Super Colorization:

A Single Pipeline for Image Colorization and Super Resolution

Mohamed M. Shehawy, 900172170
Omer Mousa, 900171920
Omar Shaheen, 900171868

***Abstract*—This paper aims at abridging the gap in the literature with respect to greyscale image processing resolution. The current state of the art aims at colorizing the greyscale images without consideration of the resolution of the resulting image. Thus, this paper proposes a pipeline of a colorization deep convolutional network that followed by a super resolution network. The colorization part utilizes the Colorful Colorization Architecture with modification to the last layers to simplify the model for computational purposes. For the super resolution, the EDSR architecture loss function was replaced with another more fitting one another network was added achieve the most visually satisfying result while maintain minimal noise.**

Keywords— Colorization, LAB Scale (SR)Super Resolution, Deep Neural Networks

I. INTRODUCTION

During the recent years, the technologies developed by scientist and engineers to do the tasks of human beings are producing better result that rivals the experts. two of these tasks are image colorization and super-resolution (SR). image colorization and super resolution are part of the image processing field that can be used in various applications such coloring old black and white movies and increasing the resolution of their frames. It could also be used in decreasing the size of images on the seder side by removing its colors and downscaling it then reconstructed again on the receiver side. It could also be utilized in improving live stream lossy videos and in surveillance cameras. Moreover, some of the SR models alone can repair images with corrupted or deleted parts. Image colorization and SR can additionally be used as a pre-stage layer in other image processing techniques. Thus, image colorization and SR are crucial to the world. The aim of this paper is to develop a pipeline model that can colorize and enhance the quality of images. Given a grayscale image X of size S_1 = N x M, this model will produce an image F(X) where F(X) is colored image with a size S_2 = s*N x s*M x 3 where "s" is the upscaling factor.

II. PREVIOUS WORK

Many deep learning researchers are working on both coloring images and increasing their resolution. For image SR, one of famous models which won the NTIRE 2017 super-resolution challenge is Enhanced Deep Residual Networks for Single Image Super-Resolution (EDSR) and the model derived from it Wide Activation for Efficient and Accurate Image Super-Resolution (WDSR); however, the most recent SR model is the super-resolution GAN.

The EDSR model follows a high-level architecture and it also uses the residual design [4]. The high-level architecture is the use of a mapping function in the LR space followed by one or more up sampling layers at the end of the model which can also be referred to as post-up sampling SR as shown in fig.1. Here, the up-sampling layer can be learned and trained together with the rest of the model.

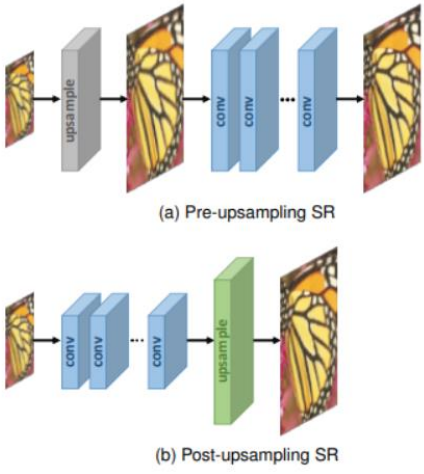


Figure 1 Pre-upsampling and Post-upsampling

Earlier approaches before EDSR used pre-up sampling then learned the mapping which resulted in more needed parameters per layer; therefore, high computational power. This limits the number of deep layers. The up-sampling layer used in EDSR is a sub-pixel convolution layer. If the input is of size H x W x C and the up-sampling factor is s, the sub-pixel convolution layer creates an activation of size H x W x s^2C. the up-sampling layer then reshapes it to s H x s W x C resulting in an output spatially scaled by a factor s as shown in fig. 2.

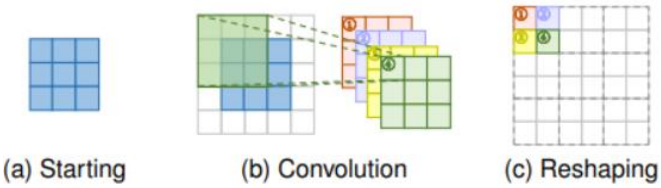


Figure 2 sub-pixel convolution layer

The construction of super-resolution images requires maintain the information of the LR in the HR images; therefore, the EDSR is using a similar design to residuals blocks to convey the identity information via a skip connection while the reconstruction of the image is done the path of the convolution.

