Group 2: Colorization Using Conditional-GANs

Arjun Singh 21111402 | Debdeep Paul Chaudhuri 21111413 | Himanshu Lal 21111403 | Shivam Tripathi 21111408
{arjuns21, debdeeppc21, himanshul21, shivamtr21}@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

Image colorization is the process of estimating the color channels of an image. With the rapid development of deep learning techniques and the availability of heavy-computing resources, this process can be automated. There are a variety of colorization models that can perform colorization on a diverse set of images [1, 2]. Generative adversarial networks are popular for image synthesis, and image-to-image translation tasks [3]. Building on the existing state-of-the-art 'pix2pix' [4], we have performed several experiments to improve the colorization quality and built a colorization-specific framework. Inspired by the super-resolution literature [5], we have made use of **perceptual-loss** to quantify the quality of colorization and optimized it via the general adversarial process. Apart from the perceptual loss, we have modified the generator architecture of 'pix2pix' with **ResidualUNet**, which is a deeper version of UNet [6] where each layer is a deep residual block in itself. In this report, we demonstrate the results of our various experiments on colorization and compare the results of our evaluation with the state-of-the-art model. The models are implemented in Python (using PyTorch), and the code is available at https://github.com/shivamt-tr/Image-Colorization-using-conditional-Generative-Adversarial-Networks

1 Introduction

Colorization of black-and-white images is an active research area in computer vision [7]. It has useful application in the restoration of historical black-and-white photographs, motion-pictures, astronomical images, and scientific and medical images.

With the advent of deep learning and the availability of computing resources, various deep architectures have been utilized for automating colorization tasks. With this project, we intend to explore such models and improve upon the quality of colorization.

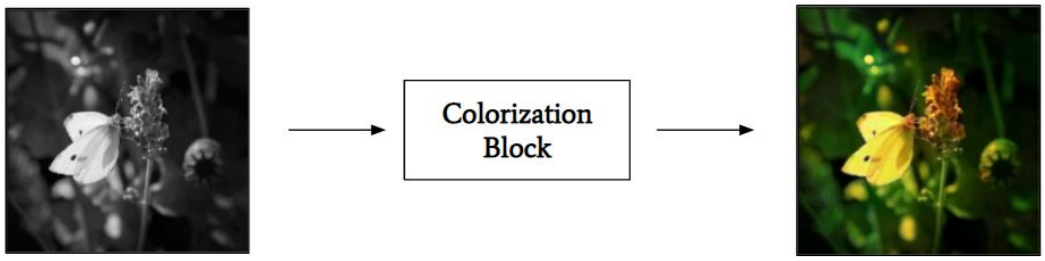


Figure 1: Colorization of black-and-white images

In this report, we will discuss the related works in this area, followed by our proposed idea. We will explain the deep ResidualUNet architecture and VGG16 [8] based content-loss [5]. We will perform ablation studies on the loss functions and generator architectures and present the result of our experiments on various evaluation metrics. The comparison table with state-of-the-art [9, 10] is present at the end of the methodology.

In the later sections, we discuss the future work and conclusion and mention the team members' contributions.

Note: The last page of report contains the qualitative results of all the experiments.

2 Related Work

Colorizing black and white images is a challenging problem that researchers have been trying to solve for quite a while. Traditionally user-guided methods were used to colorize the images [11, 12, 13, 14, 15]. These models could perform diverse colorization depending on the guided input and subject to user preference. Various deep learning models were developed for colorization [16, 17, 18] that fully-automated the colorization.

Training deep models requires a diverse dataset with roughly balanced classes. For the colorization problem, class-balancing is a challenging task. Various colorization models have difficulty in producing natural colors in the generated output. Exemplar-based colorization methods like [19, 20, 21] require an example image to guide the colorization process. This can be used to produce diverse colorization.

The 'pix2pix' paper [4] performs image-to-image translation by building a general framework that learns the mapping via an adversarial process using the conditional-generative adversarial networks (c-GANs). The generator tries to produce a colorized version of the input image, and the discriminator tries to identify it as real/fake. Our model builds upon a similar idea and uses conditional-GANs as a general framework for colorization.

We employ a deeper UNet [6] architecture for the generator, ResidualUNet, which follows a similar structure as UNet, but each layer is a residual block. This deeper network with skip connections can learn subtle features that result in good quality colorization. We explain the ResidualUNet architecture in detail in the later section.

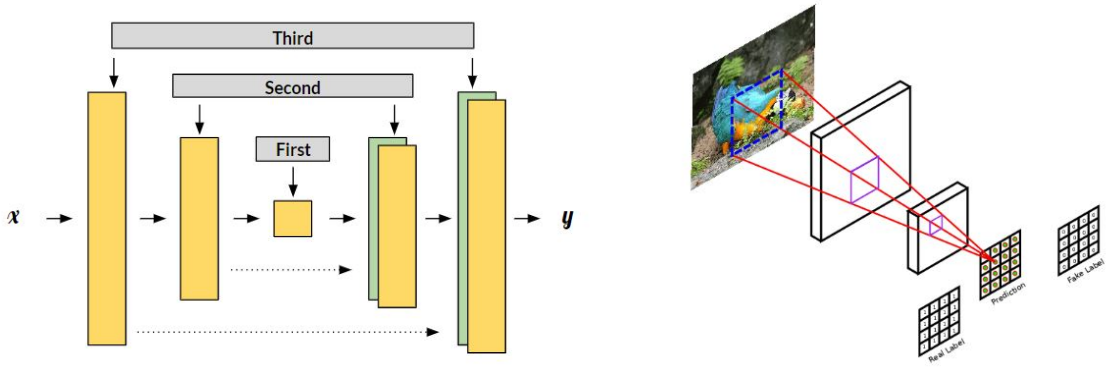
Apart from the generator architecture, we use the content-loss inspired from super-resolution literature [5] as a regularizer term in the adversarial objective. The content-loss and adversarial loss (perceptual loss is the weighted sum of both the losses) allow the model to capture high-frequency regions in the image. It uses the feature maps of a pre-trained VGG16 network [8], a popular architecture employed in various deep learning tasks. We perform ablation studies on different generator architectures and the effect of regularizers on image colorization. We report the results of each experiment in detail in the later sections.

3 Proposed Idea

3.1 Image-to-Image Translation with Conditional Adversarial Networks (Pix2Pix) [4]

Generative-adversarial networks (or GANs) are generative models that train a generator and discriminator model in an adversarial manner [22]. GANs can learn a mapping from random noise vector z to output image y, $G : z \rightarrow y$. Image colorization requires the generator to produce output related to the input image but with colors. We feed the gray-scale image (or lightness, L channel in our case). A conditional-GAN [23] learns the mapping from random noise vector z and a condition x (such as the lightness channel of the image) to the output image y, i.e., $G : \{x, z\} \rightarrow y$.

