Super Resolution Using GAN

A disquisition report submitted in attainment of the necessity

for the degree of Bachelor of Technology

by

Shambhawi (2016UCP1470)

Bhagwana Ram (2016UCP1389)

Priyanka Meena(2016UCP1437)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY

MAY 20, 2020

Certificate

We,

Shambhawi (2016UCP1470)

Bhagwana Ram (2016UCP1389)

Priyanka Meena(2016UCP1437)

Declare that this report titled, "**Super-Resolution Using GAN**" and the work presented in it are our own. I confirm that:

- This project work was done entirely while in candidature for a B.Tech. degree in the department of computer science and engineering at Malaviya National Institute of Technology Jaipur (MNIT).
- Where any part of this thesis has previously been submitted for a degree or any other qualification at MNIT or any other institution, this has been clearly stated. Where we have consulted the published work of others, this is always clearly allocated. Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this Dissertation is entirely our own work.
- We have acknowledged all the main sources of help.

Signed:

Date: MAY 20,2020

Dr. Santosh Kumar Vipparthi

Assistant Professor

Department of Computer Science and Engineering

Malaviya National Institute of Technology Jaipur

Abstract

The highly demanding task of approximate a high resolution (HR) image from its low-resolution (LR) equivalent is referred to as super-resolution (SR).In SR problems, the given image is usually suppose to be a low-pass filtered (A low-pass filter, also called a "blurring" or "smoothing" filter, averages out rapid changes in intensity) and downsampled (Downsampling is the reduction in spatial resolution while keeping the same two-dimensional (2D) representation) version of an HR image. The low-/high-resolution pairs and the down-sampling process are inaccessible.The conduct of optimization-based super-resolution methods is principally driven by the possibility of the objective function.

Acknowledgement

It is a thing of gratification and prerogative for us to present our B. Tech Project Report on “Super Resolution Using GAN”. We would like to appreciate our supervisor, Dr. Santosh Kumar Vipparthi , Assistant Professor, Department of Computer Science and Engineering, Malaviya National Institute of Technology Jaipur, for his time, exceptional counselling and inspiration all over the session. He has been the motivating force behind this and given the right superintendence to our project.

We also show our unfeigned appreciation towards all the teachers of the Department of Computer Science and Engineering and technical staff for all their counselling and support throughout our stay at MNIT Jaipur.

Shambhawi	Bhagwana Ram	Priyanka Meena
(2016UCP1470)	(2016UCP1389)	(2016UCP1437)

Contents

List of Figures

Chapter 1

Introduction

The central aim of Super-Resolution (SR) is to generate a higher resolution image from lower resolution images. High-resolution images offer a high pixel density and thereby more details about the original scene. The need for high resolution is common in computer vision applications for better performance in pattern recognition and analysis of images. High resolution is of importance in medical imaging for diagnosis. Many applications require zooming of a specific area of interest in the image wherein high resolution becomes essential, e.g. surveillance, forensic and satellite imaging applications. However, high-resolution images are not always available. This is since the setup for high-resolution imaging proves expensive and also it may not always be feasible due to the inherent limitations of the sensor, optics manufacturing technology. These problems can be overcome through the use of image processing algorithms, which are relatively inexpensive, giving rise to the concept of super-resolution. It provides an advantage as it may cost less and the existing low-resolution imaging systems can still be utilized.

.

.

Chapter 2

Important Terms and Concepts

2.1.CNN

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing, and financial time series. The name "convolutional neural network" indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.[1]

DESIGN

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with multiplication or other dot product. The activation function is commonly a RELU layer and is subsequently followed by additional convolutions such as pooling layers, fully connected layers, and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution. Though the layers are colloquially referred to as convolutions, this is only by convention. Mathematically, it is technically a sliding dot product or cross-correlation. This has significance for the indices in the matrix, in that it affects how weight is determined at a specific index point.

Convolutional

When programming a CNN, the input is a tensor with shape (number of images) x (image width) x (image height) x (image depth). Then after passing through a convolutional layer, the image becomes abstracted to a feature map, with shape (number of images) x (feature map width) x (feature map height) x (feature map channels). A convolutional layer within a neural network should have the following

attributes:

- Convolutional kernels defined by a width and height (hyper-parameters).
- The number of input channels and output channels (hyper-parameter).
- The depth of the Convolution filter (the input channels) must be equal to the number channels (depth) of the input feature map.

Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus.Each convolutional neural processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10,000 weights for each neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters. For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters. In this way, it resolves the vanishing or exploding gradients problem in training traditional multi-layer neural networks with many layers by using backpropagation

Pooling

Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2 x 2. Global pooling acts on all the neurons of the convolutional layer. In addition, pooling may compute a max or an average. Max pooling uses the maximum value from each of a cluster of neurons at the prior layer. Average pooling uses the average value from each of a cluster of neurons at the prior layer

Fully connected

Fully connected layers connect every neuron in one layer to each neuron in another layer. It is, in t heory, an equivalent because of the traditional multi-layer perceptron neural network (MLP). The fl attened matrix goes through a totally connected layer to classify the pictures.

Receptive field

In neural networks, each neuron receives input from some number of locations within the previous layer. In a fully connected layer, each neuron receives input from every element of the previous la yer. In a convolutional layer, neurons receive input from only a restricted subarea of the previous l ayer. Typically the subarea is of a square shape (e.g., size 5 by 5). The input area of a neuron is c alled its receptive field. So, during a fully connected layer, the receptive field is that the entire pre vious layer. In a convolutional layer, the receptive area is smaller than the whole previous layer.

Weights

Each neuron during a neural network computes an output value by applying a selected function to the input values coming from the receptive field within the previous layer. The function that's appli ed to the input values is decided by a vector of weights and a bias (typically real numbers). Learni ng, during a neural network, progresses by making iterative adjustments to those biases and weight s. The vector of weights and thus the bias are called filters and represent particular features of the input (e.g., a selected shape). A distinguishing feature of CNNs is that a lot of neurons can share a n equivalent filter. This reduces memory footprint because one bias and one vector of weights are used across all receptive fields sharing that filter, as against each receptive field having its own bia s and vector weighting.

2.2. RNN

A recurrent neural network (RNN) is an artificial neural network where a directed graph along a te mporal sequence is formed by connections between nodes. This allows it to exhibit temporal dyna mic behavior. Derived from feedforward neural networks, recurrent neural networks can use their i nternal state to process variable-length sequences of inputs. This makes them applicable to tasks lik e unsegmented, connected handwriting recognition or speech recognition. The term “recurrent neura l network” is employed indiscriminately to check with two broad classes of networks with an ident ical general structure, where one is finite impulse and therefore the other is infinite impulse. Both classes of networks exhibit temporal dynamic behavior.[5] A finite impulse recurrent network is a directed acyclic graph which will be unrolled and replaced with a strictly feed-forward neural netw ork, whereas an infinite impulse recurrent network may be a directed cyclic graph which will not b e unrolled. Both finite impulse and infinite impulse recurrent networks can have additional stored s tates, and therefore the storage are often under direct control by the neural network. The storage al so can get replaced by another network or graph if that comes with time delays or has feedback lo ops. Such controlled states are mentioned as gated state or gated memory and are a part of long S TM networks (LSTMs) and gated recurrent units. This is also called Feedback Neural Network.[2]

2.3.Bicubic

Bicubic interpolation is a standard method in the image interpolation field because of

its low complexity and relatively good results. But as it only interpolates in horizontal and vertical directions, edges easily suffer from artifacts such as blocking, blurring and ringing. This paper proposed a new method of image super-resolution which is named directional bicubic interpolation. According to local strength and directions, different ways are used to interpolate missing pixels. Compared with bicubic interpolation, the proposed method can preserve sharp edges and details better. Experiment results show that the proposed method is better than existing edge-directed interpolations in terms of subjective and objective measures, and its computation complexity is low.[3]

2.4.ResNet

A residual neural network (ResNet) is an artificial neural network (ANN) of a kind that builds on constructs known from pyramidal cells in the cerebral cortex. Residual neural networks do this by utilizing skip connections, or shortcuts to jump over some layers. Typical ResNet models are implemented with double- or triple-layer skips that contain nonlinearities (ReLU) and batch normalization in between.An additional weight matrix may be used to learn the skip weights; these models are known as HighwayNets. Models with several parallel skips are referred to as DenseNets. In the context of residual neural networks, a non-residual network may be described as a plain network. One motivation for skipping over layers is to avoid the problem of vanishing gradients, by reusing activations from a previous layer until the adjacent layer learns its weights. During training, the weights adapt to mute the upstream layer, and amplify the previously-skipped layer. In the simplest case, only the weights for the adjacent layer's connection are adapted, with no explicit weights for the upstream layer. This works best when a single nonlinear layer is stepped over, or when the intermediate layers are all linear. If not, then an explicit weight matrix should be learned for the skipped connection (a Highway Net should be used). Skipping effectively simplifies the network, using fewer layers in the initial training stages. This speeds learning by reducing the impact of vanishing gradients, as there are fewer layers to propagate through. The network then gradually restores the skipped layers as it learns the feature space. Towards the end of the training, when all layers are expanded, it stays closer to the manifold and thus learns faster. A neural network without residual parts explores more of the feature space. This makes it more vulnerable to perturbations that cause it to leave the manifold and necessitates extra training data to recover.[4]

