Team : HASHeD

Colorization Using Conditional-GANs

Under **Prof. Priyanka Bagade**

TAs : **Dhanajit Brahma** & **Neeraj Matiyali**

Team Members

Arjun Singh
21111402
arjuns21@iitk.ac.in

Debdeep Paul Chaudhuri
21111413
debdeeppc21@iitk.ac.in

Himanshu Lal
21111403
himanshul21@iitk.ac.in

Shivam Tripathi
21111408
shivamtr21@iitk.ac.in

Introduction and Motivation

With the advent of Deep Learning and availability of heavy-computing resources, we can automate the process of colorizing black-&-white images.

The automated framework can be used for historical black-and-white images, motion-pictures, astronomical images, Scientific and medical images.

Literature Survey

1. (Base Paper) Image-to-Image Translation with Conditional Adversarial Networks (link)
2. Image Colorization with Generative Adversarial Networks (link)
3. Learning Representations for Automatic Colorization (link)
4. Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification (link)
5. Color2Embed: Fast Exemplar-Based Image Colorization using Color Embeddings (link)
6. ChromaGAN: Adversarial Picture Colorization with Semantic Class Distribution (link)
7. Deep Colorization (link)
8. Image Colorization: A Survey and Dataset (link)

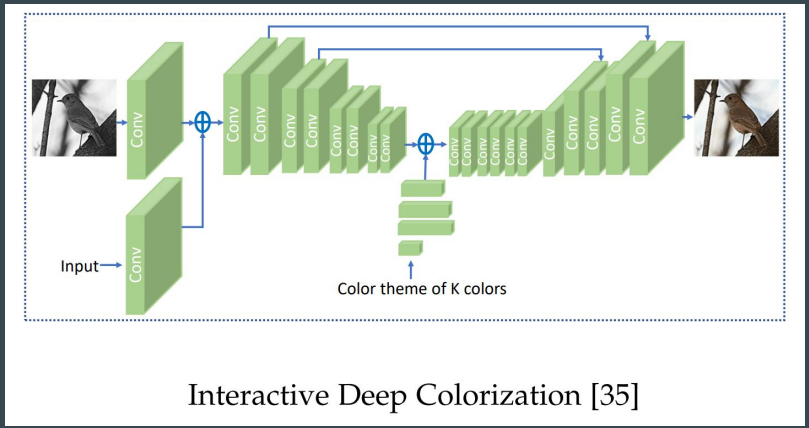
Additional Papers

9. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network (link)
10. DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks (link)

Literature Survey

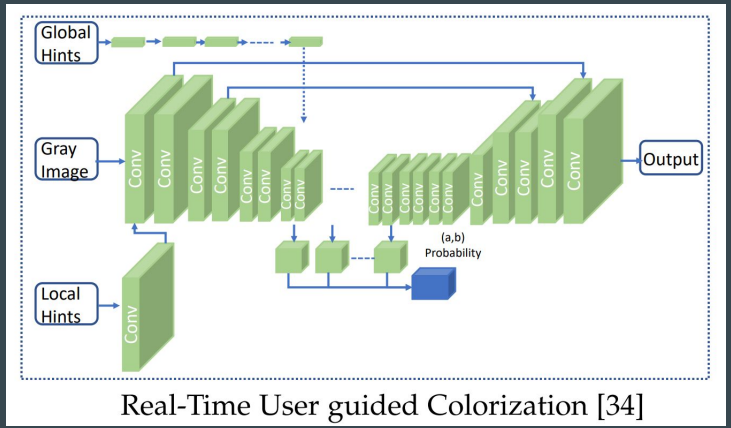
User-Guided Networks	Deep Neural Networks	Generative Models
Scribbler	Deep Colorization	Image Colorization using GANs
Real-time User-Guided Colorization	Colorful Colorization	ChromaGAN
Interactive Deep Colorization	Let There Be Color	Learning Diverse Image Colorization
Anime Line Art Colorization	Color-2-Embed	Image-to-Image Translation with c-GAN

User-Guided Networks



Interactive Deep Colorization [35]

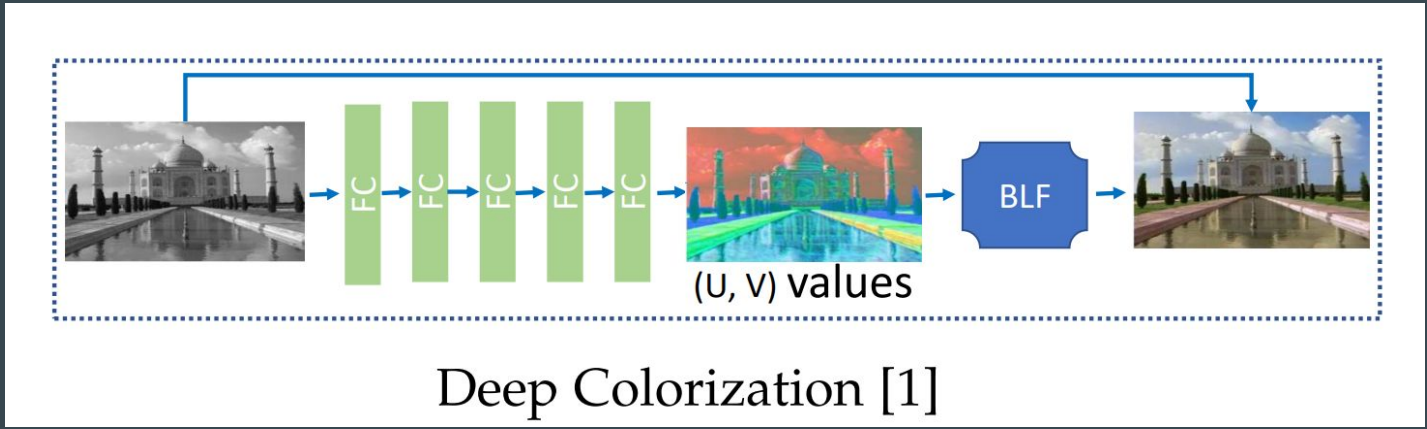
- Diverse colorization possible
- Simple and easy to implement
- Domain knowledge is used



Real-Time User guided Colorization [34]

- Colors are subject to user preference
- Selected colors may not be natural
- Requires human intervention
- Not generalizable

