applied sciences

MDPI

Article

Video Super Resolution Using a Selective Edge Aggregation Network

Samuel Kang, Young-Min Seo and Yong-Suk Choi *

Artificial Intelligence Laboratory, Hanyang University, Seoul 04763, Korea; ksml007@hanyang.ac.kr (S.K.); s0min@lguplus.co.kr (Y.-M.S.)
* Correspondence: cys@hanyang.ac.kr

Abstract: An edge map is a feature map representing the contours of the object in the image. There was a Single Image Super Resolution (SISR) method using the edge map, which achieved a notable SSIM performance improvement. Unlike SISR, Video Super Resolution (VSR) uses video, which consists of consecutive images with temporal features. Therefore, some VSR models adopted motion estimation and motion compensation to apply spatio-temporal feature maps. Unlike the models above, we tried a different method by adding edge structure information and its related post-processing to the existing model. Our model "Video Super Resolution Using a Selective Edge Aggregation Network (SEAN)" consists of a total of two stages. First, the model selectively generates an edge map using the target frame and also the neighboring frame. At this stage, we adopt the magnitude loss function so that the output of SEAN more clearly learns the contours of each object. Second, the final output is generated using the refinement (post-processing) module. SEAN shows more distinct object contours and better color correction compared to other existing models.

Keywords: computer vision; video super resolution; edge detection

Citation: Kang, S.; Seo, Y.-M.; Choi, Y.-S. Video Super Resolution Using a Selective Edge Aggregation Network. *Appl. Sci.* **2022**, *12*, 2492. https://doi.org/10.3390/app12052492

Academic Editor: Federico Divina

Received: 28 December 2021
Accepted: 17 February 2022
Published: 27 February 2022

1. Introduction

The VSR task is usually processed using information from consecutive frames. To use this information, some VSR models [1,2] adopted motion estimation and motion compensation to apply spatio-temporal features. However, in datasets that expresses large-scale motion, accurate motion estimation is challenging. On the other hand, edge maps can be generated through the widely used Canny edge detection, providing intuitive and structural information to the model. Therefore, we propose Stage 1 of our model as follows by adding structure information and a loss function to the existing model.

Before generating an edge map, it is necessary to detect the motion difference between neighboring frames. Although most of the spatio-temporal information between the consecutive frames is similar, information differences exist if some objects in the frame show large-scale motion.

In the case of a neighboring frame with little change in motion, there is no significant difference in the edge maps obtained in each frame. However, in the case of a frame with a large change in motion, there is a big difference. Therefore, we use the following three methods to obtain the edge map used in our model.

Detect blur frame. The variance of the frame calculated using the Laplacian filter can express the degree of blur. The larger the variance, the greater the degree of change in the pixel value, which means that there are many areas with a clear contour of the object on the frame. Our model considers the variance of the frame to be a blur frame if it is lower than the threshold. Therefore, our model uses only clear frames to extract an edge map.

Edge map extraction. Our model uses Canny edge detection to generate an edge map of each frame, except blur frames. The final edge map used in the model is generated by fusing the edge map of the target frame and the edge map of the neighboring frame one-to-one.

Magnitude loss function. During the training, our model uses a Sobel mask to obtain the gradient magnitude corresponding to each output of the Ground Truth (GT) and the output of our model. Using the magnitude loss function, which minimizes the difference in the magnitude between those, our model trains the contours of GT. Because artifacts exist in some frames of the TestSet, such as Vid4 [3], incomplete areas are generated. To solve this problem, there has been an attempt to use correlation between frames [4]. However, these attempts also require accurate calculations such as flow estimation. So, we use an additional module to solve this problem.

Refinement. The refinement process receives the output of Stage 1 and shows qualitatively and quantitatively improved results. This process looks similar to the refining process in the classic vision field called post-processing, but it has a different character. First, it uses deep learning, and in particular, it is possible to minimize information loss, a classic problem of convolution operation. Second, the edge and color on the output image can be processed at the same time, and it is quantitatively excellent because it tries to get as close to the GT image as possible between learning. Third, the proposed refinement was applied to the VSR (Video Super Resolution) field for the first time, and since it was designed with high versatility, it is easily compatible with other VSR models according to the user's choice.

2. Related Work

2.1. Single Image Super Resolution

Most SISR models pass single images through deep convolution layers to achieve image super resolution. In particular, VDSR [5] showed a faster convergence speed and improved performance compared to SRCNN [6] by using residual learning.

In addition to attempting to use RGB frames as input data, there was an attempt to use structural information in the model by providing an additional edge map. In particular, Ref. [7] generated an HR edge map using a low-resolution image and the corresponding edge map, which served as structural information.

However, there was a problem in which blur areas occurred in the images generated. Models using adversarial learning showed the ability to overcome these limitations. In particular, SRGAN's [8] performance was lower than that of other models, but it showed improved results when evaluated with the human eye.

2.2. Video Super Resolution

A video dataset is a set of consecutive frames, and information between neighboring frames is similar. Therefore, the VSR model attempted to combine the spatio-temporal information of the target frame and the neighboring frames.

Ref. [4] tried reconstruction without aligning the feature. Temporal information on the neighbor frame feature was combined with the target frame feature through TM-CAM, and spatial information according to the multi-scale was combined with the target frame feature through CN-CAM.

Refs. [9,10] combined spatio-temporal information by aligning the neighbor image feature using deformable convolution networks [11]. In particular, EDVR [10] showed more sophisticated alignment using PCD.

