CHAPTER 1

INTRODUCTION

1.1 Overview

In the last decade the world has seen an immense global advancement in technology, both in hardware and software. The industries took advantage of the advanced technology to produce electronic gadgets such as computers, mobile phones, PDAs and many more at affordable prices. The camera sensor manufacturing units also advanced in their manufacturing techniques to produce good quality high-resolution (HR) digital cameras. Although, HR digital cameras are available, many computer vision applications such as satellite imaging, target detection, medical imaging and many more still had a strong requisition for higher resolution imagery which very often exceeded the capabilities of these HR digital cameras. To cope up with the strong demand for higher-resolution imagery, these applications approached image-processing techniques for a solution to generate good quality HR imagery.

Super − Resolution image reconstruction is a promising technique of digital imaging which attempts to reconstruct HR imagery by fusing the partial information contained within a number of under-sampled low-resolution (LR) images of that scene during the image reconstruction process. Super-resolution image reconstruction involves up-sampling of under-sampled images thereby filtering out distortions such as noise and blur. In comparison to various image enhancement techniques, super-resolution image reconstruction technique not only improves the quality of under-sampled, low-resolution images by increasing their spatial resolution but also attempts to filter out distortions.

The central aim of Super-Resolution (SR) is to generate a higher resolution image from lower resolution images. High resolution image offers 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 concept of super-resolution. It provides an advantage as it may cost less and the existing low-resolution imaging systems can still be utilized.

1.2 Motivation

Today, people have an attention span of fewer than nine seconds, or less than a goldfish. As a result, images are more important than ever, especially on social media. A large part of the brain is focused on visual processing, allowing humans to process images at an incredible speed. The human brain can recognize a familiar object in only 100 milliseconds, making images the perfect way to communicate in today's short-attention world.

Being around since the beginning of digital photography, we no doubt understand how much progress we've made. Today's models make the first digital cameras look incredibly weak, and the reason is resolution. For a lot less than you would have paid back then, you can get a digital camera with ten times the resolution. If you remember those old computer monitors from the 90s, you'll remember how blurry the images were. These days, most basic laptops and LCD screens have a much higher resolution. You get more detail in the space of your screen because there are more dots to display the details of the images. That's all resolution is. It's the number of dots (i.e. pixels) in any given space.

But, having a keen eye on the recent technology and developments in the field of image processing and engineering, we see that a lot of focus has been given on the hardware part of the process in image enhancements. We as software engineers have taken the task of providing the industry our contribution in taking better images and photographs. We can do this by using super resolution.

Super resolution reconstruction techniques are still under development. All the existing methods of super resolution reconstruction do not provide satisfactory results; therefore, it is

required to investigate the better model which provides the better accuracy. High resolution images require higher storage, it is beneficial to store the images in low resolution and construct the HR images whenever required from low resolution images. In addition, high resolution images contain more details and information about the image and have less noise.

1.3 Challenges

- **Image Registration**: Image registration is critical for the success of multi-frame SR reconstruction, where complementary spatial samplings of the HR image are fused. The image registration is a fundamental image processing problem that is well known as ill-posed.

- **Computational Efficiency**: Another difficulty limiting practical application of SR reconstruction is its intensive computation due to large number of unknowns, which require expensive matrix manipulations. Real applications always demand efficiency of the SR reconstruction to be of practical utility, e.g., in the surveillance video scenarios, it is desired for the SR reconstruction to be real time. Efficiency is also desirable for SR systems with users in the loop for tuning parameters.

- **Robustness Aspects**: Traditional SR techniques are vulnerable to the presence of outliers due to motion errors, inaccurate blur models, noise, moving objects, motion blur etc. These inaccurate model errors cannot be treated as Gaussian noise as the usual assumption. Robustness of SR is of interest because the image degradation model parameters cannot be estimated perfectly, and sensitivity to outliers may result in visually disturbing artefacts, which are intolerable in many applications, e.g., video standard conversion.

- **Performance Limits**: The SR reconstruction has become a hot research topic since it was introduced, and thousands of SR papers have bloomed into publications. However, not much work has been devoted to fundamental understanding of the performance limits of these SR reconstruction algorithms. Such a performance limit understanding is important. For example, it will shed light on SR camera design, helping to analyse factors such as model errors, zooming factors and number of frames etc.

1.4 Applications

The applications of Super Resolution can be classified into 5 Domain Specific Applications:

1. **Depth Map Super Resolution**

   Depth maps record the distance between the viewpoint and the objects in the scene, and the depth information plays important roles in many tasks such as pose estimation semantic segmentation, etc. However, due to productive and economic limitations, the depth maps produced by depth sensors are often low-resolution and suffer degeneration effects such as noise, quantization, missing values, etc. Thus, super-resolution is introduced for increasing the spatial resolution of depth maps. Today one of the most popular practices for depth map SR is to use another economical RGB camera to obtain HR images of the same scenes for guiding super-resolving the LR depth maps. Specifically, Song et al exploit the depth field statistics and the local correlation between depth maps and RGB images to constrain the global statistics and local structure. Hui et al. utilize two CNNs to simultaneously upsample LR depth maps and down-sample HR RGB images, then use RGB features as the guidance of the up-sampling process at the same resolution. Similarly, Ni et al. and Zhou et al. use HR RGB images as guidance by extracting HR edge map and predicting missing high-frequency components, respectively. While Xiao et al. use the pyramid network to enlarge the receptive field, extract features from LR depth maps and HR RGB images, respectively, and fuse these features to predict HR depth maps. And Haefner et al. fully exploit the colour information in order to guide super-resolution by resorting to the shape-from-shading technique. In contrast to the above works, Riegler et al. combine CNNs with an energy minimization model in the form of a powerful variational model to recover HR depth maps without other reference images.

2. **Face Image Super-resolution**

   Face image super-resolution, a.k.a. face hallucination (FH), can often help other face-related tasks. Compared to generic images, face images have much more face-related structured information, so incorporating facial prior knowledge (e.g., landmarks, parsing maps, identities) into FH is a very popular and promising approach.

The most straightforward way to exploit facial prior is to constrain the generated HR images to have the identical face-related information to ground truth HR images. Specifically, the CBN utilizes the facial prior by alternately optimizing FH and dense correspondence field estimation. The Super-FAN and MTUN both introduce FAN to guarantee the consistency of facial landmarks by end-to-end multi-task learning. And the FSRNet uses not only facial landmark heatmaps but also face parsing maps as the facial prior constraints. The SICNN which aims at recovering the real identity information, adopts a super-identity loss function and a domain-integrated training approach to stable the joint training. Besides explicitly using facial prior, the implicit methods are also widely studied. The TDN incorporates spatial transformer networks for automatic spatial transformations and thus solves the face unalignment problem. Based on TDN, the TDAE adopts a decoder-encoderdecoder framework, where the first decoder learns to up-sample and denoise, the encoder projects it back to aligned and noise-free LR faces, and the last decoder generates hallucinated HR images. In contrast, the LCGE employs component-specific CNNs to perform SR on five facial components, uses k-NN search on an HR facial component dataset to find corresponding patches, synthesizes finer grained components and finally fuses them to FH results. Yang et al. also decompose deblocked face images into facial components, whose landmarks are used to retrieve adequate HR component exemplars in an external dataset, the background is fed into a generic SR network, and finally fuse them to complete HR face images. In addition to the above works, researchers also improve FH from other perspectives. Motivated by the human attention shifting mechanism, the Attention-FH resorts to a recurrent policy network for sequentially discovering attended face patches and performing local enhancement, and thus fully exploits the global interdependency of face images. The UR-DGN adopts a network similar to SRGAN with adversarial learning. And Xu et al. propose a multi-class GAN-based FH model composed of a generic generator and class-specific discriminators. Both Lee et al. and Yu et al. utilize additional facial attribute information to perform FH with the specified attributes, based on the conditional GAN.

