Multi-Frame Video Super-Resolution Using Convolutional Neural Networks

Alex Greaves
Stanford University
450 Serra Mall, Stanford, CA 94305
alexg343@stanford.edu

Hanna Winter
Stanford University
450 Serra Mall, Stanford, CA 94305
hannawii@stanford.edu

Abstract

Video super-resolution remains a challenging problem despite being a very active area of research. Even with huge strides made with single-image super-resolution, multi-frame techniques, which utilize multiple frames in improving the quality of a given frame, have yet to fully take advantage of the power of deep learning. We propose a HR for multi-frame super-resolution that outputs a higher resolution version of a given frame using pixel information from adjacent frames in the video. Unlike traditional multi-frame techniques, our proposed method has minimal data pre-processing and computation cost. Using both still image and video training data we explore different network architectures and hyperparameter combinations for both the single and multi-frame model that give the best results. Testing our solution and comparing to recently developed techniques, we observe promising results.

1. Introduction

Multi-Frame super-resolution (MFSR) is the process of taking multiple low-resolution (LR) video frames and constructing a single high-resolution (HR) video frame. A solution to this problem is a solution to the more general problem of video super-resolution (VSR), which is defined simply as improving the resolution of a given video. It has potential to help overcome the difficulties of many important tasks such as analyzing surveillance camera footage and improving low-quality cell phone videos. Autonomous vehicle technology would also benefit from success in this area. However, super-resolution (SR) is generally considered an ill-posed problem because much of the LR image pixel data was lost during the down-sampling procedure or simply not there to begin with. Thus, recovering the high-frequency details from the minimal data available requires effective data modeling and heavy inference based on common underlying properties of the natural world. Additionally, noise, motion, and blur in videos create even more challenging problems when trying to construct HR video. Thus, VSR remains a difficult and unresolved problem.

On the other hand, single image super-resolution (SISR) methods, a similar process where a HR image is created from the information given from a LR image, have progressed significantly. SISR methods have improved upon early techniques that use image priors [4] by utilizing sparse-coding techniques [6], [14], [17]. Some of the state-of-the-art techniques for SISR include methods that expand upon sparse-coding.

More recently, methods in the machine learning domain have found great success trying to tackle the SISR problem. Preliminary attempts with neural networks inspired further research using deep learning. Multi-layer cascading auto-encoders were used to match similar LR/HR pairs [2]. In an effort to take advantage of sparse-coding, experimentation with embedding sparse-coding in the layers of a neural network [15] has proven successful. Finally, traditional deep HRs (CNN) have been utilized to reflect sparse-coding and learn the non-linear mapping between an LR image and its HR counterpart [3].

We believe these methods developed for SISR can prove to be even more useful for video super-resolution (VSR). While work has been done to tackle VSR with the help of draft-ensembles and CNNs [8], many attempt to tackle VSR by dealing specifically with motion estimation [10] or using bayesian methods [9] and do not generalize well. We hope to implement a purely data driven approach to this problem using methods similar to those used for SISR.

In our approach, we utilize successful techniques from SISR to perform super-resolution using information from multiple frames. Specifically, we take advantage of Dong *et al.*'s [3] SRCNN and model a similar single-frame CNN. Our single-frame CNN takes as input a single image with 3 color channels each and outputs a HR version of the input. After we train and tune this model and achieve competitive results for SISR, we tweak this model to perform single frame super-resolution (SFSR), which differs only in that the data used to train it is now single frames instead of static images. Finally, from our SFSR model we create a model for VSR. This multi-frame CNN is initialized

with pre-trained weights from our single-frame CNN. It receives as input a set of video frames with 3 color channels and outputs a set of corresponding HR frames. To do this, for each frame of the input, our multi-frame model uses information from the frames before and after to construct a HR version. In this way, each LR frame is mapped to an HR frame based on more pixel information from surrounding frames. We find that our proposed multi-frame CNN out-performs other modern multi-frame SR methods while requiring little pre-processing.

2. Related Work

2.1. Image Super-Resolution

Generic image SR techniques utilize image priors to generate a HR version of a single LR image input. These various algorithms that use image priors can be placed into four categories: prediction models, edge-based methods, image statistical methods, and patch-based (or example-based) methods. The work of Yang *et al*. [16] thoroughly tests and evaluates most of these methods. Example-based methods such as [4], [6], [14] seemed to produce the most promising results.

