Super Resolution of occluded or unclear faces using Generative Adversarial Networks

Nair, Soni, Subramani, Yang

I. Summary

Image generation or super resolution of images always carry a big tradeoff between quantitative quality and perceptual quality of images. A lot of available approaches end up with great quantitative results but end up missing out on perceptual quality of images. Perceptual quality is defined as the ability to capture intricate details in features like texture, color and shapes of objects in an image or video. Most image processing approaches leave the question of how to preserve perceptual quality unanswered. Our team aims to address and solve this issue using the concepts of Generative Adversarial Networks by using its variant SRGAN (Super Resolution Generative Adversarial Network), which aims to recover / preserve perceptual quality (the finer details) of images while super-resolving at large upscaling factors while also ensuring that it doesnt compromise on the quantitative performance.

Estimating a high-resolution (HR) image from its low-resolution (LR) counterpart is referred to as super-resolution (SR). We experimented the SRGAN (Super Resolution Generative Adversarial Network) with multiple face images and extended our model towards video footage Super Resolution and migrated the model to AWS for scalability and performance.

1) Related Work: A variety of work has been done in the field of image super resolution. Convolutional neural network (CNN) based super resolution models have shown great performances. A CNN class GAN called Deep Convolutional Generative Adversarial Network was proposed by Radford, et al.. It generates superior results in various topics.

2) Data: The super resolution models are experimented on a widely used benchmark dataset Celeb-A. Celeb-A is a large-scale facial attributes dataset with 202,599 face images of 10,177 unique identities. The images are mostly frontal images and less occluded which might create a bias in the model. To make sure we eliminate this bias, we implemented our current model on an Indian Movie Face Database (IMFDB). IMFDB is a large unconstrained face database consisting of 34512 images of 100 Indian actors collected from more than 100 videos. Unlike the Celeb-A dataset the faces in IMFDB are collected from videos collected over the last two decades by manual selection and cropping video frames resulting a diversity in age, poses, dress patterns, expressions etc.

Once the desired Super Resolution images were obtained from the model using our image (Celeb-A) dataset, we extended our model to video (ChoKePoint) dataset, designed for person identification or verification under real-world surveillance conditions using existing technologies. This dataset is obtained from array of cameras, capturing the subjects walking through each portal, say pedestrian traffic. The faces in such dataset will have variations. The dataset consists of 25 subjects (19 male and 6 female) in portal 1. The dataset has frame rate of 30 fps and the image resolution is 800X600 pixels. We took 1 video sequence and it consists of 2292 face images. In the sequence, only one subject is presented in the image at a time. In additional to the aforementioned facial variations, the sequences were presented with continuous occlusion. This phenomenon presents challenges in identity tracking and face verification.

3) Architecture: Generative Adversarial Network is a combination of two deep neural networks working adversely against each other. They are generator and discriminator networks. The networks consists of a complex network of layers such as convolutional layers, activation layers and batch normalizations. Generator (as the name implies) attempts to generate images based on the input (here low-resolution images or down-sampled images), and the discriminator tries to distinguish between the generated images and the true images to determine whether the images are real or fake. Both networks try to get the better of each other and are each others adversaries, hence the name Generative Adversarial Networks. Super resolution of a single image can be achieved by implementing GANs as it aims to refine and convert a low-resolution image to high resolution.

For the scope of our project, we adopt a variant of GAN called SRGAN (Super Resolved GAN), that employs a deep learning network inspired from ResNet which deviates from the traditional Mean Square Error (MSE) to a perceptual loss function; which is a combination of Adversarial Loss and Content Loss Function. The model is proposed in Figure 1.

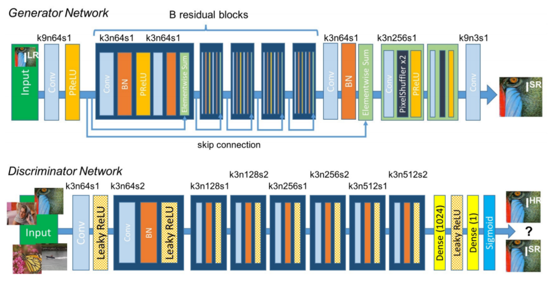


Fig. 1: Architecture

A brief overview of layers and components in the model represented by Figure 1.:

3.1) Convolution Layer: The convolution layer is the main building block of a convolutional neural network. It comprises of set of independent filters. Each filter is independently convolved with the image so we end up with filter size feature maps. The filters in the layers, through back propagation, have tuned themselves to become blobs of colored pieces and edges. As we go deeper to other convolution layers, the filters are doing dot products to the input of the previous convolution layers. So, they are taking the smaller colored pieces or edges and making larger pieces out of them.