3. Hyperspectral Image Super-resolution

Compared to panchromatic images (PANs, i.e., RGB images with 3 bands), hyperspectral images (HSIs) containing hundreds of bands provide abundant spectral features and help a variety of vision tasks. Nevertheless, due to hardware limitations, not only collecting high-quality HSIs is much more difficult than collecting PANs, but also the resolution of collected HSIs is much lower. Thus, super-resolution is introduced into this field, and researchers tend to combine HR PANs and LR HSIs to predict HR HSIs. Among them, Huang et al. present a sparse denoising autoencoder to learn LR-to-HR mappings with PANs and transfer it to HSIs. Masi et al. employ the SRCNN and incorporate several maps of nonlinear radiometric indices for boosting performance. Wei et al. propose a much deeper DRPNN based on residual learning and achieve higher spatial-spectral unified accuracy. Recently, Qu et al. jointly train two encoder-decoder networks to perform SR on PANs and HSIs, respectively, and transfer the SR knowledge in the PAN domain to the HSI domain by sharing the decoder and applying constraints such as angle similarity loss and reconstruction loss.

4. Video Super-resolution

In terms of video super-resolution, multiple frames provide much more scene information, and there are not only intraframe spatial dependency but also inter-frame temporal dependency (e.g., motions, brightness and color changes). Thus, the existing works mainly focus on making better use of the spatio-temporal dependency, including explicit motion compensation (e.g., optical flow algorithms, learning based methods) and recurrent methods, etc. Among the methods based on optical flow algorithms, Liao et al. employ various optical flow methods to generate HR candidates and ensemble them by CNNs. VSRnet and CVSRnet implement motion compensation by Druleas algorithm, and uses CNNs to take successive frames as input and predict HR frames. While Liu et al., perform rectified optical flow alignment, and propose a temporal adaptive net to generate HR frames in various temporal spaces and aggregate them adaptively. Besides, others also try to

directly learn the motion compensation. The VESPCN utilizes a trainable spatial transformer to learn motion compensation based on adjacent frames, and enters multiple frames into a spatiotemporal ESPCN for end-to-end prediction. And Tao et al. root from accurate LR imaging model and propose a sub-pixel-like module to simultaneously achieve motion compensation and super-resolution, and thus fuse the aligned frames more effectively. Another trend is to use recurrent methods to capture the spatial-temporal dependency without explicit motion compensation. Specifically, the BRCN, employs a bidirectional framework, and uses CNN, RNN, and conditional CNN to model the spatial, temporal and spatial temporal dependency, respectively. Similarly, STCN uses a deep CNN and a bidirectional LSTM to extract spatial and temporal information. And FRVSR uses previously inferred HR estimates to reconstruct the subsequent HR frame by two deep CNNs in a recurrent manner. In addition to the above works, the FAST exploits the compact description of the structure and pixel correlations extracted by compression algorithms, transfers the SR result from one frame to adjacent frames, and accelerates the state-of-the-art SR algorithms by 15 times with little performance loss (0.2dB). And Jo et al. generate dynamic up-sampling filters and the HR residual image based on the local spatio-temporal neighbourhood of each pixel, and also avoid explicit motion compensation.

5. Other Applications

Deep learning based super-resolution is also adopted to other domain-specific applications and shows great performance. Specifically, the RACNN employs SR models for enhancing the discriminability of LR image details for fine-grained classification. Similarly, the Perceptual GAN addresses the small object detection problem by super resolving representations of small objects, achieving similar characteristics as large objects and more discriminative for detection. And the FSR-GAN super-resolves small-size images in the feature space instead of the pixel space, and thus transforms the raw poor features to highly discriminative ones, which greatly benefits image retrieval. Besides, Dai et al. verify the effectiveness and usefulness of SR technology in several vision applications, including edge detection, semantic segmentation, digit and scene recognition. Huang et al. develop RS-DRL

specifically for 16 super-resolving remote sensing images. And Jeon et al. utilize a parallax prior in stereo images to reconstruct HR images with sub-pixel accuracy in registration.

The other commonly seen applications of Super Resolution can be seen in:

- Bio-medical: ECG, EEG, EMG analysis; cytological, histological and stereological applications; automated radiology and pathology, X-ray image analysis.
- Scientific applications: High energy physics, bubble chamber and other forms of track analysis, etc.
- Criminology: Finger print identification, human face registration and matching, forensic investigation, etc.
- Astronomy and space applications: Restoration of images suffering from geometric and photometric distortions, computing close-up picture of planetary surfaces, etc.
- Meteorology: Short-term weather forecasting, long term climatic change detection from satellite and other remote sensing data, cloud pattern analysis etc.
- Information technology: Facsimile image transmission, videotext, video-conferencing and videophone etc.
- Entertainment and consumer electronics: HDTV, multimedia and video-editing etc.
- Military applications: Missile guidance and detection, target identification, navigation of pilotless vehicle and range finding, etc.
- Security measure: Recognition of human objects and verification of claimed identify based on images of face, iris, palm, ear and fingerprint.

In addition to the above-mentioned areas, another important application of image processing technique is improvement of quality or appearance of a given image.

1.5 Problem Statement

Super-resolution is based on the idea that a combination of low resolution (noisy) sequence of images of a scene can be used to generate a high-resolution image or image sequence. Thus, it attempts to reconstruct the original scene image with high resolution given a set of observed images at lower resolution.

The general approach considers the low-resolution images as resulting from resampling of a high-resolution image. The goal is then to recover the high-resolution image which when resampled based on the input images and the imaging model, will produce the low resolution observed images. Thus, the accuracy of imaging model is vital for super-resolution and an incorrect modelling, say of motion, can actually degrade the image further.

The observed images could be taken from one or multiple cameras or could be frames of a video sequence. These images need to be mapped to a common reference frame. This process is registration. The super-resolution procedure can then be applied to a region of interest in the aligned composite image. The key to successful super-resolution consists of accurate alignment i.e. registration and formulation of an appropriate forward image model.

CHAPTER 2

LITERATURE SURVEY

Many studies have investigated methods of creating high-resolution images (HRIs) from low-resolution images (LRIs). Such image restoration methods were initially applied to still pictures, but were later extended in scope. These extended methods are called super resolution (SR) methods [1], one of which is super resolution image reconstruction (SRR) [2][3][4][8][9]. Among the many SR methods proposed, only SRR has so far been incorporated into commercial products [10][11][12][13]. However, SRR has outstanding practical limitations, especially when used in real-time applications.

Display devices, including LCDs and ink jet printers, have advanced to the extent that their resolution exceeds that of images taken with a film camera. Although the majority of photographs are today taken using digital cameras, focusing errors or camera shake may introduce blurring of the image. To operate effectively, SRR requires multiple LRIs. This makes it an unsatisfactory approach to improving the resolution of still photographs, as only a single image is typically available. In contrast, a video image comprises multiple adjacent frames carrying similar. Information. This makes video an ideal application for SRR, and it is unsurprising that a large number of papers have been published on the subject [5] [4][6]. However, the methods proposed so far are complex. SRR functions with self-congruency within a frame have begun to be incorporated into HDTV sets and BluRay players [11][12]. However, before we can evaluate the efficacy application of SRR to such devices, it must be borne in mind that the characteristics of video frames are different from those of still images.

TV broadcasting remains a common source of video content. Analog broadcasting, after a reign of more than 60 years, is now being supplanted by digital HDTV broadcasting in many countries. However, digital HDTV broadcasting imposes a high initial cost on the broadcaster. If SRR were able to increase the resolution of existing analog videos to meet HDTV standards, this would greatly reduce the initial cost of switching. It would allow owners of HDTVs to view their older videos at HDTV levels of resolution. Finally, it would open up new markets for HDTV receivers. In fact, HDTV sets with SRR functions have already entered the market, and SRR has been deemed a practical technology [14]. However, an HDTV equipped with

SRR functions for converting analog broadcasts has yet to be developed, and a recent study has demonstrated that current HDTV systems incorporating SRR have inferior resolution to those without SRR [29]. SRR faces a further challenge. To improve the resolution of a still image, many low-resolution images with different phases are required. Although some technologies are able to operate using a single image [7][8], iteration must be applied and the processing time will vary according to the characteristics of the image. A practical SRR technology should not be image-dependent in this way.