Internal example-based techniques utilize the observation that images have recurrent information over sub-pixel misalignments of image patches. Using these different patches from the same image allows the usage of classical multi-image SR [4], wherein multiple versions of the same image are used in combination to improve resolution. The external example-based methods such as [14], [17] train dictionaries to learn a mapping from LR to HR image patches. The nearest neighbour (NN) of the input image patch is found among the LR patches in the dictionary and its corresponding HR image patch is used to reconstruct the input. Yang *et al*. [17] improves the above technique by using a sparse coding formulation to replace the above NN strategy. Experimentation with better mapping functions such as kernel regression [6] and anchored neighborhood regression [14] has increased the speed and accuracy of the traditional example-based techniques. However, by far the most successful, state-of-the-art example-based SISR techniques expand and improve upon sparse-coding methods.

2.2. Deep Learning for Image Super-Resolution

More recent SISR techniques have taken advantage of deep learning methods. However, these methods still utilize the findings from generic SISR research. Cui *et al*. [2] uses the internal example-based method [4] to generate self-similar image patches. These image patches are used as input to each layer of a cascading multi-stacked network of collaborative auto-encoders. Sparse-coding has also been used in various deep learning techniques. Each layer in the feed-forward neural network from [15] corresponds to a step in the sparse-coding process, making the sparse-coding process integrated with the network. More recently, Dong *et al*. [3] creates a fully HR and shows the established relationship between its structure and sparse-coding-based SR methods. However, unlike [2], this CNN provides an end-to-end mapping between LR and HR images and requires little pre or post-processing.

2.3. Multi-Frame Super-Resolution

Mutli-frame super-resolution (MFSR) research has not advanced nearly as much as SISR. Most MFSR techniques involve motion estimation and noise cancellation algorithms. Ma *et al*. [10] proposed a system to estimate uniform motion blur over multiple frames and produce a HR frame. However, this solution is extremely slow requiring 20+ minutes to construct a single x4 SR frame. Liu *et al*. [9] used a Bayesian adaptive VSR approach that iteratively estimates optical flow, noise level, and blur kernel estimation in the maximum a posterior (MAP) manner. To improve upon this method, [8] proposed a non-iterative framework for MFSR using deep draft-ensemble learning. This strategy first generates a draft-ensemble from the input LR frames sequence and uses a CNN to determine the optimal draft. One of the few methods to use CNNs for MFSR, this framework reduces the majority of computation seen in [8], where each step must solve a set of nonlinear functions.

However, few to none of these MFSR techniques seem to utilize the algorithms from SISR research. In our approach, we generalize a single-frame CNN model [3] to perform MFSR.

3. Methods

3.1. Training Objective and Evaluation

For SISR, VSR, and regression problems in general, the most commonly used objective function is the Mean Squared Error (MSE) loss, at least somewhat due to its simplicity and easily interpretable gradient, but mostly due to its effectiveness in practice. As the name implies, this function measures the average squared pixel difference between the two images across all of the RGB channels. The mathematical descriptions of MSE and its gradient are described in equation 1, where we let H and W be the image height and width, and $\hat{Y} \in \mathbb{R}^{H \times W \times 3}$ be the predicted RGB image (the output of the last layer of the network).

$$\begin{aligned} \mathbf{MSE}(\hat{Y}, Y) &= \frac{1}{2}\frac{1}{3HW}\sum_{i=1}^{H}\sum_{j=1}^{W}\sum_{k=1}^{3}(\hat{Y}_{ijk} - Y_{ijk})^2 \\ (\nabla_{\hat{Y}}\mathbf{MSE})_{ijk} &= \frac{1}{3HW}(\hat{Y}_{ijk} - Y_{ijk}) \end{aligned} \tag{1}$$

Hence, with the $1/2$ factor in front of the MSE, the gradient is simply the difference between the output image of

the network and the original image, normalized by the total size of the image. While this makes for an excellent loss function, when it comes to images MSE does not directly translate into an easily interpretable metric for reconstruction quality. For this reason, within the image quality community, the most common evaluation metrics used are Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index Measurement (SSIM), which are described in equation 2. These two serve as quantitative evaluation metrics that are more easily interpretable.