2.5.GAN

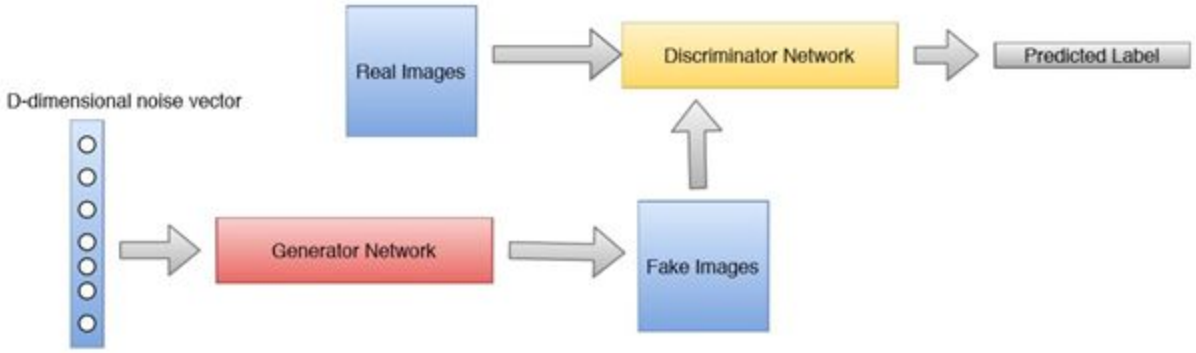

Fig 1:GAN

A generative adversarial network (GAN) is a class of machine learning systems invented by Ian Good fellow and his colleagues in 2014. Two neural networks contesting with each other in a game (in the sense of game theory, often but not always in the form of a zero-sum game). Given a training set, this technique learns to generate new data with the same statistics as the training set. For example, a GAN trained on photographs can generate new photographs that look at least superficially authentic to human observers, having many realistic characteristics. Though originally proposed as a form of generative model for unsupervised learning, GANs have also proven useful for semi-supervised learning, fully supervised learning and reinforcement learning. In a 2016 seminar, Yann Le Cun described GANs as "the coolest idea in machine learning in the last twenty years. The The generative network generates candidates while the discriminative network evaluates them. The contest operates in terms of data distributions. Typically, the generative network learns to map from a latent space to a data distribution of interest, while the discriminative network distinguishes candidates produced by the generator from the true data distribution. The generative network's training objective is to increase the error rate of the discriminative network (i.e., "fool" the discriminator network by producing novel candidates that the discriminator thinks are not synthesized (are part of the true data distribution)). A known dataset serves as the initial training data for the discriminator. Training involves presenting it with samples from the training dataset until it achieves acceptable accuracy. The generator trains based on whether it succeeds in fooling the discriminator. Typically the generator is seeded with randomized input that is sampled from a predefined latent space (e.g. a multivariate normal distribution). Thereafter, candidates synthesized by the generator are evaluated by the discriminator. Backpropagation is applied in both networks so that the generator produces better images, while the discriminator becomes more skilled at flagging synthetic images. The generator is typically a deconvolutional neural network, and the discriminator is a convolutional neural network.[5]

GAN can be used to detect glaucomatous images helping the early diagnosis which

is essential to avoid partial or total loss of vision. GANs that produce photorealistic images can be used to visualize the interior design, industrial design, shoes, bags, and clothing items or items for computer games' scenes. Such networks were reported to be used by Facebook. GANs can reconstruct 3D models of objects from images and model patterns of motion in the video. GANs can be used to age face photographs to show how an individual's appearance might change with age. GANs can also be used to transfer map styles in cartography or augment street view imagery.

A variation of the GANs is used in training a network to generate optimal control inputs to nonlinear dynamical systems. Where the discriminatory network is known as a critic that checks the optimality of the solution and the generative network is known as an Adaptive network that generates optimal control. The critic and adaptive network train each other to approximate a nonlinear optimal control. GANs have been used to visualize the effect that climate change will have on specific houses. A GAN model called Speech2Face can reconstruct an image of a person's face after listening to their voice. In 2016 GANs were used to generate new molecules for a variety of protein targets implicated in cancer, inflammation, and fibrosis. In 2019 GAN-generated molecules were validated experimentally all the way into mice.

Bidirectional GAN

Bidirectional GAN(BiGAN) aims to introduce a generator model to act as the discriminator, whereby the discriminator naturally considers the entire translation space so that the inadequate training problem can be alleviated. To satisfy this property, generator and discriminator are both designed to model thejoint probability of sentence pairs, with the difference that, the generator decomposes the joint probability with a source language model and a source-to-target translation model, while the discriminator is formulated as a target language model and a target-to-source translation model. To further leverage the symmetry of them, an auxiliary GAN is introduced and adopts generator and discriminator models of original one as its own discriminator and generator respectively. Two GANs are alternately trained to update the parameters. The resulting learned feature representation is useful for auxiliary supervised discrimination tasks, competitive with contemporary approaches to unsupervised and self-supervised feature learning.

2.6.MSE

The MSE is a measure of the quality of an estimator—it is always non-negative, and values closer to zero are better. Mean squared error (MSE) or mean squared deviation (MSD) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors—that is, the average

squared difference between the estimated values and the actual value. MSE is a risk function, corresponding to the expected value of the squared error loss. The fact that MSE is almost always strictly positive (and not zero) is because of randomness or because the estimator does not account for information that could produce a more accurate estimate. The MSE is the second moment (about the origin) of the error, and thus incorporates both the variance of the estimator (how widely spread the estimates are from one data sample to another) and its bias (how far off the average estimated value is from the truth). For an unbiased estimator, the MSE is the variance of the estimator. Like the variance, MSE has the same units of measurement as the square of the quantity being estimated. In an analogy to standard deviation, taking the square root of MSE yields the root-mean-square error or root-mean-square deviation (RMSE or RMSD), which has the same units as the quantity being estimated; for an unbiased estimator, the RMSE is the square root of the variance, known as the standard error. The MSE assesses the quality of a predictor (i.e., a function mapping arbitrary inputs to a sample of values of some random variable), or an estimator (i.e., a mathematical function mapping a sample of data to an estimate of a parameter of the population from which the data is sampled). The definition of an MSE differs according to whether one is describing a predictor or an estimator.[6]

Predictor

If a vector n prediction generated from a sample of n data points on all variables, and Y is the vector of observed values of the variable being predicted, with being the predicted values (e.g. as from a least-squares fit), then the within-sample MSE of the predictor is computed as:

$$\mathbf{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2. \qquad (1)$$

$$\left(\frac{1}{n}\sum_{i=1}^{n}\right)$$

I.e., the MSE is the mean of the squares of the errors This is an easily computable quantity for a particular sample (and hence is sample-dependent). The MSE can also be calculated on q data poin ts that weren't used in estimating the model.In this process, which is understood as cross-validation, the MSE is usually called the mean squared prediction error and is computed as

$$\mathrm{MSPE} = \frac{1}{q}\sum_{i=n+1}^{n+q}(Y_i - \hat{Y}_i)^2. \quad (2)$$

Estimator

The MSE of an estimator with reference to an unknown parameter is defined as

$$\mathrm{MSE}(\hat{\theta}) = \mathrm{E}_\theta\left[(\hat{\theta} - \theta)^2\right]. \quad (3)$$

This definition depends on the unknown parameter, but the MSE may be a priori property of an es timator. The MSE might be a function of unknown parameters, during which case any estimator of the MSE supported estimates of those parameters would be a function of the data and thus a rand om variable. If the estimator is derived as a sample statistic and is used to estimate some populatio n parameter, then the expectation is with respect to the sampling distribution of the sample statistic . The MSE is often written because of the sum of the variance of the estimator and therefore the s quared bias of the estimator, providing a useful thanks to calculating the MSE and implying that w ithin the case of unbiased estimators, the MSE and variance are equivalent.

$$\mathrm{MSE}(\hat{\theta}) = \mathrm{Var}_\theta(\hat{\theta}) + \mathrm{Bias}(\hat{\theta}, \theta)^2. \quad (4)$$