Our idea has been built on top of 'pix2pix' [4, 24], which trains a conditional-GAN for learning a generalized image-to-image translation mapping. The authors of 'pix2pix' use a popular encoder-decoder architecture, 'UNet' [6] for the generator and 'PatchGAN' for the discriminator. The PatchGAN discriminator divides the generated image into small-sized patches and classifies each patch as real/fake. This allows us to give arbitrary-sized images for the discriminator to classify.



(a) Generator (UNet) (b) Discriminator (PatchGAN)

Figure 2: Generator and discriminator architecture used in pix2pix

The objective function combines the cGAN adversarial loss along with L1 loss.

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[logD(x, y)] + \mathbb{E}_{x,z}[log(1 - D(x, G(x, z)))] \tag{1}$$

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[||y - G(x, z)||_1] \tag{2}$$

The L1 term in the loss function acts as a regularizer that forces the generator to preserve the structure of the original image and prevent it from assigning arbitrary colors to pixels just to 'fool' the discriminator. Note that L2 loss can also be used, but it tends to produce blurry results.

The final objective becomes

$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G) \tag{3}$$

The 'pix2pix' model is a generic framework for various image-to-image translation tasks. We propose multiple solutions to improve the existing model and produce better image colorization for the project.

3.2 ResidualUNet

The ResidualUNet architecture is built using the UNet architecture [6], where each layer is a deep residual block. With the deeper network, the colorization framework can learn features that capture texture details better.

The architecture is made with six residual blocks in both encoder and decoder parts with the number of filters in each block following the standard UNet architecture [6]. The residual block used in this network consists of a 3x3 convolution layer with 'same' padding followed by a 4x4 convolution layer with stride two that downsamples the image by a factor of two. The output of these layers is connected to the input via a skip-connection that includes a 4x4 convolution layer with stride two. Each convolution layer in the residual block is followed by a batch-normalization layer that solves the Helvetica scenario [3] (the generator learns that a single output is able to consistently trick the discriminator). We use dropout (with keep=0.5) in the three upsampling layers that follow the first upsampling block of the decoder. In the encoder layers, we use LeakyReLU activation with the negative slope of 0.2 and the standard ReLU activation for the decoder layers.

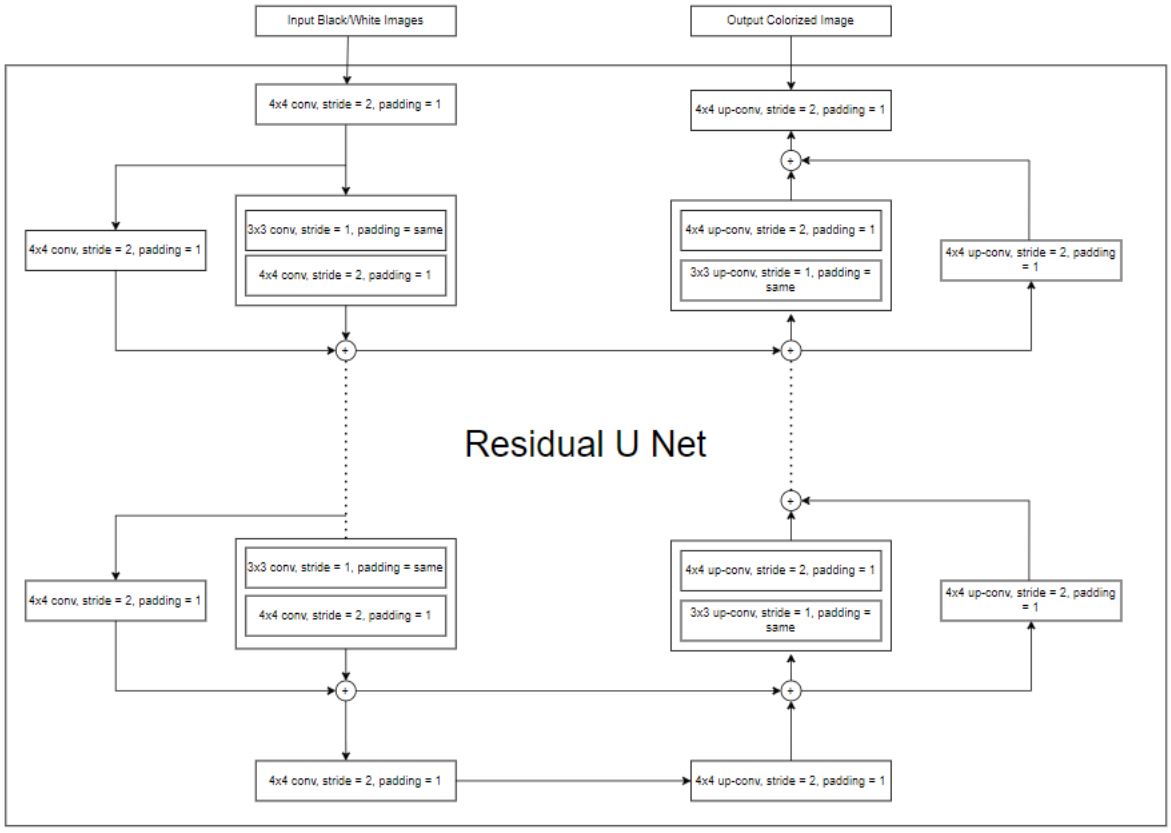


Figure 3: ResidualUNet Architecture

3.3 Content Loss

Many colorization frameworks are evaluated on the basis of the percentage of human participants that they are able to fool. This measure is usually hard to quantify. With the advent of various deep learning architectures that have been trained on large-scale datasets [25, 26], we can use the feature maps produced by such an architecture to evaluate the quality of image colorization.

Inspired by the super-resolution literature [5] we design a loss function for improving the quality of image colorization. This loss function will make use of feature maps produced by the pre-trained VGG16 network [8]. A VGG16 network's feature map represents a high-level semantic understanding of the input image.

This loss function is called 'content-loss,' It calculates the mean square difference between the VGG16 feature maps of the generated image and the ground-truth image. It can be defined as

$$\mathcal{L}_{Content} = \frac{1}{C_{i,j}H_{i,j}W_{i,j}} \sum_{x=1}^{C_{i,j}} \sum_{y=1}^{H_{i,j}} \sum_{z=1}^{W_{i,j}} (\phi_{i,j}(I^{real})_{x,y,z} - \phi_{i,j}(I^{fake})_{x,y,z})^2 \tag{4}$$

where, $\phi_{i,j}$ represent the feature map obtained by the j^{th} convolution (after activation) before i^{th} max-pooling layer within VGG-16 network. C, H, and W represent the feature map's channels, height, and width, respectively. I^{real} represents the ground-truth image, and I^{fake} represents the image generated by the generator.

