CAP 6610 Machine Learning: Super Resolution using Generative Adversarial Neural Network and Convolutional neural networks

Rahul Roy
Department of Computer Science
University of Florida
roy.rahul@ufl.edu

Ganesan Santhanam
Department of Computer Science
University of Florida
gsanthanam@ufl.edu

Sumeet Saini
Department of Computer Science
University of Florida
s.saini@ufl.edu

Abstract

The covid-19 pandemic has made us depend even more on virtual technologies to keep us going forward with our day to day activities. Even though latest cameras give high resolution images as output, in many domains due to factors like network limitations, it is very difficult to obtain high resolution images. But to get meaningful inference out of the low resolution images we will have to improve the quality. The goal of the single-image super-resolution algorithms is to generate a high resolution image from a low-resolution image. In this project, we evaluate 3 different super resolution algorithms namely generative adversarial network for single image super-resolution (SRGAN), Image Super-Resolution Using Deep Convolutional Networks (SRCNN) and Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network (ESPCN). We used the Div2k single-image super-resolution data set that contains 1,000 images with different scenes for our experiments. The data set was split into 800 for training, 100 for validation and 100 for testing.

https://github.com/Sumeet2807/Super-Resolution

1 Introduction

1.1 Super Resolution

The Super-resolution is technique wherein a high resolution image is generated for a given low resolution image. The techniques here discussed are namely generative adversarial network for single image super-resolution (SRGAN), Image Super-Resolution Using Deep Convolutional Networks (SRCNN) and Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network (ESPCN). The Div2k data set has And apart from the standard bi cubic down sampling, several types of degradations are considered in synthesizing low resolution images for different tracks of the challenges. This data set contains low resolution images with different types of degradations. The Track 2 of NTIRE 2017 contains low resolution images with unknown x4 down scaling. Track 2 and track 4 of NTIRE 2018 correspond to realistic mild ×4 and realistic wild ×4 adverse conditions, respectively. Low-resolution images under realistic mild x4 setting suffer from motion blur, Poisson noise and pixel shifting. Degradations under realistic wild x4 setting are further extended to be of different levels from image to image.

Preprint. Under review.

1.2 Related works

1.2.1 Image Super Resolution

The first methods to tackle single image super resolution was Prediction-based methods while these filtering approaches, like linear, bi-cubic or Lanczos [43] filtering, can be extremely fast, they inadvertently oversimplify the SUPER RESOLUTION problem and in turn yield solutions with overly smooth textures. The techniques that focus on edge-preservation has proposed [5, 44] and more powerful approaches are created using a complex mapping between low- resolution and high-resolution image information that usually rely on training data. Various methods that are based on example-pairs rely on low resolution training patches for which the corresponding high resolution counterparts are present. Earlier works presented by Freeman et al. [29, 40] and related approaches to the super-resolution problem originate in compressed sensing [28, 29, 31]. In Glasner et al. [232] they exploit patch redundancies across the scales within the images to drive the Super resolution, where self-dictionaries are improved by further allowing for small transformations and shape variations. Gu et al. [33] proposed a convolutional sparse coding approach that improves consistency by processing the whole image rather than overlapping patches. We can reconstruct realistic texture detail while avoiding edge artifacts, then Tai et al. [34] combine an edge-directed super resolution algorithm based on a derived gradient profile prior [35] with the benefits of learning-based detail synthesis. Zhang et al. [36] propose a multi-scale dictionary to capture redundancies of similar image patches at different scales. To super-resolve historic images, Yue et al. [37] retrieve high resolution images with similar content from the web and propose a structure-aware matching criterion for alignment. We then move on to convolutional neural network (CNN) based SUPER RESOLUTION algorithms have shown excellent performance. In Wang et al. [38] encodes a sparse representation prior into their feed-forward network architecture based on the learned iterative shrinkage and thresholding algorithm. Dong et al. [40, 41] used bi cubic interpolation to up-scale an input image and trained a five layer deep fully convolutional network end-to-end to achieve state of-the-art super resolution performance. Subsequently, it shows that enabling the network to learn the up-scaling filters directly can further increase performance both in terms of accuracy and speed with their deeply-recursive convolutional network (DRCN), Kim et al. [42] presented a highly performing architecture that allows for long-range pixel dependencies while keeping the number of model parameters small, who rely on a loss function closer to perceptual similarity to recover visually more convincing High Resolution images generated by the SRGAN. The Efficient Sub-Pixel Convolutional Neural Network (ESPCN) technique allows us to increase the resolution from low resolution to high resolution only at the very end of the network and super-resolve HR data from LR feature maps. This eliminates the need to achieve a lot of the super resolution operation in the far larger HR resolution. For this purpose, we propose an efficient sub-pixel convolution layer to learn the up-scaling operation for image and video super-resolution.

1.2.2 Design

The state of the art for several computer vision problems is set by specifically designed convolutional neural network architectures following the success of the work by ImageNet classification with deep convolutional neural networks. It was demonstrated that deeper network architectures can be difficult to train but have the potential to increase the network's accuracy as they allow modeling mappings of remarkably high complexity [24, 27]. To efficiently train these deeper network architectures, batch normalization [32] is frequently used to counteract the internal co-variate shift. Deeper network architectures have also proven to increase performance for SISR, e.g., Kim et al. [34] formulate a recursive CNN and present state-of-the art results. Another powerful design choice that eases the training of deep CNNs is the recently introduced concept of residual blocks [29] and skip-connections [30, 34]. Skip connections relieve the network architecture of modeling the identity mapping that is trivial in nature, however, potentially non-trivial to represent with convolutional kernels. In the context of SISR it was also demonstrated that learning up scaling filters is beneficial in terms of accuracy and speed [11, 14]. This is an improvement over Image super-resolution using deep convolutional networks where bi cubic interpolation is employed to upscale the LR observation before feeding the image to the CNN. The SRCNN architecture is simple yet provides superior accuracy compared with state-of-the-art example-based methods. Secondly, among the variety of popular numerical evaluations like the Peak Signal-to-Noise Ratio (PSNR), structure similarity index (SSIM), multi-scale SSIM, information fidelity criterion, we will be taking the PSNR as the metric to evaluate the result of the proposed SR algorithms. Super-Resolution Convolutional Neural Network (SRCNN) surpasses

the bi cubic baseline with just a few training iterations and outperforms the popular sparse-coding-based method with moderate training. But surprisingly outperforms the SRGAN. Yet the SRGAN provides visually appealing reconstructed image. With numbers of filters and layers, the SRCNN achieves fast speed for practical on-line usage even on a CPU. Our method is faster than a series of example-based methods because it is fully feed-forward and does not need to solve any optimization problem on usage. Thirdly, results show that the restoration quality of the network could be enhanced when larger and more diverse datasets like Div2k is used with a larger and deeper model is used. On the contrary, a larger data set can present with immense challenges, furthermore SRCNN can cope with three channels of color images simultaneously to achieve improved super-resolution performance. Overall, the SRCNN is a fully convolutional neural network for image super-resolution wherein the network directly learns an end-to-end mapping between low resolution and high-resolution images, with little pre-processing beyond the optimization. This algorithm can establish a relationship between our deep learning-based SRGAN. This relationship provides a guidance for the design of the network structure. So, to improve the SRCNN by introducing larger filter size in the non-linear mapping layer, and explore deeper structures by adding nonlinear mapping layers, then, we extend the SRCNN to process three color channels (either YCbCr or RGB) In addition, we include the main metric of the comparison apart from PSNR is the HR image having the perceptual quality as an indicator for the performance of the algorithm compared to computational efficiency itself. The ESPCN algorithm tackles a problem common for convolutional networks, wherein the processing speed directly depends on the input image resolution. Also, the interpolation methods commonly used to accomplish this specific task, such as bi cubic interpolation does not bring any additional information to solve the ill-posed reconstruction problem. So, learning the up scaling filters was briefly suggested in paper named Jointly optimized regressors for image super-resolution. Dong et.al [6]. Nevertheless, the importance of integrating onto a CNN as part of the SR operation was not completely recognized and is not explored. So, the ESPCN proposes to increase the resolution from LR to HR only at the very end of the network and super-resolve HR data from LR feature maps eliminating the need to accomplish most of the operation for super resolution. To achieve this goal, we incorporate an efficient sub-pixel convolution layer to learn the up scaling operation for a single image super resolution that even be useful for a video super-resolution. So, in ESPCN, up scaling is managed by the last layer meaning each LR image is also directly fed to the network and feature extraction occurs through nonlinear convolutions in LR space. With various degradations holding reduced input resolution images of Div2k data set, we will be using smaller filter size to integrate the information while maintaining the given contextual area. The assumed network has L layers, so for n_{L-1} up scaling filters for the n_{L-1} feature maps as opposed to one up scaling filter for the input image which the case for SRCNN. Additionally, by not using an explicit interpolation filter meaning a network that implicitly learns the processing necessary for SR operations. Therefore, the network is capable of learning a better and more complex LR to HR mapping compared to a single fixed filter up scaling at the first layer.