2.7. MOS

Mean opinion score (MOS) is utilized in the domain of Quality of Experience and telecommunicati ons engineering, denoting the general quality of a system. It is the first moment overall individual "values on a predefined scale that a topic assigns to his opinion of the performance of a system qu ality". Such ratings are usually gathered during a subjective quality evaluation test, but they will ev en be algorithmically estimated.MOS may be a commonly used measure for video, audio, and audi ovisual quality evaluation, but not restricted to those modalities. ITU-T has defined several ways of pertaining to a MOS in Recommendation P.800.1, counting on whether the score was obtained fro m audiovisual, conversational, listening, talking, or video quality tests. The MOS is expressed as o ne real number, typically within the range, 1–5, where 1 is the lowest perceived quality, and 5 is t hat the highest perceived quality. Other MOS ranges also are possible, counting on the rating scale that has been utilized in the underlying test. The Absolute Category Rating scale is very commonl y used, which maps ratings between Bad and Excellent to numbers between 1 and 5.[7]

The MOS is calculated because of the first moment over single ratings performed by human subjec ts for a given stimulus during a subjective quality evaluation test. Thus:

$$MOS = \frac{\sum_{n=1}^{N} R_n}{N} \quad (5)$$

Where R the individual ratings for a given stimulus by N subjects.

it is not meant to directly compare MOS values produced from separate experiments unless those experiments were explicitly designed to be compared, and even then the data should be statistically analyzed to ensure that such a comparison is valid.

Obtaining MOS ratings may be time-consuming and expensive as it requires the recruitment of human assessors. For various use cases such as codec development or service quality monitoring purposes – where quality should be estimated repeatedly and automatically – M OS scores can also be predicted by objective quality models, which typically have been de veloped and trained using human MOS ratings. A question that arises from using such mo dels is whether the MOS differences produced are noticeable to the users. For example, wh en rating images on a five-point MOS scale, an image with a MOS equal to 5 is expecte d to be noticeably better in quality than one with a MOS equal to 1. Contrary to that, it is not evident whether an image with a MOS equal to 3.8 is noticeably better in quality than one with a MOS equal to 3.6. Research conducted on determining the smallest MOS difference that is perceptibl to users for digital photographs showed that a MOS difference of approximately 0.46 is required in order for 75% of the users to be able to detect the higher quality image. Nevertheless, image quality expectations, and hence MOS, change ov er time with the change of user expectations. As a result, minimum noticeable MOS differe nces determined using analytical methods such as in may change over time.

2.8.VGG Network

It usually refers to a deep convolutional network for object recognition developed and train ed by Oxford's renowned Visual Geometry Group (VGG), which achieved very good perfor

mance on the ImageNet dataset. [8]

Input

VGG takes in a 224x224 pixel RGB image. For the ImageNet competition, the authors cro pped out the center 224x224 patch in each image to keep the input image size consistent.

Convolutional Layers

The convolutional layers in VGG use a very small receptive field (3x3, the smallest possib le size that still captures left/right and up/down). There are also 1x1 convolution filters whi ch act as a linear transformation of the input, which is followed by a ReLU unit. The co nvolution stride is fixed to 1 pixel so that the spatial resolution is preserved after convolut ion.

Fully-Connected Layers

VGG has three fully-connected layers: the first two have 4096 channels each and the third has 1000 channels, 1 for each class.

Hidden Layers

All of VGG's hidden layers use ReLU (a huge innovation from AlexNet that cut training t ime). VGG does not generally use Local Response Normalization (LRN), as LRN increases memory consumption and training time with no particular increase in inaccuracy.

2.9.Common DATASETS: set5, set14, BSD 100, BSD300, ImageNet

A data set (or dataset) is a collection of data. In the case of tabular data, a data set corr esponds to one or more database tables, where every column of a table represents a partic ular variable, and each row corresponds to a given record of the data set in question. The data set lists values for each of the variables, such as the height and weight of an objec t, for each member of the data set. Each value is known as a datum. Data sets can also consist of a collection of documents or files.

Set5: Set5 test dataset

Set14: Set14 test dataset

BSD100: A subset (test) of BSD500 for testing

BSD300: A subset (test) of BSD500 for testing

DIV2K: proposed in NTIRE17 (800 train and 100 validation

Flickr 2K: 2650 2K images from Flickr for training

ImageNet

The ImageNet project is a large visual database designed for use in visual object recognitio n software research. More than 14 million images have been hand-annotated by the project to indicate what objects are pictured and in at least one million of the images, bounding boxes are also provided.ImageNet contains more than 20,000 categories with a typical cate gory, such as "balloon" or "strawberry", consisting of several hundred images. The database of annotations of third-party image URLs is freely available directly from ImageNet, thoug h the actual images are not owned by ImageNet. Since 2010, the ImageNet project runs an annual software contest, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) , where software programs compete to correctly classify and detect objects and scenes. The challenge uses a "trimmed" list of one thousand non-overlapping classes. a convolutional n eural network (CNN) called AlexNet achieved a top-5 error of 15.3% in the ImageNet 201 2 Challenge, more than 10.8 percentage points lower than that of the runner up. This was made feasible due to the use of graphics processing units (GPUs) during training, an esse ntial ingredient of the deep learning revolution. According to The Economist, "Suddenly peo ple started to pay attention, not just within the AI community but across the technology in dustry as a whole. In 2015, AlexNet was outperformed by Microsoft's very deep CNN with over 100 layers, which won the ImageNet 2015 contest.[9]

2.10. PSNR

Peak signal-to-noise (PSNR) is an engineering term for the ratio between the maximum possible po wer of a symbol and the corrupting noise that affects its representation. Because many signals have a really wide dynamic range, PSNR is typically expressed in terms of the logarithmic decibel scal e.PSNR is most easily defined via the mean squared error (MSE). MSE for a noise free monochro matic m*n image with noisy approximation K is defined as:

$$MSE = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \tag{6}$$

The PSNR (in dB) is defined as:

$$\begin{aligned} PSNR &= 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \\ &= 20 \cdot \log_{10} \left(\frac{MAX_I}{\sqrt{MSE}} \right) \\ &= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE) \end{aligned} \tag{7}$$

Here, MAXI is the maximum possible pixel value of the image. When the pixels are represented u sing 8 bits per sample, this is often 255.

PSNR most ordinarily wont to measure the standard of reconstruction of lossy compression codecs(e.g., for image compression). The signal, during this case, is that of the original data, and therefore the noise is that of the error introduced by compression. When comparing compression codecs, PS NR is an approximation to the human perception of reconstruction quality. Typical values for the P SNR within the lossy image and video compression are between 30 and 50 dB, provided the bit de pth is 8 bits, where higher is best . Although a better PSNR generally indicates that the reconstruct ion is of upper quality, in some cases it's going to not. One has got to be extremely careful with th e range of validity of this metric; it's only conclusively valid when it's wont to compare results fro m an equivalent codec (or codec type) and the same content. Generally, PSNR shows poor quality than other quality metrics when it involves estimating the quality of images and particularly videos as perceived by humans.[10]

2.11. SSIM

The difference with regard to other techniques mentioned previously like MSE or PSNR is that the

se approaches estimate absolute errors; on the other hand, SSIM could also be a perception-based model that considers image degradation as perceived change in structural information, while also in corporating important perceptual phenomena, including both luminance masking and contrast maski ng terms. Structural information is that the concept of the pixels have strong inter-dependencies esp ecially once they are spatially close. These dependencies carry important information about the stru cture of the objects within the visual scene. Luminance masking could also be a phenomenon wher eby image distortions (in this context) tend to be less visible in bright regions, while contrast mask ing could also be a phenomenon whereby distortions subsided visible where there's a big activity o r "texture" within the image. The SSIM index is calculated on various windows of a picture . The SSIM between two windows(call it x and y) of common size N×N is:

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (8)$$

SSIM has applications during a sort of different problems. Some examples are:

Image Compression:

In lossy image compression, information is deliberately discarded to decrease the storage space of i mages and videos. The MSE is usually utilized in such compression schemes. According to its auth ors, using SSIM rather than MSE is usually recommended to supply better results for the decompre ssed images.[12]

Image Restoration:

Image restoration focuses on solving the problem $y = h * x + n$ (9) where y is the blurry image that should be restored, h is the blur kernel, n is the additive noise and x is the original image we wish to recover. The traditional filter which is employed to unravel this problem is that the Wiener Filter. However, the Wiener filter design is predicated on the MSE . Using an SSIM variant, specifically Stat-SSIM, is claimed to supply better visual results, consiste nt with the algorithm's authors.

Pattern Recognition:

Since SSIM mimics aspects of human perception, it might be used for recognizing patterns.The alg orithm's authors claim that it's better to use CW-SSIM.Since data-driven pattern recognition approa

ches may produce better performance when an outsized amount of knowledge is out there for traini ng, the authors suggest using CW-SSIM in data-driven approaches, when issues like image scaling, rotation and translation, etc are faced .