Refs. [1,2,12] applied motion estimation and motion compensation. Motion estimation is a process of extracting motion information between neighboring frames and has an important influence on VSR model performance. However, motion estimation does not guarantee accuracy because it is vulnerable to large-scale motion. Therefore, our model adds structure information and additional modules to improve this vulnerability.

2.3. Post-Processing

First, post-processing used in the existing SISR or VSR operates based on CNN. Park et al. proposed the In-loop Filter CNN (IFCNN) to replace SAO [13]. In addition, Dai et al. proposed a Variable-Sized Filter Residual Learning Convolutional Neural Net-

work (VRCNN) to replace both DB and SAO in HEVC intra-coding [14]. In their experiments, VRCNN was reported to achieve a promising result of an average reduction in BD speed of 4.6%. Both VRCNN and IFCNN are shallow networks with less than five layers. Kang et al. proposed a Multi-Modal/Multi-Scale CNN (MMS-net) to replace the existing DB and SAO in HEVC using a deeper network of 30 layers [15]. Wang et al. proposed a Deep CNN-based Auto-Decoder (DCAD) for post-processing [16]. In this paper, DCAD claims that there are average BD-rate reductions of 5.0%, 6.4%, 5.3%, and 5.5%, respectively, for AI, LDP, LDB, and RA configurations. Ma et al. also proposed a residual-based video restoration network (residual VRN) for video post-processing [17].

3. Method

Our model consists of a total of two stages, as shown in Figure 1. Stage 1 consists of progressive recurrent structure to restore LR_t to SR_t inspired by RBPN.

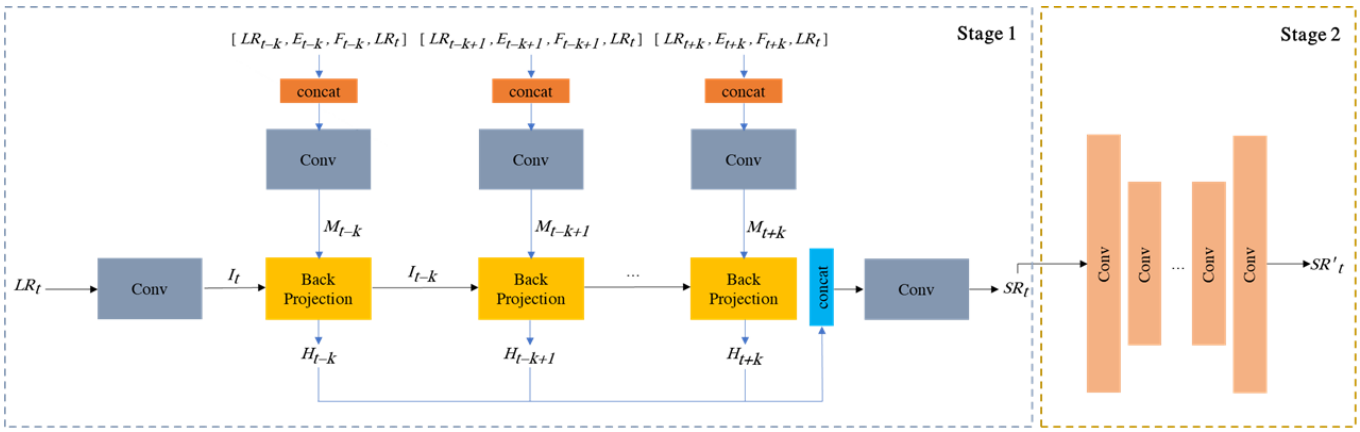

Figure 1. Architecture of our Video Super Resolution Using Selective Edge Aggregation Network (SEAN). It contains two stages: image teconstruction (Stage 1) and refinement (Stage 2).

In Stage 1, LR_t is the target frame and 2k neighboring frames are $\{LR_{t-k}, \ldots, LR_{t+k}\}$. The optical flow F_{t-k} is precomputed between LR_t and LR_{t-k} by Pyflow [18]. Edge map e_t of LR_t is generated using Canny edge detection, E_{t-k} is the edge map extracted through edge map extraction (Section 3.1). The size of LR_t, F_{t-k}, and E_{t-k} is w × h, and the size on SR_t is W × H (w < W, h < H).

Stage 2 is the method of refining SR_t to generate SR'_t, which is the final output of the model. Inspired by VDSR, the refinement module reduces the size of SR_t and then extends the size to generate SR'_t that is W × H.

3.1. Edge Map Extraction

Just as F_{t-k} is generated using LR_t and LR_{t-k}, we use e_t and e_{t-k} to extract E_{t-k}. Before extracting E_{t-k}, we consider the motion difference between LR_t and LR_{t-k}. If large-scale motion exists, noise occurs in E_{t-k} generated by fusing e_t and e_{t-k}. Therefore, we proceed with the detection of blur frames to reduce noise.

We use the Laplacian filter to get the variance of LR_{t-k}, and if the variance is less than the threshold, we consider LR_{t-k} to be a blur frame. If LR_{t-k} is a blur frame, our model does not use e_{t-k} when generating E_{t-k}, and e_t is used instead of e_{t-k}. After detecting the blur frames, our model generates E_{t-k} as shown in Equation (1).

$$E_{t-k}^{i,j} = \begin{cases} 0 & \text{if, } e_{t-k}^{i,j} + e_t^{i,j} = 0 \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

$E^{i,j}_{t-k}$ and $e^{i,j}_{t-k}$, which contain all the information on two edge maps (or a single edge map), provide the structural information corresponding to time $t-k$ to our model. i and j are horizontal and vertical locations, respectively.