1.2.3 Loss Function

The SRGAN uses pixel-based loss functions as MSE struggle to handle the uncertainty while recovering lost high-frequency details such as texture, etc. Hence, by minimizing MSE encourages the model to find pixel-wise averages of plausible solutions that are typically overly-smooth and thus have poor perceptual quality. Reconstructions of varying perceptual quality are exemplified with corresponding PSNR in Figure. We tackled this problem by employing generative adversarial networks (GANs) for the application of image generation. Yu and Porikli [66] augment pixel-wise MSE loss with a discriminator loss to train a network that super-resolves face images with large up scaling factors (8). Using GANs to learn the mapping from one manifold to another for style transfer and inpainting. To minimize the squared error in the feature spaces of VGG19 and scattering networks. Dosovitskiy and Brox [7] use loss functions based on Euclidean distances computed in the feature space of neural networks in combination with adversarial training. In the proposed loss that allows visually superior image generation and can be used to solve the ill-posed inverse problem of decoding nonlinear feature representations. Similar to SRGAN, Johnson et al. [9] and Bruna et al. [16] propose the use of features extracted from a pretrained VGG network instead of sub pixel-wise error measures followed by ESPCN. They formulate a loss function based on the Euclidean distance between feature maps extracted from the VGG19 network. That results in a perceptually convincing images is obtained. Recently, Li and Wand [8] also investigated the effect of comparing and blending

patches in pixel or VGG feature space. Whereas in SRCNN , the loss function L is the average of mean square error (MSE) for the training samples (n), which is a kind of standard loss function.

For the SRCNN, during the training process, the original HR images will be ground truth data. The mean squared error (MSE) is used to measure the difference between the generated SR images and the ground truth HR images. The pixel-wise MSE loss function of the network is represented in equation 1, and for ESPCN, it's modified MSE loss where I(HR) represents each original image in the data set; I(LR) represents each down sampled LR image; r represents the up scaling factor; H represents the image's height value; W represents the image's width value, W(1:L) represents all the learnable network weights and b(1:L) represents all the learnable biases as shown in the equation 2.

$$L(\Theta) = \frac{1}{n}\sum_{i=1}^{n} \|F(\mathbf{Y}_i;\Theta) - \mathbf{X}_i\|^2 \tag{1}$$

$$l(W_{1:L}, b_{1:L}) = \frac{1}{r^2 HW}\sum_{x=1}^{rH}\sum_{x=1}^{rW}(\mathbf{I}_{x,y}^{HR} - f_{x,y}^{L}(\mathbf{I}^{LR})^2) \tag{2}$$

2 Method

2.1 SRGAN Architecture

The SRGAN algorithm aims is to estimate a super resolved image I_{SR} from a low-resolution input image I_{LR}. Here I_{LR} is the low-resolution version of its high resolution counterpart I_{HR}. In the training data, I_{LR} is obtained by applying a Gaussian filter to I_{HR} followed by down sampling with a factor of r. For an image with C color channels, we have I_{LR} by a real-valued tensor of size $W \times H \times C$ and I_{HR}, I_{SR} by $rW \times rH \times C$ respectively. Our goal is to train a generating function G that estimates for a given LR input image its corresponding HR counterpart. To achieve this, we train a generator network as a feed-forward CNN GθG parameterized by θG. Here GθG = $W1 : L; b1 : L$ denotes the weights and biases of a L-layer deep network and is obtained by optimizing a SR-specific loss function l_{SR}. For training images I_{HR} n , n = 1, . . . , N with corresponding I_{LR} n , n = 1, . . . , N,
we solve:

In this work we will specifically design a perceptual loss l SR as a weighted combination of several loss components that model distinct desirable characteristics of the recovered SR image. The task of SISR is to estimate a HR image I_{SR} given a LR image I LR down scaled from the corresponding original HR image I_{HR}. The down sampling operation is deterministic and known: to produce I_{LR} from I_{HR}, we first convolve I_{HR} using a Gaussian filter - thus simulating the camera's point spread function - then down sample the image by a factor of r. We will refer to r as the up scaling ratio. In general, both I_{LR} and I_{HR} can have C colour channels, thus they are represented as real-valued tensors of size H × W × C and $rH \times rW \times C$, respectively. To solve the SISR problem, the SRCNN proposed in [7] recovers from an up scaled and interpolated version of I_{LR} instead of I_{LR}. To recover I_{SR}, a 3 layer convolutional network is used. The SRGAN has novel network architecture, as implemented from [22] to avoid up scaling I_{LR} before feeding it into the network. In our architecture, we first apply a l layer convolutional neural network directly to the LR image, and then apply a sub-pixel convolution layer that up scales the LR feature maps to produce I_{LR}. For a network composed of L layers, the first L-1 layers can be described as follows: is a 2D convolution tensor of size $nl_1 \times nl \times kl \times k$, where nl is the number of features at layer l, $n0 = C$, and kl is the filter size at layer l. The biases bl are vectors of length nl . The nonlinearity function (or activation function) Φ is applied element-wise and is fixed. The last layer f_L has to convert the LR feature maps to a HR image.

2.1.1 Adversarial network architecture

The implementation according to the paper [1] is the formulation that it trains a generative model G with the goal of fooling a differentiable discriminator D that is trained to detect SR images from real images. And we implemented our generator to learn to create solutions that are highly similar

to original images and thus difficult to classify by Discriminator D. This encourages perceptually superior solutions residing in the HR subspace, the manifold, of original images. This is in contrast to SR solutions obtained by minimizing pixel-wise error measurements, such as the MSE. At the core of the generator network G, in figure 4 are B residual blocks with identical layout. Inspired by Johnson et al. [9] we employ the block layout proposed by Gross and Wilber [10]. Specifically, we use two convolutional layers with small 3×3 kernels and 64 feature maps followed by batch-normalization layers [32] and ParametricReLU [11] as the activation function. We increase the resolution of the input image with two trained sub-pixel convolution layers as proposed by Shi et al. [48]. To discriminate real HR images from generated SR samples we train a discriminator network. The architecture is shown in Figure 4. We follow the architectural guidelines summarized by Radford et al. [13] and use LeakyReLU activation (α = 0.2) and avoid max-pooling throughout the network. The discriminator network is trained to solve the maximization problem in Equation 2. It contains eight convolutional layers with an increasing number of 3 × 3 filter kernels, increasing by a factor of 2 from 64 to 512 kernels as in the VGG network [14]. Strided convolutions are used to reduce the image resolution each time the number of features is doubled. The resulting 512 feature maps are followed by two dense layers and a final sigmoid activation function to obtain a probability for sample classification.

