Video Upscaling Using Deep Neural Network

Aditya Rathi, Jaydutt Kulkarni, Ruchita Sinha, Vatsal Joshi

Abstract

Video super-resolution has recently become one of the most important problems due to the rise of video communication and streaming services. While many solutions have been proposed for this task, the majority of them are too computationally expensive to run on portable devices with limited hardware resources. We propose the upscaling of a lower resolution input video stream to one that is at a significantly higher resolution while maintaining visual fidelity using deep learning networks. The aim is to have the network contextually interpolate the video frame producing a larger output of a higher resolution. Using still image training data, we explore different network architectures and hyperparameter combinations for the model that gives the best results. Testing our solution on both images and videos, and comparing to recently developed techniques, we observe promising results. Our tests show comparable metrics to a modern Generative Adversarial Network (GAN) based architecture with limited training on a specialized dataset.

1. Introduction

Single image super-resolution (SISR), as a fundamental low-level vision problem, has attracted increasing attention in the research community and AI companies. SISR aims at recovering a high-resolution (HR) image from a single low-resolution (LR) one. The key enhancements that should be offered by a super-resolution deep neural network are superior image quality and lower storage requirements. For applications that involve rendering frames such as video games, the network should offer image quality comparable to native resolution while only having to render one-quarter to one-half of the pixels, resulting in significantly higher performance without loss in visual fidelity. For applications that do not involve rendering such as network-based video streaming, this can result in decreased video usage as the footage can be transmitted at a lower resolution and then locally upscaled on the client device. Video files can also be stored in lower resolutions, saving space.

There needs to be training data, where low-resolution images are provided to the deep learning model, while comparing the output of the model to ground truth, i.e., higher resolution versions of the same images. There also needs to be validation and test data sets for benchmarking of the neural network. We plan to generate video footage of our own and compare it to an existing dataset[1], with respect to the performance of the model achieved after training. We also plan to test our model to understand its generalizability in terms of domain-specific training sets and performance outside the domain.

We plan to change the network architecture given in Dong et al.[2] to achieve a good balance between visual fidelity and the performance of the network. We also plan to tweak the hyper-parameters of the model to see what affects the performance the most and find an optimum solution in the Pareto set. We plan to compare the trained model with the results obtained using bicubic interpolation. The videos obtained from our deep learning architecture will be displayed alongside the video obtained from the original method. Furthermore, the results obtained from our model will be benchmarked using standard performance techniques such as Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index (SSIM).[3]

2. Related Work

There have been several works published recently that use deep learning for the super-sampling of input images. Since the pioneer work of Super Resolution CNN (SRCNN) proposed by Dong et al.[2], deep convolution neural network (CNN) approaches have brought prosperous development. SRCNN extracts feature maps which are then mapped nonlinearly to high-resolution patch representations. The features are extracted using convolution layers with different layer sizes. Multiple convolution layers are stacked in conjunction with a non-linear activation function to extract features at multiple scales from the original input image. The features are then mapped to create a high-resolution map of the features from the original image. The final layer averages the high-resolution map and combines it with the spatial neighborhood to generate the final image.

Huang et al.[4] showcases comparisons of deep-learning-based networks with traditional techniques such as bicubic interpolation. Lundkvist[5] shows how humans perceive these super-resolution outputs and shows a clear preference of people towards CNN-based approaches. Watson[6] shows a practical application of these techniques as used by the NVIDIA Deep Learning Super-Sampling (DLSS) network for super-resolution in video games.