$$\begin{aligned}\textbf{PSNR}(\hat{Y}, Y) &= 20\log(s) - 10\log\textbf{MSE}(\hat{Y}, Y) \\ \textbf{SSIM}(\hat{Y}, Y) &= \frac{(2\mu_{\hat{Y}}\mu_Y + c_1)(2\sigma_{\hat{Y}Y} + c_2)}{(\mu_{\hat{Y}}^2 + \mu_Y^2 + c1)(\sigma_{\hat{Y}}^2 + \sigma_Y^2 + c2)}\end{aligned} \tag{2}$$

where s is the maximum possible pixel value (255 in our case), μ_Y denotes the mean of image Y, σ_Y^2 the variance, $\sigma_{\hat{Y}Y}$ the covariance of the two images, and $c1, c2$ constants typically set to $0.01s^2$ and $0.03s^2$, respectively [1]. In practice, SSIM is computed over several windows of an image and reported as the average value over all windows. Generally, PSNR gives a metric of image quality that scales more closely to our subjective experience than MSE, but SSIM often serves as a better evaluation metric than PSNR. Hence, when hyper-tuning parameters, we used mean SSIM of the validation set to select our best models.

3.2. Framework

3.2.1 Network Architecture

Images and videos come in many different sizes and resolutions, so we seek to create a model that can handle any type of input. In addition, because we want our model to serve as a better alternative to conventional methods of resizing images, we need to structure it such that it can increase the resolution by an arbitrary factor. However, the nature of CNNs make it difficult to create layers that can scale their input to an arbitrary size. With these goals in mind, as well as this restriction, we elected to take an approach similar to [3] wherein we create a network that takes in the input of an image already upscaled using bicubic interpolation, and outputs a higher quality output (see figure [1]). In order to generalize to any sized input, the network is fully convolutional, and because the input and outputs have the same size, we can set it to have any number of layers.

In our approach, we considered a class of networks with up to 9 convolutional layers, each with a ReLu activation function, and employing dropout for training. In addition, we performed L2 regularization for each weight matrix W, which limits the size of the weights at each layer by adding a term to the loss equal to some hyperparameter $lambda$ times the sum of the squares of each weight in the weight matrix. We describe these in equation 3, where we let x be the output of a single neuron in the network and p the dropout probability.

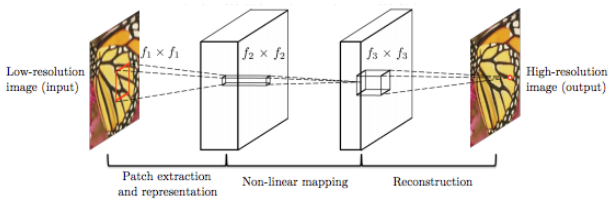


Figure 1: The architecture of SRCNN as presented in [3]. We use this architecture for our single-image CNN.

$$\begin{aligned}&\textbf{ReLu}(x) = \max(0, x) \\ &\textbf{Dropout}(x, p) = \begin{cases} x, & \text{with prob. } p \\ 0, & \text{with prob. } 1-p \end{cases} \\ &\textbf{L2}(W) = \lambda \|W\|_2^2\end{aligned} \tag{3}$$

3.2.2 Training Algorithms

We experimented with both Momentum [11] and Adam updates [7], which are two common algorithms to perform parameter updates based on the gradient of the loss function. With momentum update, parameters experience an increasing velocity in the directions of consistent gradients. Equation 4 describes this update, where X_t is our parameter matrix at iteration t, $\gamma \in [0, 1)$ is our momentum hyperparameter (commonly set to 0.9), v_t is the velocity vector at iteration t, and α is the learning rate.

$$\begin{aligned}v_t &= \gamma v_{t-1} - \alpha \nabla X_{t-1} \\ X_t &= X_{t-1} + v_t\end{aligned} \tag{4}$$

Equation 5 shown below illustrates the Adam update, which improves upon RMSProp [5] by combining it with the momentum update.

$$\begin{aligned}m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla X_{t-1} \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)\nabla (X_{t-1})^2 \\ X_t &= X_{t-1} - \frac{\alpha m_t}{\sqrt{v_t + \epsilon}}\end{aligned} \tag{5}$$

Here, $\beta_1, \beta_2 \in [0, 1)$ and ϵ are are hyperparameters commonly set to 0.9, 0.999, and $1e^{-8}$ respectively, m_t is the momentum vector at iteration t, v_t is the velocity vector at iteration t, and α is the learning rate. Adam is an effective update algorithm because it uses information of the first and second moments of the gradient for each parameter to make an update.