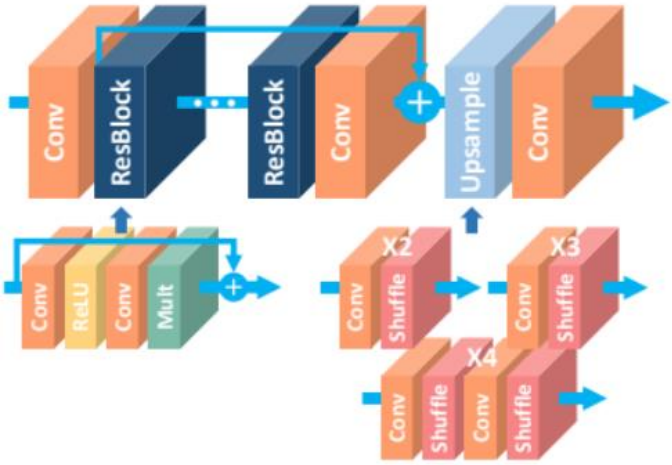


Figure 3 skip connection

The design of residual blocks in EDSR is not identical to the ResNet as batch normalization layers were removed along with the final ReLU activation. The authors of EDSR argue that the batch normalization reduces the range of the activation and loses scale information of the image. In fact, the removal of batch normalization does not only increase the performance of SR, but it also reduces the GPU memory up to 40% allowing it to fit larger models.

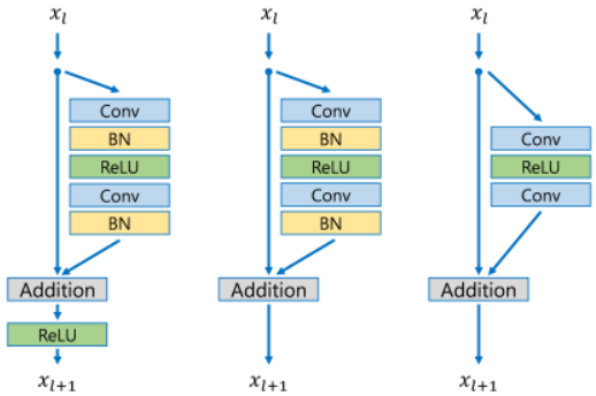


Figure 4 EDSR modifications on ResNet

The WDSR which was derived from the EDSR model makes more changes to the residual blocks [5]. It reduces the number of channels on the identity mapping path and increase it in each residual block, but they did not increase the total number of parameters in the model. The authors of WDSR asserted that increasing the number of channels before the ReLU allows the path of more information leading to a better performance. They also argued that adding a weight normalization parameter makes the learning process easer since it allows the use of higher order learning rates than the ones used in EDSR.

DATA:

Both EDSR and WDSR used the data set DIV2K which consists of pairs of LR and HR images with a variety of contents. LR images are available for many downgrading functions, and every downgrading function set has 800 HR training images and 100 validation images.

PIXEL LOSS:

The pixel wise L^1& L^2losses are frequently used in training SR models. L^1calculates the pixel-wise mean absolute error and L^2 measures the pixel wise mean square error between SR image $I^{SR}$and the ground truth high resolution image I^{HR}. The H, W and C are the height, width and number of channels.

$$\mathcal{L}_{pixel,L^2}(I^{HR}, I^{SR}) = \frac{1}{HWC}\|I^{HR} - I^{SR}\|_2^2$$

$$\mathcal{L}_{pixel,L^1}(I^{HR}, I^{SR}) = \frac{1}{HWC}\|I^{HR} - I^{SR}\|_1$$

L^2 optimizes the peak signal to noise ratio (PSNR) which is a well-known metric used to evaluate models in SR competition. On the other

hand, L^1 achieves better results, thus it was used in EDSR and WDSR.

SRGANs:

One of the major problems of pixel loss is the poor perceptual quality of the generated image. They usually lack good texture and are perceived as blurry. This problem is addressed by Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network (SRGAN) [6]. The authors are using what is called content loss function to compare the features of SR to HR images.

$$\mathcal{L}_{content}(I^{HR}, I^{SR}; \phi, l) = \frac{1}{H_l W_l C_l} \|\phi_l(I^{HR}) - \phi_l(I^{SR})\|_2^2$$

ϕl(I) is the feature map at layer l, H, W & C are the height, width and number of channels of the same feature map. They train the SR model as generator by optimizing the discriminator of generative adversarial network (GAN) to discriminate SR images from HR. The generator loss is shown below.

