DS 5500: Project Report
Super Resolution of Images and Videos using SRGAN

Srikanth Babu Mandru
MS Data Science
Northeastern University
Boston, MA, USA
mandru.s@husky.neu.edu

I. SUMMARY

Super-resolution (SR) of images refers to the process of generating or reconstructing the high-resolution (HR) images from low-resolution images (LR). This project mainly focuses on dealing with this problem of super-resolution using the generative adversarial network, named SRGAN, a deep learning framework. In this project, SRGAN was trained and evaluated using 'DIV2K', 'MS-COCO' and 'VID4' [6] which are the popular datasets for image resolution tasks.

In total, datasets were merged to form:

1. 5800 training images
2. 100 validation images
3. 4 videos for testing

Apart from the datasets mentioned above, 'LFW', 'Set5' and 'Set14' datasets [6] were used to get inferences and compare the performance of models implemented in this project with the models from Ledig et al. [2].

Most of this project is built upon the ideas of Ledig et al [2]. Apart from that, I did some research on comparing the results obtained using different objective functions available in TensorFlow's "TFGAN" library for loss optimizations of SRGAN. Different model implementations were evaluated for pixel quality through the peak signal-to-noise ratio (PSNR) scores as a metric. Intuitively, this metric does not capture the essence of the perceptual quality of an image. However, it is comparatively easy to use PSNR when evaluating the performance while training the model compared to mean-opinion-score (MOS) that has been used by Ledig et al [2]. To evaluate the perceptual quality of images, I have compared the generated images from both the models. This paper also proposes a method of super-resolution using SRGAN with "Per-Pix loss" which I defined in the losses section of this paper. Based on results from [2] and [5], I have combined both MSE and VGG losses, named it "Per-Pix loss" that stands for 'Perceptual and Pixel' qualities of the image, which resulted in preserving the pixel quality besides improving the perceptual quality of images. Finally, I have compared the models built in this project with the models from Ledig et al. [2] to know the performance and effectiveness of models implemented in this project.

II. METHODS

Methods overview:

Our ultimate goal is to train a generating function (G) that estimates the HR image for a given LR input image. For this purpose of learning G, I have implemented SRGAN which is a GAN based neural network framework optimized for "perceptual loss" and "Per-Pix loss" functions. In this section, I describe methods that I have used for Image preprocessing, SRGAN model implementation, training and deployment.

Image preprocessing:

Image super-resolution model in this paper is a single-image-super-resolution (SISR) which means that it estimates the high-resolution, super-resolved image (denoted by I^{SR} $and\ SR$), from its low-resolution input image (denoted by $I^{LR}$and LR). The SRGAN model expects the input in the form of both low-resolution (fed to generator network) and high-resolution (fed to discriminator network). Thus, we initially need to preprocess the raw data to get LR and HR of input raw image. During training, high-resolution images (denoted by I^{HR} and HR, also called real images) are present and those images were down-sampled by a factor of "r = 4" with the bicubic kernel to obtain low-resolution images.

The original images are of different resolution sizes. For each mini-batch of training images, I have randomly crop 16 HR images of shape 96 × 96 × 3 (in the general case, rW × rH × C) and formed the LR images of size 24 × 24 × 3 (in the general case, W × H × C) as described in the above down-sampling method. The images size was reduced from 256 × 256 × 3 (image size considered during phase 1 of this project) due to mainly two reasons:

1. Super-resolution task is about learning the kind of interpolation operation rather than the content learning

2. For faster training of models

However, there is a possibility of getting better results with increased image size due to the availability of more parameters.

Further, the LR images and HR images were normalized to scale of [0, 1] and [-1, 1] respectively. I have followed the similar preprocessing steps that were used for training images to prepare the validation images in the form of LR and HR.

Example LR and HR images that were obtained from the image preprocessing stage are as shown in the below figures, fig 1 and fig 2:

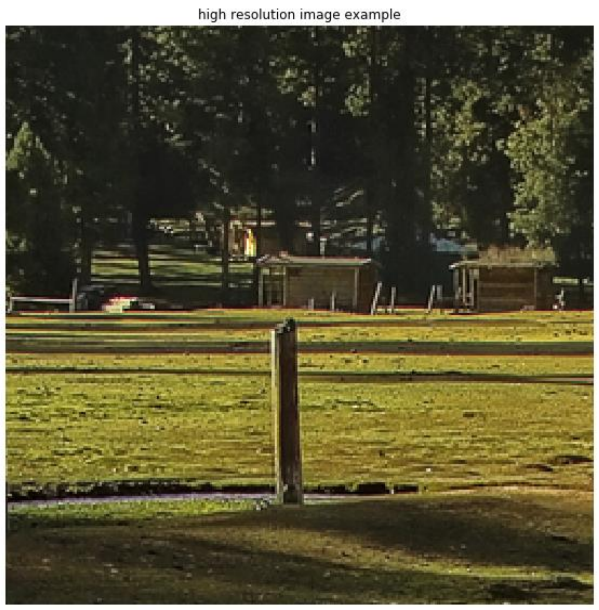


Fig 1: High resolution image (HR) example

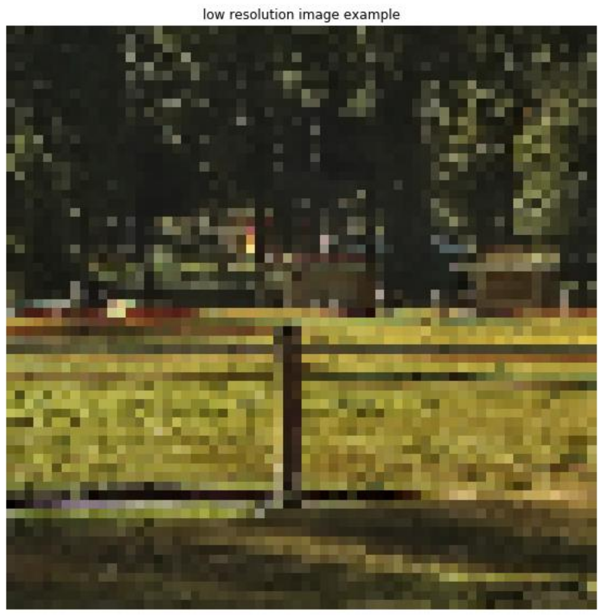


Fig 2: Low resolution image (LR) example

Generative Adversarial Network (GAN):

In general, Generative Adversarial Networks (GAN) [1] constitutes of two main blocks, namely generator block (G) and discriminator block (D). Often, these blocks correspond to neural networks with different architectures.