Deep Neural Networks and Exemplar Based

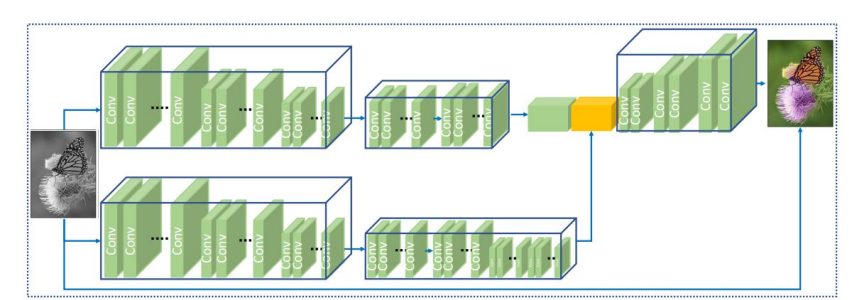


Deep Colorization [1]

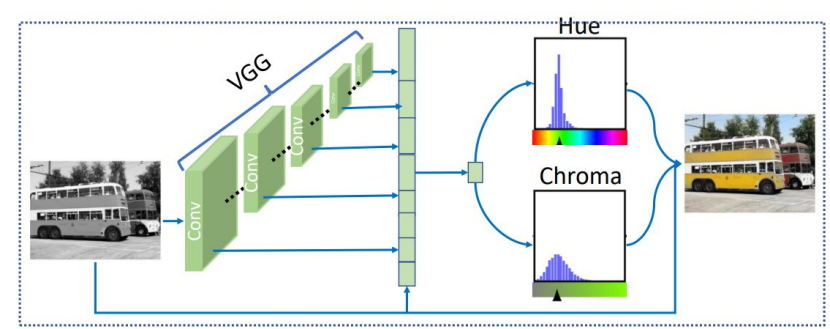
- Employs supervised training for automated colorization
- Guided via exemplar images

- Difficult to obtain original colors
- Dependent on training data
- Challenging to quantify results
- Example images may not be useful for natural colorization

Complex Deep Neural Networks and GAN Based Models



Let there be color [63]



Learning Automatic Colorization

- Fully-convolutional skip-connection architecture learns semantically meaningful features
- GAN based architectures can perform several tasks related to image-to-image translation

- Mis-colorization is common
- Inconsistent background colorization
- Un-natural color shifts
- Complex architectures require several days of training on high-end GPUs

conditional-Generative Adversarial Networks (cGANs)

Trains two models in an adversarial manner.

The generator tries to perform the task in hand and the discriminator learns to differentiate between the generated (fake) and real output.

The mini-max game produces a trained generator.

The lightness channel (L) acts as a condition that facilitates learning as the semantic nature of grayscale image matches with that of ground-truth image.

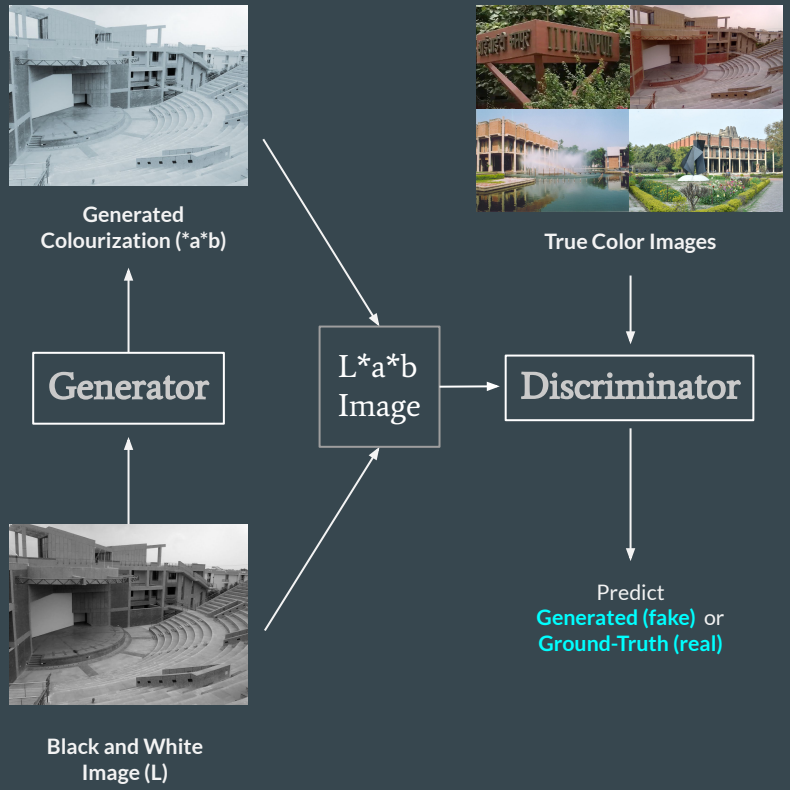
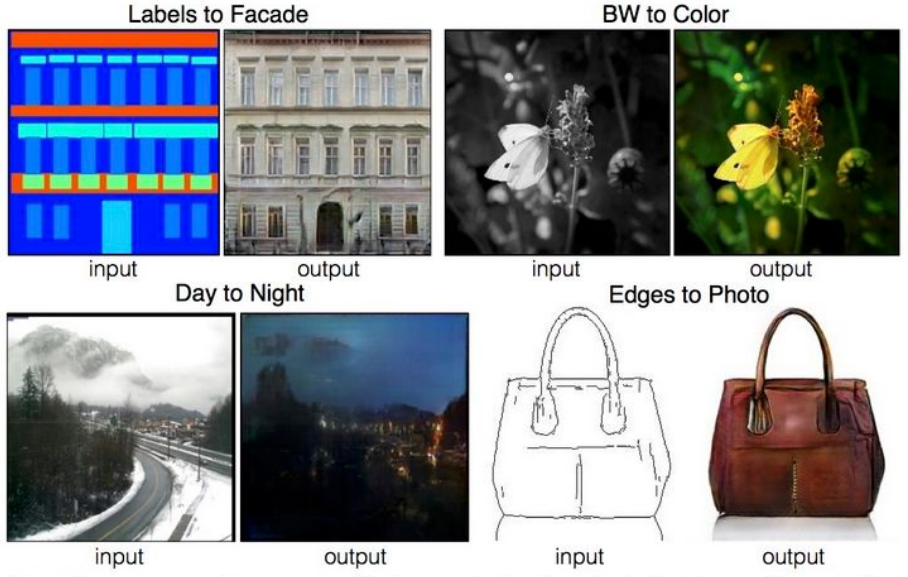
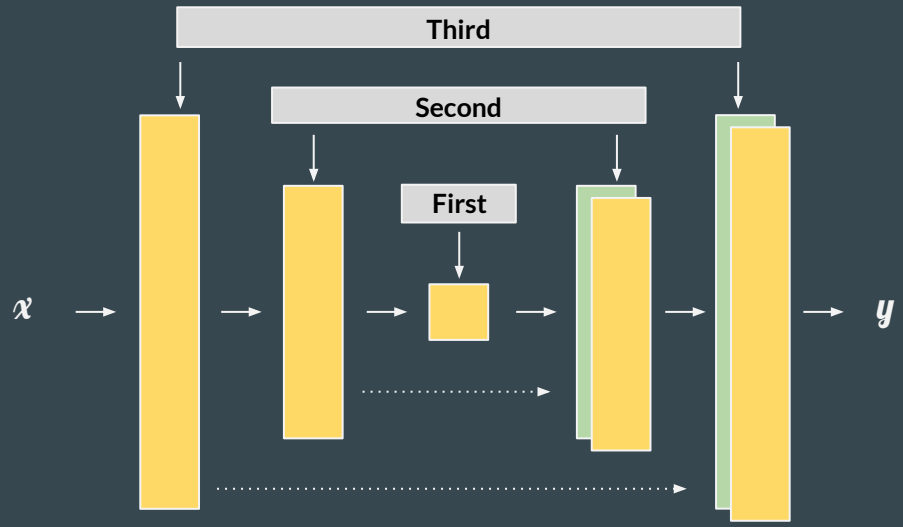