Despite the breakthroughs in accuracy and speed of single image super-resolution using faster and deeper convolutional neural networks, one central problem remains largely unsolved: how do we recover the finer texture details when we super-resolve at large upscaling factors? The behaviour of optimization-based super-resolution methods is principally driven by the choice of the objective function. Recent work has largely focused on minimizing the mean squared reconstruction error. The resulting estimates have high peak signal-to-noise ratios, but they are often lacking high-frequency details and are perceptually unsatisfying in the sense that they fail to match the fidelity expected at the higher resolution.

In paper [16], they present SRGAN, a generative adversarial network (GAN) for image super resolution (SR). To our knowledge, it is the first framework capable of inferring photo-realistic natural images for 4× upscaling factors. To achieve this, a perceptual loss function which consists of an adversarial loss and a content loss is presented. The adversarial loss pushes the solution to the natural image manifold using a discriminator network that is trained to differentiate between the super-resolved images and original photo-realistic images. In addition, use of a content loss motivated by perceptual similarity instead of similarity in pixel space is done. The deep residual network is able to recover photo-realistic textures from heavily down sampled images on public benchmarks. An extensive mean-opinion-score (MOS) test shows hugely significant gains in perceptual quality using SRGAN. The MOS scores obtained with SRGAN are closer to those of the original high-resolution images than to those obtained with any state-of-the-art method.

The highly challenging task of estimating a high-resolution (HR) image from its low-resolution (LR) counterpart is referred to as super-resolution (SR). SR received substantial attention from within the computer vision research community and has a wide range of applications [17, 18, 19].

The ill-posed nature of the underdetermined SR problem is particularly pronounced for high upscaling factors, for which texture detail in the reconstructed SR images is typically absent. The optimization target of supervised SR algorithms is commonly the minimization of the mean squared error (MSE) between the recovered HR image and the ground truth. This is convenient as minimizing MSE also maximizes the peak signal-to-noise ratio (PSNR), which is a common measure used to evaluate and compare SR algorithms [20]. However, the ability of MSE (and PSNR) to capture perceptually relevant differences, such as high texture detail, is very limited as they are defined based on pixel-wise image differences [21, 22, 23]. This is illustrated in Figure 2.1, where highest PSNR does not necessarily reflect the perceptually better SR result. The perceptual difference between the super-resolved and original image means that the recovered image is not photorealistic as defined by Ferwerda [24].

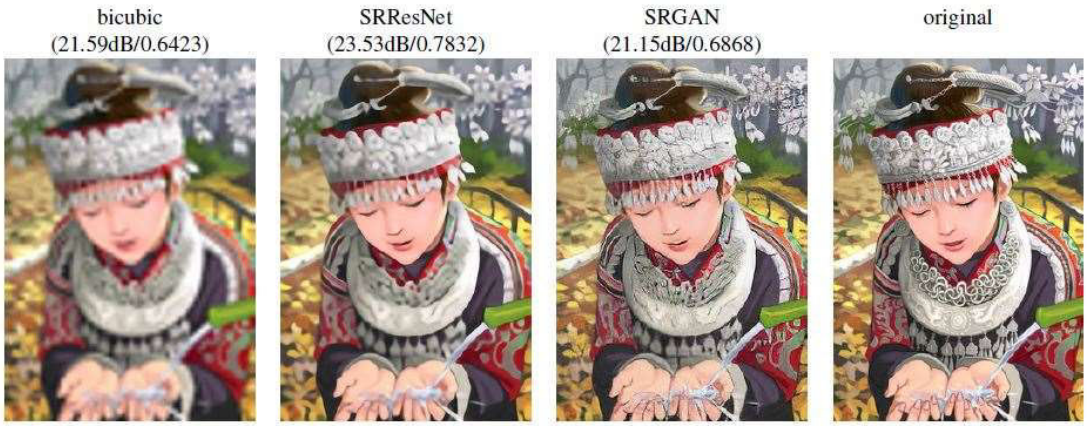

Figure 2.1: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4x upscaling]

In work [16], a generative adversarial network (SRGAN) for which we employ a deep residual network (ResNet) with skip-connection and diverge from MSE as the sole optimization target is proposed. Different from previous works, [16] define a novel perceptual loss using high-level feature maps of the VGG network [25, 26, 27] combined with a discriminator that encourages solutions perceptually hard to distinguish from the HR reference images.

2.1. Related work

2.1.1 Image super-resolution

Recent overview articles on image SR include Nasrollahi and Moeslund [19] or Yang et al. [20]. Here we will focus on single image super-resolution (SISR) and will not further discuss approaches that recover HR images from multiple images [28, 29]. Prediction-based methods were among the first methods to tackle SISR. While these filtering approaches, e.g. linear, bicubic or Lanczos [30] filtering, can be very fast, they oversimplify the SISR problem and usually yield solutions with overly smooth textures. Methods that put particularly focus on edge-preservation have been proposed [31, 32].

More powerful approaches aim to establish a complex mapping between low- and high-resolution image information and usually rely on training data. Many methods that are based on example-pairs rely on LR training patches for which the corresponding HR counterparts are known. Early work was presented by Freeman et al. [33, 34]. Related approaches to the SR problem originate in compressed sensing [35, 36, 37]. In Glasner et al. [38] the authors exploit patch redundancies across scales within the image to drive the SR. This paradigm of self-similarity is also employed in Huang et al. [39], where self-dictionaries are extended by further allowing for small transformations and shape variations. Gu et al. [40] proposed a convolutional sparse coding approach that improves consistency by processing the whole image rather than overlapping patches.

To reconstruct realistic texture detail while avoiding edge artifacts, Tai et al. [41] combine an edge-directed SR algorithm based on a gradient profile prior [50] with the benefits of learning-based detail synthesis. Zhang et al. [43] propose a multi-scale dictionary to capture redundancies of similar image patches at different scales. To super-resolve landmark images, Yue et al. [44] retrieve correlating HR images with similar content from the web and propose a structure-aware matching criterion for alignment. Neighbourhood embedding approaches up sample a LR image patch by finding similar LR training patches in a low dimensional manifold and combining their corresponding HR patches for reconstruction [45, 46]. In Kim and Kwon [47] the authors emphasize the tendency of neighbourhood approaches to overfit and formulate a more general map of example pairs using kernel ridge regression. The regression problem can also be solved with Gaussian process regression, trees or Random Forests. In Dai et al. [48]

a multitude of patch-specific regressors is learned and the most appropriate regressors selected during testing.

Recently convolutional neural network (CNN) based SR algorithms, have shown excellent performance. In Wang et al. [49] the authors encode a sparse representation prior into their feed-forward network architecture based on the learned iterative shrinkage and thresholding algorithm (LISTA) [50]. Dong et al. [13, 14] used bicubic interpolation to upscale an input image and trained a three layer deep fully convolutional network end-to-end to achieve state-of- the-art SR performance. Subsequently, it was shown that enabling the network to learn the upscaling filters directly can further increase performance both in terms of accuracy and speed. With their deeply-recursive convolutional network (DRCN), Kim et al. [34] presented a highly performant architecture that allows for long-range pixel dependencies while keeping the number of model parameters small. Of particular relevance for our paper are the works by Johnson et al. [26] and Bruna et al. [27], who rely on a loss function closer to perceptual similarity to recover visually more convincing HR images.

2.1.2 Design of convolutional neural networks

The state of the art for many computer vision problems is meanwhile set by specifically designed CNN architectures following the success of the work by Krizhevsky et al. [37]. It was shown that deeper network architectures can be difficult to train but have the potential to substantially increase the network's accuracy as they allow modelling mappings of very high complexity [49, 51]. To efficiently train these deeper network architectures, batch normalization [32] is often used to counteract the internal co-variate shift. Deeper network architectures have also been shown 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 modelling 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 shown that learning upscaling filters is beneficial in terms of accuracy and speed [11, 48, 27]. This is an improvement over Dong et al. [10] where bicubic interpolation is employed to upscale the LR observation before feeding the image to the CNN.