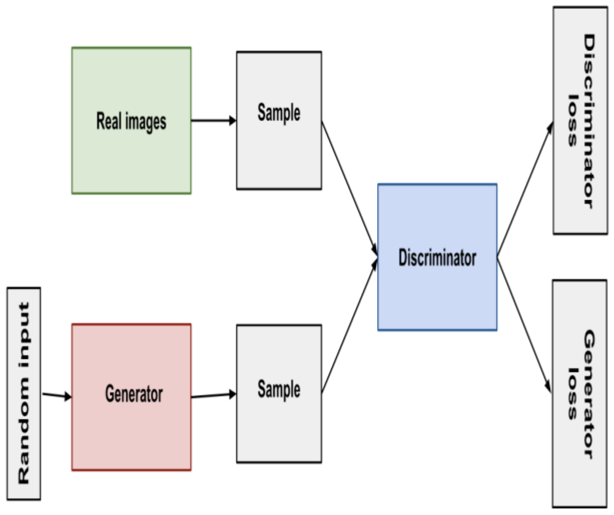


Fig 3: GAN block diagram

The basic idea of GAN is that it allows training a generative model (G) with the intention of fooling a discriminator (D) that is trained to classify super-resolved images from real images. Thus, in GANs, a generator can learn to create outputs that are highly similar to real images which makes difficult for the discriminator to classify. This encourages perceptually superior solutions residing in the subspace, the manifold, of natural images.

While training GANs, generator and discriminator are optimized alternatively with separate loss functions as shown in Fig 3. Generator (G) learns the distribution from the training data and uses that for predictions on future data. Whereas, discriminator (D) learns to extract the feature details to classify the images.

Super-Resolution GAN (SRGAN):

SRGAN is a specific type of GAN model which is built with networks particularly to solve the super-resolution problem. The generator network (G) of SRGAN is comprising of mainly B residual blocks which are identical as shown in Fig 4. In this project's implementation, I have considered 16 blocks to implement the generator network. Each residual block contains two convolutional layers with small 3×3 size kernels and 64 feature filters followed by batch-normalization layers and "Parametric ReLU" as the activation function. Later in the network following residual blocks, there will be two up-sampling blocks with a convolutional layer, pixel shuffler and P-ReLU layers. Here, up-sampling blocks will ensure that a low-resolution input image that is the down-sampled version of a real image will be forwarded through the generator network and produces the super-resolved output image with the same dimensions as real HR image. Generator (G) is trained to optimize the "MSE loss", "VGG loss" and "Per-Pix loss" which will be described in the later section of this paper.

The other major part of SRGAN is discriminator network (D) which is shown in figure 5. At the core of its structure, it is having a block constituting of convolutional layer, batch normalization layer and leaky ReLU activation layers. In the deeper part of the network, feature maps are passed through dense layers and a sigmoid activation layer for final classification results. This discriminator network is trained to maximize the adversarial minimax loss [1] which can be given as in below formula:

$$E_x\left[log\left(D(I^{HR})\right)\right] + E_z\left[log\left(1 - D\left(G(I^{LR})\right)\right)\right] \quad (1)$$

Where, (these definitions also apply to other equations in this paper)

(i) D(..) represents the discriminator's estimate

(ii) E_x is expected value over all real data instances.

(iii) G(..) represents the generator's output

(iv) E_z is the expected value over all LR input images to the generator

The uniqueness of the SRGAN framework lies in the perceptual loss function (l^{SR}) that is used for generator training.

Perceptual Loss:

Perceptual loss is the weighted sum of content loss (l_x^{SR}) and adversarial loss (l_{gen}^{SR}), which is as follows:

$$l^{SR} = l_x^{SR} + 10^{-3} l_{gen}^{SR} \quad (2)$$

Here, 'x' in (l_x^{SR}) represents the type of content loss which are described in the following sub-sections of content loss.

In general terms, it can be defined as follows:

$$\text{Perceptual loss} = \text{Content loss} + \text{Adversarial loss}$$

Content loss:

1. Pixel-wise MSE loss:

MSE loss is widely used for super-resolution and many state-of-art approaches were based on this loss as an objective function. This loss often results in giving high PSNR values. However, this loss lacks the high-frequency details of images and result in an unsatisfactory perceptual quality of the reconstructed image. MSE loss function can be given as:

$$l_{MSE}^{SR} = \frac{1}{r^2WH}\sum_{x=1}^{rW}\sum_{y=1}^{rH}\left(I_{x,y}^{HR} - G(I^{LR})_{x,y}\right)^2 \quad (3)$$

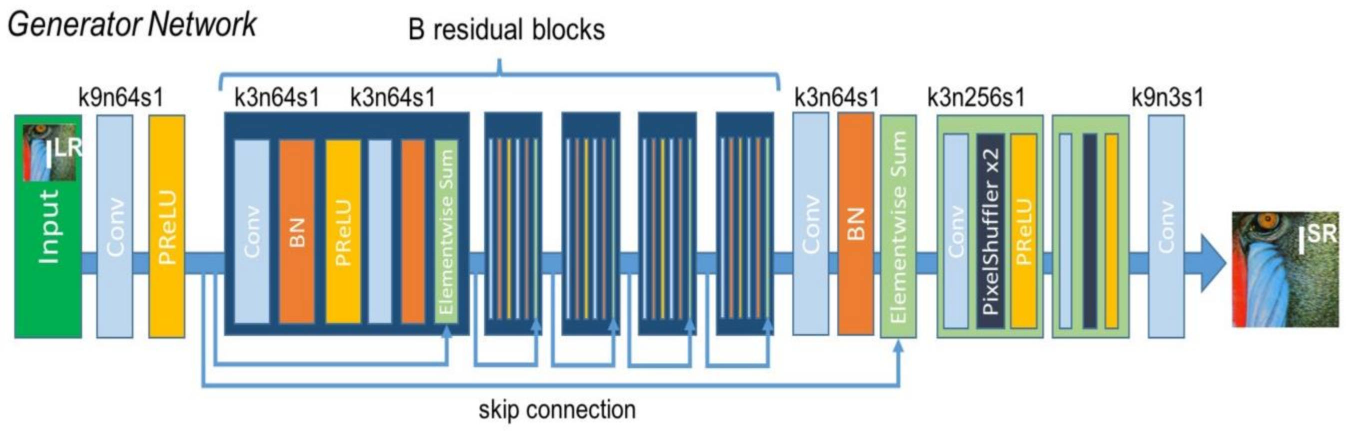


Fig 4: Generator network in SRGAN (source: SRGAN paper [2])