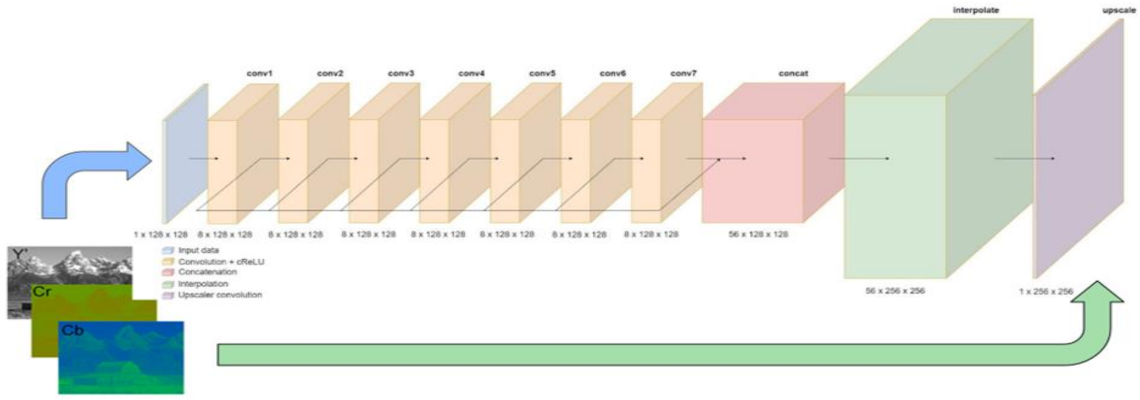

Figure 1: Neural Network Architecture

Finally, Dong[7] introduces a modern reference architecture that showcases very promising results on low-resolution input images. These readings showcase clearly that CNN-based approaches are significantly better than traditional upscaling approaches both quantitatively and qualitatively; these approaches are being used by real-world applications and also provide reference models on which to base our strategy.

3. Methodology

Our aim is to improve the visual fidelity of a low-resolution image while maintaining adequate performance for a real time framework. Towards this end, in this section, we describe the network architecture, explain our pipeline, and introduce the loss function that we have used for our method.

3.1. Pipeline

- The first step of preprocessing data (before training) is the color space conversion. Real-time images and videos are stored in RGB color space, because it is based on the sensitivity of color detection cells in the human visual system. In digital image processing the YCbCr (the luminance Y, the blue difference Cb and the red difference Cr) color space is often used in order to take advantage of the lower resolution capability of the human visual system for color with respect to luminosity. Thus, RGB to YCbCr conversion is widely used in image and video processing[8].
- The next step involves sending the luminance component of the converted image vector, Y, to the neural network for upscaling. The neural network architecture is explained in Section 3.2.
- In order to upscale within the neural network, we need some way to go from a lower resolution image to a higher one. We generally do this with the transposed convolution operation. Roughly, transposed convolution layers allow the model to use every point in the small image to "paint" a square in the larger one. Unfortunately, deconvolution can easily have "uneven overlap," putting more of the metaphorical paint in some places than others[9]. In particular, deconvolution has uneven overlap when the kernel size (the output window size) is not divisible by the stride (the spacing between points on the top). The overlap pattern also forms in two dimensions. The uneven overlaps on the two axes multiply together, creating a characteristic checkerboard-like pattern of varying magnitudes. In order to overcome these artifacts, we use nearest-neighbor interpolation for upscaling instead.
- A final convolution layer is used after the interpolation operation to combine all the channels.
- The processed Y-channel is combined back with b and Cr using bicubic interpolation on the chroma channels.

3.2. Network Architecture

The neural network consists of 8 convolution layers, as shown in Fig. 1. The Y-channel of the image is passed through each of these layers, which combines convolution with a concatenated-ReLU operation to preserve the shape of the image vector. The output from each convolution operation is concatenated at the end to form a vector with 7-time the number of channels. This is then passed through an interpolation operation to upscale the image. The final convolution layer is used to reduce the number of channels to 1. The training loss is then calculated between this generated batch of images and the ground truth in order to train the model.

3.3. Content Loss

One vital point to keep in mind during upscaling is that you also need to preserve the content of an along with the desired style. Mean Squared Error (MSE) doesn't express the human perception of image fidelity well. For example, there can be distorted images that are equally distant from the original image in terms of MSE, but they don't look equally good. This is because MSE cares only about pixel-wise intensity differences, but not the structural information about the contents of an image.

So instead of using the naïve MSE loss function between prediction and ground truth, we construct a loss function that is akin to a feature map. If you visualize what is learnt by a neural network, there's evidence that suggests that different feature maps in higher layers are activated in the presence of different objects. If two images to have the same content, they should have similar activations in the higher layers. A preset layer of a pretrained model on ImageNet (in this case, the 34th layer of VGG-19 as is seen in Fig. 2) is taken, and the difference between feature maps produced by that layer is computed. The difference between the feature maps can be minimized to train another model, just like any other loss function. The layers that generate those feature maps stay frozen during training and act as a fixed feature extractor. This loss is commonly referred to as content loss, which can be expressed mathematically as follows:

$$L_{content} = \frac{1}{2}\sum_{i,j}(A^{l}_{ij}(g) - A^{l}_{ij}(c))^2$$

where $A^{l}_{ij}(X)$ is the activation of the lth layer, ith feature map, and jth position obtained using the image X.