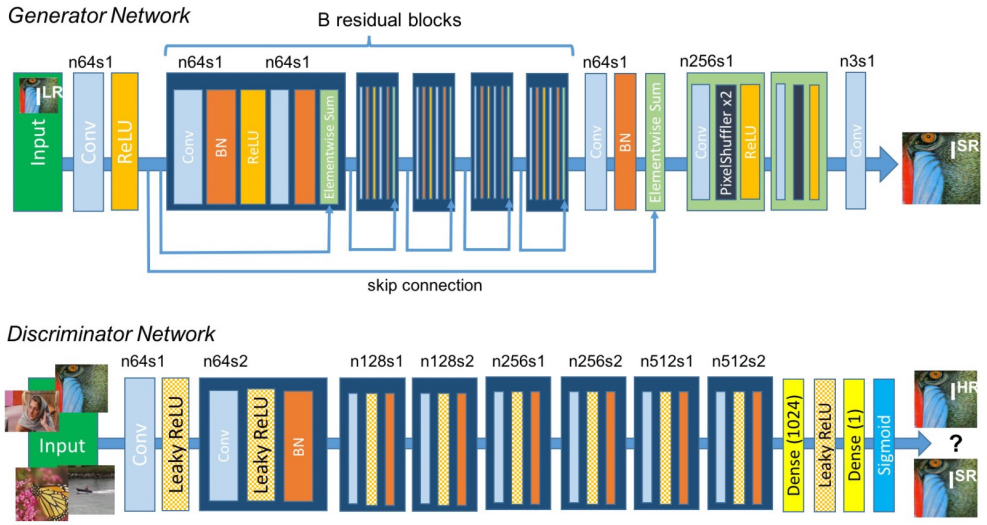


Figure 1: SRGAN Architecture

2.1.2 Perceptual loss function

The definition of our perceptual loss function l SR is critical for the performance of our generator network. While l_{SR} is commonly modeled based on the MSE [10, 48], it's improved from the implementation followed in the Johnson et al. [9] and Bruna et al. [16] and have designed a loss function that assesses a solution with respect to perceptually relevant characteristics. We formulate the perceptual loss as the weighted sum of a content loss (l_{SR} and an adversarial loss component as:

$$l^{SR} = \underbrace{\underbrace{l_X^{SR}}_{\text{content loss}} + \underbrace{10^{-3} l_{Gen}^{SR}}_{\text{adversarial loss}}}_{\text{perceptual loss (for VGG based content losses)}}$$

Figure 2: SRGAN Perceptual Loss

2.1.3 Content loss

The pixel-wise MSE loss is calculated and shown in figure 3,as a widely used optimization target for image super resolution[10,48]. Nevertheless,by achieving high PSNR,solutions of MSE optimization problems often lack high frequency content that results in perceptually unsatisfying solutions with overly smooth textures.Instead of relying on pixel-wise losses we build on the ideas of Gatys et al. [15], Bruna et al. [16] and Johnson et al. [9] and use a loss function that is closer to perceptual similarity. We define the VGG loss based on the ReLU activation layers of the pre-trained 19 layer VGG network described in Simonyan and Zisserman [14]. With $\phi_{i,j}$ we indicate the feature map

obtained by the j-th convolution (after activation) before the i-th maxpooling layer within the VGG19 network, which we consider given. We then define the VGG loss as the euclidean distance between the feature representations of a reconstructed image GΘG (I_{LR}) and the reference image I_{HR}

MSE Loss

$$l^{SR}_{MSE} = \frac{1}{r^2WH}\sum_{x=1}^{rW}\sum_{y=1}^{rH}(I^{HR}_{x,y} - G_{\theta_G}(I^{LR})_{x,y})^2$$

VGG Loss

$$l^{SR}_{VGG/i,j} = \frac{1}{W_{i,j}H_{i,j}}\sum_{x=1}^{W_{i,j}}\sum_{y=1}^{H_{i,j}}(\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2$$

Figure 3: SRGAN Content Loss

2.1.4 Adversarial loss

Apart from the content losses, we also include the generative component of our GAN to the perceptual loss. This enables our network to favor solutions that reside on the manifold of natural images, by trying to fool the discriminator network. The generative loss l^{SR}_{Gen} is defined based on the probabilities of the discriminator DθD (GθG (I_{LR})) over all training samples.

Adversarial Loss

$$l^{SR}_{Gen} = \sum_{n=1}^{N} -\log D_{\theta_D}(G_{\theta_G}(I^{LR}))$$

Here, $D_{\theta_D}(G_{\theta_G}(I^{LR}))$ is the probability that the reconstructed image $G_{\theta_G}(I^{LR})$ is a natural HR image. For better gradient behavior we minimize $-\log D_{\theta_D}(G_{\theta_G}(I^{LR}))$ instead of $\log[1 - D_{\theta_D}(G_{\theta_G}(I^{LR}))]$

Figure 3: SRGAN Adversarial Loss

2.1.5 Training details and parameters

2.2 SRCNN Architecture

There have been a few studies of using deep learning techniques for image restoration. The multi-layer perceptron (MLP), whose all layers are fully-connected (in contrast to convolutional), is applied for natural image denoising [23] and post-deblurring denoising [18]. More closely related to our work, the convolutional neural network is applied for natural image denoising [19] and removing noisy patterns (dirt/rain) [23]. These restoration problems are more or less denoising-driven. Cui et al. [16] propose to embed auto-encoder networks in their superresolution pipeline under the notion internal examplebased approach [16]. The deep model is not specifically designed to be an end-to-end solution, since each layer of the cascade requires independent optimization of the self-similarity search process and the auto-encoder. On the contrary, the proposed SRCNN optimizes an end-toend mapping. Further, the SRCNN is faster at speed. It is not only a quantitatively superior method, but also a practically useful one.

2.2.1 Formulation

1. Patch extraction and representation: this operation extracts (overlapping) patches from the lowresolution image Y and represents each patch as a high-dimensional vector. These

vectors comprise a set of feature maps, of which the number equals to the dimensionality of the vectors.

2. Non-linear mapping: this operation nonlinearly maps each high-dimensional vector onto another high-dimensional vector. Each mapped vector is conceptually the representation of a high-resolution patch. These vectors comprise another set of feature maps.

3. Reconstruction: this operation aggregates the above high-resolution patch-wise representations to generate the final high-resolution image. This image is expected to be s

2.2.2 Patch extraction and representation

The patch based extraction follows a strategy to restore image is to densely extract patches and then represent them as a set of pre-trained bases such as PCA, DCT etc. This is similar to convolving the image by a set of filters, each of which has a basis. In SRCNN, we involve the optimization of these bases onto the optimization of the network. Initially, our first layer is expressed as an operation F1: F1(Y) = max (θ, W1 × Y + B1), (1) where W1 and B1 represent the filters and biases respectively, and '×' denotes the convolution operation. Here, W1 corresponds to n1 filters of support $c \times f1 \times f1$, where c is the number of channels in the input image, f1 is the spatial size of a filter. Intuitively, W1 applies n1 convolutions on the image, and each convolution has a kernel size $c \times f1 \times f1$. The output is composed of n1 feature maps. B1 is an n1-dimensional vector, whose each element is associated with a filter.

2.2.3 Non-linear mapping

The first layer extracts a n1-dimensional feature for each patch. The next operation is that we map each of these n1-dimensional vectors into an n2-dimensional one. This is exactly like applying n2 filters that have a trivial spatial support 1 × 1. This interpretation is valid for 1×1 filters. Yet it's easy to generalize to larger filters like 3 × 3 or 5 × 5. In that case, our non-linear mapping is not on a patch of the low resolution image; instead, it is on a 3 × 3 or 5 × 5 "patch" of the feature map. The second layer operates using the formula: F2(Y) = max (0, W2 × F1(Y) + B2). (2) Here W2 contains n2 filters of size n1 ×f2 ×f2, and B2 is n2-dimensional. Each of the output n2-dimensional vectors is conceptually a representation of a high-resolution patch that shall be used for reconstruction. It's possible to add more convolutional layers to increase the non-linearity. So the complexity of the model increases (n2 × f2 × f2 × n2 parameters for one layer), resulting in more training time. We can explore deeper structures by introducing additional non-linear mapping layers onto the model.

2.2.4 Reconstruction

In the old methods, the predicted overlapping high-resolution patches are averaged to produce the resultant full image. The operation of averaging is considered as a pre-defined filter on a set of feature maps wherein each position is the "flattened" vector form of a high resolution patch. So, we can define a convolutional layer to produce the final high-resolution image: F(Y) = W3 × F2(Y) + B3.The ReLU activation function is part of the Non-linear mapping and the patch extraction and representation becomes purely linear convolution. Hence, for all c filters of a size n2 × f3 × f3, and B3 is a c-dimensional vector. High resolution patches of the image domain. So, the filters act like an averaging filter; if the representations of the high-resolution patches are in some other domains. The W3 is a set of linear filters. The three operations are motivated by different thinking, yet they all lead to the same form as a convolutional layer. So, we can implement all three operations together and form a convolutional neural network (Figure 4).