$$\mathcal{L}_{generator}(I^{LR}; G, D) = -\log D(G(I^{LR}))$$

they combined the generator loss with the content loss to get their optimization target.

$$\mathcal{L}_{perceptual} = \mathcal{L}_{content} + 10^{-3}\mathcal{L}_{generator}$$

The second part of this project is concerned with coloring grayscale images. Colorfull image colorization: One of the most remarkable milestones towards coloring images using deep learning is colorful image colorization [7]. This paper uses CNN to analyze RGB images relative to its grayscale versions. The authors used single-stream, VGG-styled network with added depth and dilated convolutions. They also used classification loss with rebalanced rare class.

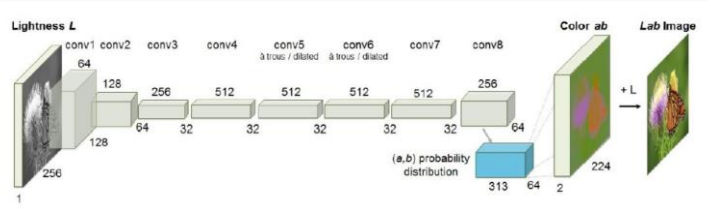


Figure 4 colorful image colorization architecture

Since the authors wanted to measure the difference between the predicted color and the ground truth color, they couldn't use the normal Euclidean distance. If color degree can take different combination of values, taking the average of every cell would be an optimal addition to the Euclidean distance; however, this addition favors grayish, desaturated colors. Thus, they used multinomial cross entropy loss to solve this problem.

$$\mathrm{L}_{cl}(\hat{\mathbf{Z}}, \mathbf{Z}) = -\sum_{h,w} v(\mathbf{Z}_{h,w}) \sum_{q} \mathbf{Z}_{h,w,q} \log(\hat{\mathbf{Z}}_{h,w,q})$$

LEARNING REPRESENTATIONS FOR AUTOMATIC COLORIZATION:

Another paper that influenced the field of image coloring is Learning Representations for Automatic Colorization [8]. The authors used the semantic composition of the image (what is the objects) as well as a localization (where are these objects) to color any arbitrary image. Convolutional neural network is a fit tool to incorporate the semantics and the localization into the coloring system. The second part of the project is a system that predicts a histogram instead of a single color for every location.

The loss function makes use of the histogram of the ground truth image Y, and $D_{KL}$is the KL-divergence.

$$L_{\text{hist}}(\mathbf{x}, \mathbf{y}) = D_{\text{KL}}(\mathbf{y} \| f(\mathbf{x}))$$

The base architecture in this project is similar to a fully convolutional version of VGG-16. They concatenate the features at each spatial location in all layers to create a hyper column description. They feed the output to a fully connected layer.

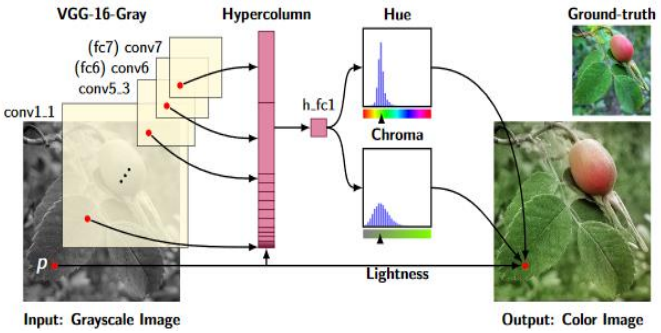


Figure 5 learning representation architecture

III. SUPER RESOLUTION

We aim to do 2x upscaling for the image as well as enhancing the resolution.

1. *Data Preperation*

We used mainly as subset of Microsoft MS-COCO images dataset to train on. The training set of the super resolution network was about 16k images. Due to their size not fitting the GPU memory, we built a TensorFlow data pipeline to read images on the run from the secondary storage (hard drive). The idea is basically to make the feeding of the data part of the computation graph to load it on the go and throw it at the end so that it does not occupy the RAM. The data pipeline consists of three stages:

- Get a TensorFlow generator flowing data from directory as .jpg images and it fetches them in parallel to make it faster. Pre-process the input and output images.
- Out pre-processing was only to resize the input images (x) to 64*64*3 RGB images and the output (y) images to 128*128*3 RGB. This is done because our purpose was to increase the resolution by 2x.
- Construct a generator that yields a batch of variable size containing (x,y) where x are the input batch and y are the target batch. This tuple yielded was then passed for training to the TensorFlow.keras fit generator function.