Essentially, content loss captures the root MSE between the activations produced by generated image and content image. We can use this loss function to train a model to enhance images and get good results. Comparatively,

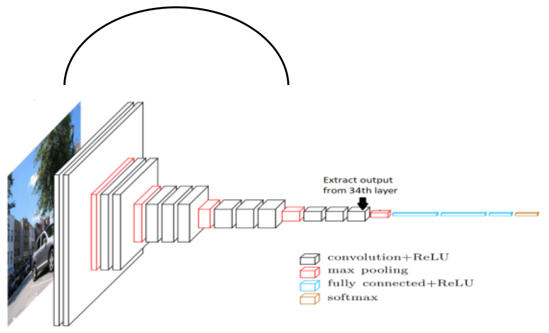

Figure 2: VGG-19 network architecture[10]

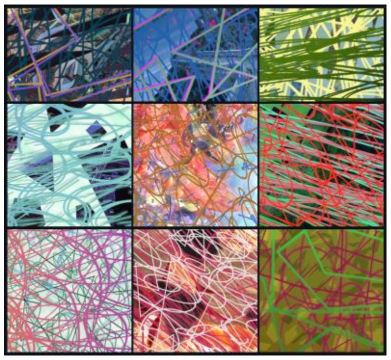

Figure 3: SYNLA+ Dataset training samples[11]

content loss preserves high frequency features better and produces better "visually pleasing" images.

4. Experiments

We conducted a number of trials with various loss functions, hyperparameters, and neural network architectures before settling on a combination that worked best for the Anime test images.

4.1. Dataset

Our focus was on upsampling Anime-themed media which uses line art. The Synthetic Line Art + (SYNLA+) dataset is designed to simulate complex line art[11]. It is useful for training machine learning models which performs tasks such as deblurring, super-resolution, denoising, artifact removal, etc. Most line art are licensed and have copyright. This dataset offers an open alternative and is released under MIT license. The full dataset contains 65536 (2^{16}) images of size 256 x 256. A smaller preview dataset (4096 images: SYNLA-4096) was used as the training dataset, while another mutually exclusive dataset was created for validation (1024 images: SYNLA-1024).

The training images have backgrounds that are real images - scenes from different Anime or even normal pictures. As shown in Fig.3, the images have lines drawn on top of them, which helps the neural network learn to maintain fidelity of line samples. The focus on animated image upscaling which is heavy in complex line art making this the perfect training dataset.

4.2. Training Details

All experiments are performed with a scaling factor of x2 between low resolution (LR) and high resolution (HR) images. We obtain LR images by down-sampling HR images using bicubic interpolation. The mini-batch size is set to 96. The spatial size of cropped HR patch is 128 x 128. We observe that training a deeper network benefit

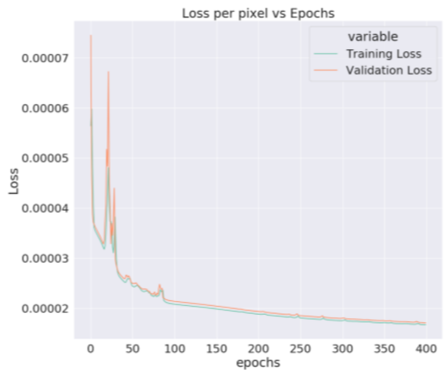

Figure 4: Loss per pixel vs Epochs for training and validation

from a larger patch size, since an enlarged receptive field helps to capture more semantic information. However, it costs more training time and consumes more computing resources.

The neural network constructed as shown in Fig. 1 is trained using a loss function. The learning rate is initialized as 1×10^{-4} and decayed by a factor of 10 after the learning rate plateaus. This is achieved using the ReduceLROnPlateau learning rate scheduler. The training loop is run for 400 epochs. The number of epochs is determined using a convergence criterion. It was determined that the model starts overfitting the training data after 400 epochs. The validation loop is run on the mutually exclusive dataset, SYNLA-1024. We evaluate our model on a widely used benchmark – Set14.

Initially, pixel-shuffle is used for upscaling, which results in checkerboard artifacts in the output images. In order to overcome this limitation, we changed from a pixel shuffle function to a nearest neighbor interpolation for upsampling. This overcomes the artifact limitation and results in a much better, generalizable model. This can be verified by visualizing loss per pixel in the training and validation sets against the number of epochs, as shown in Fig. 4.

For optimization, we use Adam optimizer with $\beta 1 = 0.9$, $\beta 2 = 0.999$. We implement our models with the PyTorch framework and train them using NVIDIA RTX 3070 GPU[1].