2.2.5 Relationship to Sparse-Coding-Based Methods

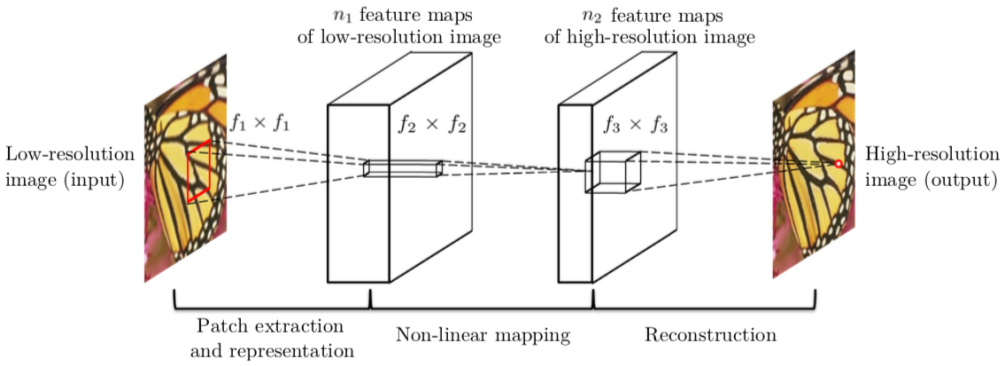


Figure 4: Sparse coding Architecture

2.3 Convolutional Network

Each learned filter has its specific functionality, for example, the filters g and h are like Laplacian/Gaussian filters, the filters a - e are like edge detectors at different directions, and the filter f is like a texture extractor. So in general, the performance would improve if we increase the network width that would be adding more filters, which in turn costs running time. Specifically, based on [2]the settings are of n1 = 64, n2 = 32, So they explore by varying the layers as follows (i) one is with a larger network with n1 = 128 and n2 = 64, and (ii) the other is with a smaller network with n1 = 32 and n2 = 16. The network sensitivity to different filter sizes to validate if a reasonably larger filter size could grasp richer structural information, which in turn lead to better results. Specifically, the mean PSNR values achieved by all variation of SRCNN suggest that utilizing neighborhood information in the mapping stage is beneficial. Nevertheless, the deployment speed will in turn drop with a larger filter size. For instance, if we take the number of parameters for a SRCNN to be 9-1-5, 9-3-5, and 9-5-5 is 8,032, 24,416, and 57,184 respectively. The complexity of 9-5-5 is nearly twice of 9-3-5, although the performance improvement isn't much. Therefore, the network scale should always be a trade-off between performance and speed. So, when we try to implement [17], that is having deeper structures just by adding another non-linear mapping layer, which has 16 filters with size 1. Resulting in, i.e., 9-1-1-5, 9-3-1-5, 9-5-1-5, SRCNN just by adding an additional layer on 9-1-5, 9-3-5, and 9-5-5, respectively. The initialization scheme and learning rate of the additional layer are the same as the second layer. So it is observed that the four-layer networks converge slower than the three-layer network. However, if given enough training time, the deeper networks will finally catch up and converge to the three-layer ones. Furthermore, it is understood that deeper networks do not necessarily result in improved performance. Specifically, if an additional layer with n22 = 32 filters on 9-1-5 network is added, the performance actually degrades and fails to measure upto the performance of the three-layer network.The paper [2] indicates that it is not "the deeper network the better" in this deep model for super-resolution. Our SRCNN network contains no pooling layer or full-connected layer, thus it is sensitive to the initialization parameters and learning rate. With deeper layers(e.g., 4 or 5 layers),setting appropriate learning rates that guarantees convergence is really difficult. Even with convergence, the network may fall into a worse local minimum, and the learned filters are of less diversity even given enough training time where improper increase of depth leads to accuracy saturation or degradation for image classification.

2.4 ESPCN Architecture

2.4.1 Deconvolutional layer

By adding a deconvolution layer is a good choice to recover the resolution from max-pooling and other image down-sampling layers. The approach is a successful way to implement in visualizing layer activations [14] and for generating the semantic segmentations using high level features from the network [10]. It is trivial to show that the bicubic interpolation used in SRCNN is a special case of the deconvolution layer, as explored already in [24, 7]. The deconvolution layer proposed in [50]

can be seen as multiplication of each input pixel by a filter element-wise with stride r, and sums over the resulting output windows also known as backwards convolution.

2.4.2 Efficient sub-pixel convolution layer

A common technique used to upscale a low resolution image is convolution with fractional stride of 1 r in the LR space that can be naively implemented by interpolation, un-pooling from LR space to HR space followed by a convolution with a stride of 1 in HR space. We should be able to increase the computational cost by a factor of r by 2 , since convolution happens in HR space. Interestingly, a convolution with stride of 1 r in the LR space with a filter W_s of size ks with weight spacing 1 r would activate different parts of W_s for the convolution. The weights that are between the pixels are simply not activated and are not calculated. The number of activation patterns is exactly r 2 . with each activation pattern, is in respect to the location. The patterns are periodically activated during the convolution of the filter across the image depending on different sub-pixel location: $\mod(x, r)$, $\mod(y, r)$ where x, y are the output pixel coordinates in HR space. According to [17], an effective way to implement the above operation when mod (ks, r) = 0 . So, for the convolution operator W_L has shape $n_{L-1} \times r_{2C} \times kL \times kL$ does need non linearity added to the outputs of the convolution at the last layer as $k_L = k_s$ r and $\mod(ks, r) = 0$ meaning it is equivalent to sub-pixel convolution in the LR space with the filter Ws. The last layer that is the sub-pixel convolution layer and implemented network is as efficient sub-pixel convolutional neural network (ESPCN). The last layer produces a HR image from LR feature maps directly with one upscaling filter for each feature map shown in figure 6. Given a training set consisting of HR image examples I HR n , n = 1 . . . N, we generate the corresponding LR images I LR n , n = 1 . . . N, and calculate the pixel-wise mean squared error (MSE) of the reconstruction as an objective function to train the network: ‘(W1:L, b1:L) = 1_r $2HW$ rH, x=1 I_{HR} with x,y having f_L(x,y) of $I_L R$). The periodic shuffling can be avoided in training time, and should be part of the output as part of the layer, we can preshuffle the training data to match the output of the layer before PS. Thus the model becomes log2r 2 times faster compared to deconvolution layer in training and r 2 times faster compared to implementations using various forms of upscaling before convolution.

$$\mathbf{I}^{SR} = f^L(\mathbf{I}^{LR}) = \mathcal{PS}\left(W_L * f^{L-1}(\mathbf{I}^{LR}) + b_L\right),$$

Periodic Shuffling

$$\mathcal{PS}(T)_{x,y,c} = T_{\lfloor x/r \rfloor, \lfloor y/r \rfloor, C \cdot r \cdot mod(y,r) + C \cdot mod(x,r) + c}$$

MSE for reconstruction

$$\ell(W_{1:L}, b_{1:L}) = \frac{1}{r^2 HW} \sum_{x=1}^{rH} \sum_{x=1}^{rW} \left(\mathbf{I}^{HR}_{x,y} - f^L_{x,y}(\mathbf{I}^{LR})\right)^2$$

Figure 6

2.4.3 Implementation details

For the ESPCN, we have set l = 3, (f1, n1) = (5, 64), (f2, n2) = (3, 32) and f3 = 3 in our evaluations. The choice of the parameter has been taken from SRCNN’s 3 layer 9-5- 5 model and in the training phase, 17r × 17r pixel sub-images are extracted from the training ground truth images I_{HR}, where r is the upscaling factor. To process the low-resolution samples I_{LR}, we blur I_{HR} using a Gaussian filter and sub-sample it by the upscaling factor. The sub-images are extracted from original images with a stride of (17 - Pmod (f, 2)) × r from I_{HR} and a stride of 17 - Pmod (f, 2) from I_{LR}. All pixels in the original image will appear once and only once as the ground truth of the training data. We choose tanh instead of relu as our activation function for the final model motivated by our experimental results. The training stops after no improvement of the

cost function is observed after 20 epochs. Initial learning rate is set to 0.01 and final learning rate is set to 0.0001 and it updates gradually when the improvement of the cost function is smaller than a threshold μ. The final layer learns 10 times slower that training takes roughly four to five hours on a GPU Colab. We use the PSNR as the performance metric to evaluate our models.