The same can be done for validation data as well. This is one of the fastest ways to get the data from disk.

2. *EDSR Modificed Archticture and Training*

The first thing tried to tackle the problem is the proposed architecture of the EDSR Network. It has 16 Residual blocks with the components shown below. Each residual block has two conv layers of 64 filters and no batch normalization layers and no pooling layer in the whole network. Of course, there is a skip connection in the res blocks, but there is also a skip connection globally over res blocks. Then there is an upsample layer followed by a Conv layer and that is it. We introduced some pre-processing before feeding the image to the network; we normalize the Image with the mean and the standard deviation of Imagnet then denormalize it before calculating the loss. We increased the number of res blocks (tried 24 and 32 and found 24 res blocks to be best.)

The implementation of the upsample layer was done by using a Conv layer with 256 filters then using tensorflow.nn.depth_to_space to make it double the input size with 64 filters.

We increased the number of Conv layers after the upsampling layer to increase the complexity and the level of details in the output image. The architecture is shown in figure 6.

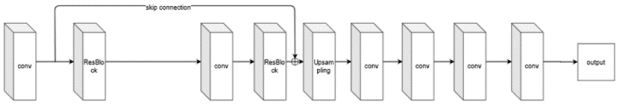


Figure 6 EDSR Modified Architecture

a. *EDSR Loss Function*

Originally, the paper suggests that we use Mean Absolute Error loss. We proposed a new loss function that is the sum of the MAE loss and the MSE content loss coming from passing the images to a VGG19 network. We use the 5th layer of the VGG19 network to get the content loss. The Loss equation is shown below where y is the ground truth image and y' is the predicted image.

$$L = |y - y'| + (vgg(y) - vgg(y'))^2$$

b. *EDSR Training*

This proposed variation of the EDSR network has about 3M parameters. We train the model using tensorflow.keras *fit_generator* function that takes the tensorflow data generator. We trained it for 300 epochs. The loss dropped from 400K to 10 in this period. We use Adam optimizer with an exponential decaying learning rate with initial value of 1e-4 and it is halved every 200000 steps. The runtime of the 300 epochs is about 6-8 hours on Google Colab Tesla V100 16GB GPU. We

failed to report the continuous loss curve because of the continuous runtime disconnection of Google Colab.

c. EDSR Performance

The network performs well on relatively simple images and poorly on images with human faces or lots of variations in them. Two examples from chosen images are shown in figure 7. In the first one, the network performs well. It removes the distorted pixels in the lighting part, leading to a clearer image that is twice in size. In the second image, it has a human face in it, we note that the network slightly enhances the quality of the picture, but we still can see the distortion in the image. In this case, it does almost as poorly as the classic interpolation.

