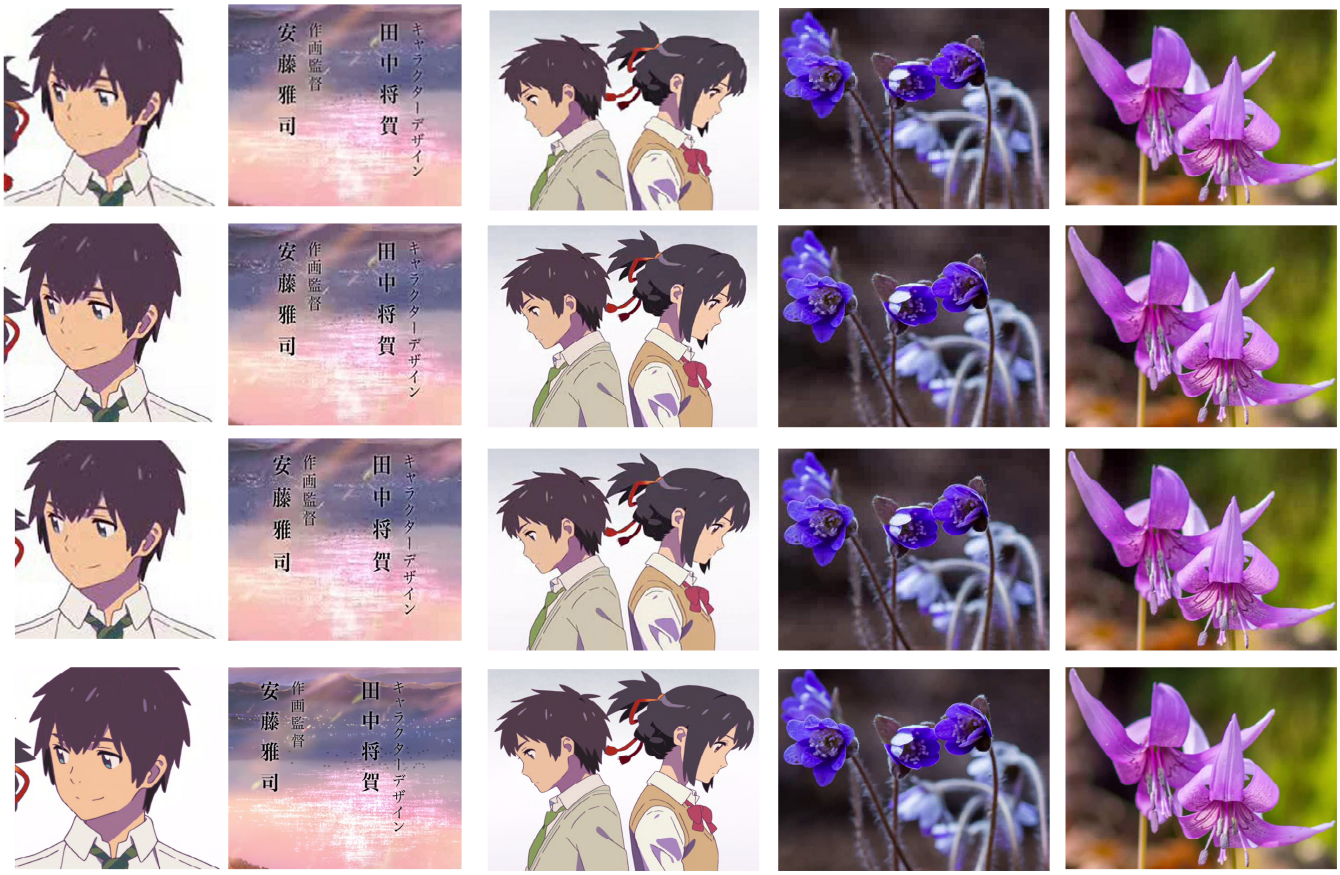


Fig. 9: Qualitative results from the video clip. From top to bottom are the bicubic method, espcn, ours and original HD images, respectively. More details can be enlarged to view the picture.