Image-to-Image Translation with conditional-GAN (pix2pix)

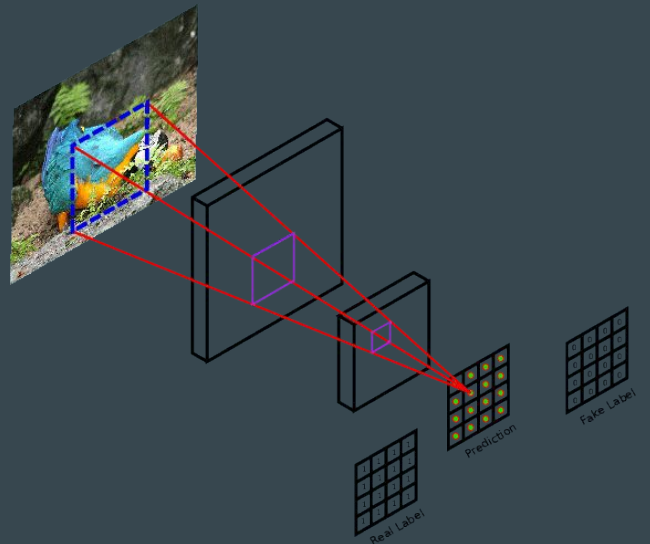


A general image-to-image translation model based on conditional GAN that can perform tasks such as converting maps to satellite photographs, black and white photographs to color, and sketches of products to product photographs.

Image-to-Image Translation with conditional-GAN (pix2pix)



Generator (UNet)



Discriminator (PatchGAN)

Objective Function

Adversarial Loss →

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

L1 Loss →

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

Final Objective →

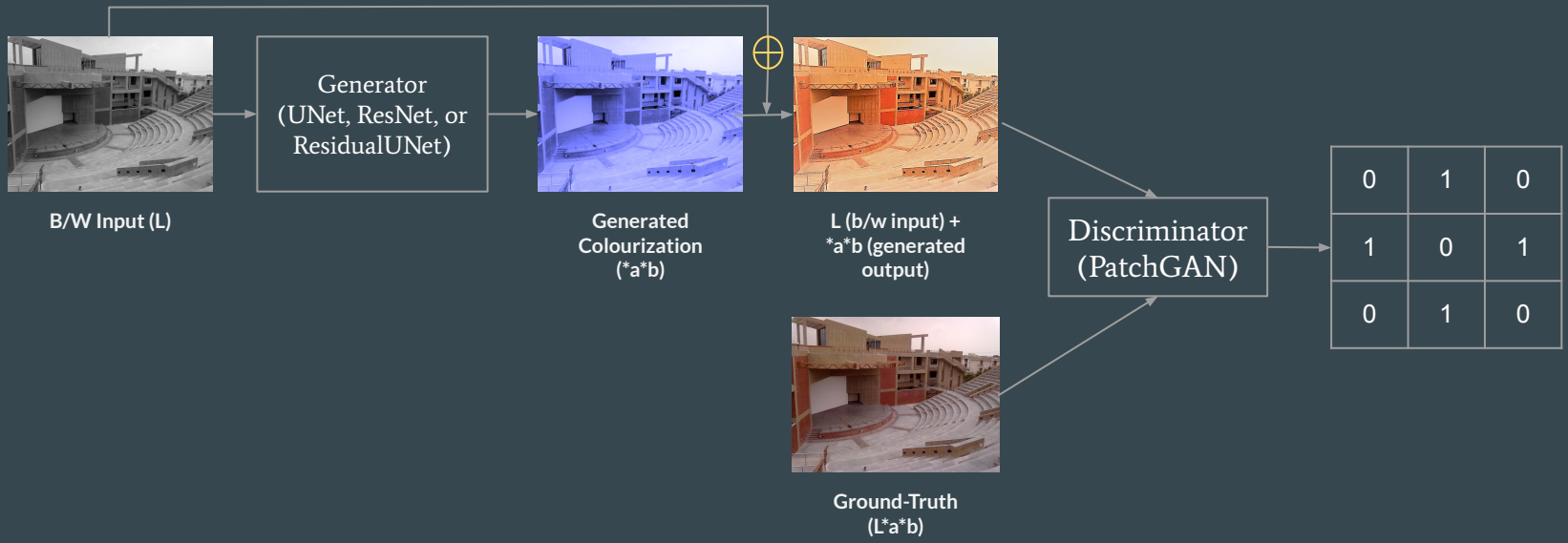
$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

Image-to-Image Translation with conditional-GAN (pix2pix)

Mis-Colorization: Regions with high fluctuations are frequently colored green. This is likely caused by majority of green regions in training data (like, grasslands).