Note: Based on the terminology of SRGAN paper [5], perceptual loss is the weighted sum of content loss $l_{content}$ and adversarial loss l_{gen}.

$$l_{perc} = l_{content} + 10^{-3} l_{gen}$$

4 Methodology

4.1 Dataset

MS-COCO 2017 [25] is a popular large-scale object detection, segmentation, key-point detection, and captioning dataset. Due to resource constraints, we have experimented with 10k randomly sampled images from the train set of MSCOCO-2017. The train and test data are split into 8k and 2k images.

We have also evaluated our models on 5k images randomly sampled from the validation set of ImageNet-2012 [26].

The MSCOCO dataset contains diverse scenes that might help in building a robust colorization model. However, to make the comparison fair, we report the accuracy on ImageNet, which has been used in many colorization papers.

4.2 L*a*b Colorspace

We use the L*a*b colorspace instead of RGB for training the models. The LAB colorspace contains the lightness channel (L) and two-color channels *a (green-red) and *b (yellow-blue). The lightness or L channel represents the gray-scale image. This is fed as conditioned input to the generator, which produces *a*b channels as the output.

This way, the generator has to predict only two channels instead of three (red, blue, and green).

4.3 Evaluation Metrics

To evaluate the result of our experiments we use the following evaluation metrics:

- **Mean Absolute Error (MAE):** Mean of absolute difference between corresponding pixel values.

$$MAE = \frac{1}{N * C * H * W} \sum_{n_i}^{N} \sum_{c_j}^{C} \sum_{h_k}^{H} \sum_{w_l}^{W} |y_{i,j,k,l} - \hat{y}_{i,j,k,l}| \tag{5}$$

- **Epsilon Accuracy ($\mathcal{E}$):** percent of the correct pixels (within certain threshold, $\mathcal{E}$)

$$Accuracy\ \mathcal{E} = \frac{1}{N * H * W} \sum_{n_i}^{N} \sum_{h_k}^{H} \sum_{w_l}^{W} \prod_{c_j}^{C} 1_{[0,\mathcal{E}_l]} \left(|y^{(i,j,k,l)} - \hat{y}^{(i,j,k,l)}| \right) \tag{6}$$

- **Peak Signal to Noise Ratio (PSNR):** Log of the ratio of maximum possible value to the root mean square error. A higher value means lesser noise, i.e., a better quality of colorization.

$$PSNR = 10log_{10} \left(\frac{(L-1)^2}{MSE} \right) = 20log_{10} \left(\frac{L-1}{RMSE} \right) \tag{7}$$

- **Frames per Second (FPS):** Number of images that can be processed in a second.
- **Frechet Inception Distance (FID):** FID score is a method for measuring the quality of generated samples [27, 28]. It uses the output of the inception v3 [29] network up to the last pooling layer to generate image-specific activation vectors for both real and synthetic images and returns the difference in distribution between them. Consequently, a lower FID score translates to a better quality image generation.

$$FID = ||\mu - \mu_w||_2^2 + Tr(\Sigma + \Sigma_w - 2(\Sigma\Sigma_w)^{1/2}). \tag{8}$$

4.4 Experimental Setup

All the experiments are performed on 8k randomly sampled images from the training set of MSCOCO-2017. For training we use Adam optimizer with learning rate $\eta = 0.0002$, $\beta_1 = 0.5$, and $\beta_2 = 0.999$. For pre-training the generator we use Adam optimizer with learning rate $\eta = 0.0001$, and default values of β.

We train our models for 50 epochs on the 8k images, taking a batch size of 16. Due to resource constraints, we were unable to train for longer but **we ensure that the model has sufficiently converged**. The experiments performed follow the general architecture as explained in section 4.5

4.5 General Architecture for Experiments

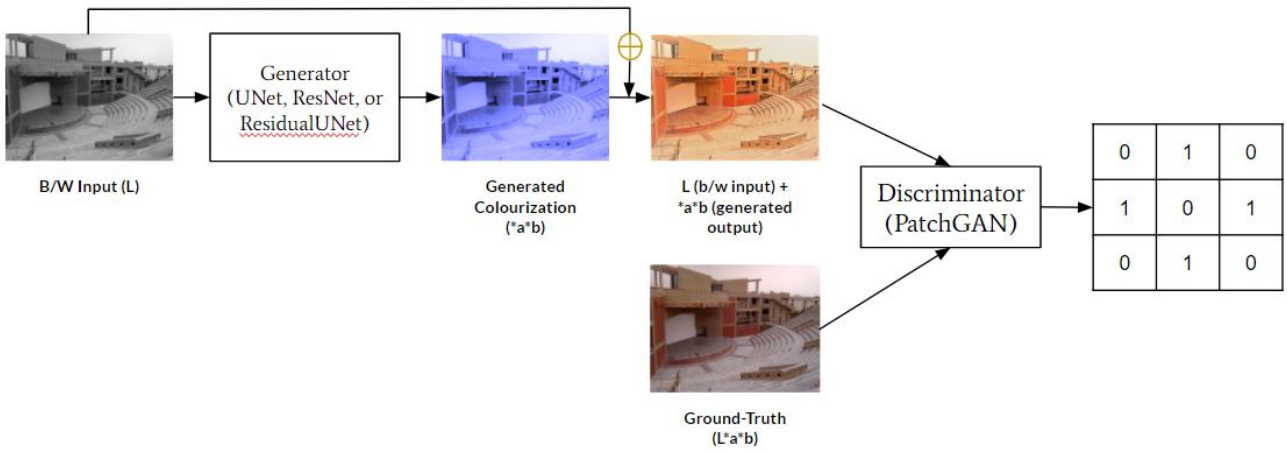


Figure 4: General architecture for experiments, inspired from pix2pix [4]

The model architecture is based upon the 'pix2pix' [4]. We feed the lightness channel (gray-scale input) to the generator, which generates two-channel output (*a and *b color channels). The lightness channel (L) is concatenated with color channels (*a*b) to make a three-channel colored image (L*a*b). We pass the generated output and ground truth to the discriminator, which classifies them as real/fake. Since we are using PatchGAN, the discriminator outputs a matrix of zeros and ones, where zero denotes fake/synthetic/generated patch and one denotes real patch.

The discriminator loss is evaluated using binary cross-entropy loss. Since the output of the discriminator is a matrix of zeros and ones (PatchGAN), the training labels are as follows: a matrix of one's is used to train the model with real images, and a matrix of zero's is used to train the model with fake images.