Due to its popularity, SSIM is usually compared to other metrics, including more simple metrics li ke MSE and PSNR, and other perceptual image and video quality metrics. SSIM has been repeated lyshown to significantly outperform MSE and its derivatives in accuracy, including research by its own authors and others. A paper by Dosselmann and Yang claims that the performance of SSIM is "much closer to that of the MSE" than usually assumed. While they do not dispute the advantage of SSIM over MSE, they state an analytical and functional dependency between the two metrics. According to their research, SSIM has been found to correlate as well as MSE-based methods on subjective databases other than the databases from SSIM's creators. As an example, they cite Reibman and Poole, who found that MSE outperformed SSIM on a database containing packet-loss–impaired video. In another paper, an analytical link between PSNR and SSIM was identified.[11]

2.12.Image Scaling: Downsampling and Upsampling

Downsampling and upsampling are two fundamental and widely used image operations, with applications in image display, compression, and progressive transmission. Downsampling is the reduction in spatial resolution while keeping the same two-dimensional (2D) representation. It is typically used to reduce the storage and/or transmission requirements of images. Upsampling is the increasing of the spatial resolution while keeping the 2D representation of an image. It is typically used for zooming in on a small region of an image, and for eliminating the pixelation effect that arises when a low-resolution image is displayed on a relatively large frame. More recently, downsampling and upsampling have been used in combination: in lossy compression, multiresolution lossless compression, and progressive transmission.

The standard methods for down/upsampling are decimation/duplication and bilinear interpolation, which yield low visual performance. The increasing use of

down/upsampling, especially in combination, warrant the development of better methods for them. In this paper, we examine the existing methods and propose new down/upsampling combination methods, and formulate a frequency-response approach for evaluating them. The approach is validated experimentally. Our findings show that the best down/upsampling filters are what we term binomial filters and some well-chosen biorthogonal wavelets. Bilinear interpolation was found significantly inferior, and decimation duplication came last.[12]

2.13.Image Quality

Image quality can refer to the level of accuracy in which different imaging systems capture, process, store, compress, transmit and display the signals that form an image. Another definition refers to image quality as "the weighted combination of all of the visually significant attributes of an image". The difference between the two definitions is that one focuses on the characteristics of signal processing in different imaging systems and the latter on the perceptual assessments that make an image pleasant for human viewers.

Image quality should not be mistaken with image fidelity. Image fidelity refers to the ability of a process to render a given copy in a perceptually similar way to the original (without distortion or information loss), i.e., through a digitization or conversion process from analog media to digital image. The process of determining the level of accuracy is called Image Quality Assessment (IQA). Image quality assessment is part of the quality of experience measures. Image quality can be assessed using two methods: subjective and objective. Subjective methods are based on the perceptual assessment of a human viewer about the attributes of an image or set of images, while objective methods are based on computational models that can predict perceptual image quality. Objective and subjective methods aren't necessarily consistent or accurate between each other: a human viewer might perceive stark differences in quality in a set of images where a computer algorithm might not.

Subjective methods are costly, require a large number of people, and are impossible

to automate in real-time. Therefore, the goal of image quality assessment research is to design algorithms for objective assessment that are also consistent with subjective assessments. The development of such algorithms has a lot of potential applications. They can be used to monitor image quality in control quality systems, to benchmark image processing systems and algorithms and to optimize imaging systems.[13]

Chapter 3

Related work

3.1. Deep learning-based image super-resolution

Deep learning has shown its higher performance in different computer vision tasks including image Super Resolution. Dong et al. [14] firstly introduced a three-layer CNN in image SR to learn a complicated LR-HR mapping. Kim et al. [15] increased the layer of CNN to twenty layers for more contextual information usage in LR images. In [15], a jump link beat the problem of optimization when the network became contain more layers. latest studies have various types of jump links to attain improvement in image Super Resolution. SRResNet[16] and EDSR[17] applied residual jump links from [18]. SRDenseNet[19] applied dense jump links from [20]. Zhang et al. [21] integrated local or global residual and dense jump links in their RDN. Since the jump links in these network architectures use or integrate hierarchical features in bottom-up way, the low-level features can only obtain the knowledge from forehand layers, lacking enough contextual information because of the limitation of small receptive fields. These low-level features are again used within the following layers, and thus stop the reconstruction ability of the network. So we propose a superresolution feedback network (SRFBN) to mend this issue, within which high-level information Stream through feedback links in a top down manner to accurate low-level features using more contextual information. The assistance of jump links, neural networks go deeper and hold more parameters . Such huge capacity networks acquire large amounts of storage resources and tolerate overfitting. To emphatic lower network parameters and gain better generalization power, the recurrent structure employed. Particularly, the recurrent structure plays a crucial role to understand the feedback process within the proposed SRFBN .

3.2. Feedback mechanism

The feedback mechanism allows the network to hold a notion of output to accurate former states. Latest, the feedback mechanism has been used by many network architectures for

various vision tasks. For image SuperResolution, a couple of studies also gave efforts to introduce the feedback mechanism. supported back-projection, “Haris et al.” [22] constructed up and down-projection units to realize repetitive error feedback. “Han et al.” [23] employed a delayed feedback mechanism that transmits the knowledge between two recurrent states during a dual-state RNN. However, the flow of data from the LR image to the ultimate SR image remains feedforward in their network architectures unlike ours. The most relevant work to ours is [24], which transfers the hidden state with high-level information to the knowledge of an input image to understand feedback during a convolutional recurrent neural network. However, it aims at solving high-level vision tasks, e.g. classification. to suit a feedback mechanism in image SR, we elaborately design a feedback block (FB) because the basic module in our SRFBN, rather than using ConvLSTM as in [24]. the knowledge in our FB efficiently flows across hierarchical layers through dense jump jumps. Experimental results indicate our FB has superior reconstruction performance than ConvLSTM 1 and thus is more suitable for image SR tasks.

3.3.Image super-resolution.

Early attempts on superresolution using CNNs used standard L p losses for training which led to blurry super-resolved images. To alleviate this, instead of using an MSE over pixels (between the super-resolved and therefore the ground truth HR image), the authors of [25] proposed an MSE over feature maps, coined perceptual loss. Notably, we also use a perceptual loss in our method. More recently, in [20], the authors presented a GAN-based approach which uses a discriminator to differentiate between the super-resolved and therefore therefore the original HR images and the perceptual loss. In [26], a patch-based texture loss is proposed to enhance reconstruction quality. Notice that each one the aforementioned image superresolution methods are often applied to all or any sorts of images and hence don't incorporate face-specific information, as proposed in our work. Also, in most cases, the aim is to supply high-fidelity images given a picture that's already of excellent resolution (usually 128 × 128) while face superresolution methods typically report results on very low resolution faces (16 × 16 or 32 × 32). From all the above mentioned methods, our work is more closely associated with [25]. One of our contributions is to explain an improved GAN-based architecture for super-resolution, which we used as a robust baseline on top of which we

built our integrated face superresolution and alignment network.

3.4. Perception Oriented Super-Resolution

The problem of distortion-oriented models recently drew the eye of researchers that the super-resolved results often lack the high-frequency details and aren't perceptually satisfying. Also, Blau et al. [27] showed that there's a trade-off between the perceptual quality and distortion, and a few perception-oriented models are proposed accordingly. for instance , Johnson et al. [28] have shown that the loss within the pixel domain isn't optimal for the perceptual quality, and instead, the loss within the feature space could be closer to the human perception model. Then, Ledig et al.[29] introduced the SRGAN which adopted the generative model with GAN and employed the perceptual loss as in [28]. Hence, unlike the distortion-oriented methods that produce the typical of possible HR images, the SRGAN generates one among the candidates within the multi-modal target HR space. EnhanceNet [26] goes one step further by exploiting the feel loss [30] for better producing image details. However, thanks to the inherent property of one-to-many inverse problem, it's required to think about the semantics for the generated pixels. In this respect, SFT-GAN [31] restricts the feature space by conditioning the semantic categories of target pixels**.**

Chapter 4

Proposed Method

In our work we target to produce a corresponding superresolved image I^{SR} from a low-resolution input image I^{LR}. I^{LR} (the LR image) is obtained by downsampling of its high resolution version I^{HR} using bicubic kernel by a factor of 4.If $I^{LR}$is of size $W \times H \times C$,then we have I^{HR}, I^{SR} of size $4*W \times 4*H \times 4*C$ .We aim to train a network that produces corresponding HR version for a its LR input image.In We train a feed-forward CNN $G_{\theta G}$ generator network with parameter θG,where θG = {W1:L;b1:L} are weights and biases.They are obtained by converging the model using our own loss function. For training images I^{HR} and I^{LR} , we solve the following optimization function:

$$\hat{\theta}_G = \arg\min_{\theta_G} \frac{1}{N} \sum_{n=1}^{N} l^{SR}(G_{\theta_G}(I_n^{LR}), I_n^{HR}) \tag{10}$$

Now we will design our own perceptual loss function.The loss functions are described in Section 4.2.