3.2. Feature Extraction

To extract target frame feature I_t, LR_t is passed through a convolution layer. Then, LR_t,F_{t-k},E_{t-k}, and LR_{t-k} are concatenated with nine channel-sized features. To provide our model with multi-scale information, the concatenated map is passed through a convolution layer that yields M_{t-k}. So M_{t-k} that has spatio-temporal and structure information can fill in the insufficient feature of I_t.

3.3. Back Projection

In the back projection block, features extracted in Section 3.2 can be upscaled and combined to make final features. By projecting I_t to each neighboring feature $\{M_{t-k}, \ldots, M_{t+k}\}$, I_t is provided with multi-scale features, and then our model generates $\{H_{t-k}, \ldots, H_{t+k}\}$. In Net_{down}, H_{t-k} is downscaled to I_{t-k} for providing next back projection with H_{t-k} feature. Net_m, Net_s, Net_{res}, and Net_{down} are networks on RBPN. The entire structure is in Figure 2 and equations are described as follows:

$$H^m_{t+k} = Net_m(M_{t+k}) \tag{2}$$
$$H^s_{t+k-1} = Net_s(I_{t+k-1}) \tag{3}$$
$$H_{t+k} = Net_{res}(H^m_{t+k} - H^s_{t+k-1}) \tag{4}$$
$$I_{t+k} = Net_{down}(H_{t+k}) \tag{5}$$

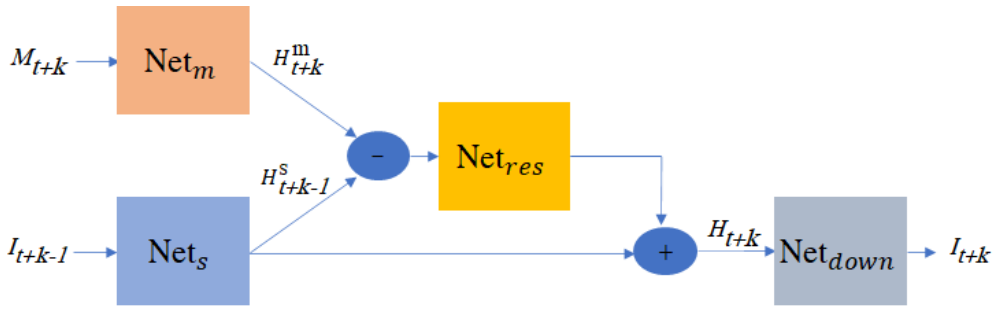

Figure 2. Structure of back projection.

3.4. Reconstruction

Finally to make LR_t into SR_t, our model concatenates $[H_{t-k}, \ldots, H_{t+k}]$ to combine all features. This combined feature is generated as SR_t through a convolution layer f_{SR} with kernel size of 1, and can be described as follows:

$$SR_t = f_{SR}([H_{t-k}, \ldots, H_{t+k}]) \tag{6}$$

3.5. Refinement

Figure 3 is the architecture of the Refinement module we propose and explains how it was made from now on. We add a series of purification steps in this paper. We propose a refining process to obtain a higher quantitative/qualitative score, and excellent color correction effects and contour processing effects can be obtained. In the existing field of super resolution, there was a limitation in which it was difficult to process clear color correction and contour processing at once. This is because it was basically a method of enlarging a small image size to a large size. Therefore, we add a refinement stage that learns while maintaining the original image size for efficient processing. The input of the proposed method is the enhanced image output from Stage 1. After that, the edge map of the input image is extracted using "Canny edge detection" as a pre-processing process, and then

merged in units of channels to learn together. In the first convolution step, the channel size, width, and height are reduced once, and then, after learning the same size in 17 layers, restored to the original size again in the last layer. At this time, the calculation of reducing and increasing the size once solves the problem of the number of parameters and at the same time facilitates learning better than a layer without compression or restoration at all. This is based on the characteristics of the convolution operation. Finally, a total of 19 layers are used, 3D convolution is used to calculate the time axis, and an experimentally proven optimal layer is created.

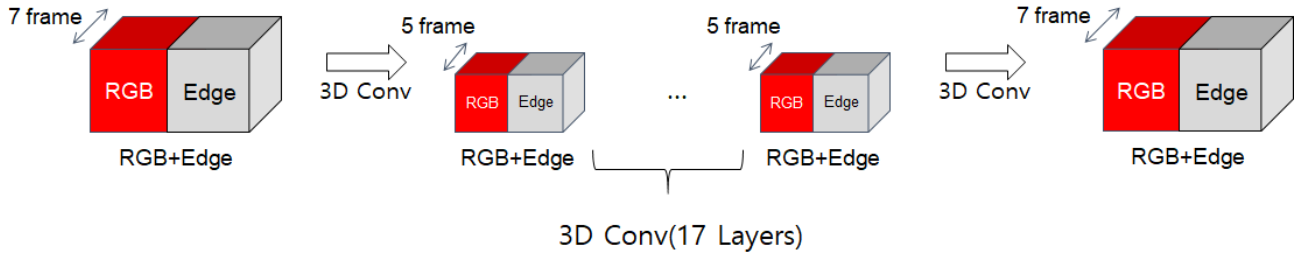

Figure 3. This is the architecture of our proposed refinement module. The RGB image and the edge image are input as a pair, and compression and restoration occur once at the beginning and end through 3D convolution. The central 17 3D convolution layers were maintained without any size change to minimize the loss.