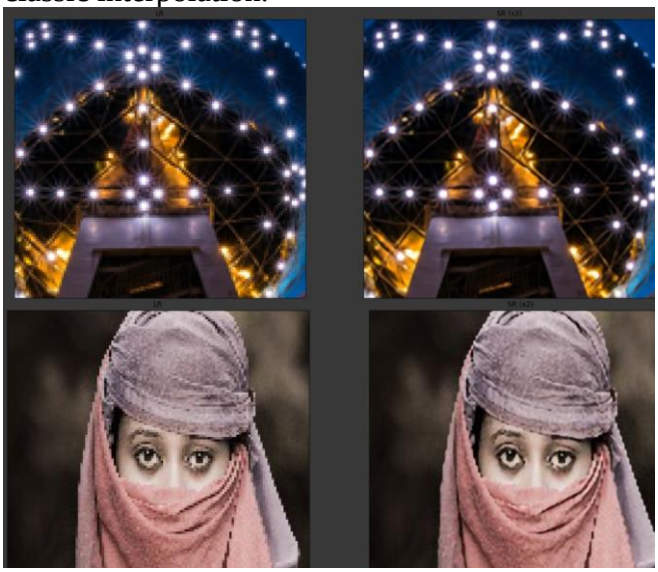


Figure 7 EDSR Modified Sample Output

3. Proposed Architecture (Modificed EDSR + QED) Archticture and Training

We note that the problem with our version of EDSR network even with a better loss function is its inability to add much detail after upscaling the image. This was evident on the loss saturating while giving similar results to the image above even on the training set. In other words, given the data, the model cannot do better. We believe that the problem was not the lack of data, but it was that the upsampling layer was at the end of the network, making it hard for the network to add fine details after upscaling the image. Therefore, we propose a modified architecture based on the idea of the division of labour. We add a new network similar to the EDSR network but without any upsampling layer or normalization. This network sole job will be to take the upscaled image and try to enhance its quality by filling in details and making it clearer. This second network takes the upscaled 2x images (128*128*3) in this case. It will not do any upscaling to the image; it will just learn what the EDSR network misses of details so that it makes the output image clearer with minimal distortion. We call this second network DQE (Deep Quality Enhancement). The architecture of DQE and the proposed SR model is shown in figure 8.

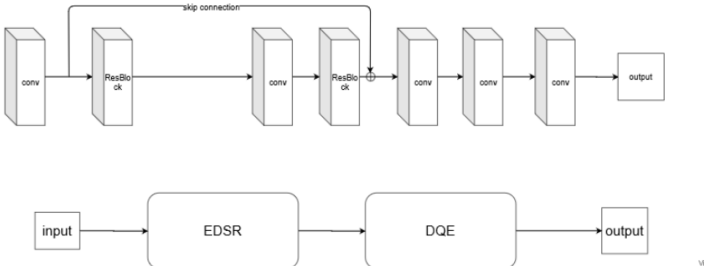


Figure 8 DQE Architecture + SR model flow.

a. DQE Loss Function

The loss function of the EDSR network is the same as above (content loss + mean absolute error). The loss of the DQE network is a simple mean absolute error.

b. Training

The total number or parameters of the DQE is about 2.8 M parameters. The total number of parameters then is 3M for the EDSR+2.8M for the DQE network ~ 6M trainable parameters. We train every one of the two networks independently. First, we train the EDSR first, then we train DQE network. We trained the DQE network for 50 epochs only. The loss dropped from 1000 to 10 in this period. We use Adam optimizer with an exponential decaying learning rate with initial value of 1e-4 and it is halved every 200000 steps. The runtime of the 50 epochs is about 4 hours on Google Colab Tesla V100 16GB GPU. We failed to report the continuous loss curve because of the continuous runtime disconnection of Google Colab.

c. *Performance*

Some of the results for this proposed approach can be shown in figure 9. We can already see a noticeable improvement over using only the EDSR. The image is much clearer and closer to the natural view of images, which is a descent progress in enhancing the quality of images.

Figure 9 New SR Model Sample Output

IV. COLORIZATION

1. *Data Preperation*

To accommodate the available resources, we started by using the flowers dataset in [1]. The state-of-the-art training set is usually Image-Net; however, the size of ImageNet is extremely huge for us to accommodate. Furthermore, we tried a subset of MS-COCO found in [2] which represents common objects in context; however, it resulted in a memory explosion and bad training results on our architecture. Finally, we tried to train our model on MIRFLICKR-25000 [3] dataset which contains 25000 diverse images downloaded from the social photography website Flicker. We tried different preprocessing techniques for our experimentation. All preprocessing included converting the image from the RGB space to the CIELAB color space as done by [7] and [8]. The CIELAB provides a convenient way to separate the color channels from the lighting channel; thus, the lighting channel could be considered as the greyscale image. This L channel passed through a Deep ConvNet to deduce the AB channel. We tried additional preprocessing like normalizing the data around the mean and the standard deviation of the training data; however, no notable effect on the training progress or the final results.

2. *Deep Colorizer Archticture and Training*

To train the colorizer we utilized a deep convolutional neural network. The structure is the same as that in the work done by [7]; however, we removed the last layers that included probability distributions for the colors to simplify the model. Instead, we converted the problem into a regression problem where we added a convolutional layer at the end of the net that has dimensions equal to that of the input image and with two neurons. We used the output of this layer, which is equal to the size of the AB channels of the CIELAB image, to compare it with the ground truth AB channels. Then, the Root Mean Square Error is calculated so that we can find a regression model able to achieve the colorization regression problem. The architecture is shown in figure 10. The model is about 24M trainable parameters.

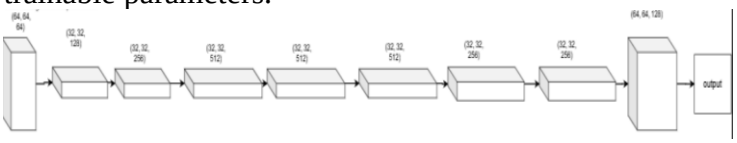


Figure 10 Deep Colorizer Architecture

3. *Performance*

The colorization architecture had problems with saturating to colors that are close to the mean of the images that it was trained on. Thus, the model had quite low performance with types of images different from those that it was trained on.