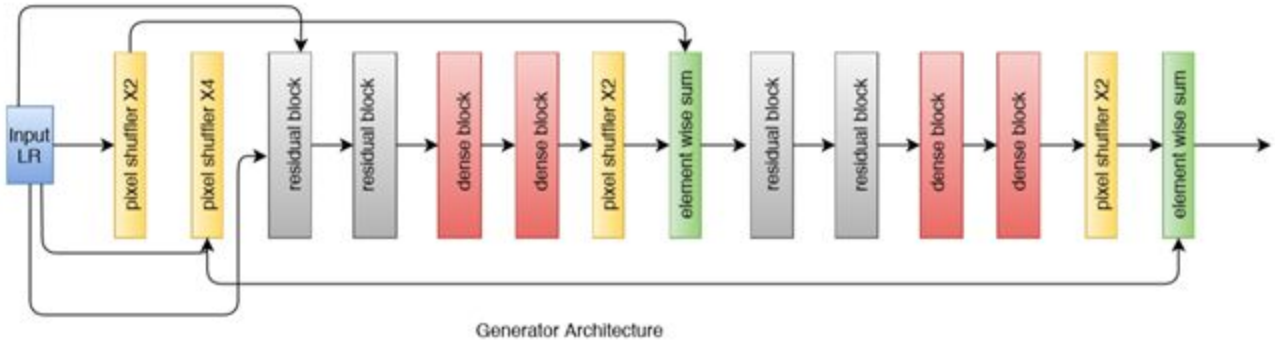

Fig 2: GeneratorArchitecture

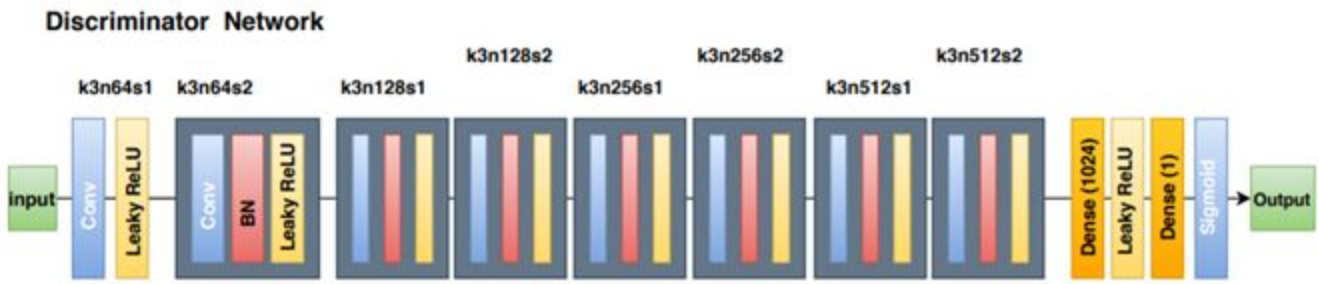

Fig 3: Discriminator Architecture

4.1.Adversarial network architecture

We define a discriminator network DθD(fig 3) inspired by Goodfellow et al. [32] which we optimize in an alternating manner along side GθG (fig 2) to solve the adversarial min-max problem:

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{I^{HR} \sim p_{\text{train}}(I^{HR})}[\log D_{\theta_D}(I^{HR})] + \mathbb{E}_{I^{LR} \sim p_G(I^{LR})}[\log(1 - D_{\theta_D}(G_{\theta_G}(I^{LR}))] \quad (11)$$

The general idea behind this formulation is that it allows one to coach a generative model G with the goal of fooling a differentiable discriminator D that's trained to differentiate super-resolved images from real images. With this approach our generator can learn to make solutions that are highly almost like real images and thus difficult to classify by D. This encourages perceptually superior solutions residing within the subspace, the manifold, of natural images. This is often in contrast to SR solutions obtained by minimizing pixel-wise error measurements, like the MSE. At the core of our very deep generator network G,which is illustrated in Fig4,Fig5 are dense blocks and residual blocks with identical layout. Inspired by Johnson et al. [33]we employ the block layout proposed by Gross and Wilber [34]. We increase the resolution of the input image with two trained sub-pixel convolution layers as proposed by Shi et al. [35]. We train a discriminator network in order to identify real HR images from generated SR images . The architecture is shown in Figure. We follow the architectural guidelines summarized by Radford et al. [36] and use LeakyReLU activation ($\alpha = 0.2$) and avoid max-pooling throughout the network. The discriminator network is trained to unravel the maximization problem in Equation 11. It contains eight convolutional layers with an increasing number of three × 3 filter kernels, increasing by an element of two from 64 to 512 kernels as within the VGG network [37]. Strided convolutions are wont to reduce the image resolution whenever the amount of features is doubled. The resulting 512 feature maps are followed by two dense layers and a final sigmoid activation function to get a probability for sample classification.

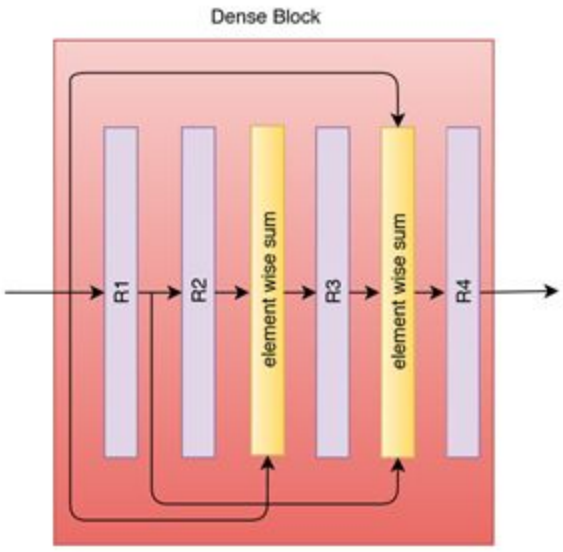

Fig 4: Dense Block

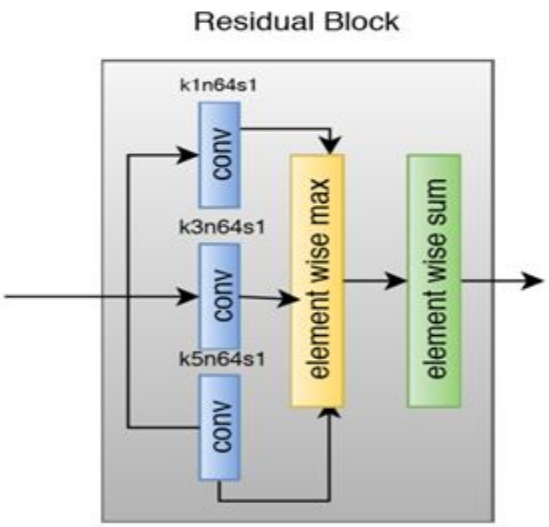

Fig 5: Residual Block

4.2.Perceptual loss function

The loss function is crucial to the performance of our model. Most of the super resolution work uses MSE pixel wise loss [38, 39], we would use Johnson et al. [33] and Bruna et al. [40] work and add our own extra components in the loss function so as to improve .The perceptual loss is defined as the weighted sum of a content loss (lSR X) and an adversarial loss and logit loss component as:

$$l^{SR} = l_X^{SR} + 10^{-3} l_{Gen}^{SR} + 10^{-3} L_G$$

content loss adversarial loss logit loss

perceptual loss (for VGG based content losses) (12)

4.2.1 Content loss

We calculate MSE loss on pixel wise basis in most of the cases as follows :

$$l_{MSE}^{SR} = \frac{1}{r^2WH}\sum_{x=1}^{rW}\sum_{y=1}^{rH}(I_{x,y}^{HR} - G_{\theta_G}(I^{LR})_{x,y})^2 \quad (13)$$

This is a most commonly used objective function used by many state-of-the-art methods[38, 39]. But we would like to bring the limitation of results calculated on pixel wise MES loss.It lacks high-frequency texture(noise) and produces over smooth results with looks unplausible .We are replacing the pixel-wise MSE loss with the work Gatys et al. [41], Bruna et al. [40] and Johnson et al. [33] and adding extra components to our loss function so that the results produced are of good perceptual quality .We will calculate VGG based loss from the output of a pretrained 19 layers VGG network inspired from the work of Simonyan and Zisserman [42]. VGG loss is defined as L2 norm distance between the feature representations of are produced image $G_{\theta G}(I_{LR})$ the target image I_{HR}:

$$l_{VGG/i.j}^{SR} = \frac{1}{W_{i,j}H_{i,j}}\sum_{x=1}^{W_{i,j}}\sum_{y=1}^{H_{i,j}}(\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2 \quad (14)$$

where $\varphi_{i,j:}$ the feature vector produced by the j-th convolution prior to the i-th max pooling layer in the VGG19 network and

$W_{i,j}$ and $H_{i,j}$: dimensions of the corresponding feature vectors of the VGG network.