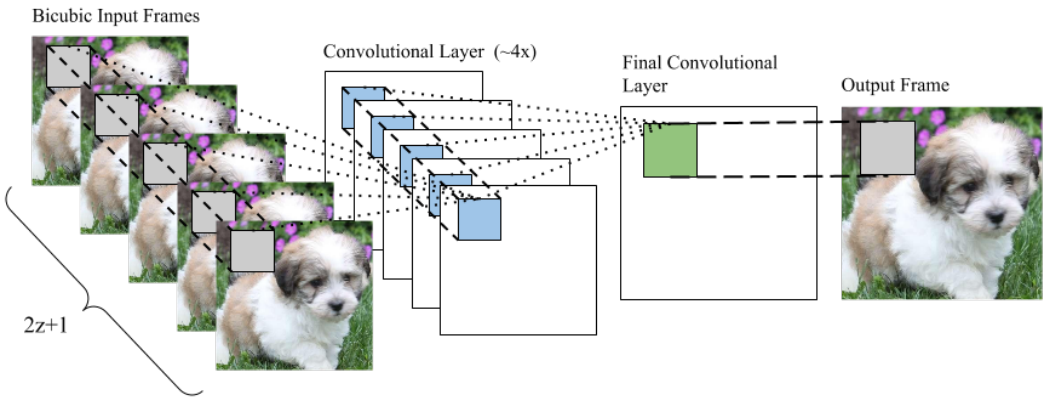


Figure 2: The architecture of our multi-frame CNN. The input is the target bicubic interpolated frame patch and the z frames before and after. The output is a single HR frame patch of the output frame. This process is repeated for all frame patches for each frame in the input video.

3.2.3 Hyperparameters

For a network with k layers, let n_i denote the number of neurons (convolutional filters) at each layer, and f_i denote the filter size at each layer. In hyper-parameter tuning, we let $f_i \in [1, 11]$ and $n_i \in \{8, 16, 32, 64, 96\}$ at each layer, where the upper bound of the latter was set by memory limitations of the machine we used to train the model. The only exception was the output layer, where we set $n_k = 3$ to create the final output image. In addition, we experimented with values of α, the learning rate, λ, the regularization strength, and p, the dropout parameters follows: $\log \eta \in [-3, -6]$, $\log \lambda \in [-7, -4]$, and dropout parameter $p \in [0.9, 1]$ (we quickly discovered that $p < 0.9$ leads to poor results).

3.2.4 Weight Initialization

Within SISR literature we came across three different weight initializations for the network that had been successfully applied to achieve better convergence of the network. The first, proposed in [3], we will call the Dong initialization. The second, used in [12], we will call the Norman initialization. The third and most commonly used initialization we found is the Xavier initialization. Each of these initializations is described in equation 6.

$$\begin{aligned} &\textbf{Xavier Initialization: } W_{ij} \sim \textbf{Unif}(-\frac{1}{\sqrt{n_{\text{in}}}}, \frac{1}{\sqrt{n_{\text{in}}}}) \\ &\textbf{Dong Initialization: } W_{ij} \sim \mathcal{N}(0.001) \\ &\textbf{Norman Initialization: } W_{ij} \sim \mathcal{N}(\frac{2}{n_{\text{in}} + n_{\text{out}}}) \end{aligned} \tag{6}$$

where **Unif** denotes the uniform distribution, $\mathcal{N}$ the gaussian distribution, n_{in} is the number of neurons in the previous layer, and n_{out} is the number of neurons within the given layer. In our experiments we test all three initializations for all of the given tasks.

3.2.5 Single-Frame vs Multi-Frame CNN

We trained two different classes of models, one using only single images or frames (SICNN) and one utilizing information from multiple frames in a video (MFCNN). In the SICNN, we follow the exact architecture described above to create the network. For the MFCNN we take a different approach which allows us to take in a frame and its adjacent frames (within some distance d) and output the high quality version of the middle frame (as seen in figure [2]). In order to allow the adjacent input frames to retain all of their spatial information, we form a single training example by concatenating all of the input frames along the channel dimension. Hence, the model takes in a total of $2d + 1$ frames of size $H \times W \times 3$ (with d frames on either side of the target frame), and from them forms an input of size $H \times W \times 3(2d + 1)$. In theory, the information present in the adjacent frames will allow us to make a more accurate prediction than using the single frame alone. Because it was not initially clear how many nearby frames on average tend to be informative, d is another hyperparameter we can tune.

4. Dataset

Due to their high number of parameters, CNNs generally require a large amount of data for training. Given this, we collected roughly 20,000 HR images and 6,000 HR videos from the Flickr Creative Commons 100M dataset [13]. For our single-image CNN, we create a training set

for hyperparameter tuning consisting of 20,000 still images (ImgSet) from [13]. We use two training sets for our multi-frame CNN. One relatively small set consisting of 500 videos (SmallVidSet) and one large data set (LargeVidSet) of about 6,000 videos. To create our training examples for each dataset, we sub-sample and then up-scale each image or video frame by some factor (uniformly selected between 2 and 4) using bicubic interpolation. We break each interpolated image or video frame into 128x128x3 non-overlapping patches and use these low-resolution sub-images as our training examples. The corresponding ground truth patches from the unadjusted image or video frame are the labels, as shown in figure [3]. ImgSet consists of about 200,000 image patches, SmallVidSet has roughly 250,000 frame patches, and LargeVidSet has roughly 3M.