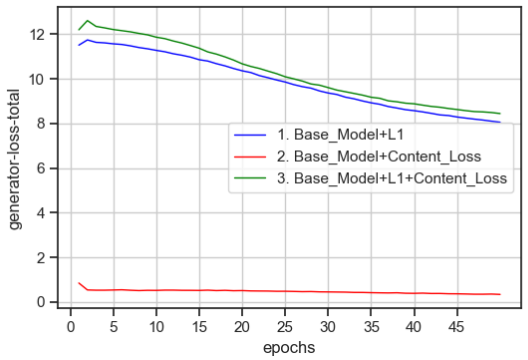
4.6 Ablation Study

Effect of regularization on colorization: To understand which loss function performs better colorization, we analyzed the results of experiments on different combinations of loss functions. As stated earlier, the L1 loss initially used by the authors forces the generator to produce good quality colors while preserving the original image structure. It prevents the generator from assigning arbitrary colors to pixels just to fool the discriminator. The authors of [4] report that L2 loss tends to produce blurry results, so we do not use that for our experiments.

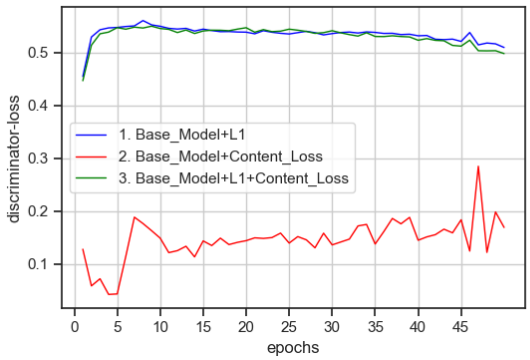
The content-loss $\mathcal{L}_{content}$ is a useful regularizer that intends to minimize the mismatch between VGG16 [8] feature maps of generated and ground-truth images. It also helps in capturing the high-frequency regions in the image.

*Model**	**FID** ↓	$\mathcal{E}$ **-Acc(5%)** ↑	**MAE** ↓	**PSNR** ↑
1. U-Net+L1	32.29	31.97	15.14	21.72
2. U-Net+Content Loss	35.66	31.13	15.74	21.36
3. U-Net+L1+Content Loss	**29.50**	**35.39**	**14.43**	**22.15**

Table 1: Experiments with different regularizer combinations show that L1 loss along with content-loss performs better than the L1 loss alone. (*evaluated on MS-COCO dataset), ↓ : lower the better, ↑: higher the better



(a) Generator+Regularizer Loss



(b) Discriminator Loss

(c) Qualitative results

Figure 5: Loss curves and qualitative comparison between models with different regularizers: The discriminator loss curve of model trained with content-loss flickers a lot and does not converge properly. The model trained with both regularizers performs slighly better colorization compared to our baseline.

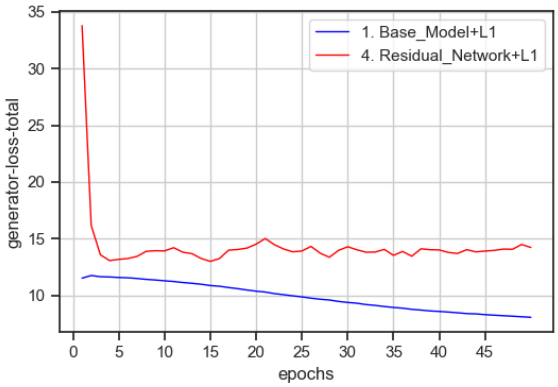
The experiments with content-loss alone along with adversarial failed to produce good results. We blame this on the absence of L1 loss that helps capture the image's low-frequency regions. Therefore, we use content-loss along with L1 loss, weighted by λ_1 and λ_2 to capture high-frequency regions as well as low-frequency regions in the image.

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda_1 \mathcal{L}_{L1}(G) + \lambda_2 \mathcal{L}_{Content} \tag{9}$$

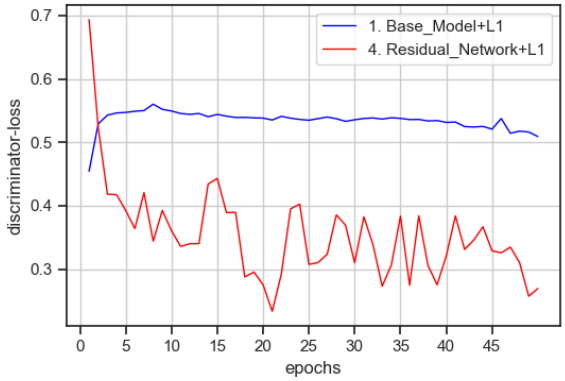
For calculating the content-loss we need the feature maps from pre-trained VGG16 network. In the experiment involving the content-loss regularizer, we use the output of 13th conv layer, whereas for the experiment with a combination of both the regularizers, we use the output of 7th conv layer. The regularizers were combined with $\lambda_1 = 100$ and $\lambda_2 = 10$.

With the experiments we performed, we see that combining both the losses is beneficial to enforce better quality colorization. However, convergence is difficult. The inclusion of both the terms helps capture the low-frequency and high-frequency regions in the image, which tends to reduce miscolorization.

Is downsampling necessary? The base model uses an encoder-decoder architecture model, i.e., UNet. It downsamples the image gradually and then upsamples the latent representation. The downsampling results in the loss of useful information that could have helped us colorize better, so we use a simple residual network [30] as the generator that does not change the image resolution while keeping the L1 regularizer as originally used in the 'pix2pix' paper.



(a) Generator+Regularizer Loss



(b) Discriminator Loss

(c) Qualitative results

Figure 6: Loss curves and qualitative comparison between ResNet based generator (without downsampling) and UNet generator: The generator loss corresponding to the simple residual network decreases first then flattens, however, the discriminator loss is noisy and does not converge. The qualitative results of the model 4 show brown-red patches in the colorized output.

The ResNet architecture used in this experiment consists of eight residual blocks, where each residual block has two 3x3 conv layers with the 'same' padding and stride of one. We use ReLU activation with each conv layer except the last layer, where we use the TanH activation. We use

batch-norm at the end of each residual block. The number of filters is 64 for each layer, and we keep the image resolution the same throughout the network.

Each residual block has skip-connections as usually used in the ResNet architecture, but additionally, we connect the network's input to the output via an extra skip-connection which will provide the local features from the input image.

This experiment shows that downsampling architecture is essential for image colorization. The image colorization problem can be viewed as a segmentation task where we make predictions at each pixel. The downsampling of feature maps helps in capturing the semantic information on a global scale.