3.2) Batch normalization: Batch normalization is used to normalize the inputs of each layer, in order to fight the internal co-variate shift problem (i.e.,) in intermediate layers, the distribution of activation's is constantly changing during training. This slows down the training process as each layer should learn to adapt to a new distribution in every training step.

3.3) Leaky ReLU and parametric ReLU: ReLU stands for rectified linear unit. It is linear for all positive values and 0 for all negative values. The downside for being 0 for all negative values is a problem called dying ReLU. A ReLU neuron is dead if its stuck in the negative side and always outputs 0. Because the slope of ReLU in the negative range is also 0, once a neuron gets negative, its unlikely for it to recover. Such neurons are not playing any role in discriminating the input and is essentially useless. Over the time you may end up with a large part of your network doing nothing. Leaky ReLU has a small slope for negative values, instead of 0 altogether. Parametric ReLU is a type of Leaky ReLU that, instead of having a predetermined slope as in Leaky makes it a parameter for the neural network to figure out, improving accuracy at negligible computational cost. Leaky ReLU fixes dying ReLU problem and speeds up training.

3.4) Skip Connection: Skip connections are extra connections between nodes in different layers of a neural network that skip one or more layers of nonlinear processing. They are essentially connections, from early layers to later layers through addition or straight up concatenation. It is feeding the output of one layer to another layer skipping a few layers in-between of a neural network, for better model convergence. Usually, some information is captured in the initial layers and is required for reconstruction during the up-sampling, done in the fully connected layer. If we would not have used the skip connection, then that information would have been lost or would have turned too abstract to be used further. So, an information that we have in the primary layers can be fed explicitly to the later layers using the skip connection. Skip connections also help traverse information faster in deep neural networks.

3.5) Pixel shuffler: Pixel shuffler layer is used in order to up-scale images. This layer essentially uses regular convolutional layers followed by a specific type of image reshaping called a phase shift. In other words, instead of putting zeros in between pixels and having to do extra computation, they calculate more convolutions in lower resolution and resize the resulting map into an upscaled image. This way, no meaningless zeros are necessary.

3.6) Flatten Layer: Flatten layer brings all levels of multi-layered image down to one plane. It basically converts the 2D or 3D matrix information (breaks spatial structure of data) into a vector that can be fed to the fully connected dense layer.

3.7) Dense Layer: A dense layer is a regular layer of neurons in a neural network. This layer has a weighted matrix w, a bias vector b and the activation's of previous layer. It finds patterns for discrimination in the pixel values that are given as input.

3.8) Residual Block: Residual block has two convolutional layers followed by batch normalization and parametricReLU layers in this architecture.

4) Loss Function: Traditionally GANs have focused on defining their loss functions based on pixel similarity in images by calculating the mean pixel value for a set of images, but that results in really smooth images where the model fails to capture the perceptual quality i.e. the texture of an image or the shape of an object in a particular image. Recent approaches have shown that defining a perceptual loss function is a better approach since it works towards preserving / recovering the features that might be lost in pixel based loss functions.

Perceptual loss function is defined as a combination of two functions Adversarial and Content, both being responsible for learning the weights of Discriminator and the Generator respectively. These two functions form a closed loop between the generator and discriminator where they effectively back-propagate after every iteration making the networks learn weights for better learning.

II. Methods

A. Data Preprocessing

Initially, to generate our input training images, we down sampled our original images using LANCZOS algorithm from 218x178 to 64x64, 32x32 i.e. high resolution (HR) and low resolution (LR) respectively. After obtaining the right set of LR and HR images we fed them into the network as features and labels respectively. To ensure that a one to one mapping between the LR and HR images were maintained, both the images were sorted (Alpha Numeric sorting sorted based on filename) in their corresponding lists.

Once pre-processing was achieved, the next step was to train our GAN with the right set of loss functions and activation functions (Leaky ReLU, PReLU, tanh, sigmoid) so that appropriate weights could be learned, and model parameters could be tuned. LR images were converted to a bit map array to represent images in a quantifiable manner. We used TensorFlows dataset input pipeline to read the images, decode them and then normalize it to values between [-1,1] as the activation functions that we use like tanh tends to perform better in that range. Finally, we zip the LR, HR images and then feed them in batches to our networks.

We used a command line tool ffmpeg for generating frames from video at the frame rate of 30 per second. Original resolution of the frame captured is 800x600 and it is further down-sampled to a low resolution of 256x256 and 64x64, so that it can be fed to the generator.