"Canny edge detection" is used for edge extraction at the refinement stage. In this case, the sigma value of 0.7 is the most optimal. This has the best effect in the learning process because it creates a distinct edge necessary for learning. Specific reasons, comparisons, and explanations will be described later in Section 4.2.

Finally, we propose a new refinement learning method. Figure 4 visually shows the method. The refinement we propose does not up-sample small-sized images, but corrects it with deep learning when an input with the same size as the final output is received. To train the refinement module, we magnify a small image by four times using the bicubic upsampling technique, and then compare it with GT to learn. However, the picture quality enlarged by bicubic upsampling is not suitable, and this affects the performance of the result. Therefore, we use SRCNN as a method to build a dataset of refinement. This makes it possible to build a high-quality training set. This is a good choice because the more good images are input, the fewer parts to consider during the learning process and the sharper edges can be built. SRCNN is a recognized SISR model, but it is very suitable for simply building datasets. For the training data of SRCNN, 400,000 IMAGENET data [19] and a small training set [20,21] are used. In subsequent experiments, we use the Vimeo90k [1] and REDS [22] training sets as the SRCNN test set, and then use the output as the refinement train sets.

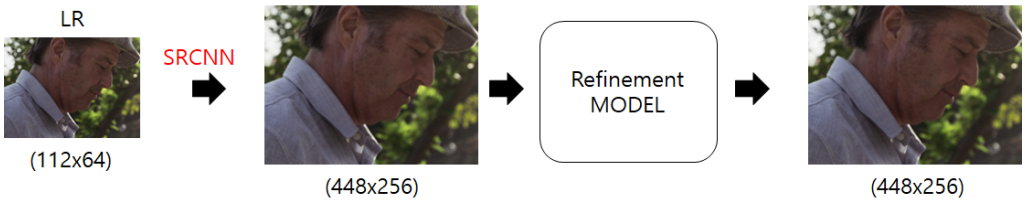

Figure 4. The proposed refinement learning method. To replace the existing bicubic upsampling, it is possible to bring improved performance and stability of the refinement module by using SRCNN to build HR required for learning.

3.6. Loss Function

The edge map of SR_t during Stage 1 training is incomplete compared to the edge map of GT frame SR_{GT}. Therefore, our model learns to minimize the difference of gradient magnitude between SR_t and SR_{GT}.

We adopt a Sobel mask as an edge detector within magnitude loss $Loss_m$. m_o and m_t are gradient magnitude of SR_t and SR_{GT}, respectively, and $Loss_m$ is defined as follows.

$$Loss_m = \sum_{i}^{H} \sum_{j}^{W} |m_o(i,j) - m_t(i,j)| \tag{7}$$

We also adopt L1 Loss $Loss_{L1}$ in Stage 1, and the the Stage 1 Loss $Loss_{stage1}$ is formed as follows. where α is a coefficient to balance the two terms.

$$Loss_{stage1} = Loss_{L1} + \alpha \times Loss_m \tag{8}$$

In Stage 2, we propose content loss, style loss, and reconstruction loss additionally, except for the existing L1 loss and magnitude loss between refinement learning, and perform optimal learning by learning the parameters together in front of each added loss formula.

First of all, $L_{content}$, the content loss, refers to the two images we want to compare with p and x, respectively, and calculates each feature map to determine the p feature representation as P_l and the feature representation of x as F_l. It is the square of the Frobenius norm for the difference.

$$L_{content}(p, x, l) = \frac{1}{2} \sum (F_l - P_l)^2 \tag{9}$$

In this case, content reconstruction can be solved by finding x that minimizes the loss using the feature map of the deep layer where only the in-depth information remains. L_{style}, the style loss, is the sum of multiplying the weights after finding the Gram matrix for the feature map and squaring the Frobenius norm. The reason why the Gram matrix is used here is that style defines the correlation between feature maps. Here too, if the two images to be compared are a and x, respectively, the values obtained by the Gram matrix for each feature map can be called A_l and G_l, and the effect E_l of the layer on the total loss is defined as Equation (10).

$$E_l = \frac{1}{4{N_l}^2 {M_l}^2} \sum (G_l - A_l)^2 \tag{10}$$

$$L_{style}(a, x) = \sum (w_l - E_l) \tag{11}$$

Therefore, the final L_{style} using E_l is the same as Equation (11), and style reconstruction can be solved by finding x that minimizes loss. The reconstruction loss, L_{rec} calculates the difference between the actual value and the predicted value without taking the softmax in the cross-entropy loss.

$$L_{Stage2} = \beta \times L_{content} + \gamma \times L_{style} + \delta \times L_{rec} \tag{12}$$

Finally, the Stage 2 loss to be minimized by using all three losses we propose is the same as Equation (12), and each given beta, gamma, and delta is a factor that adjusts the reflection weight of each loss function. In this case, these arguments are self-learned.

4. Experiment

Baseline. We use the RBPN structure as Stage 1. These differences between RBPN and our model are that our model uses edge map extraction (Section 3.1) before motion compensation and adopts $Loss_m$.

Dataset. We use Vimeo90k and REDS as the training dataset and use Vid4 and REDS4 as the test dataset for each training dataset.