In order to properly compare our single-image CNN model against our reference model [3] and others, we use *Set14* [18], which is a common data set for testing SISR algorithms. Unfortunately, such a data set for VSR does not exist, but following the guidelines of [8] we constructed a test set of 48 sequences from 12 HR videos, each 31 frames long. We call this test set *VidSet12*.

(a) (b) (c)

Figure 3: An example of a single video frame and 3 frame patches from our dataset. (a) shows the ground truth video frame above and the bicubic interpolated video frame below. (b) shows the ground truth frame patches (labels) and (c) has the corresponding bicubic interpolated frame patches (training examples).

5. Experiments and Results

We performed three separate experiments using our two models. For the SICNN, we trained it on both ImageSet and LargeVidSet in order to produce models that perform the slightly different tasks of SISR and single frame super resolution (SFSR). SFSR in general is a more difficult task than SISR because while most images are of static scenes, videos tend to contain much more motion that can result in blurring. This leads to lower quality frames and drastically expands the set of possible high quality interpretations from those frames, as blurring means that there are more possible ways the objects are positioned and oriented in the high quality frame. For the MFCNN, we trained it on SmallVidSet and LargeVidSet and thereby produce a model that takes a frame and its surrounding frames and returns a super resolution prediction.

5.1. Single Image Super Resolution

Our first experiment was creating a working SISR model using static images (ImageSet) in order to verify that our network architecture was a viable choice and had the capacity to perform some super resolution tasks. Most of the research into the best network architecture took place at this stage because the network architectures that did well at this task tended to do well at the other two tasks.

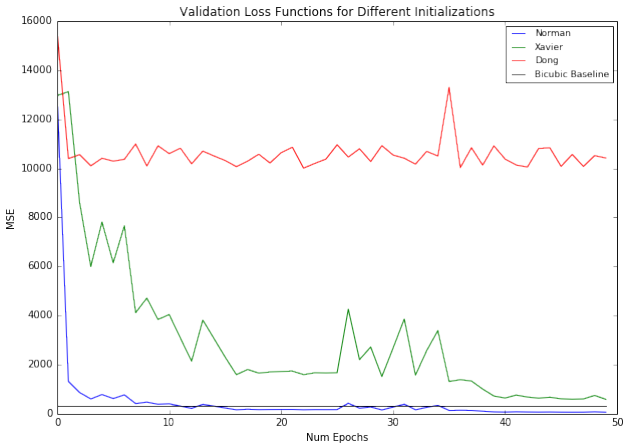


Figure 4: The validation loss (MSE) of our SICNN for different weight initializations. The blue, green, and red depict the loss for Norman, Xavier, and Dong initializations respectively.

5.1.1 Hyperparameter Tuning

As described above, there is a large space of hyperparameters to choose from. These include the number of layers, number of neurons for each layer, filter size at each layer, training algorithm, learning rate, regularization strength, dropout probability, and choice of weight initialization method. In order to explore as much of this space as possible, we started by training 100 different networks on a subset of ImageSet and evaluating the success of each network based on the mean SSIM of the validation set. From

this initial search, we were able to narrow down our hyperparameter search significantly in four ways: 1) neither form of regularization affected the network's success, 2) Adam performed much better than momentum, 3) networks with number of layers between 5 and 7 performed significantly better, and 4) Norman initialization drastically outperformed all others.

In figure 4 we plot the loss functions of the best performing networks for each initialization, demonstrating exactly how much better Norman initialization performed than both the Dong and Xavier initializations.

In addition to the above observations, we were also able to narrow down the ranges of some of the other hyperparameters, such as filter size and learning rate, leaving us with a significantly reduced space of likely optimal hyperparameters. Using this reduced set of possible hyperparameters, we trained 200 networks on the full ImageSet and for each network saved the weights and associated hyperparameters of the model that achieved the highest validation SSIM during training. Table 1 summarizes the results of this search, giving the sets of hyperparameters for the best three resulting networks.