B. Implementation

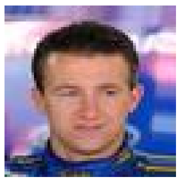
The initial focus of our implementation was to build a framework for single image super resolution and then extend the framework for video super resolution. We used TensorFlow 2.0 for building our GAN model. We ran our model locally to see if it was able to obtain the required results on our machines, but due to the complexity of our model and heavy computations we had to migrate to AWS for scalability and improved performance. In AWS, we used AMI (DLAMI) in an EC2 instance (GPU based) of type p2.xlarge (4 cores, 61 GB memory).

The generator takes the low-resolution images as input and tries to identify the shape, color and texture of objects in our images, generating a new fake image (called SR) based on what was learned from the LR images. These fake / generated images were then fed to the discriminator which also takes in the actual HR images as an input. Based on their quantitative values, discriminator classifies each image as fake (close to 0) or real (close to 1). Initially the generator model is quite naive, but as training progresses the generator becomes better at generating fake images and discriminator gets worse at distinguishing between the generated images and the actual images. Over time, the feedback provided by discriminator becomes less meaningful making the convergence of GAN unstable as it starts to take in random feedback from the discriminator. Initially, the model was experimented with image dataset. After migrating it to AWS instance and obtaining improved SR images, we fed the preprocessed frames extracted from the video sequence to our model. After multiple iterations, we obtained a series of SR frames, which were later combined using command line ffmpeg to convert these frames into a video, hence producing a new Super Resolved video.

Various hyper parameters like kernel size, padding, stridges, optimizers and initializers were tuned for optimizing the model.

III. RESULTS

Phase 1: Once the models were built, we trained it for 500 iterations and observed the following results:


(a) High Resolution

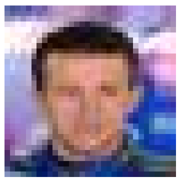
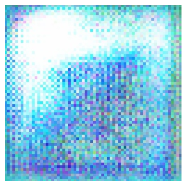
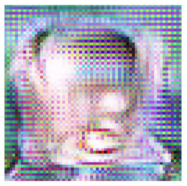

(b) Low Resolution

Fig. 2: Input Images


(a) Super Resolution Iteration 1


(b) Super Resolution Iteration 100

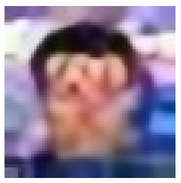
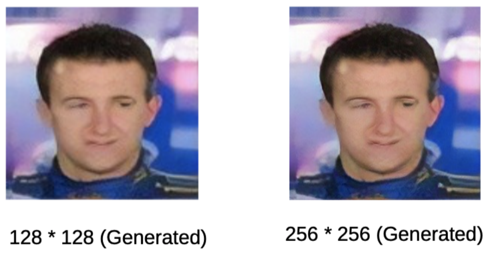
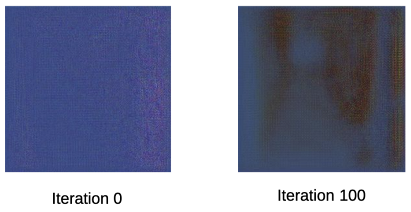

(c) Super Resolution Iteration 500

Fig. 3: Result Images

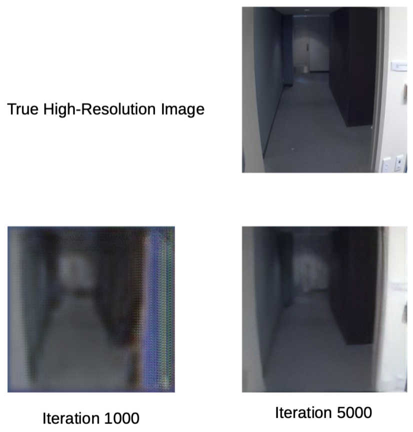
As clear from the Figure 3a, Iteration 1 produces a terrible image showing that the generator hasn't learned much from the input image and it still has to learn all the features of our input image. There is a definite improvement at the 100th iteration (Figure 3b) as now the generator finally starts to learn and recognize features like shape of the object and colour of that object in our image. At 500th iteration (Figure 3c) its clear that there is a significant improvement from the first 100 iterations as we see a face like structure with good amount of colours and a decent background, showing that the generator has finally started to learn the features and is getting better at generating images based on what is being fed. The quality of image generated at the 500th iteration is nowhere near the quality of the actual image (Figure 2a) which shows that the generator still has a long way to go, but was restricted due to the computational challenges faced while running the model on a local machine. This is further discussed in the next section.