2.1.3 Loss functions

Pixel-wise loss functions such as MSE struggle to handle the uncertainty inherent in recovering lost high-frequency details such as texture: minimizing MSE encourages finding pixel-wise averages of plausible solutions which are typically overly-smooth and thus have poor perceptual quality [42, 33, 13, 5]. Reconstructions of varying perceptual quality are exemplified with corresponding PSNR in Figure 2.1. We illustrate the problem of minimizing MSE in Figure 2.2 where multiple potential solutions with high texture details are averaged to create a smooth reconstruction.

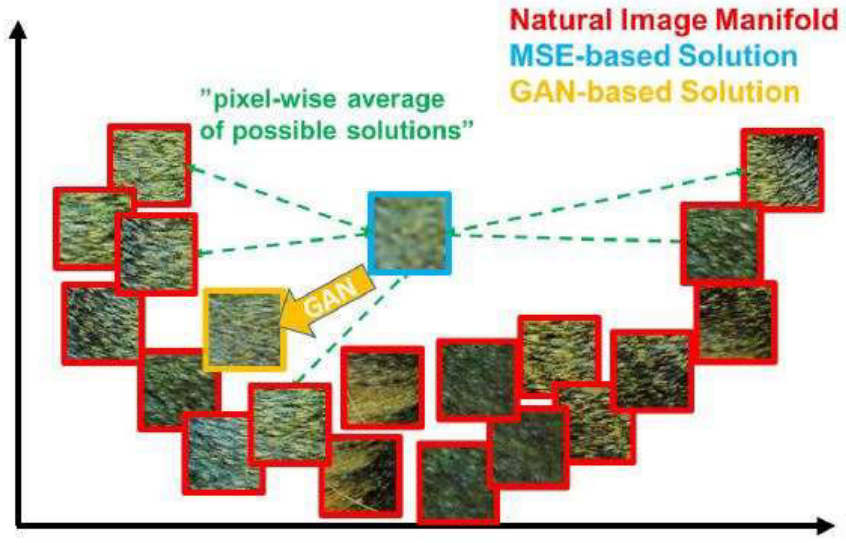

Figure 2.2: Illustration of patches from the natural image manifold (red) and super-resolved patches obtained with MSE (blue) and GAN (orange). The MSE-based solution appears overly smooth due to the pixel-wise average of possible solutions in the pixel space, while GAN drives the reconstruction towards the natural image manifold producing perceptually more convincing solutions.

In Mathieu et al. [42] and Denton et al. [7] the authors tackled this problem by employing generative adversarial networks (GANs) [22] for the application of image generation. Panda and Prasad [6] augment pixel-wise MSE loss with a discriminator loss to train a network that super-resolves face images with large upscaling factors (8x). GANs were also used for unsupervised representation learning in Radford et al. [44]. The idea of using GANs to learn a mapping from one manifold to another is described by Li and Wand [38] for style transfer and Yeh et al. for inpainting. Bruna et al. [5] minimize the squared error in the feature spaces of VGG19 [49] and scattering networks.

Dosovitskiy and Brox [13] use loss functions based on Euclidean distances computed in the feature space of neural networks in combination with adversarial training. It is shown that the proposed loss allows visually superior image generation and can be used to solve the ill-posed inverse problem of decoding nonlinear feature representations. Similar to this work, Johnson et al. [33] and Bruna et al. [5] propose the use of features extracted from a pretrained VGG network instead of low-level pixel-wise error measures. Specifically, the authors formulate a loss function based on the euclidean distance between feature maps extracted from the VGG19 [49] network. Perceptually more convincing results were obtained for both super-resolution and artistic style-transfer [19, 20]. Recently, Li and Wand [38] also investigated the effect of comparing and blending patches in pixel or VGG feature space.

2.2. Contribution

GANs provide a powerful framework for generating plausible-looking natural images with high perceptual quality. The GAN procedure encourages the reconstructions to move towards regions of the search space with high probability of containing photo-realistic images and thus closer to the natural image manifold as shown in Figure 2.2. [16] provides:

- A new state of the art for image SR with high upscaling factors (4x) as measured by PSNR and structural similarity (SSIM) with our 16 blocks deep ResNet (SRResNet) optimized for MSE.
- Propose SRGAN which is a GAN-based network optimized for a new perceptual loss. Here we replace the MSE-based content loss with a loss calculated on feature maps of the VGG network [49], which are more invariant to changes in pixel space [38].
- Confirm with an extensive mean opinion score (MOS) test on images from three public benchmark datasets that SRGAN is the new state of the art, by a large margin, for the estimation of photo-realistic SR images with high upscaling factors (4x).

CHAPTER 3

DESIGN AND ARCHITECTURE

3.1 Flowchart/Development life cycle for a neural network

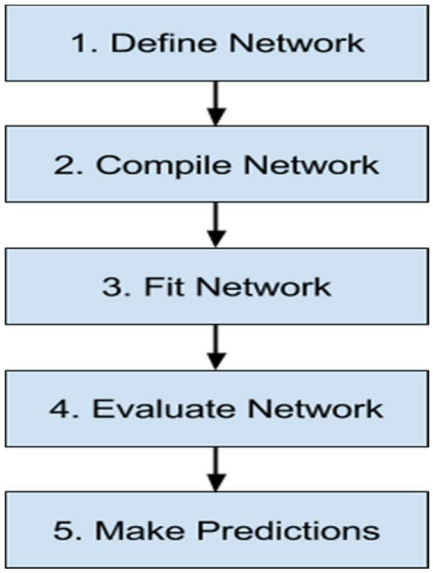

Fig 3.1: Neural Network Life Cycle

Neural networks are defined in Keras as a sequence of layers. The container for these layers is the Sequential class.

The first step is to create an instance of the Sequential class. Then you can create your layers and add them in the order that they should be connected.But we can also do this in one step by creating an array of layers and passing it to the constructor of the Sequential.

The first layer in the network must define the number of inputs to expect. The way that this is specified can differ depending on the network type, but for a Multilayer Perceptron model this is specified by the input_dim attribute.

Think of a Sequential model as a pipeline with your raw data fed in at the bottom and predictions that come out at the top. This is a helpful conception in Keras as concerns that were traditionally associated with a layer can also be split out and added as separate layers, clearly

showing their role in the transform of data from input to prediction. For example, activation functions that transform a summed signal from each neuron in a layer can be extracted and added to the Sequential as a layer-like object called Activation. The choice of activation function is most important for the output layer as it will define the format that predictions will take.

3.2 COMPILE NETWORK

Compilation is an efficiency step. It transforms the simple sequence of layers that we defined into a highly efficient series of matrix transforms in a format intended to be executed on your GPU or CPU, depending on how Keras is configured. Once we have defined our network, we must compile it.

Compilation is always required after defining a model. This includes both before training it using an optimization scheme as well as loading a set of pre-trained weights from a save file. The reason is that the compilation step prepares an efficient representation of the network that is also required to make predictions on your hardware.

Compilation requires a number of parameters to be specified, specifically tailored to training your network. Specifically, the optimization algorithm to use to train the network and the loss function used to evaluate the network that is minimized by the optimization algorithm.

For example, below is a case of compiling a defined model and specifying the stochastic gradient descent (sgd) optimization algorithm and the mean squared error (mse) loss function, intended for a regression type problem

The type of predictive modelling problem imposes constraints on the type of loss function that can be used.

3.3 FIT THE NETWORK

Once the network is compiled, it can be fit, which means adapt the weights on a training dataset. Fitting the network requires the training data to be specified, both a matrix of input patterns X and an array of matching output patterns y.

The network is trained using the backpropagation algorithm and optimized according to the optimization algorithm and loss function specified when compiling the model. The backpropagation algorithm requires that the network be trained for a specified number of epochs or exposures to the training dataset.

Each epoch can be partitioned into groups of input-output pattern pairs called batches. This define the number of patterns that the network is exposed to before the weights are updated within an epoch. It is also an efficiency optimization, ensuring that not too many input patterns are loaded into memory at a time.