Hyperparameters	SICNN 1	SICNN 2	SICNN 3
Num Layers	5	5	6
Num Neurons	32-32-64-32-16	64-32-32-16-16	16-32-64-32-32-32
Learning Rate	1.14e-4	0.91e-5	1.03e-4
Filter Sizes	9-5-5-3-3	7-5-5-3-3	7-5-5-3-5-3

Table 1: Hyperparameter combinations for the top 3 models.

5.1.2 Results

We then used the best resulting network in terms of validation SSIM and tested it on the Set14 test set in order to compare it to other SR algorithms. The results of this test are summarized below in Table 2.

Method	PSNR (dB)	SSIM
Bicubic	31.00	0.859
SC [17]	28.31	0.7954
KK [6]	32.11	0.9026
SRCNN [3]	32.45	0.9067
CSCN-MV [15]	32.71	0.9095
SICNN	**32.38**	**0.9060**

Table 2: Results of our Single-Frame model (SICNN) on the *Set14* dataset.

From the S14 test, we conclude that our model performs much better than the standard methods, though not quite as well as the state of the art, neural network based methods. This serves as a litmus test that allows us to go forward with applying this architecture to video. Visually, the model is successful in that it produces more subjectively higher quality images, as seen in figure 5.

5.2. Video Super Resolution

5.2.1 Single-Frame CNN

After performing the above hypertuning for the SICNN, we took the best 50 networks and from them made 100 different networks to solve the single-frame SR task on the SmallSet; 50 of these networks were exact copies of the SICNN network and 50 were these networks perturbed slightly so that $W_{ij} \sim \textbf{Unif}(0.99W_{ij}, 1.01W_{ij})$ for each W in the network. We call these networks Single Frame CNNs (SFCNN). While mostly similar, upon training the perturbed group converged slightly faster than the other group (and much faster than not using any pre-trained weights), and thus we only used these 50 to hypertune our SFCNN. Generally, the same network architectures that did the best at SISR did the best at SFSR. The sets of hyperparameters of best three networks are summarized below in Table 3.

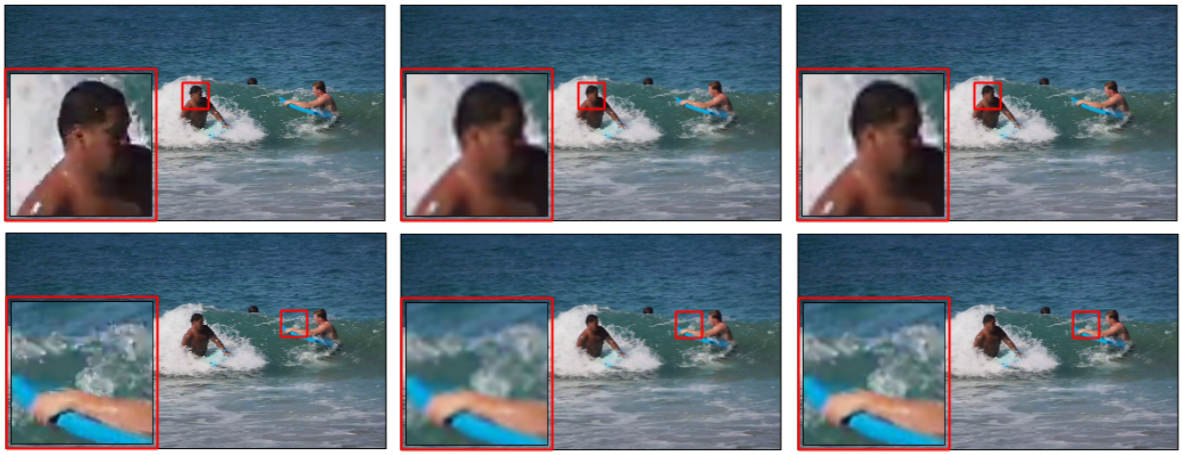
Hyperparameters	SFCNN 1	SFCNN 2	SFCNN 3
Num Layers	5	5	5
Num Neurons	32-32-64-32-16	64-32-32-16-16	64-64-32-32-16
Learning Rate	1.14e-4	0.91e-5	1.21e-4
Filter Sizes	9-5-5-3-3	7-5-5-3-3	9-7-3-5-3

Table 3: Hyperparameter combinations for the top 3 models.