Figure 11 Colorization Model Sample Output

On data close to the training data, the color favorized certain colors to others again due to the existence of those colors in the background. Sample outputs of the colorization model can be found in figure 11. We can see that it performs well on certain colors and poor on images requiring other pool of colors than the one it saturated on.

V. Colorization and SR Results

Finally, we achieve the requirement by cascading the two models together to colorize an input grey image and then enhancing its quality. Below are some samples. We can see that applying both models brings live to the image as was intended. Samples of this output are shown in figure 12. We start by a low-quality grey image, ending with a colored one that is close to natural clear images.

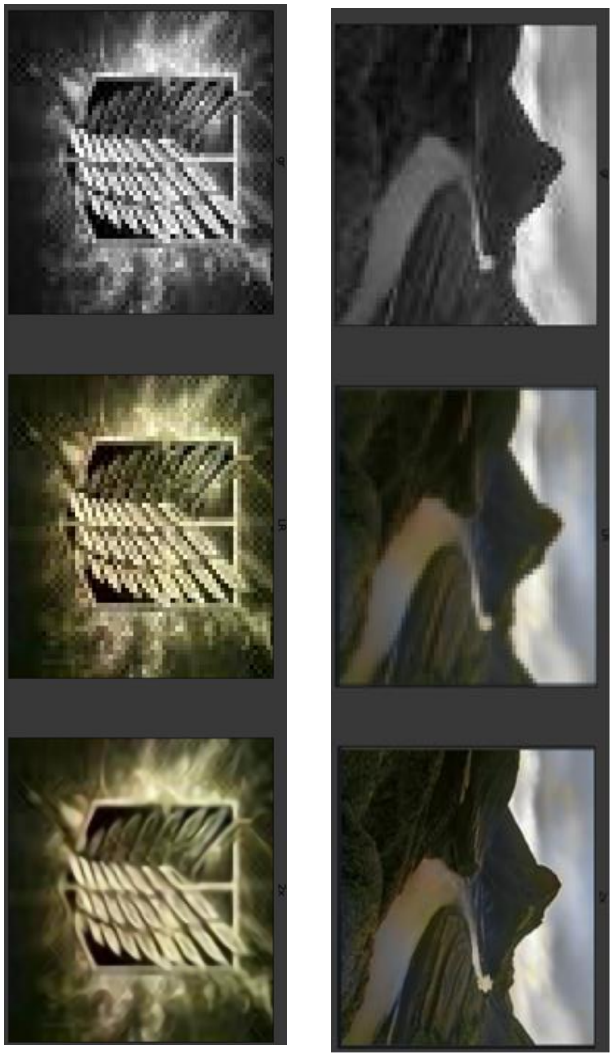


Figure 12 Colorization +SR Sample Output

VI. Conclusion

The intended task was given a low-quality grey scale image, our model will be able to colorize the image and increase its quality (resolution) by 2x so that it looks more natural and cleared. Due to lack of time and lack of resources to train the colorization image on a huge dataset, we put most of our focus on the super resolution task which can be achieved with less resources and less data. It can be seen that the super resolution with our proposed architecture works well and gives results that are close to natural. The colorization network is hungry for data; it gives acceptable results on images it's familiar with its color distribution. Therefore, training it on data as huge as ImageNet would have given much more promising results.

VII. Future Work

This work can be extended in the future by adopting GANs for this purpose and apply the same architecture proposed (ESDR+DQE) above as the generator of the GAN. The existence of the discriminator should further enhance the quality of the output images. We can try also Nvidia Conditional GAN which has two generators on the super resolution task hoping to achieve even better results. Moreover, trying to upscale the images by 4x and 8x instead of 2x would be a significant step. For the colorization, we can use GAN techniques that is suitable for the nature of the colorization problem as colorization is by nature a generative problem. Furthermore, given the needed resources we can train the colorization model on more diverse and huge datasets like ImageNet.