4.2.2 Adversarial loss

We add the generative component of our GAN to loss function. This encourages our network to produce solutions that look similar to real images more plausible to we humans, hence trying to keep the discriminator in illusion. The following loss l^{SR} Gen is defined as:

$$l_{Gen}^{SR} = \sum_{n=1}^{N} -\log D_{\theta_D}(G_{\theta_G}(I^{LR})) \quad (15)$$

Here, $D_{\theta D}(G_{\theta G}(I^{LR}))$ is the probability/chances of the output(produced) image $G_{\theta G}(I^{LR})$ if it is a real HR

image. For better gradient behavior we minimize $-\log D_{\theta D}(G_{\theta G}(I^{LR}))$ rather than $\log[1-D_{\theta D}(G_{\theta G}(I^{LR}))]$ [43].

4.2.3 Logit loss

In the third component of our loss function,we add the logit component of our model to the our loss function. This encourages our network to support the results that are more plausible of human eyes and trying to keep the discriminator network in illusion.

$$L_G = -E_{xr\sim Pr}[\log(\ D(x_r))] - E_{xf\ \sim Pg}[\log(1 - D(x_f))]$$

Chapter 5

Experimental Results

5.1.Data and similarity measures

We will perform experiments on three widely used benchmark datasets Set5[44], Set14[45] and BSD100, the testing set of BSD300 [46]. We performed all the experiments with a scale of 4× between LR and HR images which is equivalent to a 16× increase in image pixels. The PSNR and SSIM [47] measures are calculated as metrices. Super-resolved images for the reference methods, including nearest neighbor,bicubic,SRCNN[48]and SelfExSR[31,49],were obtained from material of Huang et al.2 [49] .DRCN was available on internet from work of Kim et al.3 [50].

5.2.Training details

We trained our model on a GeForce RTX 2080 Ti/PCIe/SSE2 with the help of 8156 images from the RAISE dataset and 2650 images from DIV2K dataset.Training and testing images are different. We obtained the corresponding LR images by downscaling the corresponding HR version by a factor of r with the help of a bicubic interpolation. We cropped 16 random 96 × 96 HR sub images of distinct training images for each min batch. We can apply our model to images of any dimension as it is FC. The LR input images are scaled to range 0 to 1 and the HR images are scaled to range -1 to 1. The MSE loss calculated on images are of intensity range -1 to 1.We rescaled VGG feature maps by a factor of 1 /12.75 so that the obtained VGG losses are in the range of MSE loss,which is equivalent to multiplying Equation 14 with a factor of ≈ 0.006. Adam optimizer [51]was used having the regular values of beta i.e, $\beta 1 = 0.9$. We trained SRGAN at a learning rate of 10^{-4} for 105 update iterations and at a reduced learning rate of 10^{-5} next 105 iterations. Updates to discriminator network and the generator network are alternated, inspired by Goodfellow et al. [43].

5.3. Generator

5.3.1. Residual Block

```
# The Bx residual blocks
def residual_block(inputs, output_channel, stride, scope):
    with tf.variable_scope(scope):
        net1 = conv2(inputs, 1, output_channel, stride, use_bias=False, scope='conv_1')
        net2 = conv2(inputs, 3, output_channel, stride, use_bias=False, scope='conv_2')
        net3 = conv2(inputs, 5, output_channel, stride, use_bias=False, scope='conv_3')
        net=tf.maximum(net1,net2)
        net=tf.maximum(net,net3)
        net=net+net1
    return net
```

Fig 6 : Residual block snippet

5.3.2. Dense Block

```
    # The Dx dense blocks

    def dense_block(inputs, output_channel, stride, scope):
        with tf.variable_scope(scope):
            name_scope = 'resblock_1'
            net = residual_block(inputs, 64, 1, name_scope)
            residual1_output=net

            name_scope = 'resblock_2'
            net = residual_block(net, 64, 1, name_scope)
            residual2_output=net
            net=net+inputs

            name_scope = 'resblock_3'
            net = residual_block(net, 64, 1, name_scope)
            residual3_output=net
            net=net+residual1_output+inputs

            name_scope = 'resblock_4'
            net = residual_block(net, 64, 1, name_scope)
            residual4_output=net

        return net
```

Fig 7 : Dense block snippet

```
# Definition of the  my_generator
def generator(gen_inputs, gen_output_channels, reuse=False, FLAGS=None):
# Check the flag
    if FLAGS is None:
        raise  ValueError('No FLAGS is provided for generator')


    with tf.variable_scope('generator_unit', reuse=reuse):
        # The input layer

        gen_inputs=conv2(gen_inputs, 3, 64, 1, use_bias=False, scope='conv_0')
        net=gen_inputs

        for i in range(1, 3 , 1):
            name_scope = 'resblock_%d'%(i)
            net = residual_block(net, 64, 1, name_scope)

        for i in range(1, 3 , 1):
            name_scope = 'denseblock_%d'%(i)
            net = dense_block(net, 64, 1, name_scope)
        a,w,h,b = gen_inputs.get_shape()


        upsampled_2_gen_inputs = pixelShuffler(gen_inputs, scale=2)
        upsampled_2_gen_inputs= conv2(upsampled_2_gen_inputs, 3, 64, 1, use_bias=False, scope='conv_sh')


        a,w,h,b = net.get_shape()

        upsampled_2_net = pixelShuffler(net, scale=2)
        upsampled_2_net= conv2(upsampled_2_net, 3, 64, 1, use_bias=False, scope='conv_sh_0')

        net=upsampled_2_gen_inputs+upsampled_2_net

        for i in range(1, 3 , 1):
            name_scope = 'resblock_2_%d'%(i)
            net = residual_block(net, 64, 1, name_scope)

        for i in range(1, 3 , 1):
            name_scope = 'denseblock_2_%d'%(i)
            net = dense_block(net, 64, 1, name_scope)

        a,w,h,b = gen_inputs.get_shape()

        upsampled_4_gen_inputs = pixelShuffler(gen_inputs, scale=4)
        upsampled_4_gen_inputs= conv2(upsampled_4_gen_inputs, 3, 64, 1, use_bias=False, scope='conv_sh_1')

        a,w,h,b = net.get_shape()

        upsampled_2_net = pixelShuffler(net, scale=2)
        upsampled_2_net= conv2(upsampled_2_net, 3, 64, 1, use_bias=False, scope='conv_sh_2')

        net=upsampled_4_gen_inputs+upsampled_2_net
        net=conv2(net, 3, gen_output_channels, 1, use_bias=False, scope='conv_6')

    return net
```

Fig 8 : Dense block snippet

5.4. Discriminator

```
# Define the discriminator block
def discriminator_block(inputs, output_channel, kernel_size, stride, scope):
    with tf.variable_scope(scope):
        net = conv2(inputs, kernel_size, output_channel, stride, use_bias=False, scope='conv1')
        net = batchnorm(net, FLAGS.is_training)
        net = lrelu(net, 0.2)

    return net

with tf.device('/gpu:0'):
    with tf.variable_scope('discriminator_unit'):
        # The input layer
        with tf.variable_scope('input_stage'):
            net = conv2(dis_inputs, 3, 64, 1, scope='conv')
            net = lrelu(net, 0.2)

        # The discriminator block part
        # block 1
        net = discriminator_block(net, 64, 3, 2, 'disblock_1')

        # block 2
        net = discriminator_block(net, 128, 3, 1, 'disblock_2')

        # block 3
        net = discriminator_block(net, 128, 3, 2, 'disblock_3')

        # block 4
        net = discriminator_block(net, 256, 3, 1, 'disblock_4')

        # block 5
        net = discriminator_block(net, 256, 3, 2, 'disblock_5')

        # block 6
        net = discriminator_block(net, 512, 3, 1, 'disblock_6')

        # block_7
        net = discriminator_block(net, 512, 3, 2, 'disblock_7')

        # The dense layer 1
        with tf.variable_scope('dense_layer_1'):
            net = slim.flatten(net)
            net = denselayer(net, 1024)
            net = lrelu(net, 0.2)

        # The dense layer 2
        with tf.variable_scope('dense_layer_2'):
            net = denselayer(net, 1)
            logit=net
            net = tf.nn.sigmoid(net)

return net,logit
```