Once fit, a history object is returned that provides a summary of the performance of the model during training. This includes both the loss and any additional metrics specified when compiling the model, recorded each epoch.

3.4 EVALUATE AND MAKE PREDICTIONS

After the neural network has been trained, the model is tested and evaluated against the actual expected output. If the model's output is found to be satisfactory then the model is finalised. New predictions can be made from the trained model and be tested against new inputs.

CHAPTER 4

METHODOLOGY

4.1 Generative Adversarial Network (GAN)

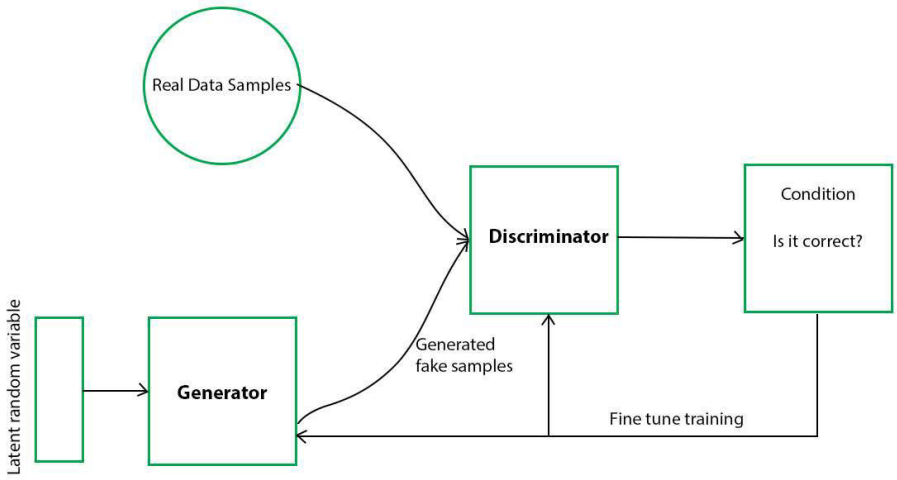

Fig 4.1: GANs block architecture

Generative Adversarial Networks (GANs) are a powerful class of neural networks that are used for unsupervised learning. It was developed and introduced by Ian J. Goodfellow in 2014. GANs are basically made up of a system of two competing neural network models which compete with each other and are able to analyse, capture and copy the variations within a dataset.

Generative Adversarial Networks (GANs) can be broken down into three parts:

- **Generative:** To learn a generative model, which describes how data is generated in terms of a probabilistic model.
- **Adversarial:** The training of a model is done in an adversarial setting.
- **Networks:** Use deep neural networks as the artificial intelligence (AI) algorithms for training purpose.

In GANs, there is a **generator** and a **discriminator**. The Generator generates fake samples of data (be it an image, audio, etc.) and tries to fool the Discriminator. The Discriminator, on the other hand, tries to distinguish between the real and fake samples. The Generator and the Discriminator are both Neural Networks and they both run in competition with each other in the training phase. The steps are repeated several times and in this, the Generator and Discriminator get better and better in their respective jobs after each repetition. The working can be visualized by the diagram given below:

Here, the generative model captures the distribution of data and is trained in such a manner that it tries to maximize the probability of the Discriminator in making a mistake. The Discriminator, on the other hand, is based on a model that estimates the probability that the sample that it got is received from the training data and not from the Generator. The GANs are formulated as a minimax game, where the Discriminator is trying to minimize its reward **V(D, G)** and the Generator is trying to minimize the Discriminator's reward or in other words, maximize its loss. It can be mathematically described by the formula below:

$$\min_G \max_D V(D,G) \tag{4.1}$$

$$V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z))]$$

where,

G = Generator

D = Discriminator

Pdata(x) = distribution of real data

P(z) = distribution of generator

x = sample from Pdata(x)

z = sample from P(z)

D(x) = Discriminator network

G(z) = Generator network

So, basically, training a GAN has two parts:

- **Part 1:** The Discriminator is trained while the Generator is idle. In this phase, the network is only forward propagated and no back-propagation is done. The Discriminator is trained on real data for n epochs, and see if it can correctly predict them as real. Also, in this

phase, the Discriminator is also trained on the fake generated data from the Generator and see if it can correctly predict them as fake.

- **Part 2:** The Generator is trained while the Discriminator is idle. After the Discriminator is trained by the generated fake data of the Generator, we can get its predictions and use the results for training the Generator and get better from the previous state to try and fool the Discriminator.

The above method is repeated for a few epochs and then manually check the fake data if it seems genuine. If it seems acceptable, then the training is stopped, otherwise, its allowed to continue for few more epochs.

Algorithm/pseudocode for Generative Adversarial Network (GAN)

for number of training iterations **do**
  for k steps **do**
    • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]. \tag{4.2}$$

  end for
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right). \tag{4.3}$$

end for

Fig 4.2: Algorithm/pseudocode for Generative Adversarial Network (GAN)

4.2 Using GANs for Super Resolution (SR)

Generative adversarial networks (GANs) have found many applications in Deep Learning. One interesting problem that can be better solved using GANs is super-resolution. Super-resolution is a task concerned with upscaling images from low-resolution sizes such as 90 x 90, into high-resolution sizes such as 360 x 360. In this example, 90 x 90 to 360 x 360 is denoted as an upscaling factor of 4x. One solution to super-resolution is to train a Deep Convolutional Network that takes in data where the input is a Low-Resolution Patch and the labeled output is a High-Resolution Patch. This is different from many supervised learning problems where the output is either 1 or 0 or a vector of class predictions. In this case, the output is an image. These networks learn a mapping from the low-resolution patch through a series of convolutional, fully-connected, or transposed convolutional layers into the high-resolution patch. For example, this network could take a 30 x 30 low-resolution patch, convolve over it a couple times such that the feature map is something like 22 x 22 x 64, flatten it into a vector, apply a couple of fully-connected layers, reshape it, and finally, upsample it into a 30 x 30 high-resolution patch through transposed convolutional layers.

One problem with doing this is that it is difficult to design an effective loss function. The common loss function used for this is the MSE (Mean Squared Error) between the network output patch and the ground truth high-resolution patch. A solution for this loss function is the use of a perceptual loss function developed by Johnson et al. [1]. This loss function is computed by taking the difference in feature map activations in high layers of a VGG network [2] between the network output patch and the high-resolution patch. The feature map activations are thus denoted as a perceptual loss metric.

GANs further improve the development of this loss function as demonstrated by Ledig et al. [3]. In addition to the perceptual, content loss, an adversarial loss is added to further push images towards a natural image manifold. The GAN framework is integrated to denote if the patch created by the generator is similar to a ground truth set of high-resolution patches. The error of the generator network is then calculated through the adversarial loss, as well as the perceptual loss (denoted by the difference in VGG feature map activation from the outputted patch and ground truth patch).

4.3 METHOD

Flowchart for using GANs to accomplish super resolution:

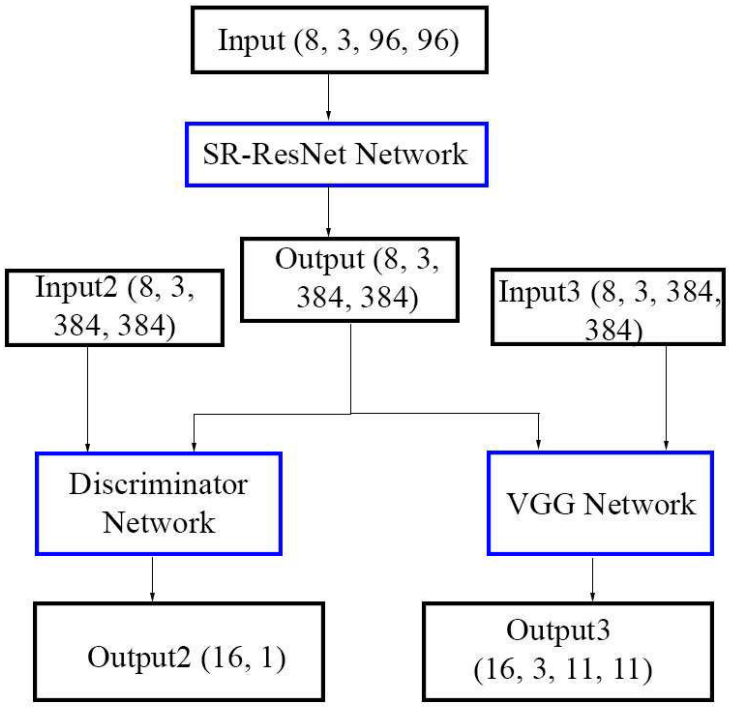

Fig 4.3: Project input and output flow

Training procedure is shown in following steps:

- We process the HR(High Resolution) images to get down-sampled LR(Low Resolution) images. Now we have both HR and LR images for training data set.
- We pass LR images through Generator which up-samples and gives SR(Super Resolution) images.
- We use a discriminator to distinguish the HR images and back-propagate the GAN loss to train the discriminator and the generator.