5. Results

The term peak signal-to-noise ratio (PSNR) is an expression for the ratio between the maximum possible value (power) of a signal and the power of distorting noise that affects the quality of its representation. We compare our final model output with the HR version of the test images. We also compare our model with the state-of-the-art (SOTA) PSNR-oriented methods as well as perceptual-driven approaches such as ESRGAN[12].

Table 1: Quantitative comparison between different neural network models using PSNR

PSNR	**Set14 Baboon**	**Set14 Face**
Our Method	27.99	27.75
ESRGAN	20.35	30.50

Set14 is a standard set of images used to evaluate SR models. It consists of 14 diverse images on which PSNR is calculated for comparison. Fig. 5 shows a comparison between the up-scaled image and HR image for one of the images in Set 14. As can be seen clearly, the neural network does a great job of up-scaling the LR image, as can also be seen in the PSNR table.

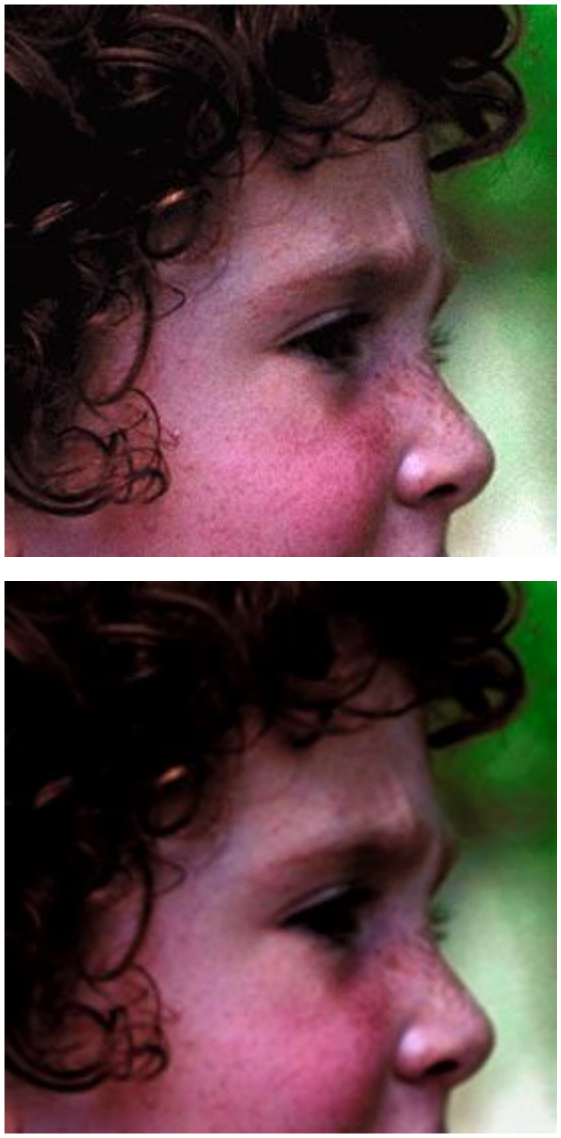

Figure 5: (Top) – Up-sampled Set14 image vs (Bottom) – Ground Truth HR image

[1] GitHub code can be found at github.com/aditya-rathi/Video-Upscaling

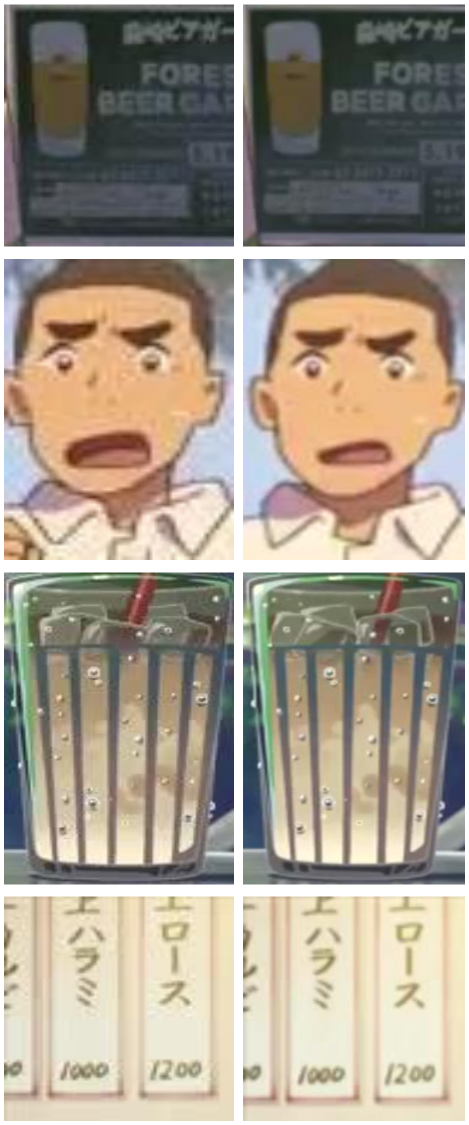

Figure 6: 500% crop of still frames from test videos, where left column represents the LR images, while the right column represents the HR images generated by our model