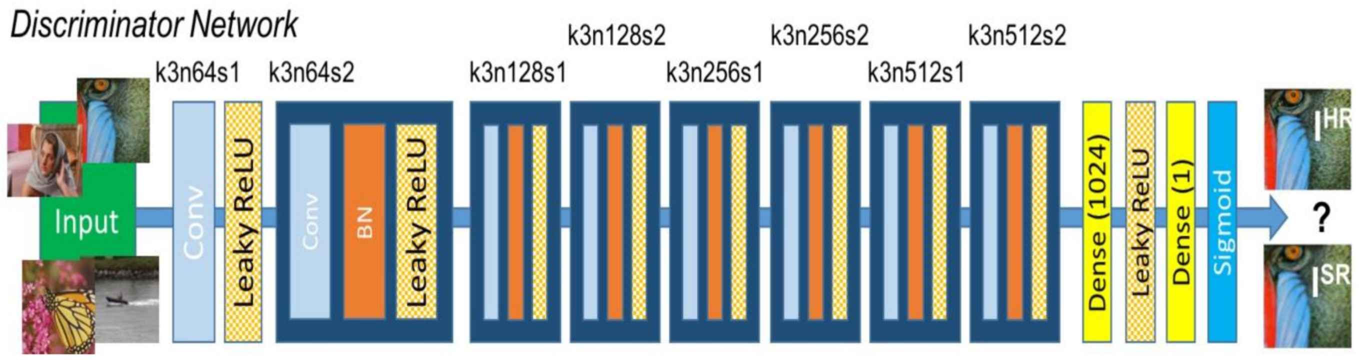


Fig 5: Discriminator network in SRGAN (source: SRGAN paper [2])

2. VGG loss:

It is now clear that MSE based training is not accurate enough to obtain photo-realistic natural images. Thus, to get better super-resolved images, authors from SRGAN paper [2] implemented the loss using the VGG19 network (namely VGG loss). VGG loss is defined as the Euclidean distance between feature representations of the reconstructed image and real image and can be written as in below formula:

$$\left(l_{VGG}^{SR}\right) = \frac{1}{WH}\sum_{x=1}^{W}\sum_{y=1}^{H}\left(F(I^{HR}) - F\left(G(I^{LR})\right)\right)^2 \quad (4)$$

Where, 'F(..)' indicates the feature map obtained from the "block5_conv4" layer output of the VGG network. 'W' and 'H' are the width and height of images or feature maps.

Authors of Ledig et al. [2] have used a pre-trained SR-ResNet model that was optimized with respect to MSE loss as the generator network. Thus, VGG loss will be used after the MSE loss. In general, VGG loss is utilized to make the SRGAN model learn high-frequency details from feature maps obtained using the VGG19 model, which improves the perceptual quality rather than PSNR value.

Adversarial loss:

In addition to content loss, there is generative loss function (adversarial loss component) used in GANs which makes the generator train such that it produces the images that are harder for the discriminator to classify. Generative loss can be defined as:

$$l_{gen}^{SR} = -\sum_{n=1}^{N} log D\left(G(I^{LR})\right) \quad (5)$$

Where, N is the mini-batch-size of training set

Per-Pix loss:

With experimentation of using different losses for generator optimization and observations drawn from results of [2] and [5], to obtain better performance of the model, formed a new loss with the weighted sum of losses, named it as “Per-Pix loss”, as shown below:

$$l^{SR} = w1 * l^{SR}_{MSE} + w2 * l^{SR}_{VGG} + 10^{-3} * l^{SR}_{gen} \quad (6)$$

Where, w1 – weight given to MSE loss

w2 – weight given to VGG loss

Evaluation Metric:

PSNR – Peak signal-to-noise ratio:

For evaluating the pixel quality of the super-resolution image, I choose to use the PSNR as a metric. It is defined as the ratio of the maximum pixel value in the image to the mean-squared error between the real image and the reconstructed image. In general, the higher the PSNR score, the better is the reconstructed image quality. As a formula notation, PSNR can be stated as:

$$PSNR(in\ dB) = 10log_{10}\left(\frac{MAX^2}{MSE}\right) \quad (7)$$

Where, $MSE = \frac{1}{WH}(I^{SR} - I^{HR})^2$

Training details:

I have trained the models using the Adam optimizer [4] with $\beta_1 = 0.9$ and learning rate of 10^{-4} for generator and discriminator respectively. For each training step, model training was alternated between the generator and discriminator that is k=1 as described in [1]. In total, I have trained both models (MSE loss and PERPIX loss trained models) for $3.5 * 10^4$ steps (where each step is training over a mini-batch of training data). The figure illustrating the basic workflow of this project is provided in Appendix E. To describe the workflow, both the models were trained in parallel with NVIDIA Tesla P100 GPUs, one on Google Colaboratory and the other on Google Cloud’s AI platform Deep Learning virtual machine.

While training the model with PERPIX loss, as a part of experimentation, I have set the weights w1 and w2 as follows:

(i) from steps 1 - 15k: w1 = 10 and w2 = 1

(ii) from steps 15k – 20k: w1 = 10 and w2 = 10

(iii) from steps 20k - 35k: w1 = 1 and w2 = 10

Deployment details:

Now, the trained models (best models described in results section) were deployed in the Google Cloud AI platform and can be used for future predictions from almost any application through a REST API call sending JSON payload of shape 4-Dimensional array or tensor, where the first dimension corresponds to the batch size of images and fourth dimension represents the number of color channels of images. The link for the deployment website has been provided in APPENDIX D.

III. RESULTS

SRGAN models were evaluated in terms of pixel and perceptual qualities using PSNR values and visualization of the images respectively. Firstly, I describe the pixel quality evaluation and then explain the perceptual quality evaluation of models.

Pixel quality evaluation:

Obtained the PSNR score on both training and validation datasets using SRGAN with MSE loss as the content loss and results are reported as shown in below figure 6:

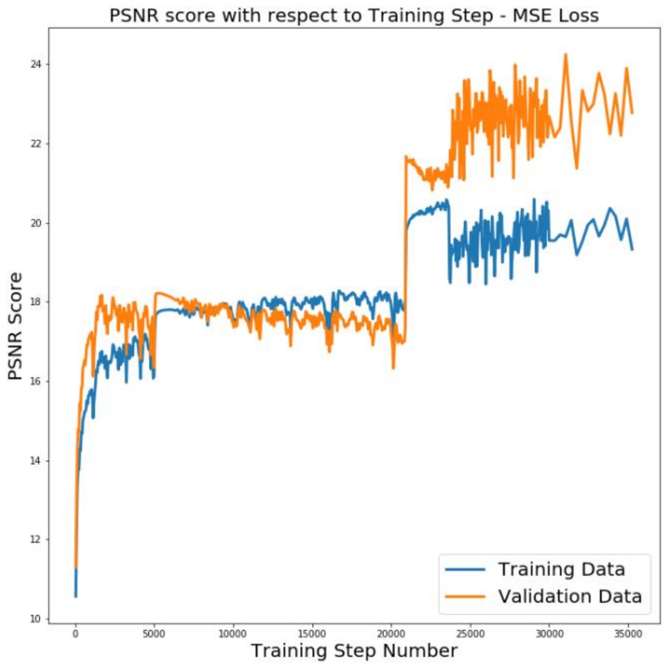


Fig 6: PSNR across training steps for MSE trained model

From figure 6, it can be inferred that the PSNR score has been increasing with training steps (where each step is training over a mini-batch of training data) on both the training and validation data. Since the PSNR score is only on the part of training data and training data is varied due to random cropping of images, it can be considered just to know whether the PSNR scores are improving on training data (this applies to other models in this project). On the validation dataset, the model's PSNR score peaked at approximately "24.0" around the $3.2 * 10^4$ training step. Thus, the best model in the case of MSE training corresponds to the highest PSNR value.

During the experimentation with different losses as the content loss, the SRGAN model was trained with only the VGG loss and obtained the PNSR scores on training and validation datasets as shown in below figure 7. From figure 7, it can be observed that VGG loss alone does not improve the PSNR scores of the SRGAN model. Thus, it is mostly intended to use after the model is trained with MSE loss.

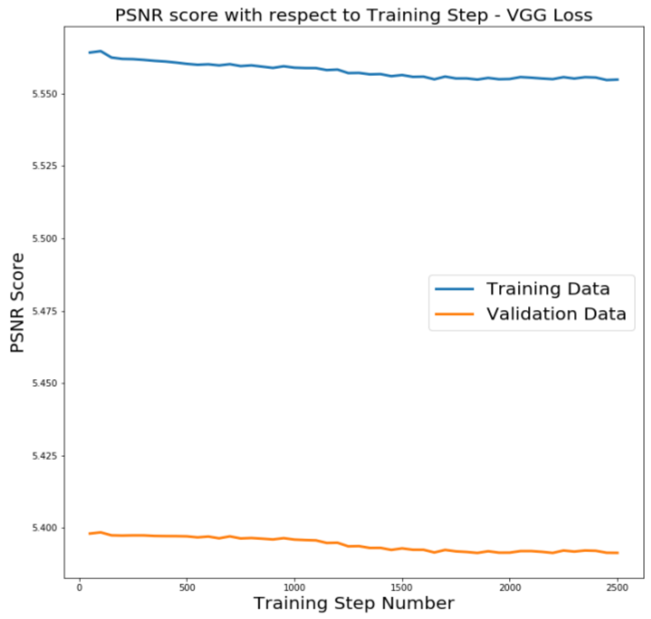


Fig 7: PSNR across training steps for VGG loss trained model

After the experiments with other losses and observations drawn from the results of [2] and [5], I have combined the MSE and VGG losses (Per-Pix loss) giving weights to both loss components. The results in terms of PSNR for the SRGAN model trained with PERPIX loss are plotted in figure 8.

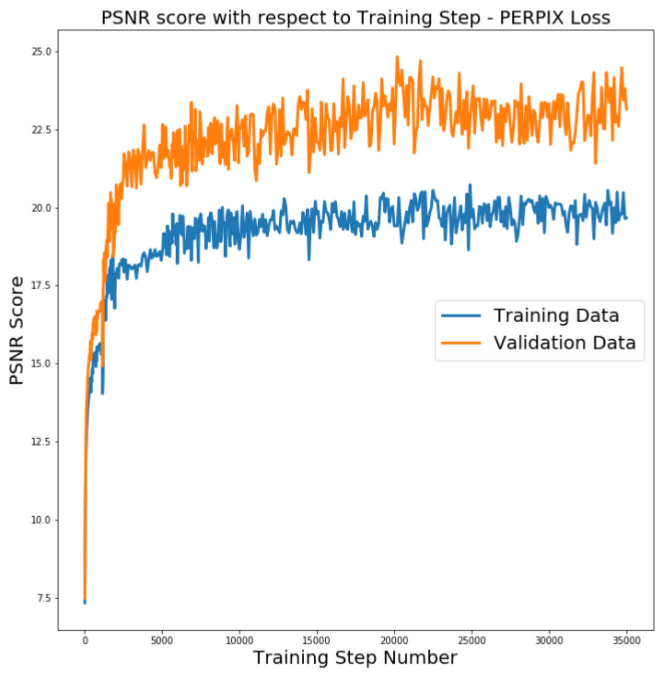


Fig 8: PSNR across training steps for PERPIX trained model

As seen from the figure 8, it can be inferred that due to more weight on the MSE loss from the steps 1 to 15k, SRGAN model has achieved good PSNR values in less number of steps compared to MSE loss trained model (which took around 30k steps). This curve can be extended further with more training steps corresponding to high weight on the MSE component. However, after the 15k training step, equal weights for both MSE and VGG loss components in PERPIX loss resulted in slower improvement in PSNR causing the SRGAN model to reach the highest PSNR value which is approximately "25.0" (on validation dataset) around 20k training step. Later, from the 20k training step, the weight of MSE loss is reduced giving more attention towards VGG loss and we can observe that the PSNR score is oscillating without any rise. However, since the intention of VGG loss is to improve the perceptual quality rather than pixel quality, we can observe the improvements in terms of perceptual quality later. Thus, PERPIX loss results in better PSNR value and can be improved by extending the training of model with more weight on MSE loss to get higher PSNR value and then switch to more weight towards VGG loss component (but "w2" should be less than MSE loss and this limitation is similar to Ledig et al. [2]) to get better photo-realistic images. Since the PSNR does not capture the essence of perceptual quality of image, the SRGAN model trained for 35k steps has been considered as the best model in the case of PERPIX loss training, even if the PSNR is lower than the best PSNR value achieved.

To better understand how efficient the models perform, I have compared the results of models implemented in this project with corresponding model results from Ledig et al. [2] using the same datasets, that is, Set5 and Set14 datasets and results were tabulated in the below figure 9. While comparing, MSE and PERPIX trained models in this project are compared with MSE and VGG54 models from Ledig et al. [2] respectively.

Set5 dataset			
	MSE	**VGG54**	**PERPIX**
SRGAN (from Ledig et al. [2])	30.64	29.40	N/A
SRGAN (In this project)	26.53	N/A	26.57

Set14 dataset			
	MSE	**VGG54**	**PERPIX**
SRGAN (from Ledig et al. [2])	26.92	26.02	N/A
SRGAN (In this project)	23.77	N/A	23.92

Fig 9: Comparison tables of PSNR values (in each table cell) on Set5 and Set14 datasets

From the tables of figure 9, we can observe that the models implemented here in this project are performing well-enough considering the number of training steps that models have been trained for. PSNR values obtained were close to the results from Ledig et al. [2] and with further training, these values will improve.