VIII. References

[1] Tf_flowers : TensorFlow Datasets. (n.d.). Retrieved December 10, 2020, from https://www.tensorflow.org/datasets/catalog/tf_flowers

[2] Common Objects in Context. (n.d.). Retrieved December 10, 2020, from https://cocodataset.org/

[3] (n.d.). Retrieved December 10, 2020, from http://press.liacs.nl/mirflickr/

[4] Lim, B., Son, S., Kim, H., Nah, S., & Mu Lee, K. (2017). Enhanced deep residual networks for single image super-resolution. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 136-144).

[5] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., ... & Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4681-4690).

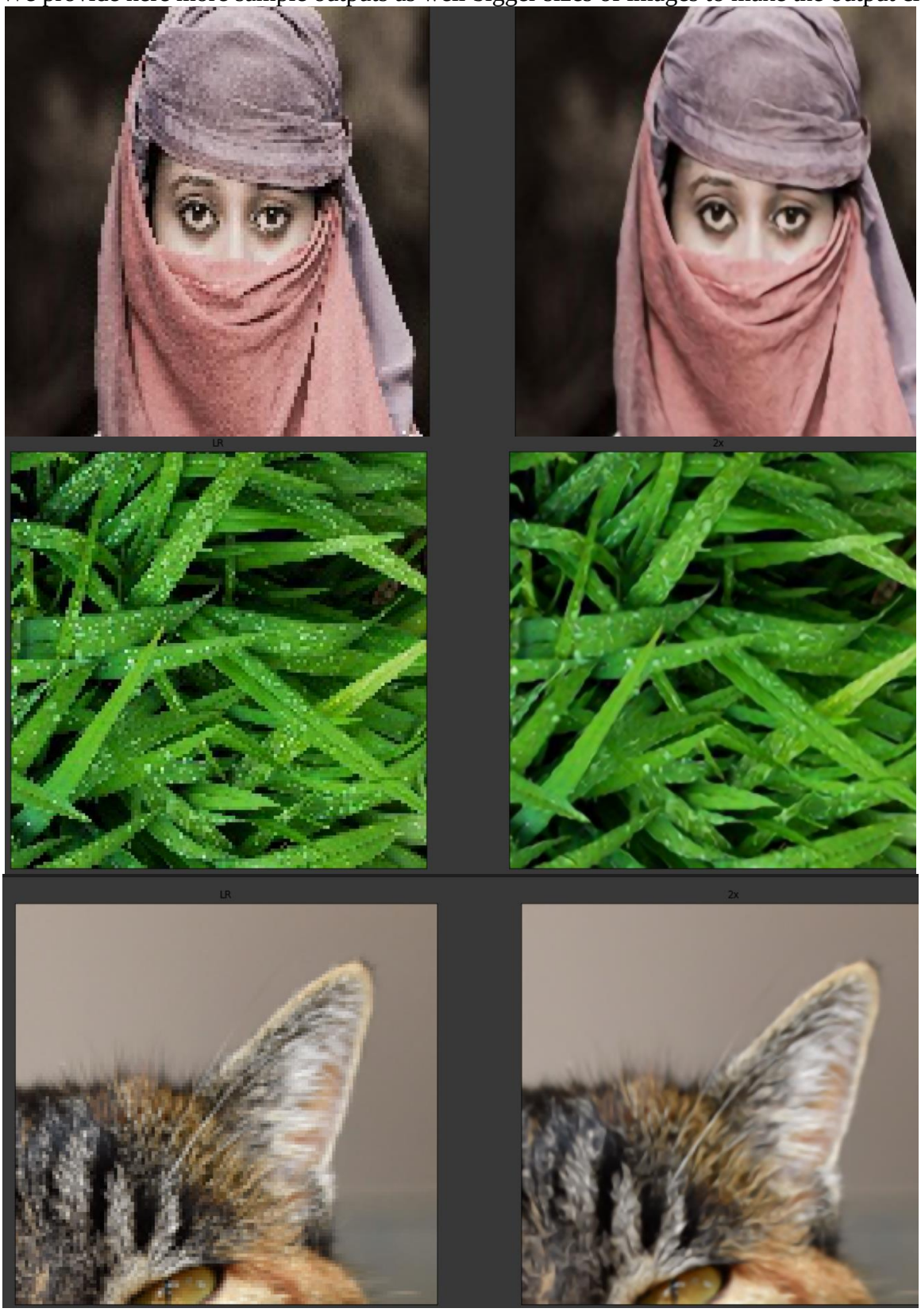
[6] Yu, J., Fan, Y., Yang, J., Xu, N., Wang, Z., Wang, X., & Huang, T. (2018). Wide activation for efficient and accurate image super-resolution. arXiv preprint arXiv:1808.08718.