5.2.2 Multi-Frame CNN

By a similar process as above, we took the 25 best networks of the SFCNN and applied it to the multi-frame SR task on the LargeSet. However, in this case we had to tweak our scheme due to the fact that the input now has channel dimension $3(2d+1)$. In order to still utilize the pre-trained weights, we set the first layer's weights to be the concatenation of $(2d+1)$ identical copies of the pre-trained weights, again subject to small random perturbation. That way, the pre-trained filter is applied to each of the $(2d+1)$ frames of the input at the first layer. In addition, we divided each of these weights, along with the biases, by $(2d+1)$ so that the mean and variance of the input to the second layer roughly match that of the SFCNN. Converting in this way, we created the MFCNN network from each of the 25 networks for each value of $d \in \{1, 2, 3\}$, resulting in 75 total networks. The best three sets of hyperparameters are summarized below in Table 4.

Overall the networks performed better when they were a bit deeper (6 layers) than in SFCNN or SISR, possibly because it takes an extra linear transformation and nonlinearity to fully take advantage of the extra information in



(a) (b) (c)

Figure 5: Comparison of the ground truth images (a) with the training examples, (b), and our results, (c).

Hyperparameters	MFCNN 1	MFCNN 2	MFCNN 3
Num Layers	6	6	6
Num Neurons	32-32-16-64-16-32	16-32-64-32-32-32	64-32-32-64-16-32
Learning Rate	0.82e-5	1.03e-4	0.90e-5
Filter Sizes	7-7-5-3-3-3	7-5-5-3-5-3	9-5-7-3-5-3
d	1	1	1

Table 4: Hyperparameter combinations for the top 3 models.

the input. However, letting $d > 1$ yielded worse results in most networks, possibly because the frame is too likely to have changed significantly from beginning to end, meaning that while some frames farther than 1 frame away add information, others add a great deal of noise. In addition, the network does not a priori know how to deal with motion, which could be in any direction (though we had hoped it would learn to do so).

5.2.3 Results

We trained the best 5 networks for both the SFCNN and MFCNN on the LargeSet and took the best resulting models for each. In order to get a decent comparison to other algorithms, we compute the mean PSNR and SSIM of Bicubic interpolation on our data set (*VidSet12*) and normalize our results such that both data sets have the same Bicubic interpolation mean PSNR and SSIM. As an example, suppose our test set PSNR is S. Then, letting B_{p0} be the mean PSNR of our data set and B_{p1} the mean PSNR of theirs, we report that our PSNR is equal to $S\frac{B_{p1}}{B_{p0}}$ (likewise for SSIM). As it happens, the ratio $\frac{B_{p1}}{B_{p0}}$ is close to one so this normalization does not have a profound effect. We tested both SFCNN, where we only consider one frame at a time, and MFCNN, considering multiple surrounding frames, and report the results in Table 5.

Method	PSNR (dB)	SSIM
Bicubic	31.80	NUM
BayesSR [9]	29.53	0.9451
DraftCNN [8]	31.87	0.9483
SFCNN	**31.62**	**0.9477**
MFCNN	**32.79**	**0.9501**

Table 5: Results of our Multi-Frame model (MFCNN) on the *VidSet12*.

Though this comparison is not exact, we can see that the MFCNN outperforms SFCNN and DraftCNN, the next best VSR algorithm we have to compare to, meaning that the network has successfully learned to incorporate information from adjacent frames, as we can seen in figure 6.

5.3. Visualization

We have seen the results of the three models above, but what is left unanswered is what exactly the networks are doing and how they do it. To partially answer both of these questions, we can take advantage of the fact that the outputs of the SISR (and SFSR) network are the same size as the inputs. Hence, we can run our SISR algorithm multiple times over the same image to see how it changes. In figure 7, we see the result of eight consecutive runs through the SISR network for the original LR image.

From this visualization, we can tell that the algorithm is sensitive to edges and contrast, seeking to decrease blurring and increase the spatial frequency of the pixels in the image. Another thing we can gather is that the network can

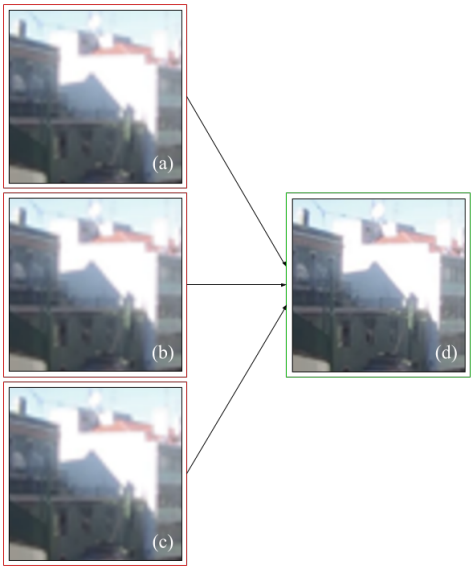


Figure 6: Results of a test video. (a)-(c) are video frame patch training examples where (b) is the target frame patch and (a), (c) are neighboring frames. (d) is our MFCNN output.