Fig 9: Discriminator block snippet

5.5. Results Screenshots

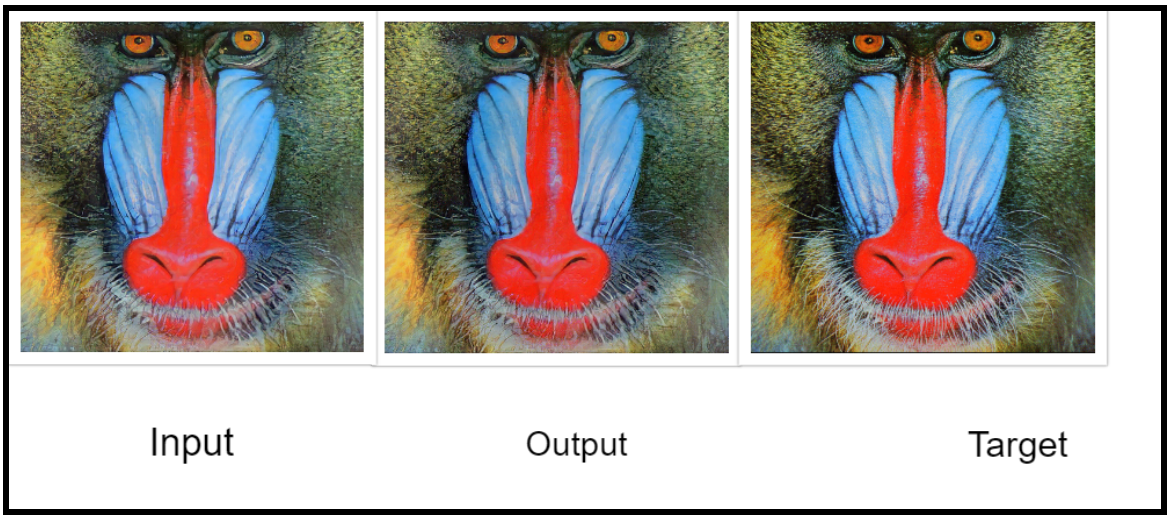

Fig 10: Results Screenshot 1

Fig 11: Results Screenshot 2

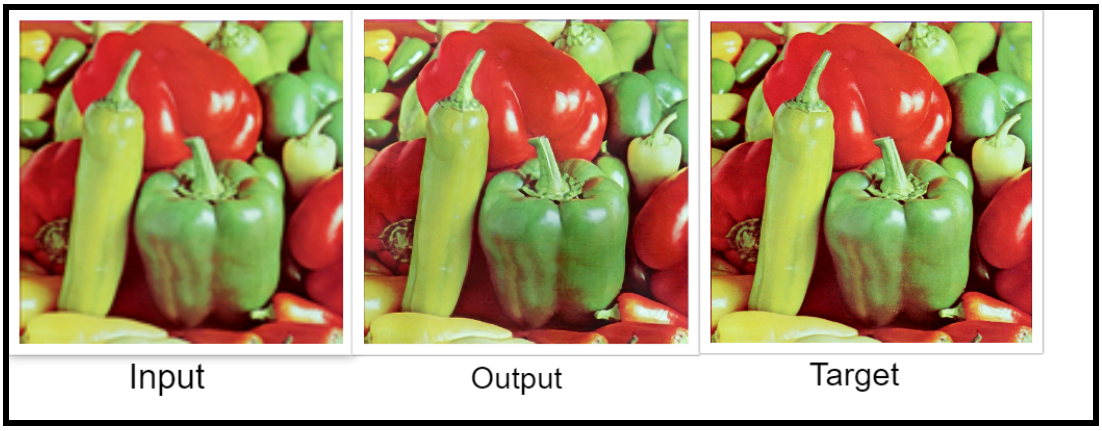

Fig 12: Results Screenshot 3

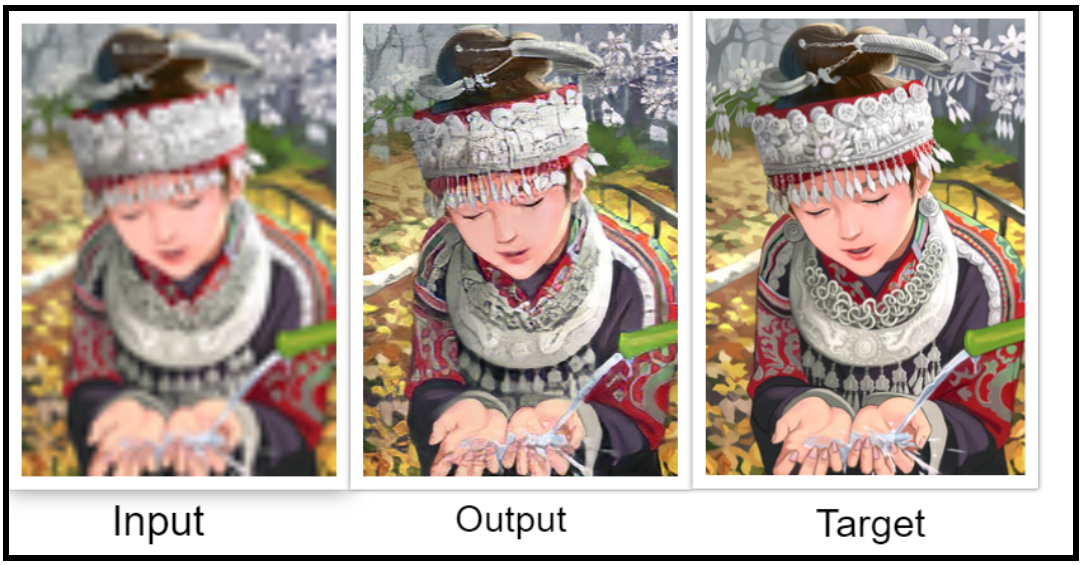

Fig 13: Results Screenshot 4

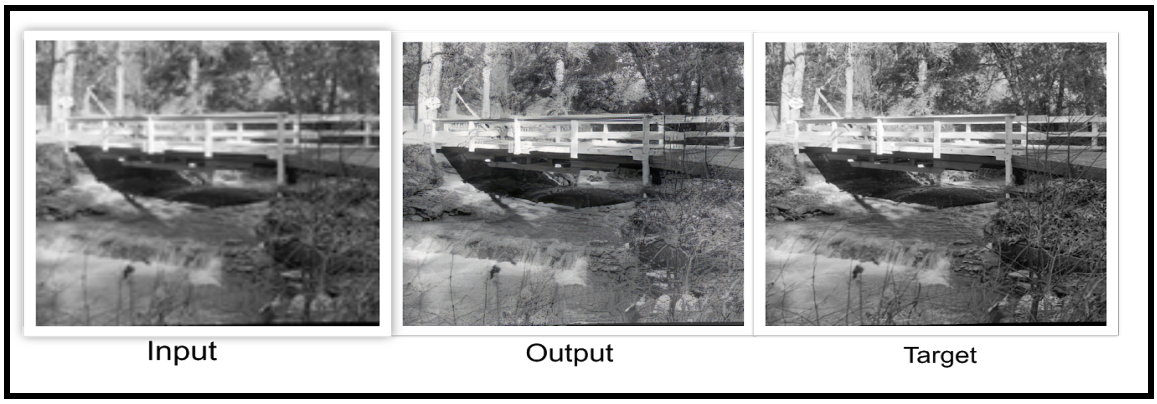

Fig 14: Results Screenshot 5

Fig 15: Results Screenshot 6

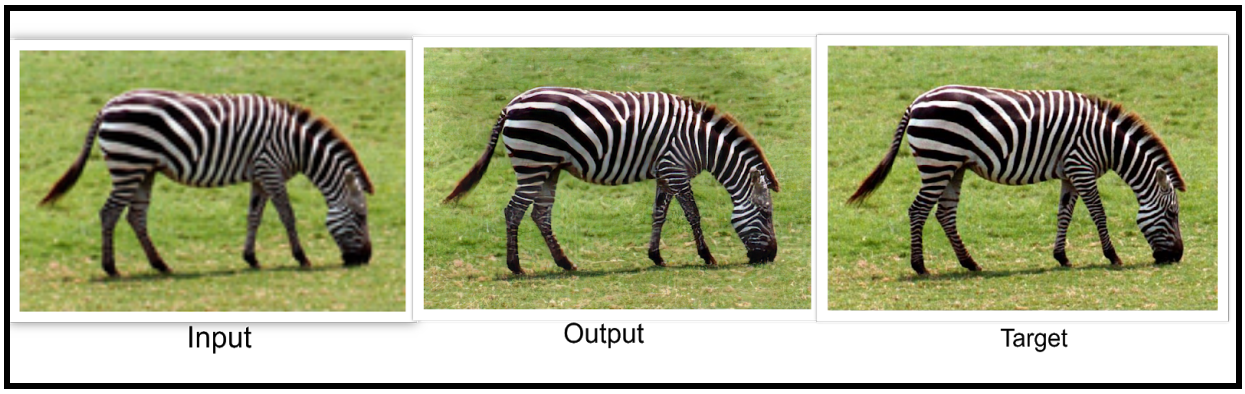

Fig 16: Results Screenshot 7

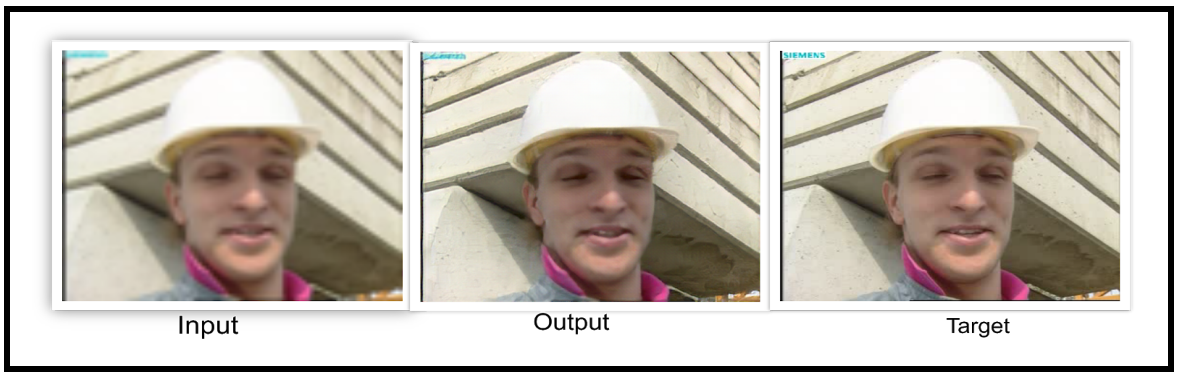

Fig 17: Results Screenshot 8

5.6. Results Table

Set5	MSE loss	VGG based loss
PSNR	22.3485	22.6545
SSIM	0.7044	0.8072

BSD100	MSE loss	VGG based loss
PSNR	22.4458	22.5871
SSIM	0.531	0.7072

Set14	MSE loss	VGG based loss
PSNR	22.1567	22.5045
SSIM	0.6943	0.6997

Manga109	MSE loss	VGG based loss
PSNR	21.2515	21.4565
SSIM	0.6994	0.7072

Table 1: Results on NTIRE 2019 Track 2 Dataset

Method	bicubic	FSRCNN	SRGAN
PSNR	22.85	22.79	23.12
SSIM	0.66	0.61	0.68

Chapter 6

Future Extension

A special setup for benchmarking RealWorld, Super-Resolution,due to the missing ground truth. Super-resolve images above camera resolution.

Zooming Texts: Setup for zooming in texts as most of the state-of-the-art methods fail to super resolve the blurred texts.

Dataset Modification:

1. Adding images with more textual content in the training data set.
2. Creation of large low-/high-resolution pairs dataset.

Chapter 7

Conclusion

We have modeled a GAN based SRGAN that tries to produce better and more photorealistic results on benchmark datasets when analyzed with SSIM and PSNR measure. We have highlighted some drawbacks of this PSNR based image SR and came with deep residual SRGAN.Our method combines the content loss ,adversarial loss and logit loss by training a Generative adversarial network. Using SRGAN results for large upscaling factors like 4 are, by a substantial difference,we tried to obtain more human plausible results than super resolved images obtained with state-of-the-art reference methods.

.

References

[1]https://en.wikipedia.org/wiki/Convolutional_neural_network

[2]https://en.wikipedia.org/wiki/Recurrent_neural_network

[3]https://www.atlantis-press.com/proceedings/icmt-13/10409

[4]https://en.wikipedia.org/wiki/Residual_neural_network

[5]https://en.wikipedia.org/wiki/Generative_adversarial_network

[6]https://en.wikipedia.org/wiki/Mean_squared_error

[7]https://en.wikipedia.org/wiki/Mean_opinion_score

[8]https://towardsdatascience.com/vgg-neural-networks-the-next-step-after-alexnet-3f91fa9ffe2c

[9]http://datasets.visionbib.com/

[10]https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio

[11]https://en.wikipedia.org/wiki/Structural_similarity

[12]https://www2.seas.gwu.edu/~ayoussef/papers/ImageDownUpSampling-CISST99.pdf

[13]https://en.wikipedia.org/wiki/Image_quality

[14]Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image superresolution using deep convolutional networks. TPAMI, 2016.

[15]Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image superresolution using very deep convolutional networks. In CVPR, 2016.

[16]Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero,Andrew Cunningham, Alejandro Acosta, Andrew P. Aitken,Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In CVPR, 2017.

[17]Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In CVPRW, 2017.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR,2016.

[19]Tong Tong, Gen Li, Xiejie Liu, and Qinquan Gao. Image super-resolution using dense skip connections. In ICCV,2017.

[20] Gao Huang, Zhuang Liu, Van Der Maaten Laurens, and Kilian Q Weinberger. Densely connected convolutional networks. In CVPR, 2016.

[21]Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution.In CVPR, 2018.

[22] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Deep back-projection networks for super-resolution.In CVPR, 2018.