The downsampling architecture also reduces computation and the inference time for such a model is lower. We report FPS of 17 with our simple ResNet architecture that does not perform downsampling, while the UNet model gives an FPS of 28.

*Model**	**FID** ↓	$\mathcal{E}$ **-Acc(5%)** ↑	**MAE** ↓	**PSNR** ↑
1. U-Net+L1	**32.29**	31.97	**15.14**	**21.72**
4. Residual Network+L1	42.26	**32.58**	16.42	21.01

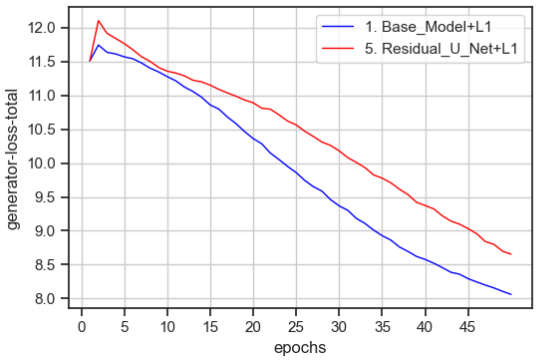
Table 2: Experiments show that an encoder-decoder architecture is better for colorization task and the ResNet architecture without downsampling performs poorly (*evaluated on MS-COCO dataset), ↓ : lower the better, ↑: higher the better

Are deeper networks better for colorization? As already stated in section 3.2, the proposed ResidualUNet architecture is deeper than the UNet used in the base model. The residual blocks make use of skip connections that helps in preventing the vanishing-gradient problem while also allowing the model to learn the identity function easily. With the deeper network, we expect it to learn essential texture details.

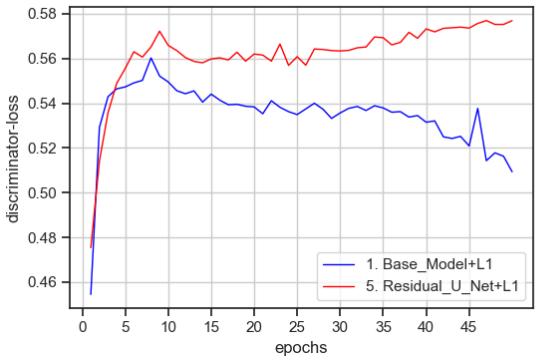
We perform experiments on ResidualUNet with a similar loss function as used in the pix2pix [4]. The convergence graph shows that the generator loss decreased identical to the base model (the base model converged more). However, the discriminator loss converged well with the ResidualUNet. The reported FID score (on 2k test images of MSCOCO-2017) is the best among all the experiments we performed.

*Model**	**FID** ↓	$\mathcal{E}$ **-Acc(5%)** ↑	**MAE** ↓	**PSNR** ↑
1. U-Net+L1	32.29	31.97	**15.14**	**21.72**
5. Residual U-Net+L1	**29.43**	**32.92**	15.56	21.55

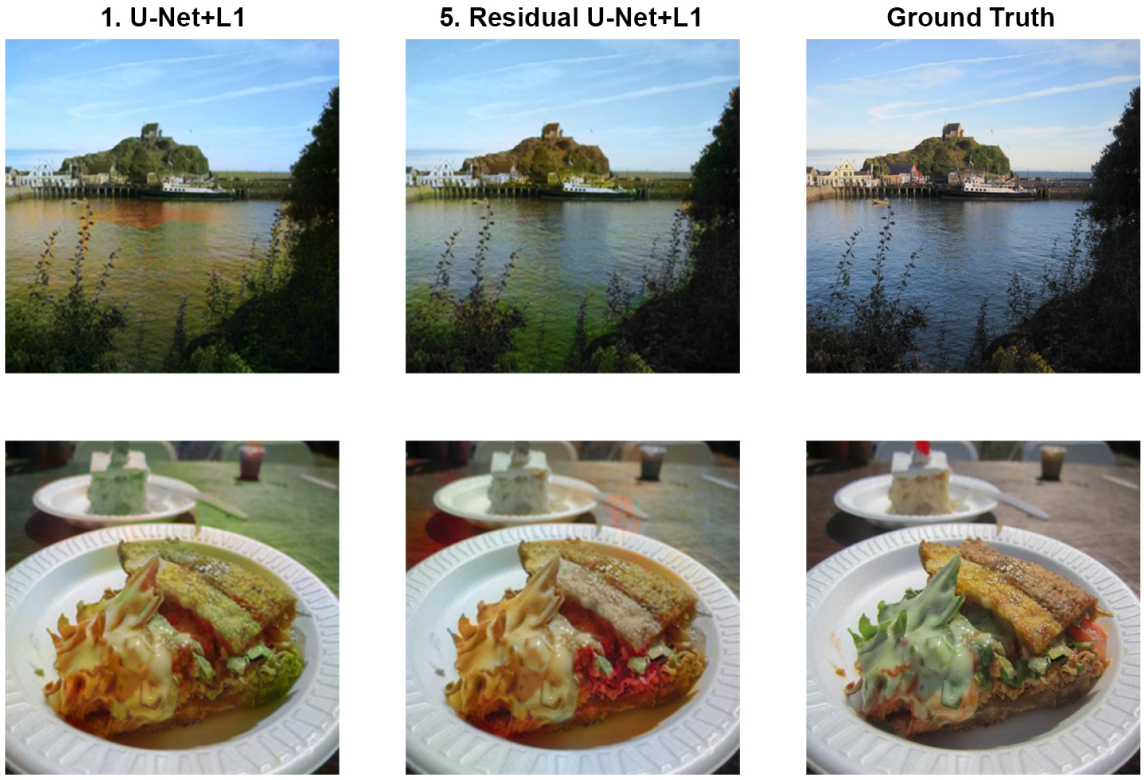
Table 3: Experiments show that a deeper encoder-decoder architecture, ResidualUNet performs better than our baseline implementation. This shows that a deeper network performs better on colorization task. (*evaluated on MS-COCO dataset), ↓ : lower the better, ↑: higher the better



(a) Generator+Regularizer Loss



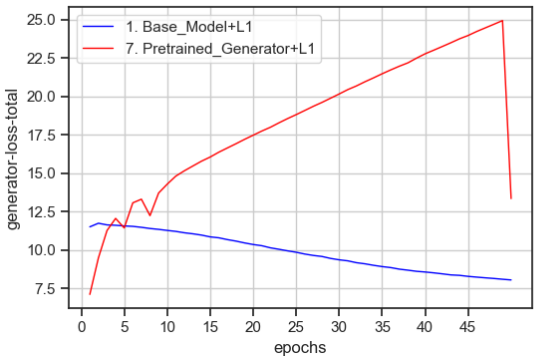
(b) Discriminator Loss



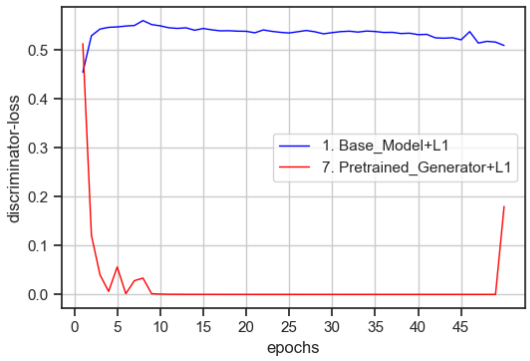
(c) Qualitative results

Figure 7: Loss curves and qualitative comparison between deep ResidualUNet and UNet: The qualitative results show that our proposed ResidualUNet based generator performs slightly better colorization.