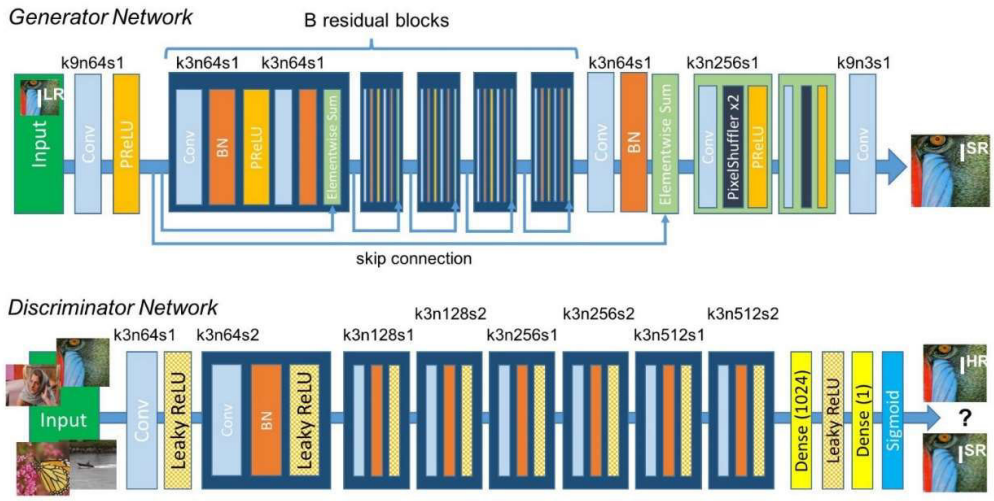

Fig 4.4: Generator and Discriminator Network

Above is the network design for the generator and the discriminator. It mostly composes of convolution layers, batch normalization and parameterized ReLU (PRelU). The generator also implements skip connections similar to ResNet.

Few things to note from Network architecture:

- Residual blocks: Since deeper networks are more difficult to train. The residual learning framework eases the training of these networks, and enables them to be substantially deeper, leading to improved performance. More about Residual blocks and Deep Residual learning can be found in paper given below. 16 residual blocks are used in Generator.
- PixelShuffler x2: This is feature map upscaling. 2 sub-pixel CNN are used in Generator. Upscaling or Upsampling are same. There are various ways to do that. In code keras inbuilt function has been used.
- PRelu(Parameterized Relu): We are using PRelu in place of Relu or LeakyRelu. It introduces learn-able parameter that makes it possible to adaptively learn the negative part coefficient.
- k3n64s:1 this means kernel 3, channels 64 and strides 1.

- Loss Function: This is most important part. As discussed we will be using Perceptual loss. It comprises of Content(Reconstruction) loss and Adversarial loss.

$$l^{SR} = \underbrace{\underbrace{l_{\mathrm{X}}^{SR}}_{\text{content loss}} + \underbrace{10^{-3} l_{Gen}^{SR}}_{\text{adversarial loss}}}_{\text{perceptual loss (for VGG based content losses)}} \tag{4.4}$$

- Adversarial loss: This pushes our solution to the natural image manifold using a discriminator network that is trained to differentiate between the super-resolved images and original photo-realistic images.

$$l_{Gen}^{SR} = \sum_{n=1}^{N} -\log D_{\theta_D}(G_{\theta_G}(I^{LR})) \tag{4.5}$$

- Content Loss: Content loss we are using so that we can keep perceptual similarity instead of pixel wise similarity. This will allow us to recover photo-realistic textures from heavily down sampled images. Instead of relying on pixel-wise losses we will and use a loss function that is closer to perceptual similarity. We define the VGG loss based on the ReLU activation layers of the per-trained 19 layer VGG network. VGG loss is defined as the euclidean distance between the feature representations of a reconstructed image and the reference image.

$$l_{VGG/i.j}^{SR} = \frac{1}{W_{i,j} H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2 \tag{4.5}$$

SRGAN uses a perceptual loss measuring the MSE of features extracted by a VGG-19 network. For a specific layer within VGG-19, we want their features to be matched (Minimum MSE for features). More about perceptual loss you can find in original paper.

4.3.1 Broad Overview of input / outputs nodes in the network

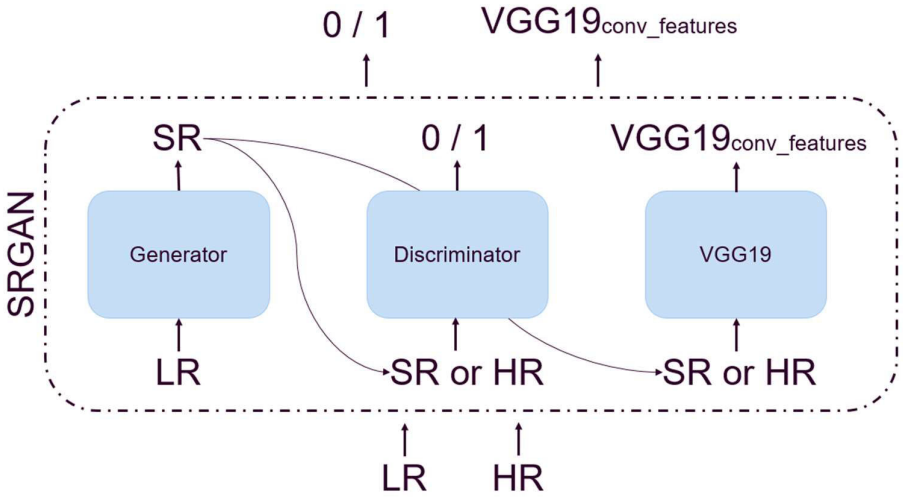

Fig 4.5: Broad overview of input/outputs nodes in the network.

Overview of the three networks; generator, discriminator, and VGG19. Generator create SR image from LR, discriminator predicts whether it's a SR or original HR, and VGG19 extracts features from generated SR and original HR images.

4.3.2 Tools and Frameworks used

1. Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of

the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

The Python 2 language, i.e. Python 2.7.x, is "sunsetting" on January 1, 2020 (after extension; first planned for 2015), and the Python team of volunteers will not fix security issues, or improve it in other ways after that date. With the end-of-life, only Python 3.5.x and later will be supported.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

AI projects differ from traditional software projects. The differences lie in the technology stack, the skills required for an AI-based project, and the necessity of deep research. To implement your AI aspirations, you should use a programming language that is stable, flexible, and has tools available. Python offers all of this, which is why we see lots of Python AI projects today.

From development to deployment and maintenance, Python helps developers be productive and confident about the software they're building. Benefits that make Python the best fit for machine learning and AI-based projects include simplicity and consistency, access to great libraries and frameworks for AI and machine learning (ML), flexibility, platform independence, and a wide community. These add to the overall popularity of the language.

Python is simple and consistent. It offers concise and readable code. While complex algorithms and versatile workflows stand behind machine learning and AI, Python's simplicity allows developers to write reliable systems. Developers get to put all their effort into solving an ML problem instead of focusing on the technical nuances of the language.

Additionally, Python is appealing to many developers as it's easy to learn. Python code is understandable by humans, which makes it easier to build models for machine learning.