(a) First phase improvement



(b) Super Resolution Iteration 1



(c) Super Resolution Iteration 1

Fig. 4: Result - Images Phase II

Phase 2: To improve the performance of image Super Resolution in Phase 1 and extend our model towards video footage Super Resolution (considering scalability and computation resources), we migrated the model to AWS. We re-trained our model in AWS for 3000 iterations on the same dataset used in Phase 1 totaling a runtime execution of around 60 hours. And observed the following results: Compared with Figure 3c (the one at 500th iteration in phase1), there is an clear improvement in Figure 4a (3000th iteration) as the quality of image generated at the 3000th iteration is close to the quality of our actual image, which captured almost all the detail of a human face in Figure 2a.

We then trained our model for 5000 iterations on the video frames extracted from ChoKePoint dataset in AWS totaling a runtime execution of around 45 hours and observed the following results: Figures 4b and 4c represent the image generated by our model at 0, 100th, 1000th and 5000th iteration. Clearly, the model didn't achieve great results initially, but by the 5000th iteration it produces images that are closer to the high resolution frame of our video and its clear that our model captured most of the details.

IV. Discussions

There were a few challenges faced through the course of this project. It was a challenge to implement the network from scratch in TensorFlow, as it required a lot of research to get the architecture right. The next major challenge was the limitations in computational resources. The model being quite huge as it had 14 million trainable parameters in total. We had to considerably reduce the number of training images from 100K to 800 and the upscaling factor to about 2X to avoid running out of memory. To ensure more scalability and better performance, we migrated our code on AWS for which we utilised the AWS Deep Learing AMI (DLAMI - which provides the infrastructure and tools to accelerate deep learning in the cloud, allowing us to define and train models at scale) in an EC2 instance (GPU of type p2.xlarge (4 cores, 61 GB memory). We observed a significant improvement in the model's performance as it took around 60 hours to run 3000 iterations on our image dataset (as compared to 50 hours for 500 iterations - locally) and 45 hours to run 5000 iterations on our video dataset.

V. Conclusions

From the results of Phase 1 and Phase 2 we saw the model performance to be good when the initial low resolution images are not extremely low and occluded. While training the model it was observed that the model leaning rate is considerably low from 2x to 4x upscaling. Finally we found video super resolution to be computationally very expensive even while running on a cloud platform and also that GAN had trouble with far off background in the frames of the video.

VI. Future Work

Observing significant results by using Deep Learning AMI, we plan on looking into AWS SageMaker that allows a more efficient image preprocessing framework and training for machine learning models with pre-trained networks on large datasets such as Imagenet. In future we would also like to try a variation of GAN called DCGAN, which has shown some excellent results recently. Once we optimize the video super resolution model to make the faces clear enough for facial recognition, we would like to build a facial recognition model coupled with a classification model to be used for demographic classification task.

VII. Statement of Contributions

1) Akhil Nair Mrinal Soni: architecture implementation, presentation, report
2) Mounica Subramani Suri Yang: model architecture research, presentation, report, loss functions

VIII. References

C. Ledig, L. Theis, F. Husz ar, J. Caballero, A. Cunningham,A. Acosta, A. Aitken, A. Te- jani, J. Totz, Z. Wang, et al.Photo-realistic single image super-resolution using a gener-ative adversarial network.arXiv preprint arXiv:1609.04802, 2016

A. Radford, L. Metz, and S. Chintala. Unsupervised repre-sentation learning with deep convo- lutional generative adver-sarial networks.arXiv preprint arXiv:1511.06434, 2015.

J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. InTheIEEE Conference on Computer Vision and Pattern Recogni-tion (CVPR), June 2016.

Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning faceattributes in the wild. InProceedings of International Con-ference on Computer Vision (ICCV), 2015.

S. Setty, M. Husain, P. Beham, J. Gudavalli, M. Kandasamy, R. Vaddi, V. Hemadri, J C Karure, R. Raju, Rajan, V. Kumar and C V Jawahar. Indian Movie Face Database: A Benchmark for Face Recognition Under Wide Variations, National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), 2013.

Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization, arXiv:1412.6980, 2014.

Wenming Yang, Xuechen Zhang, Yapeng Tian, Wei Wang, Jing-Hao Xue, Deep Learning for Single Image Super-Resolution: A Brief Review, arXiv:1808.03344, 2018.

Justin Johnson, Alexandre Alahi, Li Fei-Fei, Perceptual Losses for Real-Time Style Transfer and Super-Resolution, arXiv:1603.08155, 2016.