Does generator pre-training help in convergence? The generator-discriminator architectures train in an adversarial manner. With randomly initialized weights, it's like 'blind leading the blind'. Since we expect the generator to generate images that have a small L1 distance from the ground-truth image, we pre-train the generator [31] on the train data for 20 epochs with L1 loss [32]. This helps the generator develop an initial semantic understanding of the colorization problem and helps in easy convergence during the adversarial training.



(a) Generator+Regularizer Loss



(b) Discriminator Loss

(c) Qualitative results

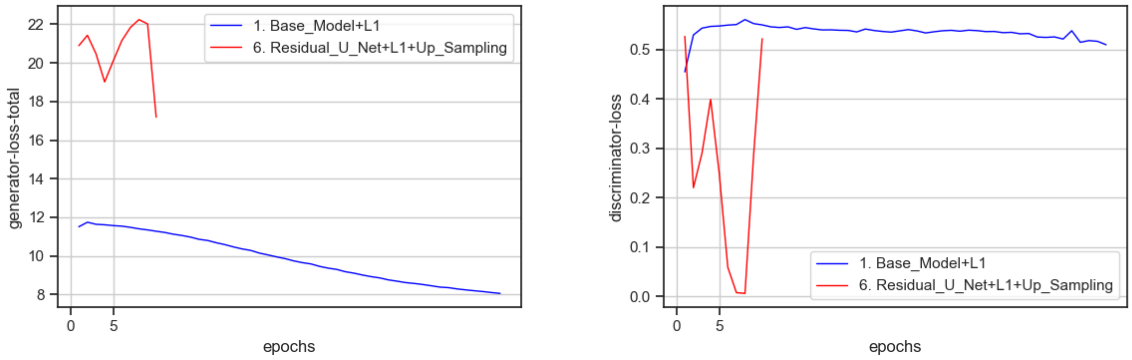
Figure 8: Loss curves and qualitative comparison between the base model and pre-trained generator: The discriminator loss curves show sudden convergence and generator loss curve seems to be anomalous. We think that pre-training the generator somehow affected the convergence of generator loss during the adversarial training. The qualitative results show that model 1 produced green patches in the railway track and model 7 chose better colors for the railway track.

After generator pre-training, we optimize adversarial and L1 loss similar to the base model. The experiments with this self-supervised learning approach show a faster convergence and better accuracy regarding MAE, PSNR, and epsilon-accuracy metrics. The FID score (on 2k test images of MSCOCO-2017) is the second-best among the experiments that we performed.

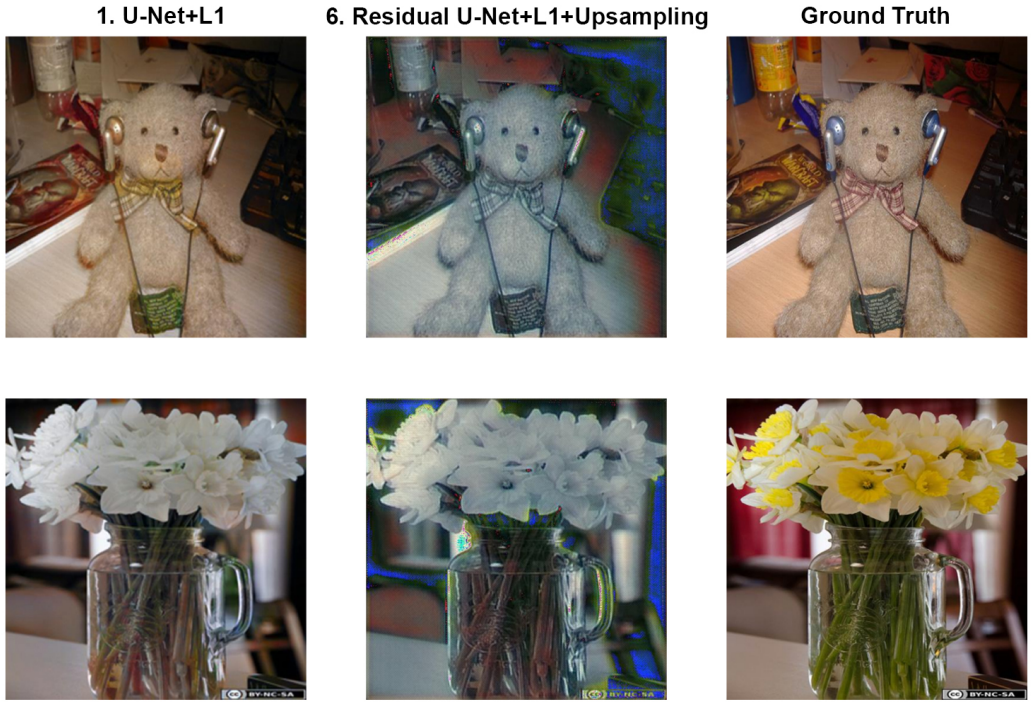
*Model**	**FID** ↓	$\mathcal{E}$ **-Acc(5%)** ↑	**MAE** ↓	**PSNR** ↑
1. U-Net+L1	32.29	31.97	15.14	21.72
7. Pre-trained Generator+L1	**31.69**	**47.18**	**11.50**	**24.19**

Table 4: Experiment performed with a pre-trained generator reports better accuracy (*evaluated on MS-COCO dataset), ↓ : lower the better, ↑: higher the better

End-to-end training of colorization with upsampling: The following experiment builds upon the previous experiments of ResidualUNet. With the intent to improve the output resolution, we append six residual blocks and an upsampling block to the pre-trained ResidualUNet and fine-tune it with the same objective function as the baseline.



(a) Generator+Regularizer Loss (b) Discriminator Loss



(c) Qualitative results

Figure 9: Loss curves and qualitative comparison between deep ResidualUNet with upsampling block and base model: The ResidualUNet with upsampling layer failed to produce good results. It did not seem to converge and the qualitative results are poor.