Details. In Stage 1 and Stage 2, an NVIDIA GeForce RTX 2080 and NVIDIA Tesla P100 GPU are used to train respectively and the batch size is set to 2. In Stage 1, we set the hyperparameters as RBPN except for the number of channels in the convolution layer of feature extraction (Section 3.2) and batch size. In Stage 2, the learning rate is 0.0001 and the Adam optimizer is used. Convolution is based on 3×3 frame size, and both horizontal and vertical strides are set to 2. The maximum iteration is set to 2 M and early stopping is performed.

4.1. Ablation Study

We conduct the experiment while removing each method to see if the proposed methods worked. The experimental results are shown in Table 1. The batch size of each experiment is set to 2, and REDS4 is used as a test set.

Table 1. Ablation study of our method and loss function on REDS4 dataset. "Baseline" means without using proposed method and $Loss_m$. "Detect Blur" represents blur frame detection before edge map extraction. "Refinement" means proposed method in Section 3.5. $Loss_m$ is the proposed magnitude loss function.

Baseline	Detect Blur	$Loss_m$	Refinement	PSNR/SSIM
√				30.15/0.8595
√	√			30.26/0.8648
√	√	√		30.40/0.8670
√	√	√	√	32.32/0.9069

Detect blur is a method of detecting the presence of a blur area in LR_{t-k} using a Laplacian filter before extracting E_{t-k}. This method eliminates the use of e_{t-k}, which can act as noise, allowing our model to learn a complete structure feature. In the experiment, when baseline and adopting detect blur, the performance is higher than when it is not adopted. Therefore, it is proved through experiments that the blur frame acts as noise when generating E_{t-k}.

Magnitude loss $Loss_m$ is a loss function that minimizes the gradient magnitude difference between SR_t and SR_{GT}. Our model learns the structure features of SR_{GT} by minimizing the gradient difference. When $Loss_m$ is adopted, that brings about a 0.14 dB improvement on PSNR.

Incomplete contour and color correction occur in some areas of SR_t due to artifacts present in the test data. To overcome these limitations, the refinement module is adopted. When the refinement module is adopted, the contours become clear and the color is improved.

4.2. Refinement

We defined some basic conditions for systematic experiments and proceeded with step-by-step experiments. First, in the process of determining the final refinement module, Vimeo90k is used as the training dataset and Vid4 is used as the test dataset. Second, the base model layer used between experiments used 3D convolution as shown in Figure 5. The procedure for conducting the experiment is as follows. First, we verify the parameter values of "Canny edge detection", which has the best performance in our model. Second, the results according to various model structures are checked. Third, after selecting the final model, the form of input data is newly proposed and tested during training. Finally, the necessity of the refinement module is shown by applying the proposed refinement to several models. When testing the sigma value of Canny edge detection and experimenting with various model structures, the refinement module learned HR through bicubic upsampling.

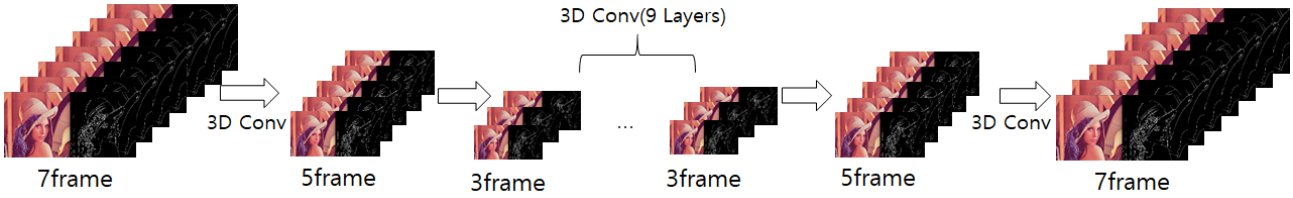

Figure 5. The architecture of the basic model used in the refinement experiments. The input data is an RGB image and the edge information is a pair. The first 7 frames are subjected to 3D convolution to reduce the channel and size. In the same way, the channel is reduced to 5 frames and then reduced to 3 frames. The data for 3 frames is 3D convolution using zero padding and goes through 9 layers without changing the size. Afterward, restore it in the opposite way to the way it was reduced.

As you can see in Table 2, we found the optimal sigma value. In "Canny edge detection", the value "Sigma" determines the strength of the edge. We test the sigma values from 0.5 to 4 on the same picture. As a result, the best performance of PSNR and SSIM are obtained at sigma = 0.7 and 1. However, when comparing the two scores, since the difference in PSNR is larger than the difference in SSIM, we conduct all subsequent experiments based on the value of 0.7.

Table 2. Quantitative results according to sigma value of Canny edge detection in refinement base models.

Sigma Value	PSNR	SSIM
0.5	26.49	0.8009
0.7	27.52	0.8178
1	27.34	0.8175
2	27.06	0.8141
3	26.65	0.8080
4	26.48	0.7980

We conduct four additional experiments on the model structure. To reduce the loss as much as possible, like the base model in Figure 6, we try to keep the 3D features of the same size without going through the compression and recovery process. We will refer to this experimental method as "VDLayer". This experiment is conducted to maintain the structural information of the data image as much as possible, and the optimal layer is found by setting the number of layers to 15, 19, and 23.