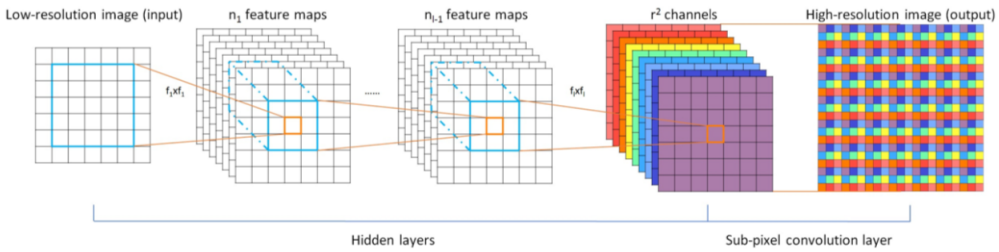


Figure 6: ESPCN Architecture

3 Training

The Div2k data set provides us with low resolution images that will be resized to 128 × 128, that will then be used as input for the model, and our output is an up-scaled with resolution 256 × 256. The data set has realistic and more complex degradation on the images that help train the model in robust manner. The SRGAN, SRCNN and ESPCN, are using the same set of images with same pre-processing step that has been implemented by us.
In SRGAN,SRCNN and ESPCN, certain layers and scales have been adjusted for the Div2K data set as they were written for ImageNet, set5 and t91 data sets.
So after we resize the image, we use channel normalization the images in each channel of a convolutional network individually. Let z_{ij} be the input of the j^{th} channel and the i^{th} layer. Channel normalization performs the transformation.where mean and var compute the empirical mean and variance, $\gamma_{i,j}$ and $\beta_{i,j}$ are parameters learned independently for each channel, and . is a fixed small constant added for numerical stability. And our optimizer is Adam Optimizer used as an replacement optimization algorithm for the stochastic gradient descent for training our models. As Adam has best properties of the AdaGrad and RMSProp algorithms, therefore provides an optimization algorithm that can handle sparse gradients on noisy problem existing in the data set and requires minimum memory as we are working with large data set.

Channel Normalization

$$\mathbf{z}'_{ij} = \frac{\mathbf{z}_{ij} - \text{mean}(\mathbf{z}_{ij})}{\sqrt{\text{var}(\mathbf{z}_{ij}) + \epsilon}} \gamma_{ij} + \beta_{ij},$$

3.1 Parameters

Trained for 20 epochs Input for Model 128 × 128 Output of Model 256 × 256 Optimizer : Adam Optimiser

Paragraphs There is also a `\paragraph` command available, which sets the heading in bold, flush left, and inline with the text, with the heading followed by 1 em of space.

4 Experiments

4.1 Dataset Div2k

DIV2K is a popular single-image super-resolution dataset which contains 1,000 images with different scenes and is splitted to 800 for training, 100 for validation and 100 for testing. It was collected for NTIRE2017 and NTIRE2018 Super-Resolution Challenges in order to encourage research on image super-resolution with more realistic degradation. This dataset contains low resolution images with different types of degradations. Apart from the standard bicubic downsampling, several types of degradations are considered in synthesizing low resolution images for different tracks of the challenges. Track 2 of NTIRE 2017 contains low resolution images with unknown x4 downscaling. Track 2 and track 4 of NTIRE 2018 correspond to realistic mild ×4 and realistic wild ×4 adverse conditions, respectively. Low-resolution images under realistic mild x4 setting suffer from motion blur, Poisson noise and pixel shifting. Degradations under realistic wild x4 setting are further extended to be of different levels from image to image. The dataset can be found at

4.1.1 PyTorch

`https://huggingface.co/datasets/eugenesiow/Div2k`

The DIV2K dataset is divided into:

1. train data: starting from 800 high definition high resolution images we obtain corresponding low resolution images and provide both high and low resolution images for 2, 3, and 4 downscaling factors

2. validation data: 100 high definition high resolution images are used for genereting low resolution corresponding images, the low res are provided from the beginning of the challenge and are meant for the participants to get online feedback from the validation server; the high resolution images will be released when the final phase of the challenge starts.

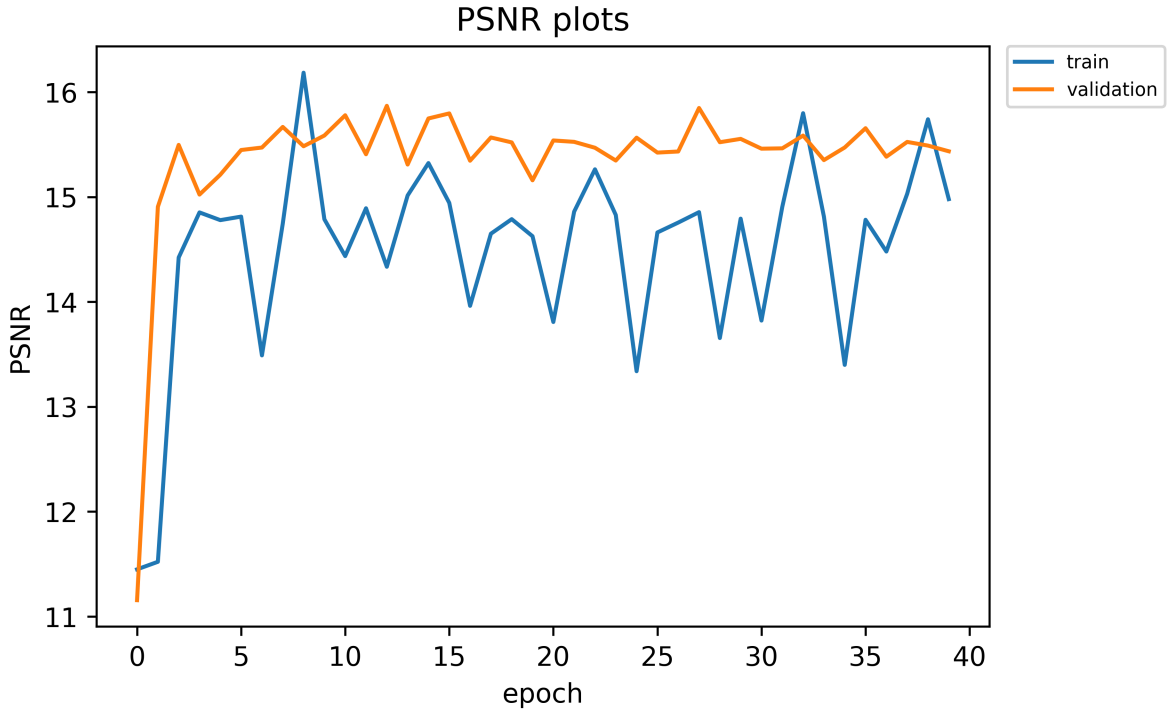
4.1.2 Tensorflow

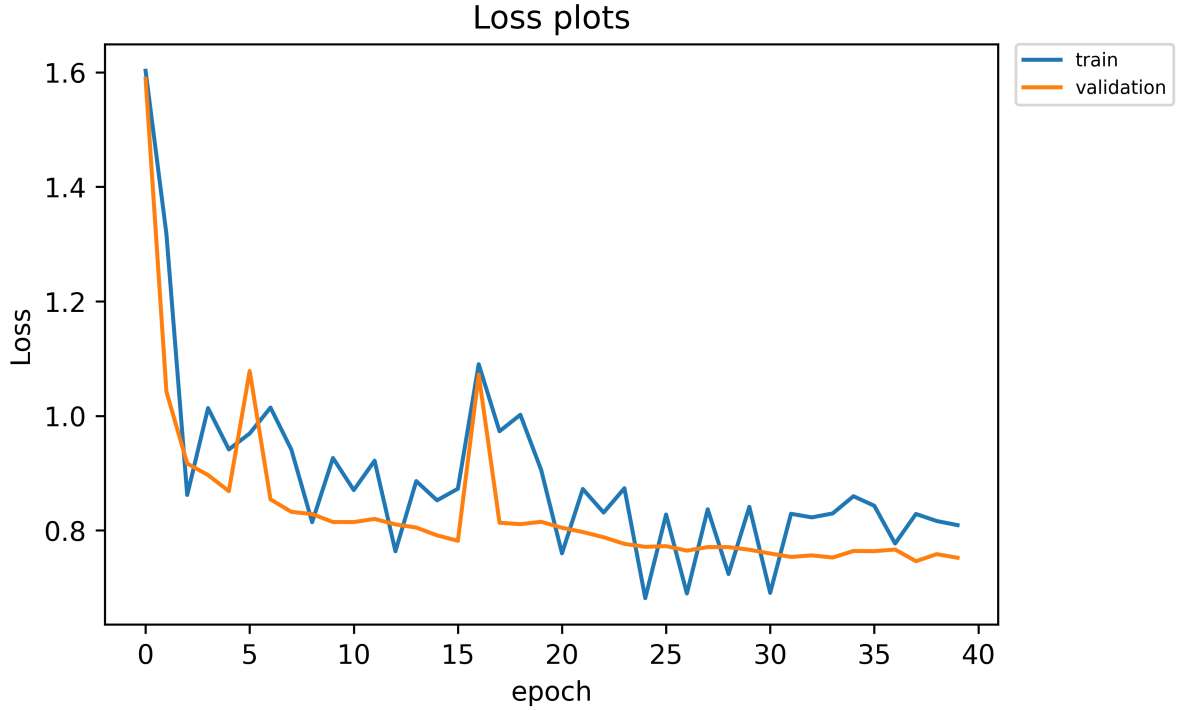
`https://www.tensorflow.org/datasets/catalog/div2k`