From figure 9, another noticeable point is that PSNR value was slightly dropped between MSE and VGG54 trained models (from Ledig et al. [2]) on both datasets. With the PERPIX loss, since we are preserving the pixel quality through the MSE loss component besides the VGG component, we can observe a slight increase in PSNR between MSE and PERPIX trained models (in this project) on both datasets. Thus, the SRGAN model trained with PERPIX loss can be considered as the better choice over the VGG54 (from Ledig et al. [2]) in terms of pixel quality that we have measured through PSNR.

Perceptual quality evaluation:

To evaluate the perceptual quality of images, I have compared the images generated from both the models with corresponding down-sampled low-resolution and the original high-resolution images. Images for comparison were included in Appendix B. From figures 10, 11 and 12, we can observe that both SRGAN models are generating better quality images (visually) from the respective down-sampled low-resolution images. However, there is a noticeable difference between generated images and the original high-resolution image. In the case of PERPIX trained model, this difference can be reduced with further iterations as it can be observed from Ledig et al. [2] that it took almost $2 * 10^5$ training steps to reach notifiable perceptual quality. We can also observe from Ledig et al. [2] that it took 20k iterations to actually diverge from the pre-trained SR-ResNet model (trained with MSE loss) and start learning the high-frequency details. Since the model here is trained for around 20k update iterations with high weight on VGG, it started learning high-frequency details and can be inferred from the figures 13 and 14. Thus, with high weight on VGG, PERPIX model possibly converge faster than just using VGG only.

From figures 13 and 14, it can be observed that the generated images from MSE trained model are consisting of pixelated boxes, whereas PERPIX trained model's generated images are more photo-realistic and contain the high-frequency details such as corners around the objects like cups, hands. Also, there are no pixelated boxes as compared to generated images from MSE trained model. Thus, we can infer that the SRGAN model trained with PERPIX loss has started learning the high-frequency details in less number of training steps with more weight towards the VGG loss component of the PERPIX loss.

From the experimentation and results, it is noteworthy to mention that using MSE loss leads to better pixel clarity with little smoothening caused to the generated images. Whereas, VGG loss shows the essence of capturing the high-frequency details in the images taking advantage of feature maps obtained from the deeper network layers of VGG architecture. Apart from these two losses, there is also a "generative or adversarial loss" which made the generator network to learn the texture details from the training images. Each of these losses has a specific functionality in terms of image quality leading to an overall better super-resolution image.

To conclude, both in terms of pixel and perceptual qualities, the PERPIX trained model performs better and considered as the best model among two models implemented in this project and showed satisfactory results compared to models from Ledig et al. [2]. PERPIX trained SRGAN model has the potential to balance between smoothening and high-frequency details of images, which is the desirable property in real-world scenarios.

Using PERPIX trained model, I have done super-resolution inferences on faces and videos, and the link to access those is provided in APPENDIX C.

IV. DISCUSSION

In this paper, focusing on the texture, perceptual and pixel qualities of the super-resolution images, I have incorporated all the losses as stated in the "Per-Pix" loss description for generator training. Using PERPIX loss, the SRGAN model preserves the pixel quality besides improving the perceptual quality of images. Thus, the goal of this project in achieving pixel and perceptual qualities has been solved through the PERPIX loss proposed in this project. From Ledig et al. [2], it can be noticed that it took 20k update iterations to get started with the learning of high-frequency details and 200k iterations or steps to get good photo-realistic super-resolution images. With further training of the SRGAN model, extending training steps with high MSE loss before switching to VGG would result in high PSNR and later with more iterations towards VGG optimization, model converges well to capture high-frequency details. Also, providing weights to the loss components resulted in faster convergence as seen from Fig 6 and Fig 8 (discussed in results section). Apart from the aforementioned benefits, there is no requirement of a pre-trained model for the generator network and providing flexibility to train the model with different weights giving importance towards pixel and perceptual qualities are the other advantages of using Per-Pix loss. To conclude, models implemented in this project perform well considering the training steps and PERPIX loss resulted in decent performance alongside the benefits (discussed above) that are being offered over the VGG only loss component. Since this project also aims at providing the SRGAN model to real-world users, both the models have been deployed into the Google Cloud AI platform and can be used through the REST API calls as mentioned in the deployment details section of this paper. This project forms a good reference to experiment with SRGAN and understand the underpinnings of how different things work and the importance of each aspect of the model. Based on the type of application, the SRGAN model implemented in this project can be fine-tuned through the transfer learning technique with data related to that specific application. In my future work, I would like to experiment with different loss functions along with modifications of the generator network to achieve higher PSNR values. Later, I will build a robust model to capture the feature space of images adding to or replacing the VGG19 architecture.

V. STATEMENT OF CONTRIBUTIONS

Contribution	**Member**
All Project stages	Srikanth Babu Mandru
Project supported by	Prof. Kylie Bemis
Project sponsored by	Quantiphi, Inc.

VI. REFERENCES

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

[2] C. Ledig, L. Theis, F. Husza ́r, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang et al., “Photo-realistic single image super-resolution using a generative adversarial network,” in CVPR, 2017.

[3] Zhihao Wang, Jian Chen, Steven C.H. Hoi, Fellow, "Deep Learning for Image Super-resolution: A Survey", IEEE, 2019.

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

[5] Nao Takano and Gita Alaghband. “SRGAN: Training Dataset Matters”, 2019. (arXiv:1903.09922).

[6] Datasets Link:

DIV2K: https://data.vision.ee.ethz.ch/cvl/DIV2K/

MS-COCO: http://cocodataset.org/#download

Vid4: https://xinntao.github.io/open-videorestoration/rst_src/datasets_sr.html

LFW: https://www.tensorflow.org/datasets/catalog/lfw

Set5 and Set14: https://www.kaggle.com/ll01dm/set-5-14-super-resolution-dataset

[7] Agustsson, Eirikur and Timofte, Radu. “NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study”, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, July 2017.

[8] Zhihao Wang, Jian Chen, Steven C.H. Hoi, Fellow, "Deep Learning for Image Super-resolution: A Survey", IEEE, 2020.

[9] 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.

[10] https://www.tensorflow.org/

VII. APPENDIX

A. GitHub link for code repository:

GitHub Link for project code implementation

B. Perceptual Quality Evaluation:

The below figures 10, 11 and 12, shows the generated images from MSE and PERPIX loss trained models besides down-sampled low-resolution image on which model generates super-resolution image. Apart from those images, the original image is also displayed as a reference for the high-resolution image. These images were part of test datasets, that is Set5 and Set14.