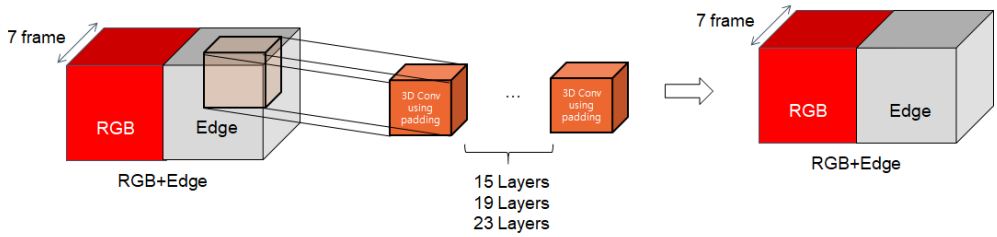

Figure 6. Using VDLayer's architecture that does not compress and recover input data, we test performance changes according to various model structures. To maintain the size when RGB + Edge pairs are input as input, zero padding is added considering the size and stride of the 3D convolution.

As you can see in Table 3, VDLayer = 19 is the highest for SSIM, but for PSNR, the existing base model performs the best. VDLayer is helpful in maintaining the structural shape of the image as expected, but learning is not smooth due to the rather large number of parameters, and accordingly, it was difficult to expect an increase in PSNR.

Table 3. Quantitative evaluation of various refinement module structures. All of the sigma values for Canny edge detection are 0.7.

Method	PSNR	SSIM
Stage1 + refinement module (BASE)	27.52	0.8178
Stage1 + refinement module (VDLayer = 15)	26.91	0.8119
Stage1 + refinement module (VDLayer = 19)	27.53	0.8180
Stage1 + refinement module (VDLayer = 23)	26.76	0.8082
Stage1 + refinement module (MVDLayers = 17)	27.54	0.8182

As can be seen in Tables 4 and 5, the combination of Stage1 to the Refinement module to which the SRCNN Refinement method was applied showed the best performance. So we select a new model structure by appropriately combining the two. Therefore, we concluded this model as the final model. In the experiment, 19 layers had the best performance. Inspired by this, we adopted a method of compressing and restoring only the first and last layers and maintaining the size of only the middle 17 layers (MVDLayers).

Table 4. Quantitative results (PSNR/SSIM) for the Vid4 test set when the refinement module is combined.

Combined Method	Stage 1	RBPN	MuCAN	EDVR
w/o Refinement	27.02/ 0.8136	27.12/ 0.8180	27.26/ 0.8215	27.35/ 0.8264
Bicubic Refinement	27.54/ 0.8182	27.15/ 0.8199	27.47/ 0.8215	27.92/ 0.8261
SRCNN Refinement	27.79/ 0.8439	27.32/ 0.8247	27.48/ 0.8312	27.76/ 0.8357

Table 5. Quantitative results (PSNR/SSIM) for the REDS4 test set when the refinement module is combined.

Combined Method	Stage 1	RBPN	MuCAN	EDVR
w/o Refinement	30.40/ 0.8670	30.09/ 0.8590	30.88/ 0.8750	31.09/ 0.8800
Bicubic Refinement	30.71/ 0.8613	30.12/ 0.8633	31.02/ 0.8771	31.01/ 0.8804
SRCNN Refinement	32.32/ 0.9069	30.34/ 0.8642	31.69/ 0.8951	31.01/ 0.8804

Vimeo90k was used as training data to test Vid4, and REDS was used as training data to test REDS4. The model trained with SRCNN shows better performance in most experiments than the refinement module learned through bicubic upsampling.

4.3. Comparison with State-of-the-Art Methods

We compared our model with a total of three models: RBPN, MuCAN, and EDVR. If you look at Tables 6 and 7, you can see the results of applying refinement (Stage 2) to the comparison model and our model. RBPN performance before using refinement in Vid4 is higher than our model performance, but lower in REDS4. This is because artifacts exist in Vid4.

Table 6. Result on Vid4. The arrows indicate how much the model we finally propose has risen compared to when only Stage 1 was used. All show a consistent rise.

	SEAN	**RBPN**	**MuCAN**	**EDVR**
PSNR	27.79 (0.24↑)	27.32 (0.19↑)	27.48 (0.45↑)	27.76 (0.46↑)
SSIM	0.8439 (0.0257↑)	0.8247 (0.0052↑)	0.8312 (0.0153↑)	0.8357 (0.0099↑)

Table 7. Result on REDS4. The arrows indicate how much the model we finally propose has risen compared to when only Stage 1 was used. All show a consistent rise.

	SEAN	**RBPN**	**MuCAN**	**EDVR**
PSNR	32.32 (1.61↑)	30.34 (0.23↑)	31.70 (0.82↑)	31.01 (0.30↑)
SSIM	0.9069 (0.0456↑)	0.0.8642 (0.003↑)	0.8951 (0.0201↑)	0.8804 (0.0078↑)

Similar to our model, an edge map was used to edge-aware loss in MuCAN, where the performance growth rate after using the refinement in Vid4 is lower than that in REDS4. The performance is also lower than EDVR performance in Vid4, but higher in REDS4.

Additionally, if you look at Figure 7, you can see the results before and after applying the Refinement module. Our model's and MuCAN's output became clearer than the output without the refinement applied, but in the case of RBPN, a blur area was created. From this point of view, it can be said that the edge map created in Vid4 was applied as noise when training Stage 1 of our model, and the edge map created in REDS4 acted as effective information.

Similar to the training of Stage 1 of our model, the edge map of each frame was used when training the refinement module, and when the experiment was conducted by applying it to the comparative model and our model, our model performance improved the most. In addition, if you look at Figures 8 and 9, it can be seen that the contour of the image generated by our model in Vid4 and REDS4 has become clear and the color has also improved compared to the comparative models. Therefore, the refinement module proved that it is effective in our model methodologically and experimentally.