[23] Wei Han, Shiyu Chang, Ding Liu, Mo Yu, Michael Witbrock,and Thomas S. Huang. Image super-resolution via dual-state recurrent networks. In CVPR, 2018.

[24]Amir R. Zamir, Te-Lin Wu, Lin Sun, William B. Shen,Bertram E. Shi, Jitendra Malik, and Silvio Savarese. Feedback networks. In CVPR, 2017.

[25]J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In European Conference on Computer Vision, pages 694–711. Springer,2016.

[26]M. S. Sajjadi, B. Schölkopf, and M. Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. ICCV, 2017

[27]Y. Blau and T. Michaeli. The perception-distortion tradeoff. In Proc. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, Utah, USA, pages 6228–6237, 2018.

[28]T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957, 2018

[29]C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2017.

[30]L. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In Advances in Neural Information Processing Systems, pages 262–270, 2015

[31]X. Wang, K. Yu, C. Dong, and C. C. Loy. Recovering realistic texture in image super-resolution by deep spatial feature transform. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2018

[32]I.Goodfellow, J.Pouget-Abadie, M.Mirza, B.Xu, D.Warde-Farley, S.Ozair,A.Courville,andY.Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems (NIPS), pages 2672–2680, 2014. 3, 4, 6

[33] J. Johnson, A. Alahi, and F. Li. Perceptual losses for real-time style transfer and super-resolution. InEuropeanConferenceonComputer Vision (ECCV), pages 694–711. Springer, 2016. 2, 3, 4, 5, 7

[34]S.GrossandM.Wilber. Training and investigating residual nets,online at http://torch.ch/blog/2016/02/04/resnets. html. 2016. 4

[35] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-Time Single Image and Video Super-ResolutionUsinganEfficientSub-PixelConvolutionalNeural Network. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1874–1883, 2016. 3, 4, 5, 6, 7, 8

[36] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In International Conference on Learning Representations (ICLR), 2016. 3, 4

[37] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In International Conference on Learning Representations (ICLR), 2015. 2, 3, 4, 5

[38] 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, 38(2):295–307, 2016. 3, 5, 7

[39] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-Time Single Image and Video Super-ResolutionUsinganEfficientSub-PixelConvolutionalNeural Network. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1874–1883, 2016. 3, 4, 5, 6, 7, 8

[40] J. Bruna, P. Sprechmann, and Y. LeCun. Super-resolution with deep convolutional sufficient statistics. In International Conference on Learning Representations (ICLR), 2016. 2, 3, 5

[41] 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. 3, 5

[42] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In International Conference on Learning Representations (ICLR), 2015. 2, 3, 4, 5

[43] I.Goodfellow, J.Pouget-Abadie, M.Mirza, B.Xu, D.Warde-Farley, S.Ozair,A.Courville,andY.Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems (NIPS), pages 2672–2680, 2014. 3, 4, 6

[44] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel. Low-complexity single-image super-resolution based on non negative neighbor embedding. BMVC, 2012. 6

[45] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In Curves and Surfaces, pages 711–730. Springer, 2012. 2, 6

[46] 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. In IEEE International Conference on Computer Vision (ICCV), volume 2, pages 416–423, 2001. 6

[47] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. IEEE Transactions on Image Processing, 13(4):600–612, 2004. 1, 6

[48] 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. 3, 6, 8

[49] J. B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5197–5206, 2015. 2, 6, 8

[50] 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), 2016. 3, 6, 8

[51] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In International Conference on Learning Representations (ICLR), 2015. 6