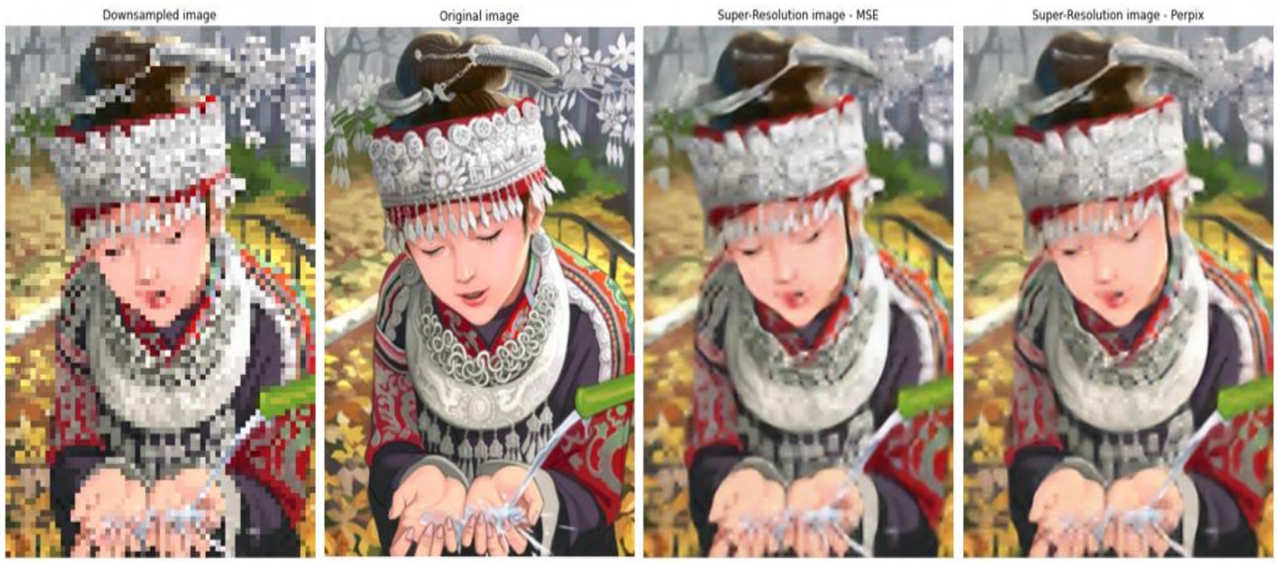


Fig 10: Down-sampled image, Original image, generated images from MSE and PERPIX trained models (from left to right)

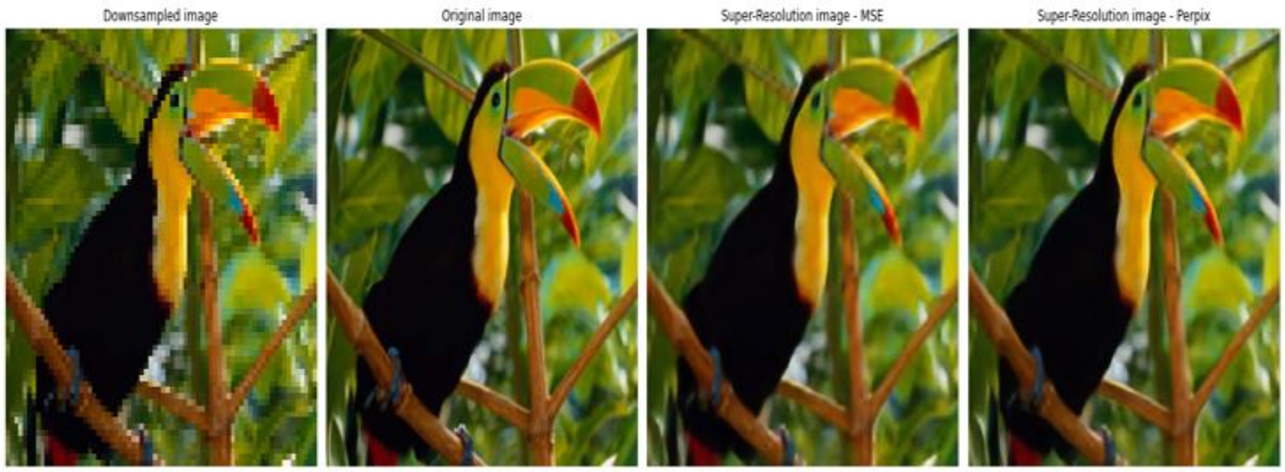


Fig 11: Down-sampled image, Original image, generated images from MSE and PERPIX trained models (from left to right)

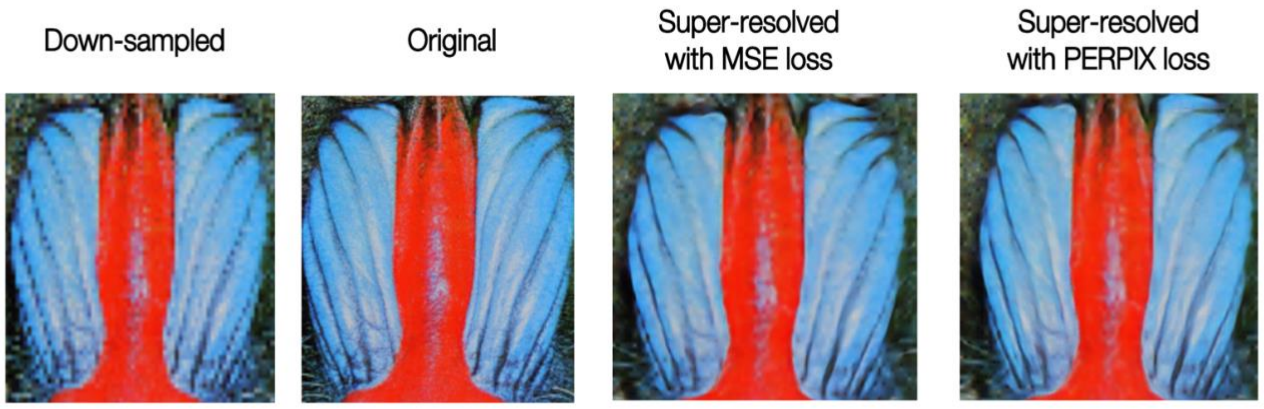


Fig 12: Down-sampled image, Original image, generated images from MSE and PERPIX trained models (from left to right)

Below figures, Fig13 and Fig 14, shows the original (real) images and generated images from MSE and PERPIX trained models on the validation dataset.