CONCLUSION In this report we have compared the performances of each algorithm for the task of super resolution. Finally,after implementing super resolution using these algorithms we conclude that SRGAN although the PSNR value is much smaller compared to the SRCNN and ESPCN, the SR images of SRGAN has better perceptual quality as it is a great indicator for the performance of the algorithm compared to computational efficiency with metric such as PSNR.

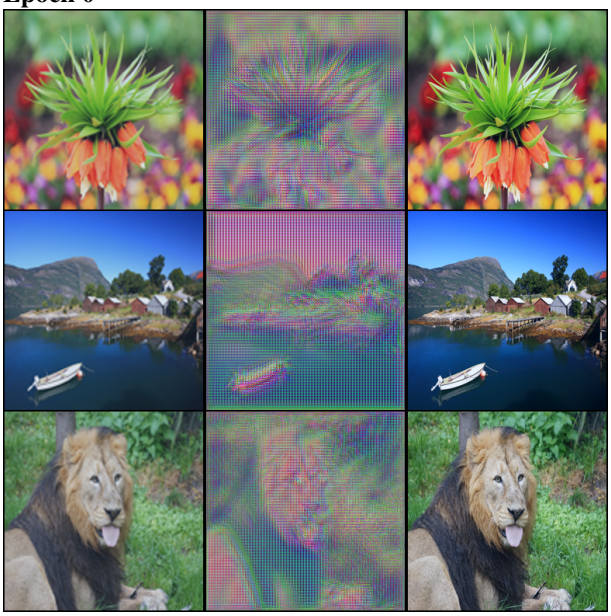
4.2 Results

4.2.1 SRGAN

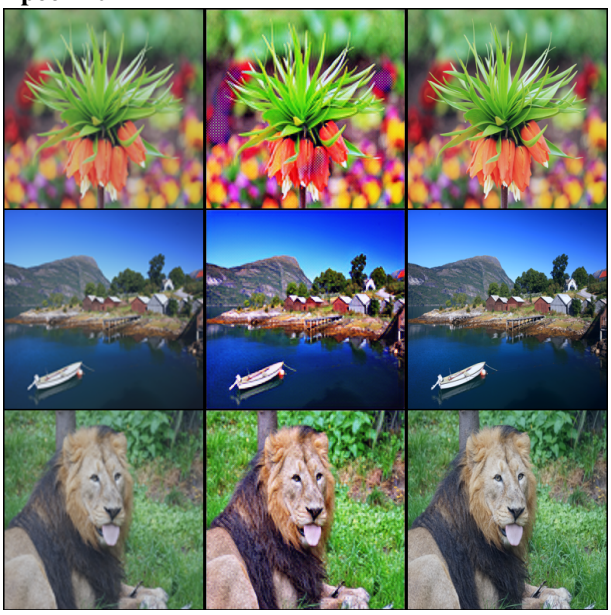




Epoch 0



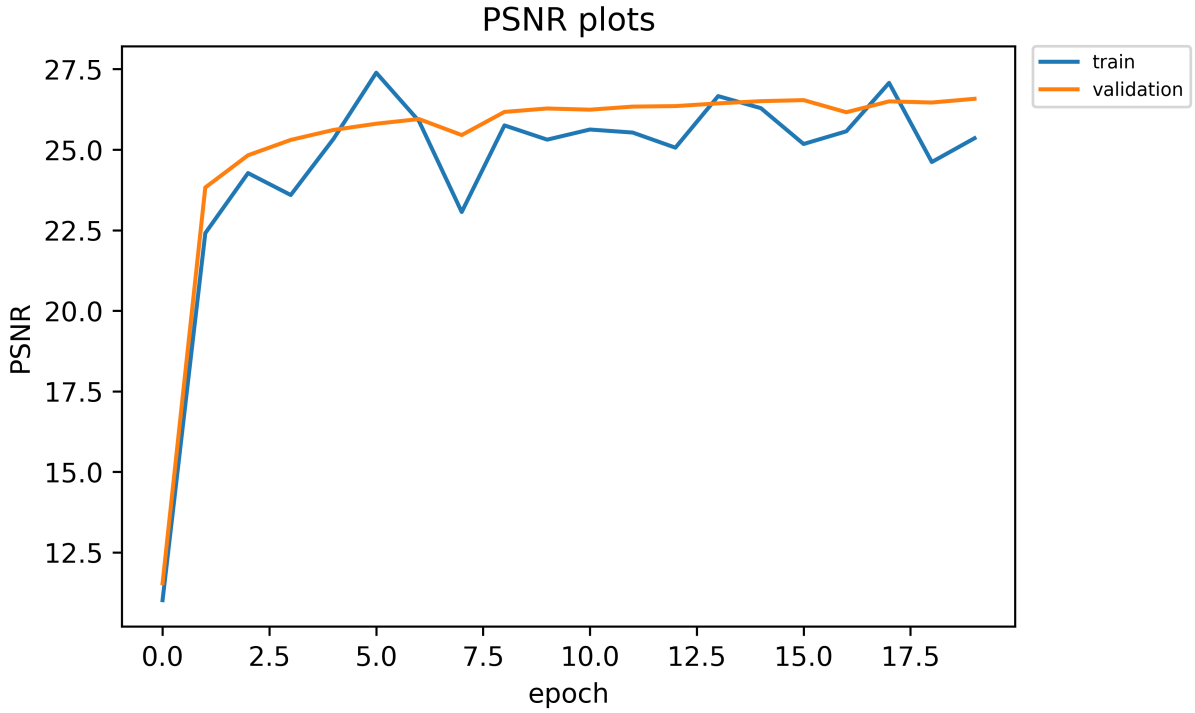
Epoch 10

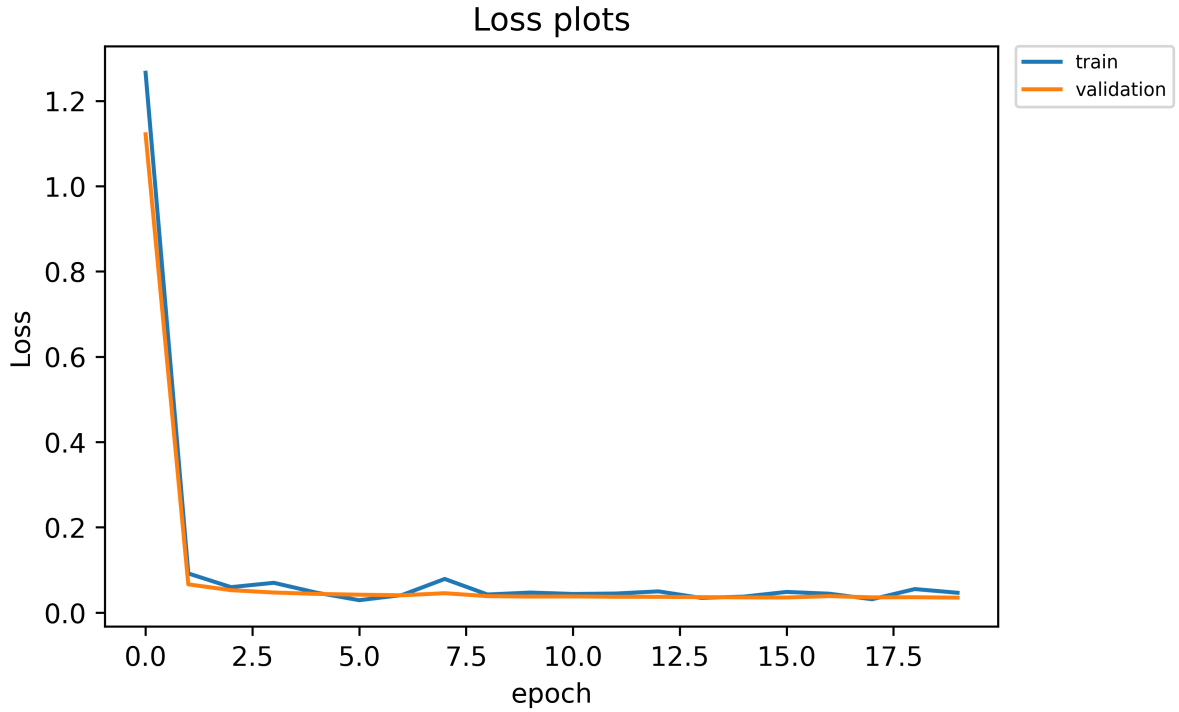


Epoch 20

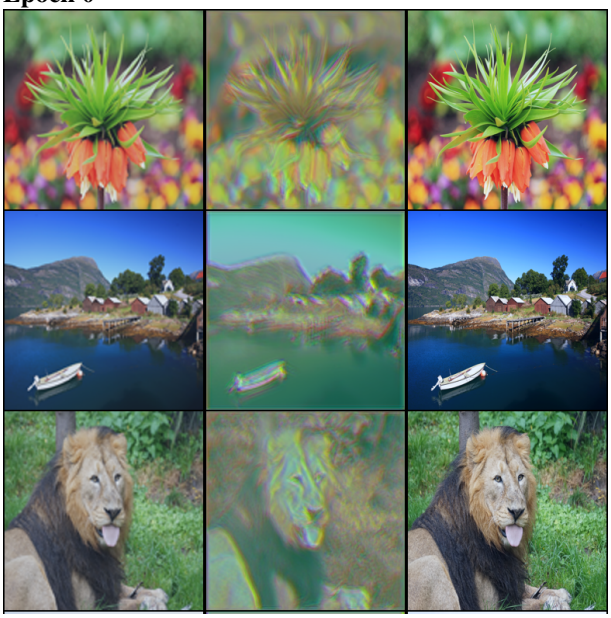
Epoch 21

4.2.2 SRCNN

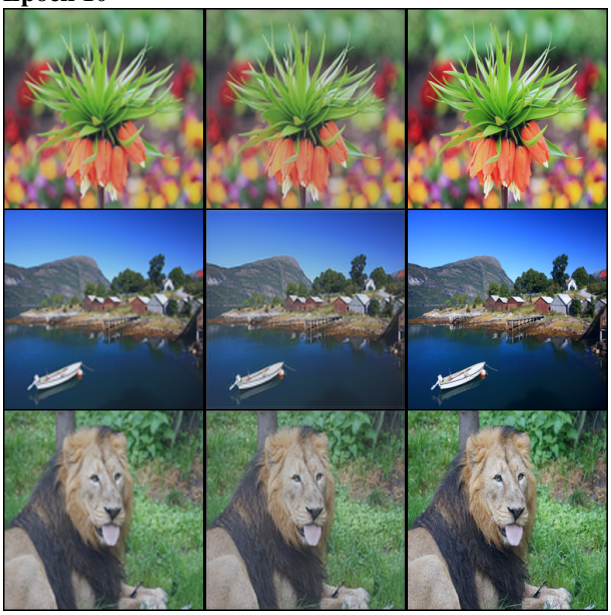

PSNR plots
PSNR
epoch
train
validation
27.5
25.0
22.5
20.0
17.5
15.0
12.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Loss plots
Loss
epoch
train
validation
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