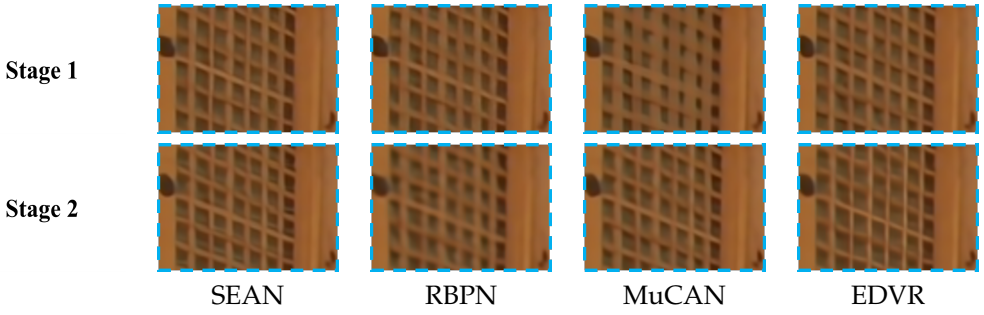

Figure 7. Comparison of applying the Stage 2 (refinement) module on REDS4. The results in the first line have not applied Stage 2, and the results in the second line have applied Stage 2. In the case of the results of the model trained using structure feature, the results are improved, and in the case of the model not used, a blur area is formed.

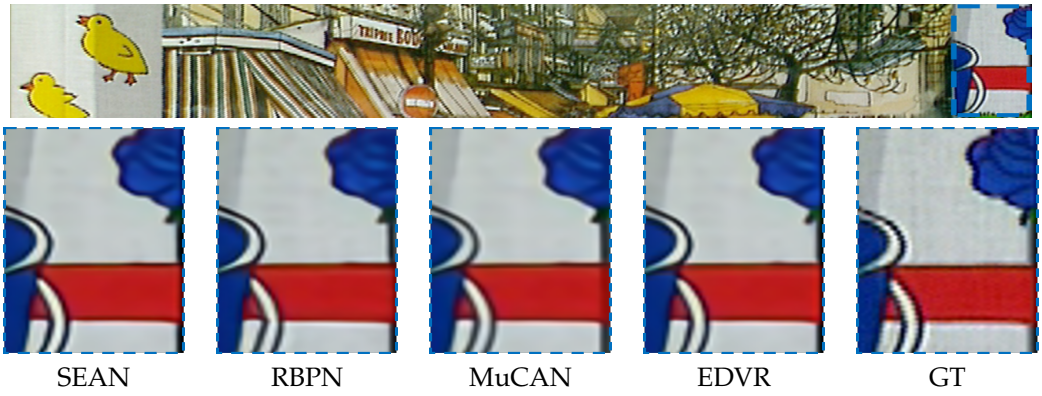

Figure 8. Visual results on Vid4 for 4× scaling factor.

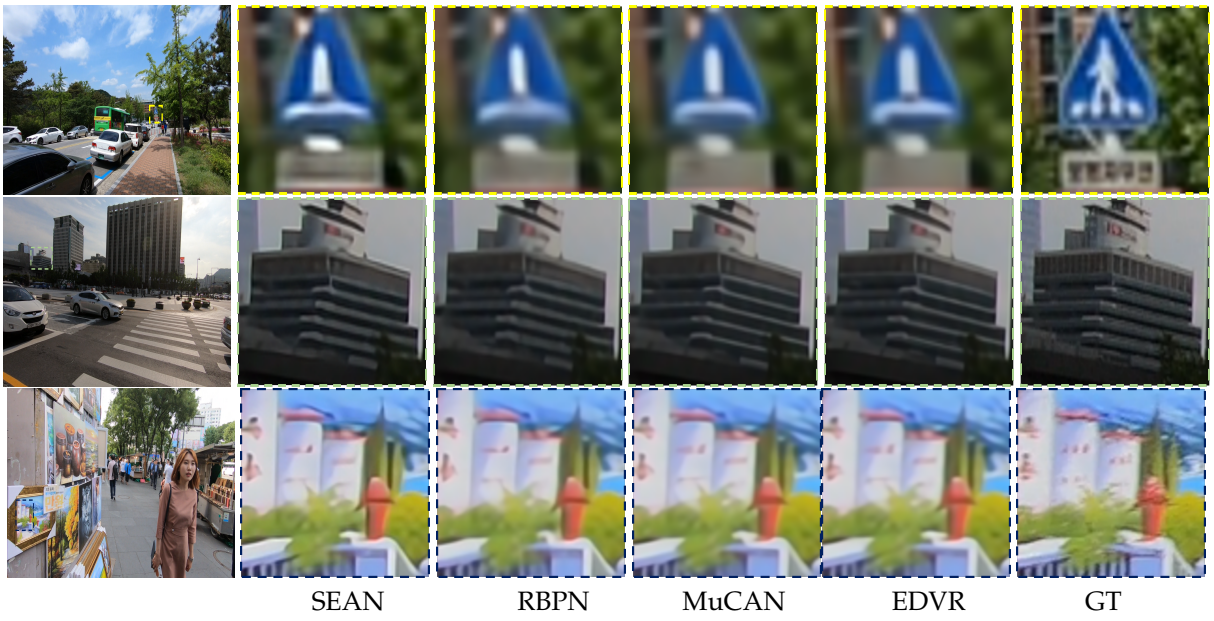

Figure 9. Visual results on REDS4 for 4× scaling factor.