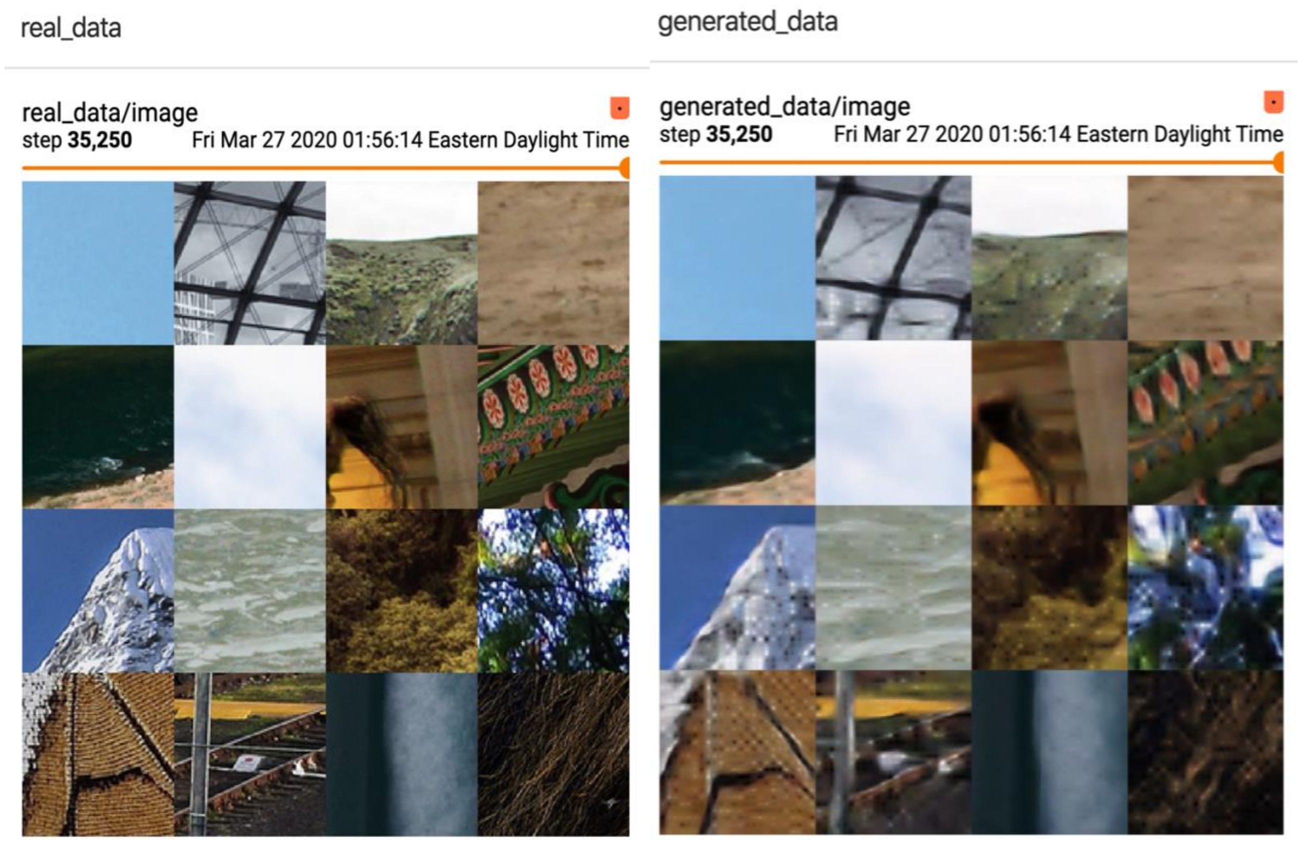

real_data
real_data/image
step 35,250 Fri Mar 27 2020 01:56:14 Eastern Daylight Time
generated_data
generated_data/image
step 35,250 Fri Mar 27 2020 01:56:14 Eastern Daylight Time

Fig 13: Original images (on left), generated images from MSE trained model (on right)

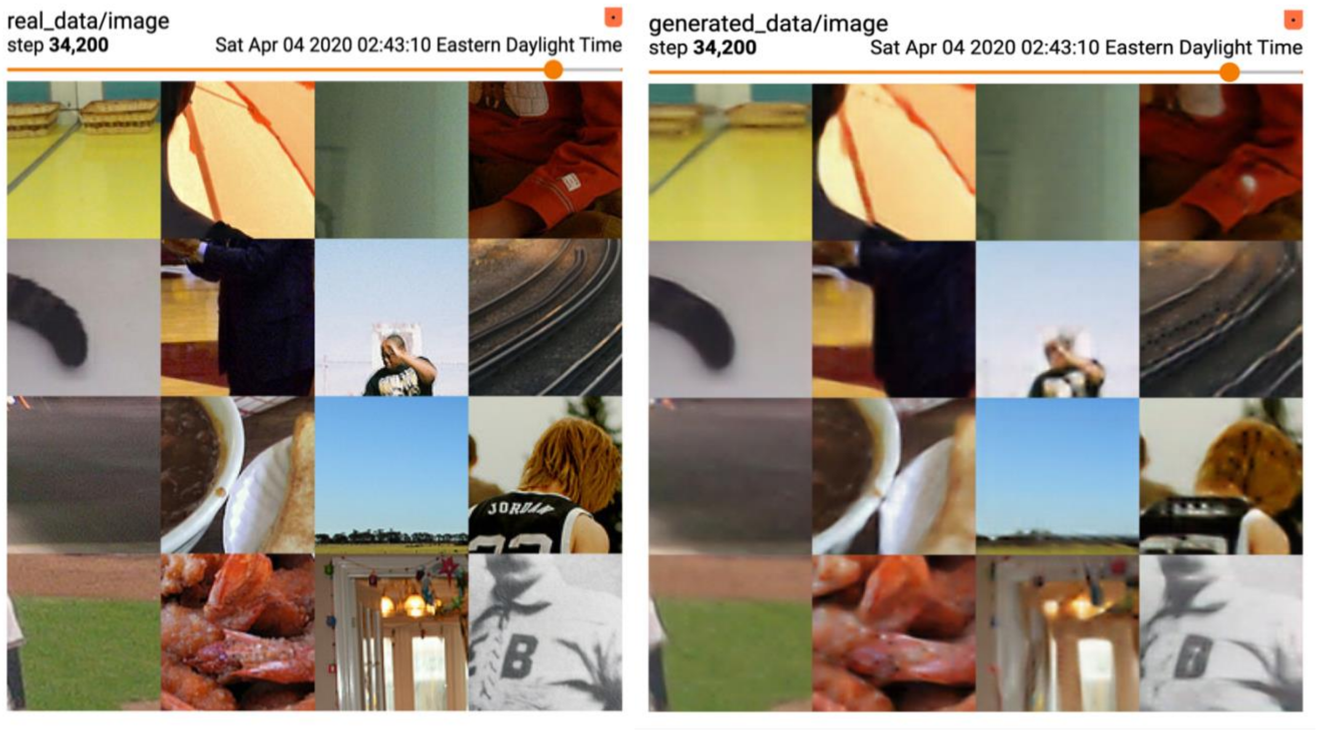

real_data/image
step 34,200
Sat Apr 04 2020 02:43:10 Eastern Daylight Time
generated_data/image
step 34,200
Sat Apr 04 2020 02:43:10 Eastern Daylight Time