Epoch 0



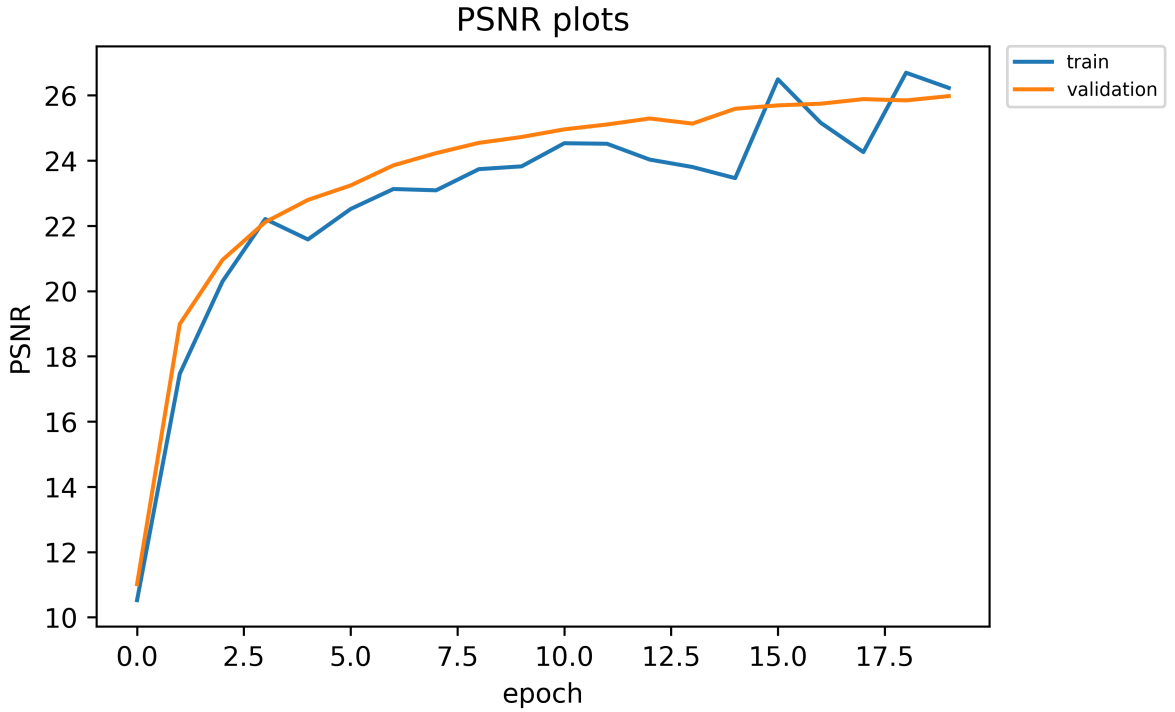
Epoch 10

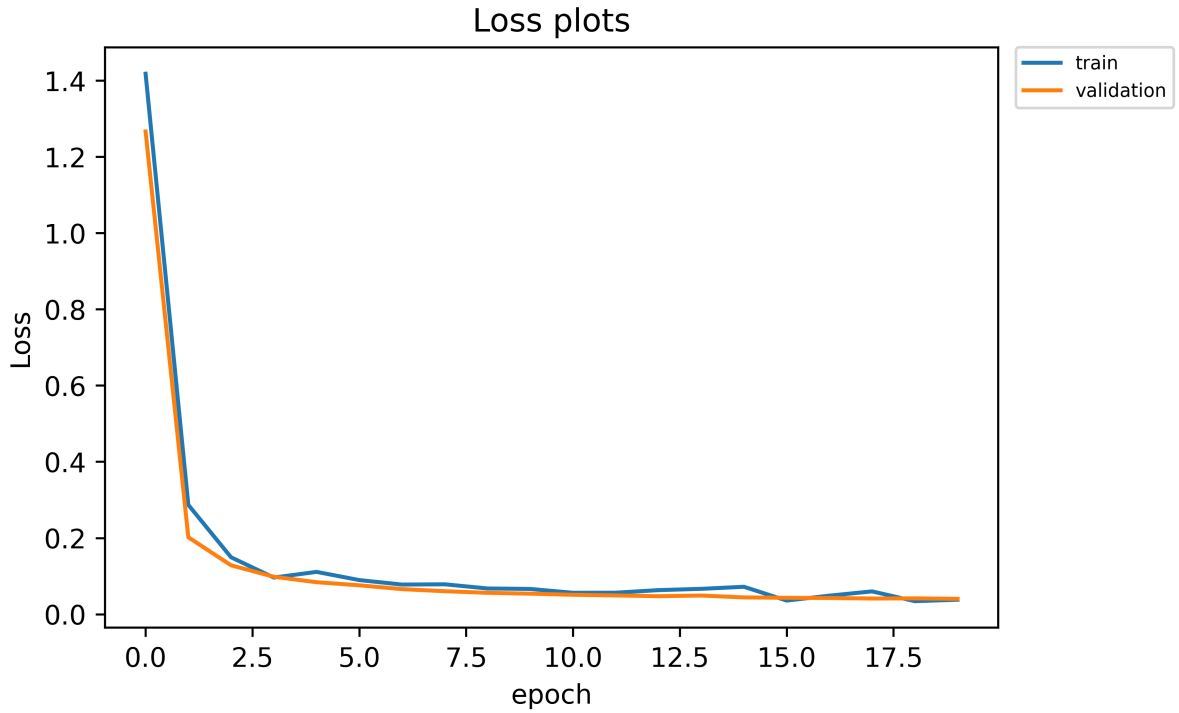


Epoch 20

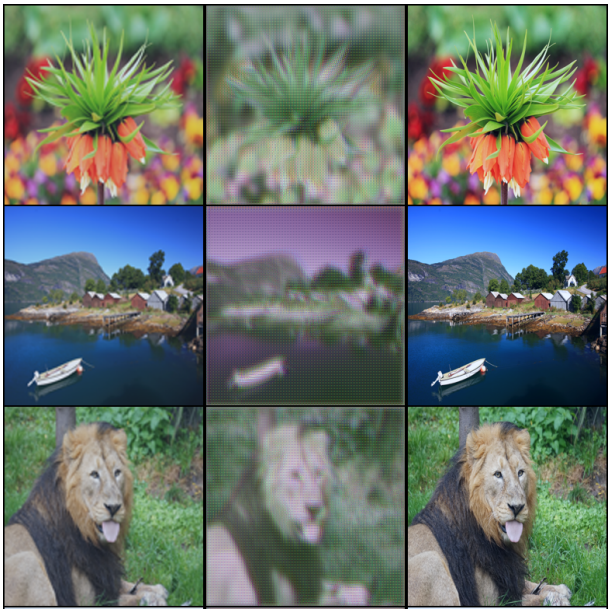
Epoch 21

4.2.3 ESPCN





Epoch 0



Epoch 10

Epoch 20

Epoch 21

References

References follow the acknowledgments. Use unnumbered first-level heading for the references. Any choice of citation style is acceptable as long as you are consistent. It is permissible to reduce the font size to `small` (9 point) when listing the references. Note that the Reference section does not count towards the page limit.

1. Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network
2. Chao Dong, Chen Change Loy, Kaiming He, Xiaoou Tang.Image Super-Resolution Using Deep Convolutional Networks
3. Wenzhe Shi1, Jose Caballero1, Ferenc Huszar, Johannes Totz1,Andrew P. Aitken1, Rob Bishop1, Daniel Rueckert1, Zehan Wang Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network

4. C. E. Duchon. Lanczos Filtering in One and Two Dimensions. In Journal of Applied Meteorology, volume 18, pages 1016–1022. 1979
5. J. Allebach and P. W. Wong. Edge-directed interpolation. In Proceedings of International Conference on Image Processing, volume 3, pages 707–710, 1996
6. S. Dieleman, J. Schluter, C. Raffel, E. Olson, S. K. Snderby, ¨ D. Nouri, D. Maturana, M. Thoma, E. Battenberg, J. Kelly, J. D. Fauw, M. Heilman, diogo149, B. McFee, H. Weideman, takacsg84, peterderivaz, Jon, instagibbs, D. K. Rasul, CongLiu, Britefury, and J. Degrave. Lasagne: First release., 2015
7. A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In Advances in Neural Information Processing Systems (NIPS)
8. C. Li and M. Wand. Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2479–2486, 2016.
9. J. Johnson, A. Alahi, and F. Li. Perceptual losses for real-time style transfer and super-resolution. In European Conference on Computer Vision (ECCV)
10. S. Gross and M. Wilber. Training and investigating residual nets, online at http://torch.ch/blog/2016/02/04/resnets. html.
11. K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In IEEE International Conference on Computer Vision (ICCV),
12.] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In International Conference on Learning Representations (ICLR), 2016
13. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In International Conference on Learning Representations (ICLR)
14. L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In Advances in Neural Information Processing Systems (NIPS), pages 262–270, 2015
15. J. Bruna, P. Sprechmann, and Y. LeCun. Super-resolution with deep convolutional sufficient statistics. In International Conference on Learning Representations (ICLR)
16. Burger, H.C., Schuler, C.J., Harmeling, S.: Image denoising: Can plain neural networks compete with BM3D? In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 2392–2399 (2012)
17. Schuler, C.J., Burger, H.C., Harmeling, S., Scholkopf, B.: A machine learning approach for non-blind image deconvolution. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1067–1074 (2013)
18. Jain, V., Seung, S.: Natural image denoising with convolutional networks. In: Advances in Neural Information Processing Systems. pp. 769–776 (2008)
19. Eigen, D., Krishnan, D., Fergus, R.: Restoring an image taken through a window covered with dirt or rain. In: IEEE International Conference on Computer Vision. pp. 633–640 (2013)