Generic framework, not specific to colorization.

General Architecture for Experiments

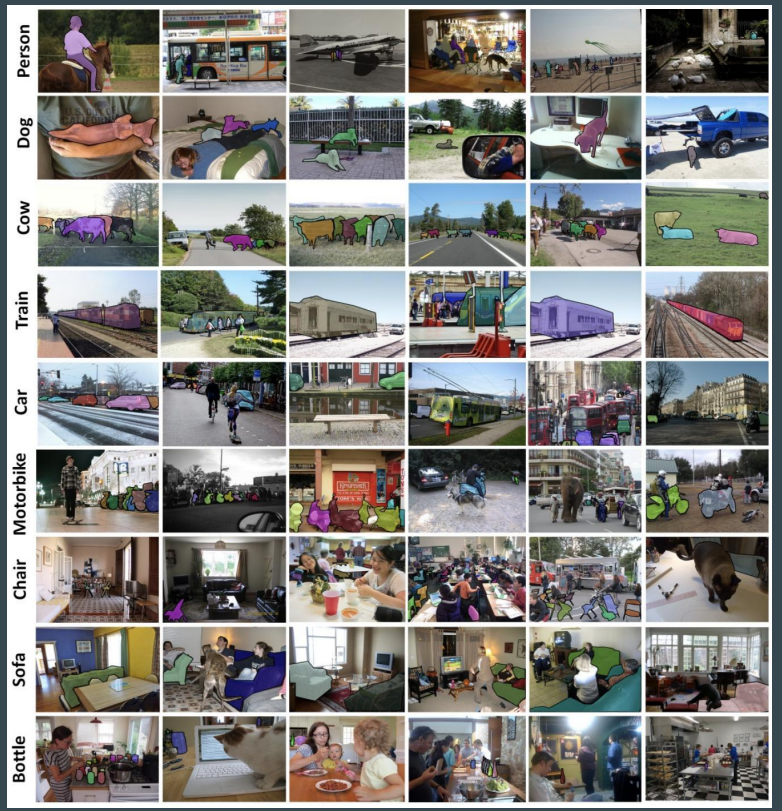


Dataset (MS-COCO 2017)

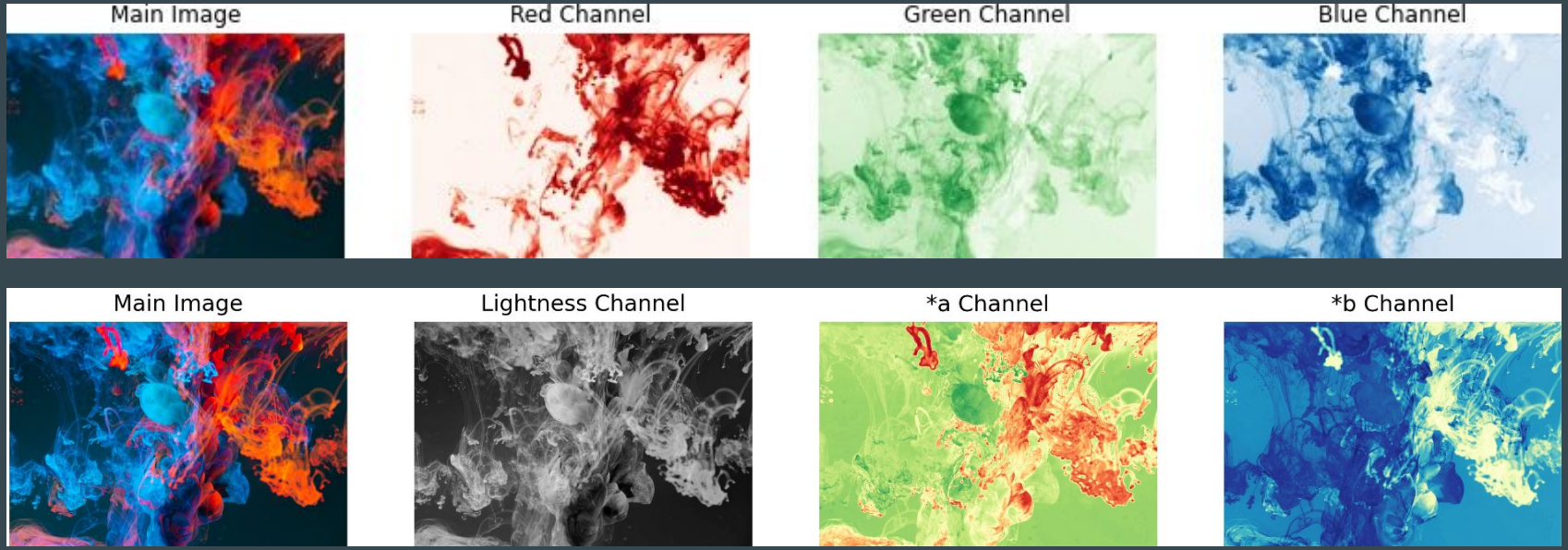
Large-scale object detection, segmentation, key-point detection, and captioning dataset.

Total Number of Images: 328k

Training Data for Experiments: 8k images randomly sampled from train2017 of MS-COCO.



L*a*b Colorspace



L channel is given as an input during training and the generator has to produce *a*b channels.

Easier for the generator to learn since RGB output requires the generator to predict 3 different color channels.

Experiments

1. UNet + L1 (base model)	• Generator: UNet • Loss: Adversarial Loss + L1
2. UNet + Perceptual Loss	• Generator: UNet • Loss: Adversarial Loss + Content Loss
3. UNet + Perceptual & L1	• Generator: UNet • Loss: Adversarial Loss + Content Loss + L1
4. Residual Generator + L1	• Generator: ResNet (no downsampling) • Loss: Adversarial Loss + L1
5. ResidualUNet + L1	• Generator: UNet with Residual Blocks • Loss: Adversarial Loss + L1
6. ResidualUNet + L1 (upsampling)	• Generator: Added upsampling layers • Loss: Adversarial Loss + L1
7. Pre-trained UNet + L1	• Generator: UNet pre-trained with L1 loss • Loss: Adversarial Loss + L1

1. UNet + L1 (base model)

- Generator: UNet
- Loss: Adversarial Loss + L1

UNet Encoder-decoder architecture with skip-connections that share low-level features in encoder with corresponding high-level features in decoder.