Since there is no effective and standard metric for perceptual quality, we present some representative qualitative result. As shown in Table 1, our model performs better than ESRGAN on some images, while it performs worse on others. Our model is trained on SYNLA-4096 which is very specific for line art while most models are trained on ImageNet.

As can be seen in Fig. 6, the model successfully converts 360p LR images into 720p HR images. This can be better demonstrated by looking at particular details of an image rather than the image as a whole. The same concept has also been applied to Anime videos. Using PIL python library, we have converted a video into image frames and run our super-sampling model over these images. Once the upscaled images have been reassembled back into a video, we can compare this with the ground truth HR video and determine the quality of the neural network output.

6. Conclusion and Future Work

In this project, we have implemented and trained a deep learning neural network model for super-resolution using bicubic interpolation, and evaluated this model compared to video upscaled using ESRGAN, by using PSNR metric. Our results show that our method is better than ESRGAN for some images, and worse for others. Also, the time required for training and HR image generation is considerably less for our model. We conclude that usage of deep learning for video scaling provides increased video quality, flexibility, and the trade-off nature of reduced storage space against difference in quality is worthwhile.

Moving ahead, the scope of work is pretty broad. One way to improve output quality is to implement a GAN in conjunction with the interpolation mechanism. Pre-training with pixel-wise loss helps GAN-based methods to obtain more visually pleasing results. The reasons are that 1) it can avoid undesired local optima for the generator; 2) after pre-training, the discriminator receives relatively good super-resolved images instead of extreme fake ones (black or noisy images) at the very beginning, which helps it to focus more on texture discrimination.

Another way forward would be to construct different loss functions to understand the impact of weighing certain aspects of an image more than others. The full SYNLA+ dataset has over 65k images, and using all of them to train the model could result in a more robust network. We could also look into using computer vision techniques such as non-local mean filter or Gaussian blurring to augment the training dataset for better results. Also, in order to reduce training and evaluating time for the neural network due to interpreter/compiler lag time, we could explore the usage of OpenGL shaders. A shader is a piece of code that is executed on the Graphics Processing Unit (GPU), usually found on a graphics card, to manipulate an image before it is drawn to the screen. They allow for various kinds of rendering effect.

References

[1] D. Martin, C. Fowlkes, D. Tal and J. Malik, "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics", ICCV, July 2001.

[2] Dong, C., Loy, C.C., He, K., Tang, X.: Learning a deep convolutional network for image super-resolution. In: ECCV. (2014)

[3] Ignatov, Andrey, et al. "Real-time video super-resolution on smartphones with deep learning, mobile ai 2021 challenge:

Report." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021

[4] S. Huang and J. Xie, "Pearl: A Fast Deep Learning Driven Compression Framework for UHD Video Delivery," ICC 2021 - IEEE International Conference on Communications, 2021, pp. 1-6, doi: 10.1109/ICC42927.2021.9500754.

[5] Lundkvist, Fredrik. "Deep upscaling for video streaming: a case evaluation at SVT." (2021).

[6] Watson, Alexander. "Deep learning techniques for super-resolution in video games." arXiv preprint arXiv:2012.09810 (2020).

[7] Dong, Chao, et al. "Image super-resolution using deep convolutional networks." IEEE transactions on pattern analysis and machine intelligence 38.2 (2015): 295-307.

[8] Keith Jack. Video demystified: a handbook for the digital engineer. Elsevier, 2011.

[9] Conditional generative adversarial nets for convolutional face generation Gauthier, J., 2014. Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester, Vol 2014.

[10] https://www.techleer.com/articles/305-vgg-16-an-advanced-approach-towards-accurate-large-scale-image-recognition/

[11] https://github.com/bloc97/SYNLA-Plus

[12] Wang, Xintao, et al. "Esrgan: Enhanced super-resolution generative adversarial networks." Proceedings of the European conference on computer vision (ECCV) workshops. 2018.