20. Cui, Z., Chang, H., Shan, S., Zhong, B., Chen, X.: Deep network cascade for image super-resolution. In: European Conference on Computer Vision, pp. 49–64 (2014)
21. Glasner, D., Bagon, S., Irani, M.: Super-resolution from a single image. In: IEEE International Conference on Computer Vision. pp. 349–356 (2009)
22. He, K., Sun, J.: Convolutional neural networks at constrained time cost. arXiv preprint arXiv:1412.1710 (2014)
23. M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In Computer Vision–ECCV 2014, pages 818–833. Springer, 2014.
24. J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. arXiv preprint arXiv:1411.4038, 2014.

25.] 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 (TPAMI), 2015
26. M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In IEEE International Conference on Computer Vision (ICCV), pages 2018–2025. IEEE, 2011
27. J. Yang, J. Wright, T. Huang, and Y. Ma. Image super-resolution as sparse representation of raw image patches. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–8, 2008.
28.] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based superresolution. IEEE Computer Graphics and Applications, 22(2):56–65, 2002.
29. W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning lowlevel vision. International Journal of Computer Vision, 40(1):25–47, 2000
30. R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In Curves and Surfaces, pages 711–730. Springer, 2012
31. D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In IEEE International Conference on Computer Vision (ICCV), pages 349–356, 2009
32. S. Gu, W. Zuo, Q. Xie, D. Meng, X. Feng, and L. Zhang. Convolutional sparse coding for image super-resolution. In IEEE International Conference on Computer Vision (ICCV), pages 1823– 1831. 2015
33. Y.-W. Tai, S. Liu, M. S. Brown, and S. Lin. Super Resolution using Edge Prior and Single Image Detail Synthesis. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2400– 2407, 2010
34. J. Sun, J. Sun, Z. Xu, and H.-Y. Shum. Image super-resolution using gradient profile prior. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–8, 2008
35. K. Zhang, X. Gao, D. Tao, and X. Li. Multi-scale dictionary for single image super-resolution. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
36. H. Yue, X. Sun, J. Yang, and F. Wu. Landmark image superresolution by retrieving web images. IEEE Transactions on Image Processing, 22(12):4865–4878, 2013.
37. Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang. Deep networks for image super-resolution with sparse prior. In IEEE International Conference on Computer Vision (ICCV),
38. K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In Proceedings of the 27th International Conference on Machine Learning (ICML-10),
39. C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In European Conference on Computer Vision (ECCV), pages 184–199. Springer, 2014
40. 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,
41. J. Kim, J. K. Lee, and K. M. Lee. Deeply-recursive convolutional network for image super-resolution. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR),
42.] C. E. Duchon. Lanczos Filtering in One and Two Dimensions. In Journal of Applied Meteorology, volume 18, pages 1016–1022
43. X. Li and M. T. Orchard. New edge-directed interpolation. IEEE Transactions on Image Processing, 10(10):1521–1527, 2001.

A Appendix

Optionally include extra information (complete proofs, additional experiments and plots) in the appendix. This section will often be part of the supplemental material.