The loss function includes an **L1 term as a regularizer** that forces generator to preserve the structure of original image and prevent it from assigning arbitrary colors to pixels just to 'fool' the discriminator.

Note: L2 loss can also be used but it produces blurry results.

2. UNet + Perceptual Loss

- Generator: UNet
- Loss: Adversarial Loss + Content Loss

Content Loss: Mean squared difference between the (vgg16) conv feature maps of generator output and ground-truth.

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

Content loss is used to capture high-frequency elements in an image.

Colorization quality is usually evaluated by using a group of persons that identify whether the colored image is fake or real. To make this quantifiable, content loss uses mean squared difference between the VGG16 feature maps. The feature maps of a pre-trained VGG16 network have a good semantic understanding of features that can mimic human-level supervision to some extent.

2. UNet + Perceptual Loss

- Generator: UNet
- Loss: Adversarial Loss + Content Loss

Insights

Perceptual loss alone as a regularizer does not produce good results.
It fails to capture low frequency regions that L1 loss does well.

3. UNet + Perceptual & L1

- Generator: UNet
- Loss: Adversarial Loss + Content Loss + L1

Perceptual loss alone as a regularizer did not produce good results.
In this experiment, we use both L1 loss and content loss, weighted by a lambda.

L1 loss maintains the low frequency elements of the image [pix2pix] and content loss maintains high frequency elements.




Insights — L1 loss maintains the low frequency elements of the image [pix2pix] and content loss maintains high frequency elements.

4. Residual Generator + L1

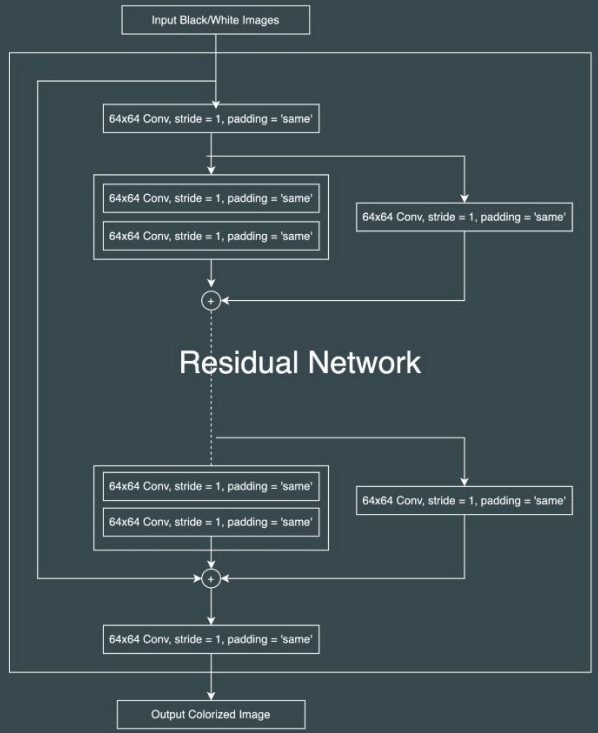
- Generator: ResNet (no downsampling)
- Loss: Adversarial Loss + L1

Base model used a UNet as a generator which has downsampling in the encoder part.

We decided to test the colorization with a ResNet architecture that does not perform downsampling.

Trained with adversarial loss and a L1 term as a regularizer.





Insights

Downsampling is important as colorization is a kind of segmentation task

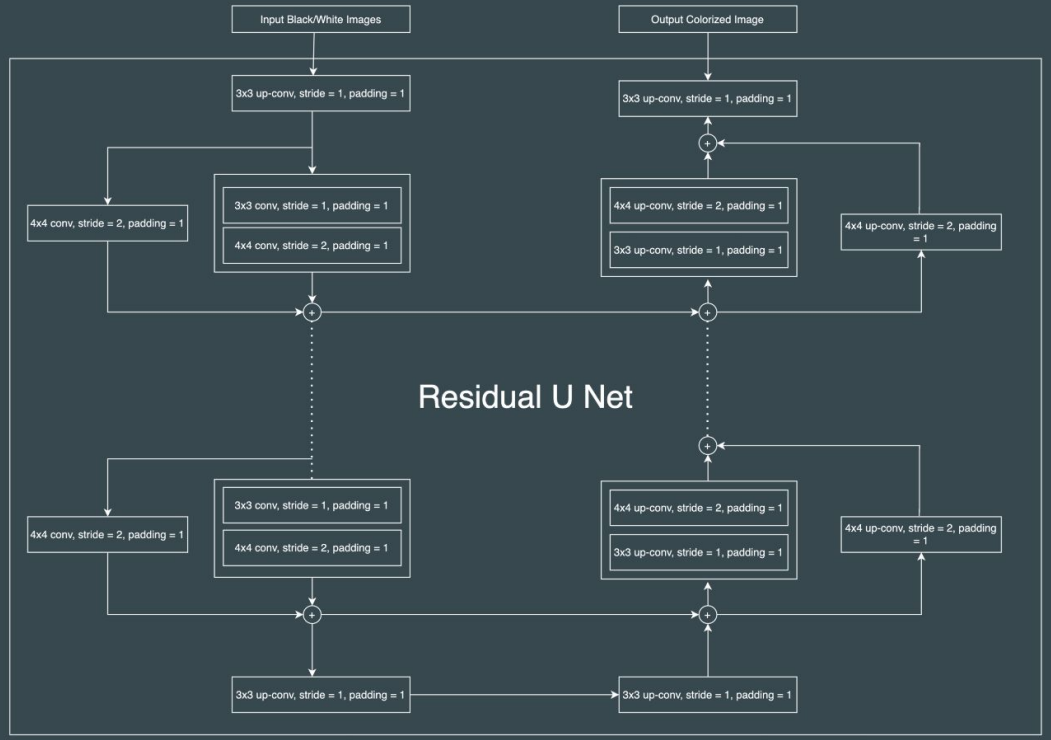
5. ResidualUNet + L1

- Generator: UNet with Residual Blocks
- Loss: Adversarial Loss + L1

ResidualUNet has the similar structure as UNet where each layer is a residual block in itself.