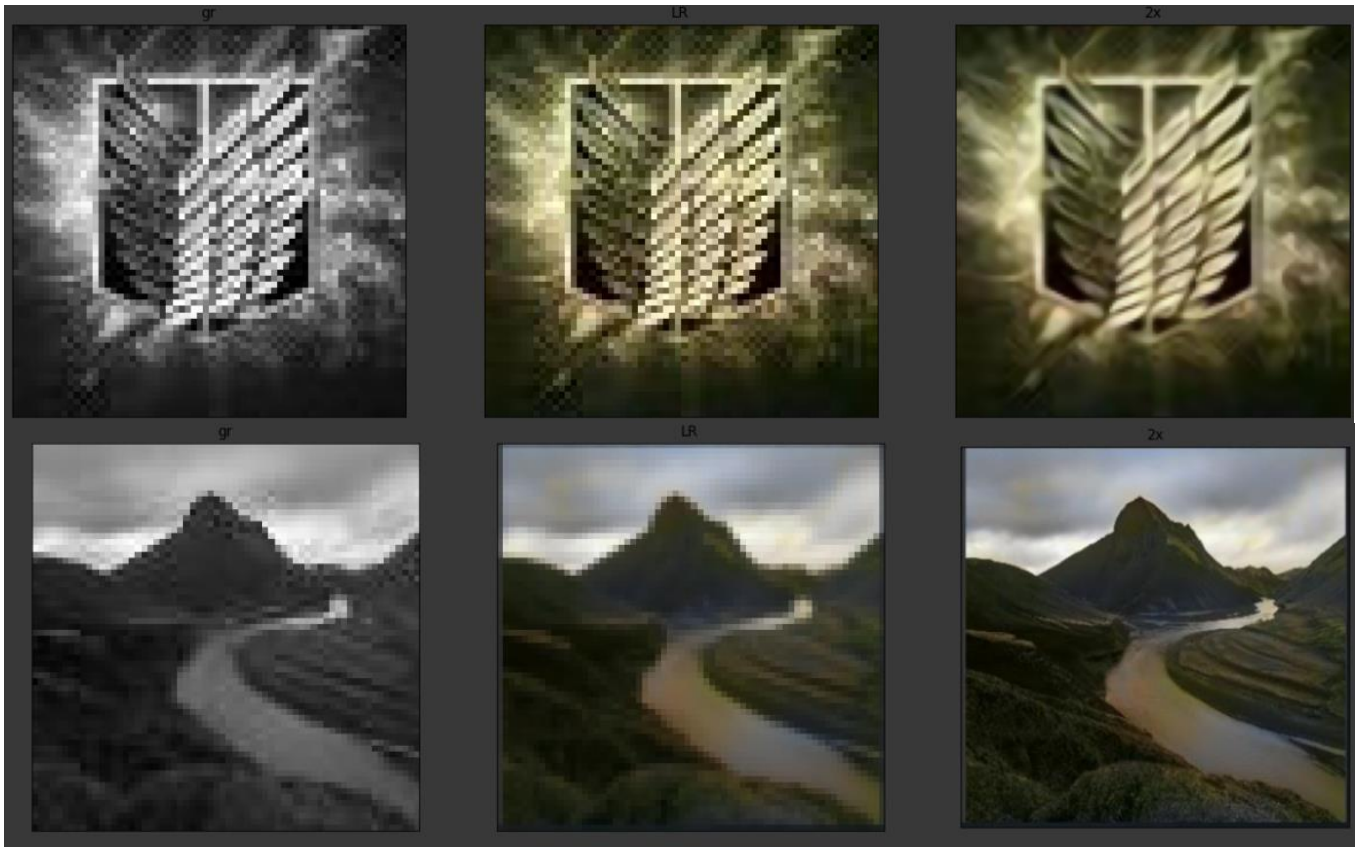
[7] Zhang R., Isola P., Efros A.A. (2016) Colorful Image Colorization. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9907. Springer, Cham. https://doi.org/10.1007/978-3-319-46487-9_40

[8] Larsson G., Maire M., Shakhnarovich G. (2016) Learning Representations for Automatic Colorization. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9908. Springer, Cham. https://doi.org/-10.1007/978-3-319-46493-0_35

IX. APPENDEX

We provide here more sample outputs as well bigger sizes of images to make the output clear.




gr
LR
2x
gr
LR
2x