(a) (b)

Figure 7: (a) is the LR frame patch. (b) shows the result of running our model several times over (a).

sometimes slightly shift some of the pixels, which is not perceivable after only one iteration, but by the eighth iteration creates somewhat of a halo effect around the edges in the image.

5.4. Areas of Possible Improvement

Although the models perform well on average, they can yield unexpected results at times, respond poorly to certain types of scenes, and can perform no better than Bicubic interpolation for particularly low quality images. As demonstrated in figure 8, the SISR model does not respond well to trees and leaves.

It is not clear exactly why the model added random,

(a) (b)

Figure 8: (a) is the LR frame patch. (b) shows the result of running our model given (a) as input.

oddly colored spots within the gaps of the tree. However, we know that the model responds strongly to contrast so it could be because areas with too much contrast per unit area cause the model to ”overreact.”

6. Conclusion and Future Work

Leveraging deep learning techniques for SISR, we have proposed a novel multi-frame VSR technique that successfully takes advantage of pixel information from adjacent frames. While the field of VSR is too young to have an clearly state of the art method, our model significantly outperforms the techniques that we were able to compare it to. Our results also confirm that VSR is an easier and thus more promising problem to solve than SISR, as we were able to improve the quality of the video considering multiple frames more than if we had only access to one frame at a time.

However, while our technique has great performance on average, it is far too inconsistent as of yet to be used generally. Future work includes fine-tuning the models in order to produce a more robust algorithm for VSR that is not subject to as many outliers and sudden or unexpected drops in performance. One idea to combat this is to add higher regularization to the models, which may reduce the mean improvement but significantly decrease its variance. Another is to ensemble multiple models and take the most popular choice among these for each pixel to reduce the probability that a single model fails on a subset of pixels.

We can also consider completely different network architectures to solve this problem, given that we know it is tractable. Recent research has shown the power of Recurrent CNNs (RCNNs) for tasks like video classification and object tracking. Given this direction of movement within the deep learning community, it seems likely that using an RCNN would yield superior results, and perhaps even be more stable.

References

[1] Y. A. Y. Al-Najjar and D. D. C. Soong. Comparison of image quality assessment: Psnr, hvs, ssim, uiqi. *International Journal of Scientific and Engineering Research*, 3(8), 2012.

[2] Z. Cui, H. Chang, S. Shan, B. Zhong, and X. Chen. Deep network cascade for image super-resolution. *European Conference on Computer Vision*, pages 49–64, 2014.

[3] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[4] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. *IEEE International Conference on Computer Vision*, pages 349–356, 2009.

[5] G. Hinton, N. Srivastava, and K. Swersky. Lecture6.5 - rmsprop, 2012.

[6] K. I. Kim and Y. Kwon. Single-image super-resolution using sparse regression and natural image prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1127–1133, 2010.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[8] R. Liao, X. Tao, R. Li, Z. Ma, and J. Jia. Video super-resolution via deep draft-ensemble learning. *IEEE International Conference on Computer Vision*, pages 531–539, 2015.

[9] C. Liu and D. Sun. On bayesian adaptive video super resolution. *IEEE TPAMI*, 2013.

[10] Z. Na, R. Liao, X. Tao, L. Xu, J. Jia, and E. Wu. Handling motion blur in multi-frame super-resolution. *CVPR*, 2015.

[11] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. *Proceedings of the 30th International Conference on Machine Learning*, pages 1139–1147, 2013.

[12] N. Tasfi. Image scaling using deep convolutional neural networks, 2015.

[13] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. The new data and new challenges in multimedia research. *arXiv preprint arXiv:1503.01817*, 2015.

[14] R. Timofte, V. D. Smet, and L. V. Gool. Anchored neighborhood regression for fast example-based super-resolution. *IEEE International Conference on Computer Vision*, pages 1920–1927, 2013.

[15] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang. Deep networks for image super-resolution with sparse prior. *CoRR*, abs/1507.08905, 2015.

[16] C.-Y. Yang, C. Ma, and M. H. Yang. Single-image super-resolution: A benchmark. *European Conference on Computer Vision*, pages 372–386, 2014.

[17] J. Yang, J. Wright, T. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873, 2010.

[18] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. *Curves and Surfaces*, pages 711–730, 2012.