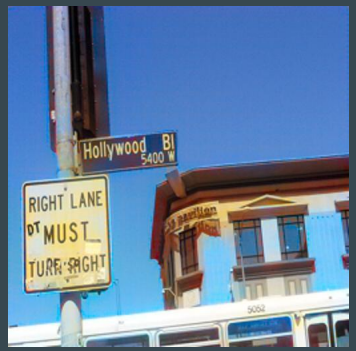
With the network more deep, it is expected to learn more subtle features.

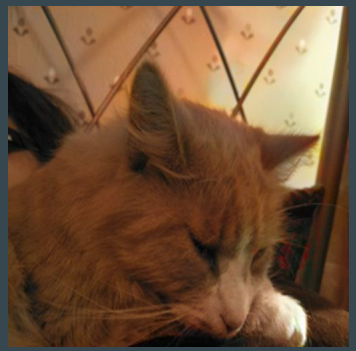
Residual blocks have skip-connections that prevent vanishing gradient problem [ResNet].



5. ResidualUNet + L1

- Generator: UNet with Residual Blocks
- Loss: Adversarial Loss + L1



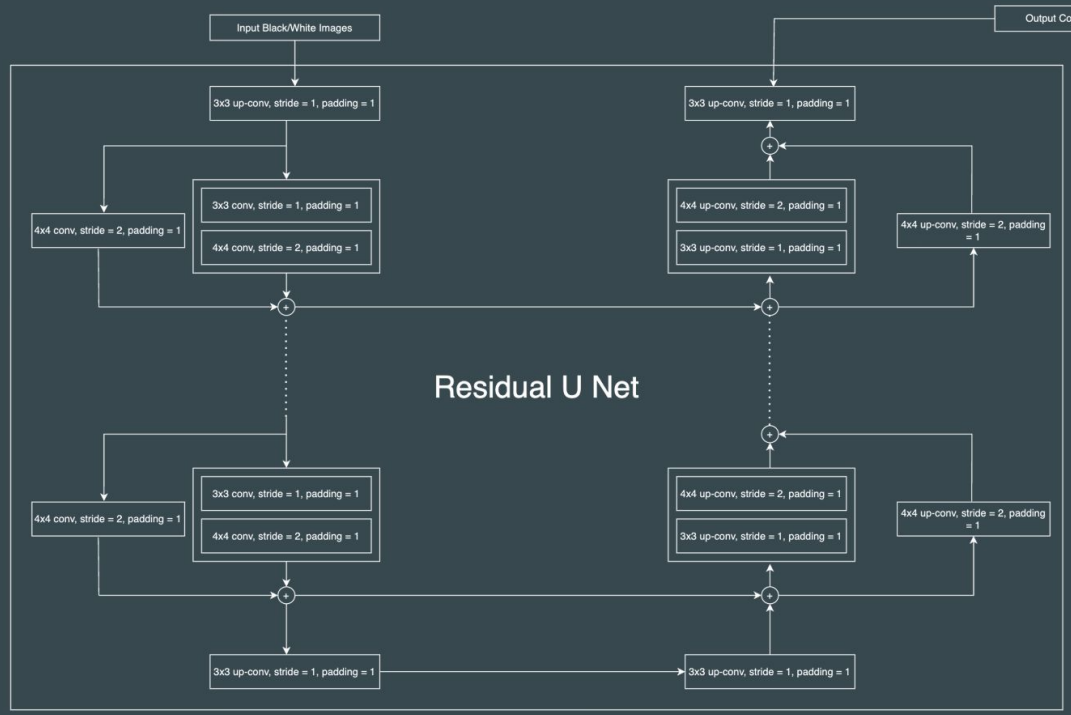


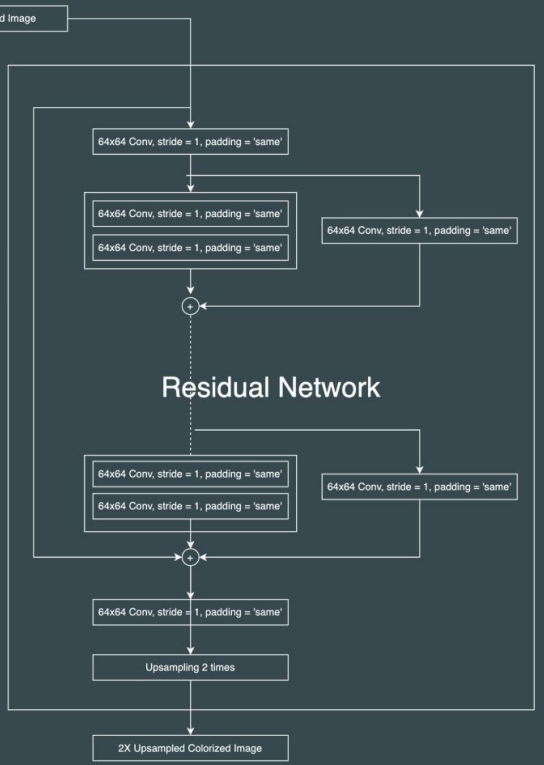
Insights

ResidualUNet is a deeper network that produces decent results. Although further improvements can be done by training with a larger training set.

6. ResidualUNet + L1 (upsampling)

- Generator: UNet with Residual Blocks
- Loss: Adversarial Loss + L1

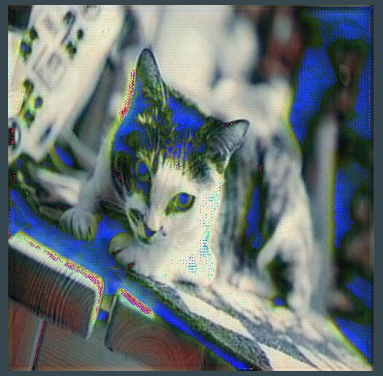


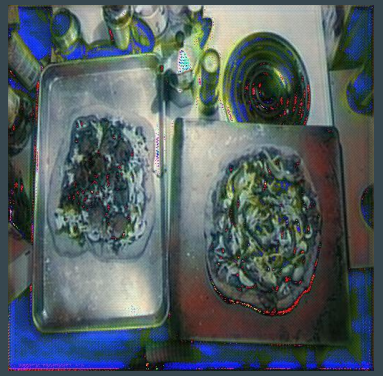


6. ResidualUNet + L1 (upsampling)

- Generator: UNet with Residual Blocks
- Loss: Adversarial Loss + L1

Upsampling layers are added to our ResidualUNet with the hope to improve the resolution of colored output.