Many programmers say that Python is more intuitive than other programming languages. Others point out the many frameworks, libraries, and extensions that simplify the implementation of different functionalities. It's generally accepted that Python is suitable for collaborative implementation when multiple developers are involved. Since Python is a

general-purpose language, it can do a set of complex machine learning tasks and enable you to build prototypes quickly that allow you to test your product for machine learning purposes.

Python has extensive selection of libraries and frameworks. Implementing AI and ML algorithms can be tricky and requires a lot of time. It's vital to have a well-structured and well-tested environment to enable developers to come up with the best coding solutions.

To reduce development time, programmers turn to a number of Python frameworks and libraries. A software library is pre-written code that developers use to solve common programming tasks. Python, with its rich technology stack, has an extensive set of libraries for artificial intelligence and machine learning. Here are some of them:

- Keras, TensorFlow, and Scikit-learn for machine learning
- NumPy for high-performance scientific computing and data analysis
- SciPy for advanced computing
- Pandas for general-purpose data analysis
- Seaborn for data visualization

Scikit-learn features various classification, regression, and clustering algorithms, including support vector machines, random forests, gradient boosting, k-means, and DBSCAN, and is designed to work with the Python numerical and scientific libraries NumPy and SciPy.

2. Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Keras (κέρας) means horn in Greek. It is a reference to a literary image from ancient Greek and Latin literature, first found in the Odyssey, where dream spirits (Oneiroi, singular Oneiros) are divided between those who deceive men with false visions, who arrive to Earth through a gate of ivory, and those who announce a future that will come to pass, who arrive through a gate of horn. It's a play on the words κέρας (horn) / κραίνω (fulfill), and ἐλέφας (ivory) / ἐλεφαίρομαι (deceive).

Keras was initially developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System).

Keras is recommended for deep learning because:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Keras guiding principles:

- **User friendliness**. Keras is an API designed for human beings, not machines. It puts user experience front and center. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.
- **Modularity**. A model is understood as a sequence or a graph of standalone, fully configurable modules that can be plugged together with as few restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions and regularization schemes are all standalone modules that you can combine to create new models.
- **Easy extensibility**. New modules are simple to add (as new classes and functions), and existing modules provide ample examples. To be able to easily create new modules allows for total expressiveness, making Keras suitable for advanced research.
- **Work with Python**. No separate models configuration files in a declarative format. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

The following is a simple code using python and keras to implement a basic neural network. The complete example below makes predictions for each example in the dataset, then prints the input data, predicted class and expected class for the first 5 examples in the dataset.

```
from numpy import loadtxt

from keras.models import Sequential

from keras.layers import Dense

# load the dataset

dataset = loadtxt('pima-indians-diabetes.csv', delimiter=',')

# split into input (X) and output (y) variables

X = dataset[:,0:8]

y = dataset[:,8]

# define the keras model

model = Sequential()

model.add(Dense(12, input_dim=8, activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

# compile the keras model

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# fit the keras model on the dataset

model.fit(X, y, epochs=150, batch_size=10, verbose=0)

# make class predictions with the model

predictions = model.predict_classes(X)

# summarize the first 5 cases

for i in range(5):

print('%s => %d (expected %d)' % (X[i].tolist(), predictions[i], y[i]))
```

3. Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 on November 9, 2015

TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.

In March 2018, Google announced TensorFlow.js version 1.0 for machine learning in JavaScript and TensorFlow Graphics for deep learning in computer graphics.

In Jan 2019, Google announced TensorFlow 2.0.

TensorFlow offers multiple levels of abstraction so you can choose the right one for your needs. Build and train models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy.

If you need more flexibility, eager execution allows for immediate iteration and intuitive debugging. For large ML training tasks, use the Distribution Strategy API for

distributed training on different hardware configurations without changing the model definition.

TensorFlow has always provided a direct path to production. Whether it's on servers, edge devices, or the web, TensorFlow lets you train and deploy your model easily, no matter what language or platform you use.

Use TensorFlow Extended (TFX) if you need a full production ML pipeline. For running inference on mobile and edge devices, use TensorFlow Lite. Train and deploy models in JavaScript environments using TensorFlow.js.

Build and train state-of-the-art models without sacrificing speed or performance. TensorFlow gives you the flexibility and control with features like the Keras Functional API and Model Subclassing API for creation of complex topologies. For easy prototyping and fast debugging, use eager execution.

TensorFlow also supports an ecosystem of powerful add-on libraries and models to experiment with, including Ragged Tensors, TensorFlow Probability, Tensor2Tensor and BERT.

4. Numpy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

NumPy is licensed under the BSD license, enabling reuse with few restrictions.

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

A powerful N-dimensional array object

- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.
Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Installation:

- Mac and Linux users can install NumPy via pip command:

>>>pip install numpy

The ndarray data structure:

The core functionality of NumPy is its "ndarray", for n-dimensional array, data structure. These arrays are strided views on memory.[7] In contrast to Python's built-in list data structure (which, despite the name, is a dynamic array), these arrays are homogeneously typed: all elements of a single array must be of the same type.

Such arrays can also be views into memory buffers allocated by C/C++, Cython, and Fortran extensions to the CPython interpreter without the need to copy data around, giving a degree of compatibility with existing numerical libraries. This functionality is exploited by the SciPy package, which wraps a number of such libraries (notably BLAS and LAPACK). NumPy has built-in support for memory-mapped ndarrays.

5. MatplotLib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter, has an active development community, and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012, and further joined by Thomas Caswell.

Matplotlib 2.0.x supports Python versions 2.7 through 3.6. Python3 support started with Matplotlib 1.2. Matplotlib 1.4 is the last version to support Python 2.6. Matplotlib has pledged to not support Python 2 past 2020 by signing the Python 3 Statement

Matplotlib ships with several add-on toolkits, including 3d plotting with mplot3d, axes helpers in axes_grid1 and axis helpers in axisartist.

Third party packages

A large number of third party packages extend and build on Matplotlib functionality, including several higher-level plotting interfaces (seaborn, holoviews, ggplot, …), and two projection and mapping toolkits (basemap and cartopy).

Citing Matplotlib

Matplotlib is the brainchild of John Hunter (1968-2012), who, along with its many contributors, have put an immeasurable amount of time and effort into producing a piece of software utilized by thousands of scientists worldwide.

Plotting graphs and images are very easy using matplotlib. Following is an example that illustrates the same :

```
>>> import matplotlib.pyplot as plt
>>> from numpy.random import normal,rand
```

```
>>> x = normal(size=200)
>>> plt.hist(x, bins=30)
>>> plt.show()
```

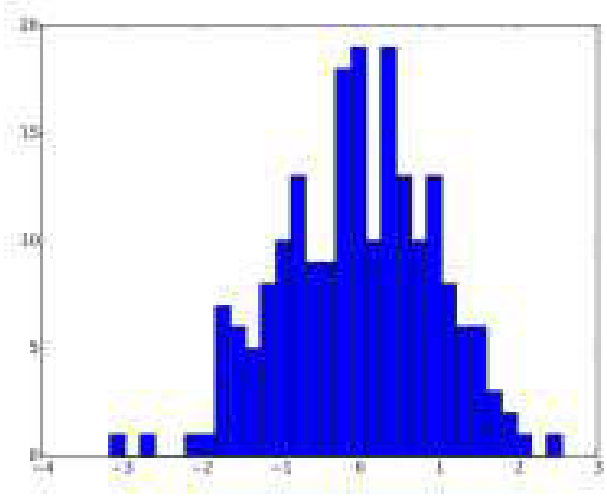

Fig 4.6 Demo graph for matplot

6. Scikit-image

Image Processing SciKit (Toolbox for SciPy).

scikit-image (a.k.a. skimage) is a collection of algorithms for image processing and computer vision.

The main package of skimage only provides a few utilities for converting between image data types; for most features, you need to import one of the following sub-packages:

Some Sub-packages examples

- Color: Color space conversion.
- Data: Test images and example data.
- Draw: Drawing primitives (lines, text, etc.) that operate on NumPy arrays.
- Exposure: Image intensity adjustment, e.g., histogram equalization, etc.
- Feature: Feature detection and extraction, e.g., texture analysis corners, etc.