Table 8 shows the quantitative performance values of the final model we propose.

Table 8. Final result.

	SEAN	RBPN	MuCAN	EDVR
Vid4	27.79/0.8439	27.12/0.8180	27.26/0.8215	27.35/0.8264
REDS4	32.32/0.9069	30.09/0.8590	30.88/0.8750	31.09/0.8800

5. Conclusions

We add structural information by using an edge map in this study and use detect blur to minimize noise before adding information. In addition, the difference between output structural information and GT structural information generated by using magnitude loss was minimized, and a refinement module was used to improve the problem caused by the artifact of the test set. We found the optimal refinement module through experiments and showed consistent performance improvement. Among any other models, when combined with the Stage 1 model we propose, the performance improvement was the greatest. Therefore, we finally propose this model.

Author Contributions: Conceptualization and methodology, S.K. and Y.-M.S.; investigation, S.K.; data curation and conceiving experiments, Y.-M.S.; performing experiments and designing results, S.K. and Y.-M.S.; writing—original draft preparation, S.K. and Y.-M.S.; writing—review and editing, S.K., Y.-M.S. and Y.-S.C.; supervision, Y.-S.C.; funding acquisition, Y.-S.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2017-0-01642) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation and supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(*MSIT) (No.2018R1A5A7059549) and supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(*MSIT) (No. 2020R1A2C1014037) and supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. 2020-0-01373, Artificial Intelligence Graduate School Program (Hanyang University)) *Ministry of Science and ICT.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Xue, T.; Chen, B.; Wu, J.; Wei, D.; Freeman, W.T. Video enhancement with task-oriented flow. *Int. J. Comput. Vis.* **2019**, *127*, 1106–1125. [CrossRef]
2. Haris, M.; Shakhnarovich, G.; Ukita, N. Recurrent back-projection network for video super-resolution. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, 16–20 June 2019; pp. 3897–3906.
3. Liu, C.; Sun, D. A bayesian approach to adaptive video super resolution. In Proceedings of the CVPR 2011, Colorado Springs, CO, USA, 20–25 June 2011; pp. 209–216.
4. Li, W.; Tao, X.; Guo, T.; Qi, L.; Lu, J.; Jia, J. Mucan: Multi-correspondence aggregation network for video super-resolution. In *Lecture Notes in Computer Science, Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020*; Springer: Cham, Switzerland, 2020; pp. 335–351.
5. 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, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 1646–1654.
6. Dong, C.; Loy, C.C.; He, K.; Tang, X. Image super-resolution using deep convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *38*, 295–307. [CrossRef] [PubMed]
7. Nazeri, K.; Thasarathan, H.; Ebrahimi, M. Edge-informed single image super-resolution. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, Seoul, Korea, 27 October–2 November 2019.
8. 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; Honolulu, HI, USA; 21–26 July 2017; pp. 4681–4690.
9. 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; Seattle, WA, USA; 14–19 June 2020; pp. 3360–3369.
10. 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, Long Beach, CA, USA, 16–20 June 2019.
11. 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, Venice, Italy, 22–29 October 2017; pp. 764–773.
12. Haris, M.; Shakhnarovich, G.; Ukita, N. Space-time-aware multi-resolution video enhancement. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA; 14–19 June 2020; pp. 2859–2868.
13. Park, W.S.; Kim, M. CNN-based in-loop filtering for coding efficiency improvement. In Proceedings of the 2016 IEEE 12th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP), Bordeaux, France, 11–12 July 2016; pp. 1–5.
14. Dai, Y.; Liu, D.; Wu, F. A convolutional neural network approach for post-processing in HEVC intra coding. In *Lecture Notes in Computer Science, Proceedings of the International Conference on Multimedia Modeling, Reykjavik, Iceland, 4–6 January 2017*; Springer: Cham, Switzerland, 2017; pp. 28–39.

15. Kang, J.; Kim, S.; Lee, K.M. Multi-modal/multi-scale convolutional neural network based in-loop filter design for next generation video codec. In Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 17–20 September 2017; pp. 26–30.
16. Wang, T.; Chen, M.; Chao, H. A novel deep learning-based method of improving coding efficiency from the decoder-end for HEVC. In Proceedings of the 2017 Data Compression Conference (DCC), Snowbird, UT, USA, 4–7 April 2017; pp. 410–419.
17. Ma, L.; Tian, Y.; Huang, T. Residual-based video restoration for HEVC intra coding. In Proceedings of the 2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM), Xi'an, China, 13–16 September 2018; pp. 1–7.
18. Liu, C. Beyond Pixels: Exploring New Representations and Applications for Motion Analysis. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA USA, 2009.
19. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F.-F. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
20. Timofte, R.; De Smet, V.; Van Gool, L. Anchored neighborhood regression for fast example-based super-resolution. In Proceedings of the IEEE International Conference on Computer Vision, Sydney, NSW, Australia, 1–8 December 2013; pp. 1920–1927.
21. Yang, J.; Wright, J.; Huang, T.S.; Ma, Y. Image super-resolution via sparse representation. *IEEE Trans. Image Process.* **2010**, *19*, 2861–2873. [CrossRef] [PubMed]
22. Nah, S.; Baik, S.; Hong, S.; Moon, G.; Son, S.; Timofte, R.; Lee, K.M. NTIRE 2019 Challenge on Video Deblurring and Super-Resolution: Dataset and Study. In Proceedings of the CVPR Workshops, Long Beach, CA, USA, 16–20 June 2019.