Insights

End-to-end training for large iterations is difficult and requires a lot of GPU resources to train. More improvements might be needed to make this work.

7. Pre-trained UNet + L1

- Generator: UNet pre-trained with L1 loss
- Loss: Adversarial Loss + L1

Pre-training the generator for self-supervised learning.

A randomly initialized generator and discriminator are like blind-leading-the-blind.

We load ResNet18 weights in the encoder part of UNet and pre-train the generator with L1 loss to induce an initial understanding of the colorization problem.

Results in good colorization from the initial epochs.

Insights
With a head-start due to pre-training, the model performs similar to base model but trains quickly.

Evaluation Metrics

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

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

Epsilon Accuracy (ϵ) : % of the correct pixels (within certain threshold, **ϵ**)

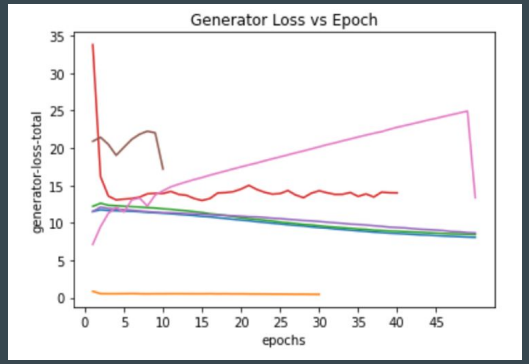
$$Accuracy\ \mathcal{E} = \frac{1}{N * C * H * W} \sum_{n_i}^{N} \sum_{c_j}^{C} \sum_{h_k}^{H} \sum_{w_l}^{W} [|y - \hat{y}|_{i,j,k,l} < \mathcal{E}]$$

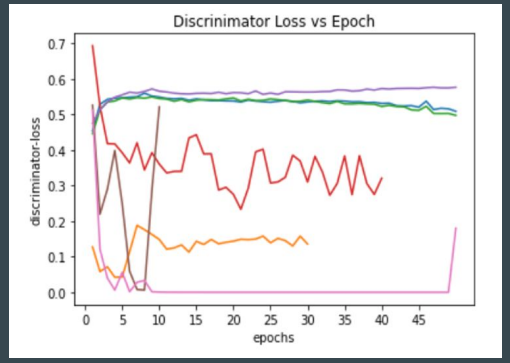
Peak Signal to Noise Ratio (PSNR) : Log of ratio of maximum possible value to the root mean square error.

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

Frames per Second (FPS) : Number of images that can be processed in a second.

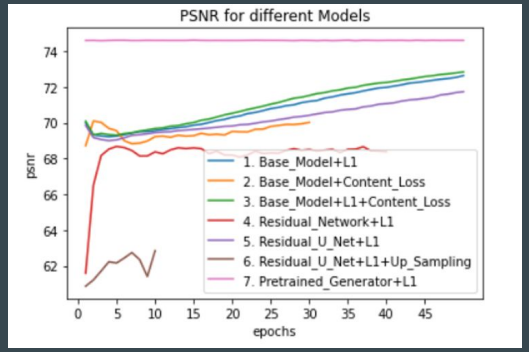
Accuracy Curves

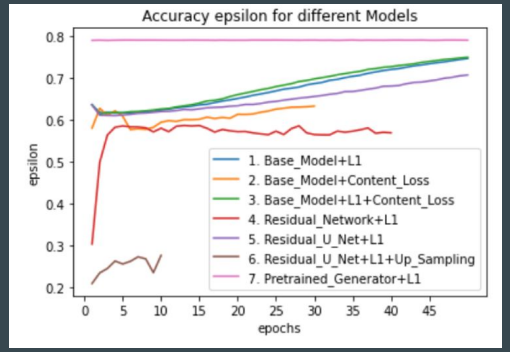


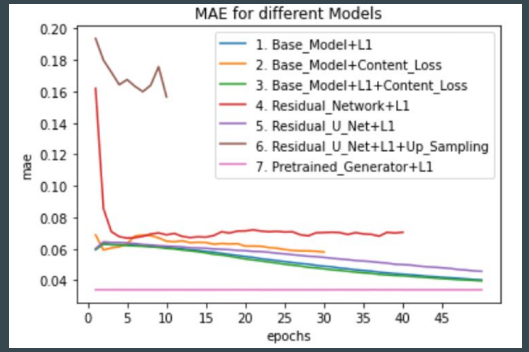


Legends

1. Base_Model+L1
2. Base_Model+Content_Loss
3. Base_Model+L1+Content_Loss
4. Residual_Network+L1
5. Residual_U_Net+L1
6. Residual_U_Net+L1+Up_Sampling
7. Pretrained_Generator+L1







Comparison Table

#	Experiment	FID↓	ϵ-Accuracy↑	MAE↓	PSNR↑	FPS↑
1	cGAN[1@]	-	65.5[+]	5.1[+]	-	-
2	Pix-to-Pix[2]	24.41	-	-	-	-
3	Palette (SOTA)[3#]	15.78[+]	-	-	-	-
4	UNet + L1 (base model)[$]	32.29	31.97	15.14	21.72	28
5	UNet + Perceptual Loss[$]	35.66	31.13	15.74	21.36	28
6	UNet + Perceptual & L1[$]	29.50	35.39	14.43	22.15	28
7	Residual Generator + L1[$]	42.26	32.58	16.42	21.01	17*
8	ResidualUNet + L1[$]	29.43*	32.92	15.56	21.55	23
9	ResidualUNet + L1 (upsampling)[$]	70.73	5.57	24.21	18.33	21
10	Pre-trained UNet + L1[$]	31.69	47.18*	11.50*	24.19*	28

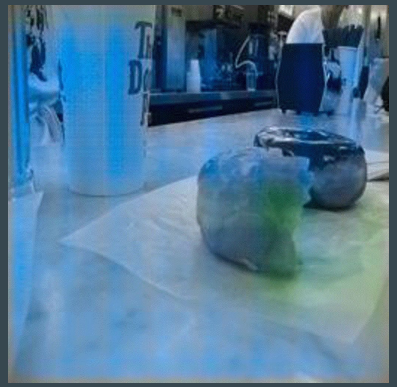
1. Image Colorization with Generative Adversarial Networks (link). 2. Image-to-Image Translation with Conditional Adversarial Networks (link). 3. Palette: Image-to-Image Diffusion Models(Link).