The six residual blocks follow the same architecture as stated in the earlier section. The upsampling is performed with PixelShuffle, and we use ParametricReLU, which was used in the SRGAN paper [5]. We also add a skip-connection from the ResidualUNet output to the final output after upsampling.

*Model**	**FID** ↓	$\mathcal{E}$ **-Acc(5%)** ↑	**MAE** ↓	**PSNR** ↑
1. U-Net+L1	**32.29**	**31.97**	**15.14**	**21.72**
6. Residual U-Net+L1+Upsampling	70.73	5.57	24.21	18.33

Table 5: End-to-end training for colorization and enhancement is difficult and our ResidualUNet (with upsampling) model performs poorly (*evaluated on MS-COCO dataset), ↓ : lower the better, ↑: higher the better

We report no better results in this model due to the unavailability of GPU resources. We stopped the training earlier because of poor results and heavy computational requirement. The end-to-end training is also complicated because we use L*a*b colorspace for colorization and the convergence to RGB breaks the gradient computation. We leave it as future work to train this model end-to-end on L*a*b colorspace with the full dataset.

5 Results

We performed several experiments with different generator architectures and combinations of regularizers. We found that our deeper ResidualUNet performs close to our baseline implementation. The experiment with pre-trained generator shows best results overall (note that, the comparison is subject to the size of test set). We have evaluated our model on 5k images from ImageNet 2012 validation set and 2k images from MSCOCO (evaluated separately). We report the results of our experiments in the table below.

	Models	FID ↓	$\mathcal{E}$-Accuracy(5%)↑	MAE ↓	PSNR ↑	FPS ↑
SOTA[@]	cGAN[+]	-	**65.5**	**5.1**	-	-
	Pix-to$_{PIX}$	24.41	-	-	-	-
	Palette (SOTA)	**15.78**	-	-	-	-
Experiments[@]	UNet + L1	16.75	32.98	16.00	21.56	28
	UNet + Perceptual Loss	17.94	30.03	16.52	21.21	28
	UNet + Perceptual + L1	16.66	32.56	16.04	21.54	28
	Residual Generator + L1	25.82	9.98	23.66	18.61	17
	Residual-U-Net + L1	16.82	30.32	17.41	20.94	23
	Residual-U-Net + L1 (Upsampling)	44.05	5.48	24.62	18.31	21
	Pre-Trained UNet + L1	**15.84**	**43.36**	**13.01**	**23.68**	28

Table 6: Comparison of our proposed models and experiments with existing state of the arts.
+ : evaluated on CIFAR @ : evaluated on 5K ImageNet 2012 val_set, ↓ : lower the better, ↑ : higher the better

Models	Dataset	FID ↓	$\mathcal{E}$-Accuracy(5%)↑	MAE ↓	PSNR ↑
U-Net+L1	ImageNet	**16.75**	**32.98**	16.00	21.56
	MS-COCO	32.29	31.97	**15.14**	**21.72**
UNet + Perceptual Loss	ImageNet	**17.94**	30.03	16.52	21.21
	MS-COCO	35.66	**31.13**	**15.74**	**21.36**
UNet + Perceptual + L1	ImageNet	**16.66**	32.56	16.04	21.54
	MS-COCO	29.50	**35.39**	**14.43**	**22.15**
Residual Generator + L1	ImageNet	**25.82**	9.98	23.66	18.61
	MS-COCO	42.26	**32.58**	**16.42**	**21.01**
Residual-U-Net + L1	ImageNet	**16.82**	30.32	17.41	20.94
	MS-COCO	29.43	**32.92**	**15.15**	**21.55**
Residual-U-Net + L1 (Upsampling)	ImageNet	**44.05**	5.48	24.62	18.31
	MS-COCO	70.73	**5.57**	**24.21**	**18.33**
Pre-Trained UNet + L1	ImageNet	**15.84***	43.36	13.01	23.68
	MS-COCO	31.69	**47.18***	**11.50***	**24.19***

Table 7: Comparison of our proposed model and experiments on 2K MS-COCO images and 5K ImageNet images :
: overall best performance ↓: *lower the better*, ↑: *higher the better*

6 Future Work

We have successfully developed a c-GANs based method to colorize black and white images. There are various shortcomings of our model that can we can work on:

- The size of the training set used in our project is small, and thus training on large-corpus is required to judge the effectiveness of our model.
- End-to-end colorization and enhancement model can be developed.
- Ensemble of multiple colorization specific loss functions can be used.
- Accuracy can also be improved by using the class balanced dataset (where no color is overpowering).

7 Conclusion

Our experiments suggest that c-GANs are a useful framework for automated colorization. With deeper architectures like ResidualUNet, we can improve the results on certain images that involve diverse scenes. Although the comparison is not entirely fair due to capped training and the small dataset used, they demonstrate the usefulness of deeper architectures and tuned regularizers that enforce the generator to produce natural colors and not select arbitrary colors just to fool the discriminator.

For the entirety of this undertaking, we had set out with a vision to colorize images and to up-sample low-resolution images in the process. While we did come up with architectures that performs reasonably well on the colorization part (even with lower epochs and lesser samples to train with), the performance certainly shudders when an up-sampling specific block is added. This issue might be tied to lesser training epochs or lesser images seen by the model.

8 Individual Contributions

The individual contribution is shown in the below table:

Team Member	Contribution
Arjun Singh (25%)	1. Perceptual Loss 2. Evaluation Metrics 3. Experiments and Report
Debdeep Paul Chaudhuri (25%)	1. ResidualUNet Implementation 2. Extensive Literature Survey 3. Experiments and Report
Himanshu Lal (25%)	1. ResidualUNet Implementation 2. Upsampling Architecture Implementation 3. Experiments and Report
Shivam Tripathi (25%)	1. Base Model Implementation 2. Generator Pre-training 3. Experiments and Report

References

[1] A. Deshpande, J. Lu, M.-C. Yeh, M. J. Chong, and D. Forsyth, "Learning diverse image colorization," 2016.

[2] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," 2016.

[3] K. Nazeri, E. Ng, and M. Ebrahimi, "Image colorization using generative adversarial networks," in *Articulated Motion and Deformable Objects*, pp. 85–94, Springer International Publishing, 2018.

[4] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," 2016.

[5] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," 2016.

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[7] S. Anwar, M. Tahir, C. Li, A. Mian, F. S. Khan, and A. W. Muzaffar, "Image colorization: A survey and dataset," 2020.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[9] C. Saharia, W. Chan, H. Chang, C. A. Lee, J. Ho, T. Salimans, D. J. Fleet, and M. Norouzi, "Palette: Image-to-image diffusion models," 2021.