TABLE 5: The performance of our streaming media system in different GPU Runtime environment.

GPU Runtime	Frames Per Second
Default	4.779
Force to use discrete graphics	5.716
CUDA	28.571

ablation experiments on the number of fire modules in the network and the size of Convβ's convolution kernel to test the impact of different settings on time and performance.

From the table Table. 1, 2, 3, we can see that the params of our model have been dramatically reduced and our processing speed is faster than ESPCN. Besides, we obtain a competitive image quality compared to ESPCN. Specifically, the super-resolution images obtained by our model and ESPCN are utterly equal in SSIM and quite close in PSNR. When scale=2,3 and 4, our processing speed increases 17%, 17% and 18%, respectively. The qualitative results can be seen in Figure 9. The results we obtained are not significantly different from those obtained by ESPCN.

We conduct experiments with different settings on the number of fire modules (hereinafter referred to as Fires) and the convolution kernel size of Convβ (hereinafter referred to as Sizeβ) in our proposed method. When the Fires or the Sizeβ increased, the quality of the super-resolution image will be improved, but the number of parameters and processing time will also increase accordingly. Using the large Sizeβ can get more quality improvement. Even when scale=2 or 4, Fires=1 and Sizeβ=5, the PSNR of ours exceeds the ESPCN, but it will also bring a larger amount of parameters and processing time. Besides, the number of output channels of Convβ is positively correlated with the upscaling factor. Therefore, a larger upscaling factor may bring an excessive amount of parameters. For example, when scale=2, the Sizeβ change from 3×3 to 5×5, the parameter amount is increased by 74%. When scale=3 or 4, it increases by 110% and 132%. Therefore, when we need a larger upscaling scale, it is not appropriate to use large Sizeβ. Overall, the various configurations of our model can ensure super-resolution image quality while having less model size and inference time. Our method is more suitable for the super-resolution streaming media system.

7.3 Benchmark in Production Environment

We have introduced a streaming media system design in section 6. With the help of modern browser technology, the super-resolution process can be run cross-platform. In this part, we will perform a benchmark in the production environment.

Environment. We test this system in a mainstream performance notebook, which has Intel Core i7-8550U CPU and NVIDIA Geforce MX150 GPU. On the browser side,

we choose Google Chrome 83, which is currently the most popular browser. Each setting is set to the default value.

Frame rate benchmark. We use frames per second (FPS) as metrics to measure method performance in the production environment. As a result, shows in Table. 4, our method has achieved 1.805 times faster than the ESPCN method.

GPU Runtime environment. When we were testing, we found that the browser would call integrated graphics, which is a default setting in NVIDIA graphics. We change the setting, force the browser to call discrete graphics. The result is shown in Table. 5. We note that using discrete graphics can improve performance. The limitations of the browser began to show because it can not effectively use the graphics card. Compared with previous experiments, the CUDA based Tensorflow implement is 5.00 times faster than the WebGL based Tensorflow.js implement. This means that we need more optimization to address browser performance loss.

8 Conclusion

In this paper, we implement a video transmission framework that uses the computing resources of the client to perform super-resolution processing of low-quality video through deep neural networks, to obtain the high-resolution video in a limited bandwidth environment for maximize the QoE of users. Moreover, we propose a fast, lightweight super-resolution model, and carried it in the system we designed. The comparison experiment with the baseline proves that we can reduce the amount of model parameters and processing time while ensuring the quality of the output super-resolution video. Besides, our model can adapt to clients with different computing capabilities by adjusting hyperparameters.

References

[1] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*, pages 184–199. Springer, 2014.

[2] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *European conference on computer vision*, pages 391–407. Springer, 2016.

[3] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[4] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016.

[5] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. 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*, pages 1874–1883, 2016.

[6] Radu Timofte, Vincent De Smet, and Luc Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *Asian conference on computer vision*, pages 111–126. Springer, 2014.