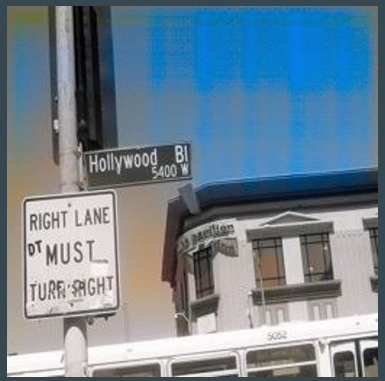
@: CIFAR, #: ImageNet, $: MSCOCO(2K subset)

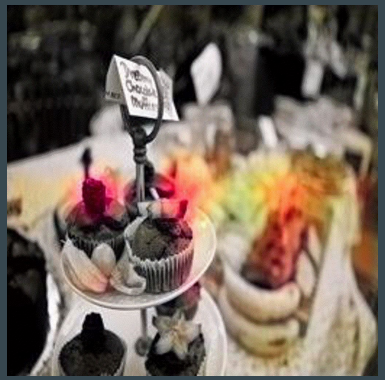
↓: Lower the better, ↑: Higher the better

+: SOTA , *: Ours

Learning to Colorize (50 epochs)







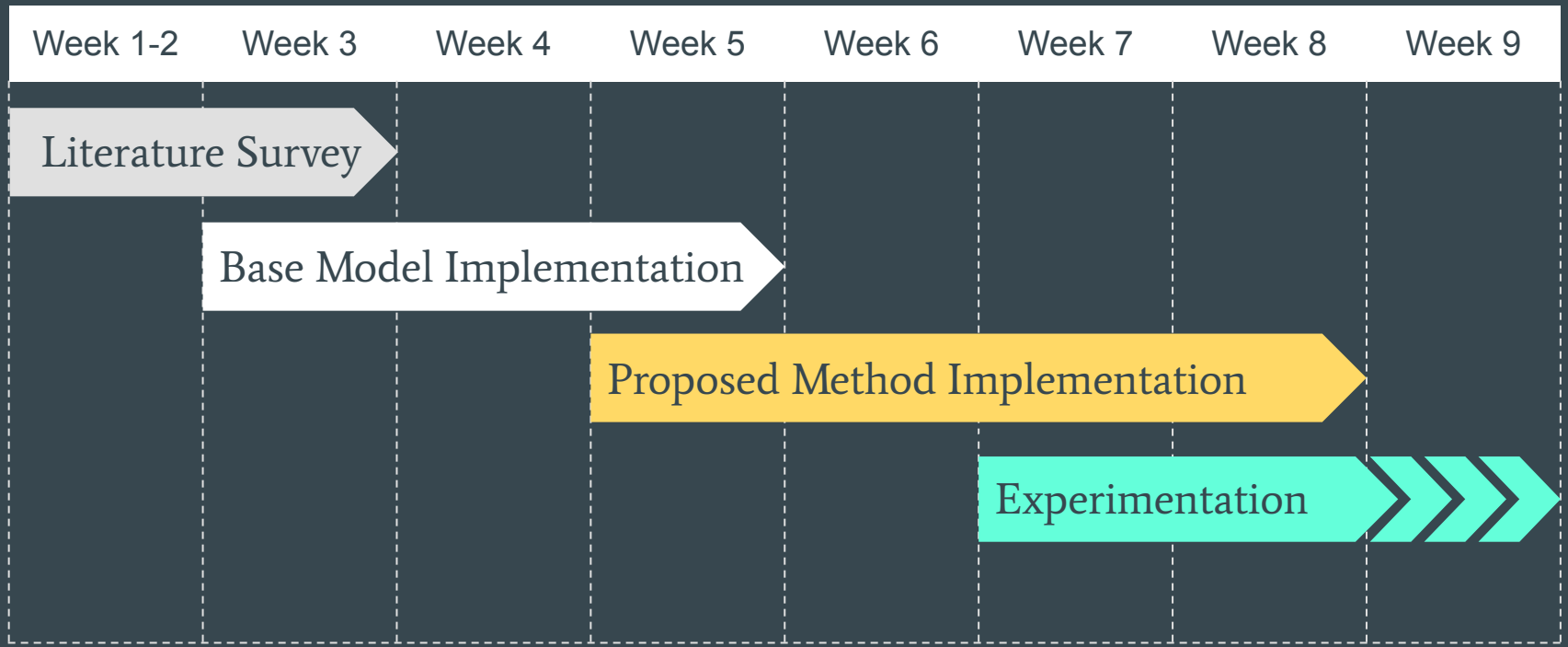
Challenges

1. Unavailability of VGG models pre-trained on images of L*a*b colorspace. General conversion of L*a*b to RGB colorspace breaks the gradient computation and can not be trained.
2. Training the architectures on large dataset requires heavy GPU resources. We trained our model on images of size 256x256 to reduce training time.
3. Training on RGB images instead of L*a*b leads to suboptimal results since the model needs to predict the three color channels.
4. The above challenges make end-to-end training computationally expensive.

Future Work

1. End-to-end training with multi-task loss
2. Ensembling multiple colorization specific loss functions
3. Training on diverse dataset with class rebalancing

Timeline (14 Feb - 18 Apr 2022)



Individual Contributions

Arjun Singh
21111402
arjuns21@iitk.ac.in

25%

Vgg16 Perceptual Loss

Evaluation Metrics

Debdeep Paul Chaudhuri
21111413
debdeeppc21@iitk.ac.in

25%

Novel Generator Architecture

Pre-training Generator

Himanshu Lal
21111403
himanshul21@iitk.ac.in

25%

Novel Generator Architecture

Experiments

Shivam Tripathi
21111408
shivamtr21@iitk.ac.in

25%

Base Model Implementation

Experiments

References

1. Colorizing black & white images with U-Net and conditional GAN — A Tutorial (link)

2. Pix2Pix - aladdinpersson (link)

3. SRGAN-PyTorch (link)

THANK YOU

Any Questions?