Fig 14: Original images (on left), generated images from PERPIX trained model (on right)

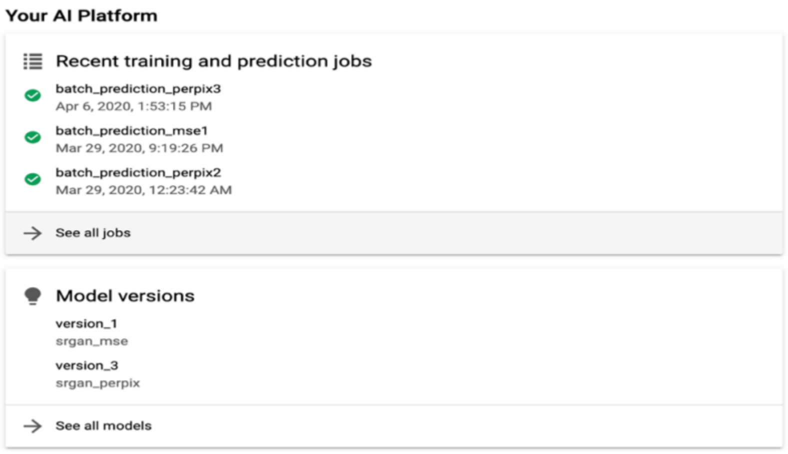
C. Inference on faces and videos:

To access the results of inferences on faces and videos, go to the following link.

Link: super resolution of faces and videos

D. Deployment Details:

From the below figure, it can be observed that both the models were deployed successfully (seen from model versions box in fig 15) and batch predictions (recent jobs box in fig 15) with deployed models are working properly, generating the super-resolution images over the batch of images.



Your AI Platform

Recent training and prediction jobs

✓ batch_prediction_perpix3
Apr 6, 2020, 1:53:15 PM

✓ batch_prediction_mse1
Mar 29, 2020, 9:19:26 PM

✓ batch_prediction_perpix2
Mar 29, 2020, 12:23:42 AM

See all jobs

Model versions

version_1
srgan_mse

version_3
srgan_perpix

See all models

Fig 15: Batch predictions performed on the deployed model

Link to deployed models: Deployed Models on Google Cloud AI platform

E. Project workflow:

During phase 1 of this project, the model was built and trained for a few training steps. In phase 2, GPU training, TensorBoard, other code implementations, and bug fixes were done. Later, the model was trained and deployed in the Google Cloud Platform. The following figure demonstrates the workflow of this project involving various tools. For Google Colab training, the data is taken from Google Drive for training. On the other hand, for Google cloud AI Deep learning VM, data was fetched from Google cloud storage. All the models were implemented using TensorFlow (python) and are compatible to run on different platforms. The trained models were then deployed on to the Google Cloud AI platform. Those deployed models have been used for performing inferences and can be used for future predictions of super-resolution on images or videos.

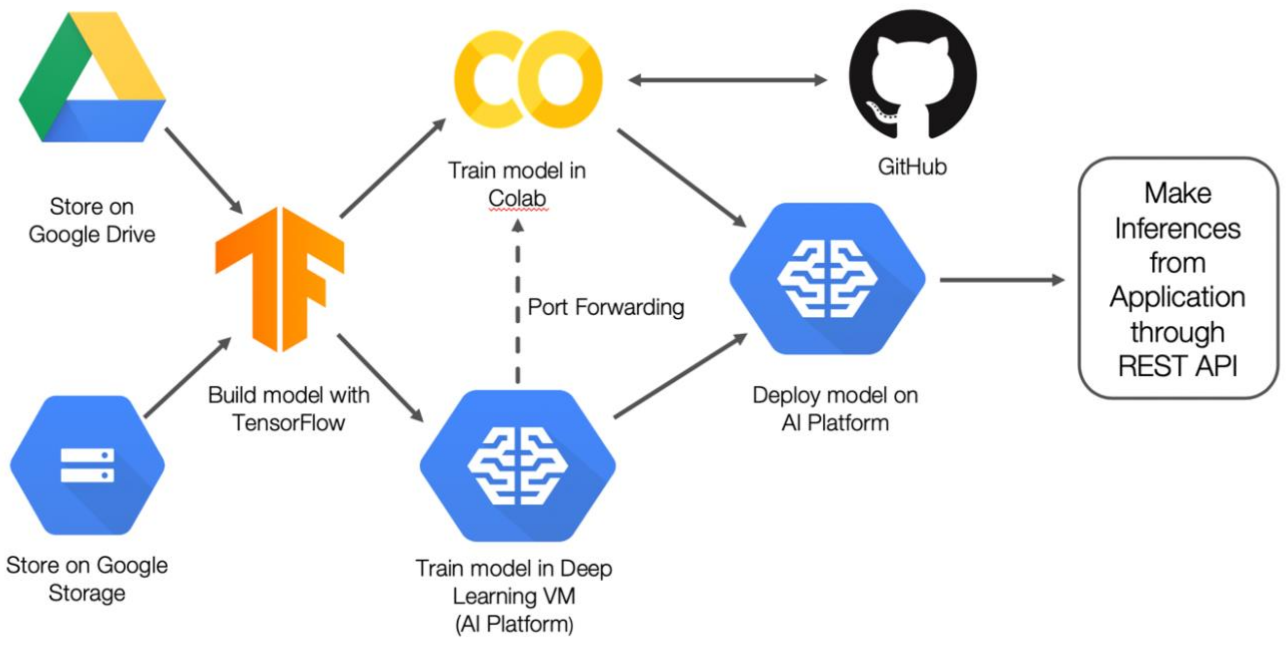

Store on Google Drive
Train model in Colab
GitHub
Build model with TensorFlow
Port Forwarding
Deploy model on AI Platform
Make Inferences from Application through REST API
Store on Google Storage
Train model in Deep Learning VM (AI Platform)

Fig 16: Project workflow