[10] "Colorization on ImageNet val." https://paperswithcode.com/sota/colorization-on-imagenet-val.

[11] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," *ACM Trans. Graph.*, vol. 23, p. 689–694, aug 2004.

[12] L. Yatziv and G. Sapiro, "Fast image and video colorization using chrominance blending," *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, vol. 15, pp. 1120–9, 06 2006.

[13] Q. Luan, F. Wen, D. Cohen-Or, L. Liang, Y.-Q. Xu, and H.-Y. Shum, "Natural Image Colorization," in *Rendering Techniques* (J. Kautz and S. Pattanaik, eds.), The Eurographics Association, 2007.

[14] Y.-C. Huang, Y.-S. Tung, J.-C. Chen, S.-W. Wang, and J.-L. Wu, "An adaptive edge detection based colorization algorithm and its applications," pp. 351–354, 01 2005.

[15] P. Vitoria, L. Raad, and C. Ballester, "Chromagan: Adversarial picture colorization with semantic class distribution," 2019.

[16] Z. Cheng, Q. Yang, and B. Sheng, "Deep colorization," 2016.

[17] G. Larsson, M. Maire, and G. Shakhnarovich, "Learning representations for automatic colorization," 2016.

[18] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification," *ACM Trans. Graph.*, vol. 35, jul 2016.

[19] H. Zhao, W. Wu, Y. Liu, and D. He, "Color2embed: Fast exemplar-based image colorization using color embeddings," 2021.

[20] X. Liu, L. Wan, Y. Qu, T.-T. Wong, S. Lin, C.-S. Leung, and P.-A. Heng, "Intrinsic colorization," in *ACM SIGGRAPH Asia 2008 Papers*, SIGGRAPH Asia '08, (New York, NY, USA), Association for Computing Machinery, 2008.

[21] A. Y.-S. Chia, S. Zhuo, R. K. Gupta, Y.-W. Tai, S.-Y. Cho, P. Tan, and S. Lin, "Semantic colorization with internet images," *ACM Trans. Graph.*, vol. 30, p. 1–8, dec 2011.

[22] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[23] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014.

[24] "CycleGAN and pix2pix in PyTorch." https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix.

[25] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2014.

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[27] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," 2017.

[28] "FID score for PyTorch." https://github.com/mseitzer/pytorch-fid.

[29] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[31] A. Lucas, S. Lopez-Tapia, R. Molina, and A. K. Katsaggelos, "Generative adversarial networks and perceptual losses for video super-resolution," *IEEE Transactions on Image Processing*, vol. 28, pp. 3312–3327, jul 2019.

[32] "Colorizing black white images with U-Net and conditional GAN — A Tutorial." https://towardsdatascience.com/colorizing-black-white-images-with-u-net-and-conditional-gan-a-tutorial-81b2df111cd8.

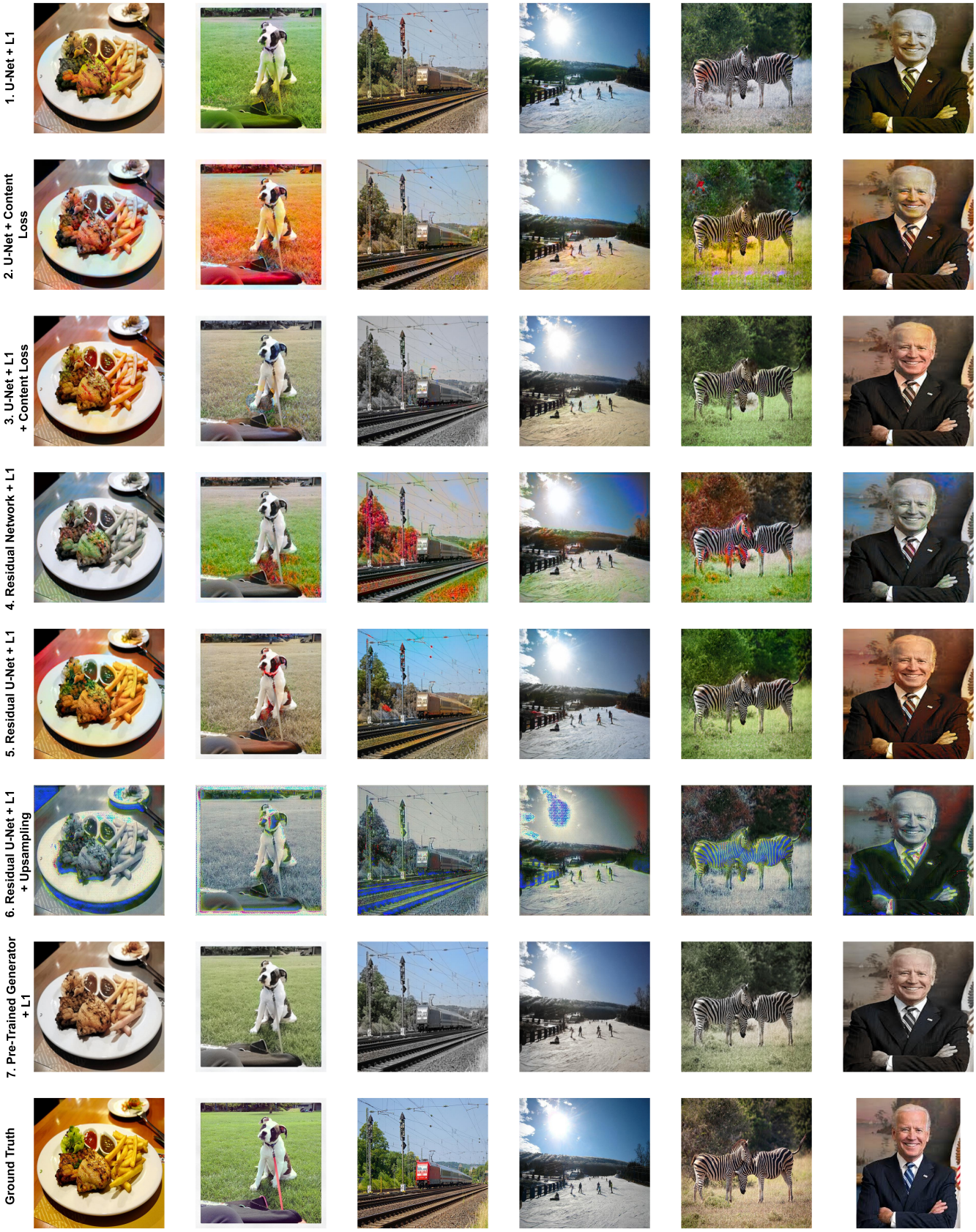


Figure 10: Qualitative result for all the experiments