- Filter: Sharpening, edge finding, rank filters, thresholding, etc.
- Graph: Graph-theoretic operations, e.g., shortest paths.

scikit-image is an image processing Python package that works with numpy arrays which is a collection of algorithms for image processing. Let's discusses how to deal with images into set of information and its some application in real world.

Important features of scikit-image:

- Simple and efficient tools for image processing and computer vision techniques.
- Accessible to everybody and reusable in various contexts.
- Built on the top of NumPy, SciPy, and matplotlib.
- Open source, commercially usable – BSD license.

Note: Before installing scikit-image, ensure that NumPy and SciPy are pre-installed. Now, the easiest way to install scikit-image is using pip:

```
>>>pip install -U scikit-image
```

Most functions of skimage are found within submodules. Images are represented as NumPy arrays, for example 2-D arrays for grayscale 2-D images.

CHAPTER 5

REFERENCES

[1] Sung Cheol Park, Min Kyu Park and Moon Gi Kang, "Super- Resolution Image Reconstruction: A Technical Overview", IEEE Signal Processing Magazine, 1053-5888/03, pp. 21-36, May, 2003.

[2] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar, "Fast and Robust Multi-frame Super-resolution", IEEE Transactions on Image Processing, vol. 13, no. 10, pp. 1327-1344, October, 2004.

[3] Adam W. M. van Eekeren, Klamer Schutte, and Lucas J. van Vliet, "Multi-frame Super-Resolution Reconstruction of Small Moving Objects", IEEE Transactions on Image Processing, pp. 2901-2912, Vol. 19, No. 11, November, 2010.

[4] Aggelos Katsaggelos, Rafael Molina, and Javier Mateos, "Super Resolution of Images and Video", Synthesis Lectures on Images, Video and Multimedia Processing, Morgan& Clayppo Publishers, 2007.

[5] Xianghua Houa and Honghai Liu, "Super-resolution Image Reconstruction for Video Sequence", 2011 International Conference on Electronic & Mechanical Engineering and Information Technology, pp. 4600-4603, 12-14 August, 2011.

[6] Matan Protter, Michael Elad, Hiroyuki Takeda, and Peyman Milanfar, "Generalizing the Nonlocal-Means to Super-Resolution Reconstruction", IEEE Transactions on Image Processing, pp. 36-51, Vol. 18, No. 1, Jan. 2009.

[7] D. Glasner, S. Bagon, and M. Irani, "Super-Resolution from a Single Image", International Conference on Computer Vision (ICCV), October 2009.

[8] S. Panda, R.S. Prasad, and G. Jena, "POCS Based Super-Resolution Image Reconstruction Using an Adaptive Regularization Parameter", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 5, No. 2, September 2011, ISSN (Online), 1694-0814.

[9] Camponez,M.O., Salles,E.O.T., Filho,M.S. "Super-Resolution Image Reconstruction Using Nonparametric Bayesian INLA Approximation", IEEE Transactions on Image Processing ,21(8), 3491-3501, 2012.

[10] Yang S., Wang M., Chen Y., Sun Y. "Single-Image Super-Resolution Reconstruction via Learned Geometric Dictionaries and Clustered Sparse Coding", IEEE Transactions on Image Processing, 21(9), 4016-4028, 2012.

[11] N. Matsumoto and T. Ida, 'A Study on One Frame Reconstruction-based Super-resolution Using Image Segmentation", IEICE Technical Report, SIP2008-6, IE2008-6 (2008- 04) (in Japanese).

[12] N. Matsumoto and T. Ida, "Reconstruction Based Super- Resolution Using Self-Congruency around Image Edges", Journal of IEICE, Vol. J93-D, No. 2, pp. 118-126, Feb. 2010 (in Japanese).

[13] 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.

[14] 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.

[15] Seiichi Gohshi and Isao Echizen, "Subjective Assessment for HDTV with Super-Resolution function", Seventh International Workshop on Video Processing and Quality Metrics for Consumer Electronics - (VPQM2013), Jan. 2013.

[16] Christian Ledig, Lucas Theis, Ferenc Husz´ar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shir. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network". In 2017, IEEE Conference on Computer Vision and Pattern Recognition.

[17] Q. Yang, R. Yang, J. Davis, and D. Nistér. Spatial-depth super resolutionforrangeimages. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–8, 2007.

[18] W. Zou and P. C. Yuen. Very Low-Resolution Face Recognition in Parallel Environment. IEEE Transactions on Image Processing, 21:327–340, 2012

[19] K. Nasrollahi and T. B. Moeslund. Super-resolution: A comprehensive survey. In Machine Vision and Applications, volume 25, pages 1423–1468. 2014

[20] C.-Y. Yang, C. Ma, and M.-H. Yang. Single-image super-resolution: A benchmark. In European Conference on Computer Vision (ECCV), pages 372–386. Springer, 2014.

[21] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multi-scale structural similarity for image quality assessment. In IEEE Asilomar Conference on Signals, Systems and Computers, volume 2, pages 9–13, 2003.

[22] 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

[23] P. Gupta, P. Srivastava, S. Bhardwaj, and V. Bhateja. A modified psnr metric based on hvs for quality assessment of color images. In IEEE International Conference on Communication and Industrial Application (ICCIA), pages 1–4, 2011

[24] J. A. Ferwerda. Three varieties of realism in computer graphics. In Electronic Imaging, pages 290–297. International Society for Optics and Photonics, 2003.

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

[26] J. Johnson, A. Alahi, and F. Li. Perceptual losses for real-time style transfer and super-resolution. In European Conference on Computer Vision (ECCV), pages 694–711. Springer, 2016.

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

[28] S. Borman and R. L. Stevenson. Super-Resolution from Image Sequences-A Review Midwest Symposium on Circuits and Systems, pages 374–378, 1998.

[29] S. Farsiu, M. D. Robinson, M. Elad, and P. Milanfar. Fast and robust multiframe super resolution. IEEE Transactions on Image Processing, 13(10):1327–1344, 2004.

[30] C. E. Duchon. Lanczos Filtering in One and Two Dimensions. In Journal of Applied Meteorology, volume 18, pages 1016–1022. 1979.

[31] J.AllebachandP.W.Wong. Edge-directedinterpolation. InProceedings of International Conference on Image Processing, volume 3, pages 707–710, 1996

[32] X. Li and M. T. Orchard. New edge-directed interpolation. IEEE Transactions on Image Processing, 10(10):1521–1527, 2001.

[33] X. Li and M. T. Orchard. New edge-directed interpolation. IEEE Transactions on Image Processing, 10(10):1521–1527, 2001.

[34] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based super-resolution. IEEE Computer Graphics and Applications, 22(2):56–65, 2002

[35] 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

[36] W. Dong, L. Zhang, G. Shi, and X. Wu. Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization. IEEE Transactions on Image Processing, 20(7):1838–1857, 2011.

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

[38] 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.

[39] 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.

[40] 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

[41] 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.

[42] 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.

[43] 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), pages 1114–1121, 2012.

[44] H. Yue, X. Sun, J. Yang, and F. Wu. Landmark image super-resolution by retrieving web images. IEEE Transactions on Image Processing, 22(12):4865–4878, 2013

[45] R. Timofte, V. De, and L. Van Gool. Anchored neighborhood regression for fast example-based super-resolution. In IEEE International Conference on Computer Vision (ICCV), pages 1920–1927, 2013.

[46] R. Timofte, V. De Smet, and L. Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In Asian Conference on Computer Vision (ACCV), pages 111–126. Springer, 2014.

[47] S. Schulter, C. Leistner, and H. Bischof. Fast and accurate image upscaling with super-resolution forests. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3791–3799, 2015

[48] D. Dai, R. Timofte, and L. Van Gool. Jointly optimized regressors for image super-resolution. In Computer Graphics Forum, volume 34, pages 95–104, 2015.

[49] 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), pages 370–378, 2015

[50] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), pages 399–406, 2010.