Colorization and Style Transfer of Images

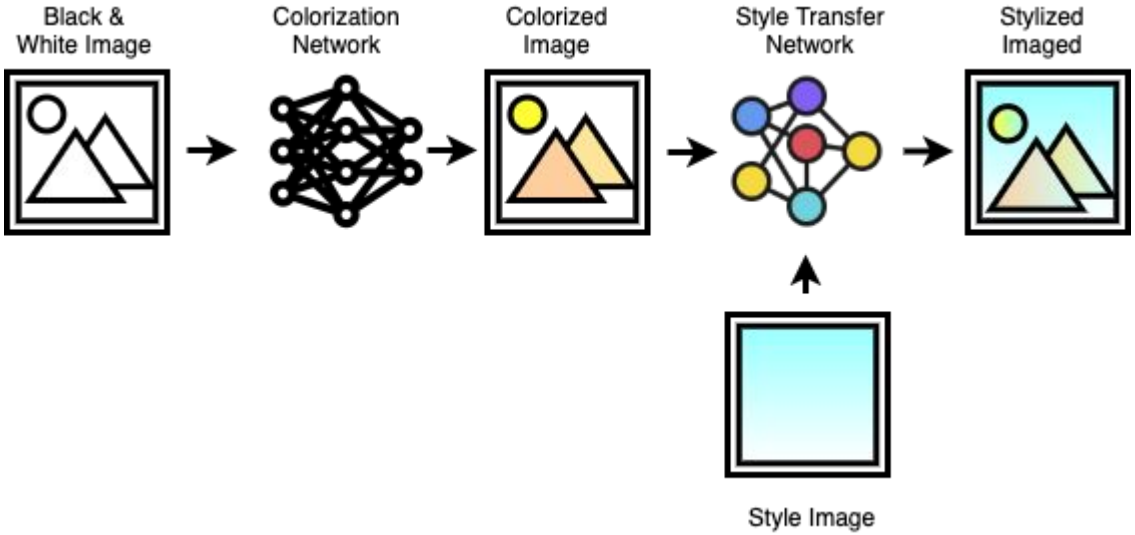
CS 6220 Big Data Sys & Analytics Demo Presentation

Group 15 : Kalyani Prasanna Jagdale, Karthik Nama Anil, Prajwal Mavinkere Revanna, Sakshi Mittal, Vandana Ramesh

https://github.com/KarthikNA/Colorisation-Style-Transfer-of-Images

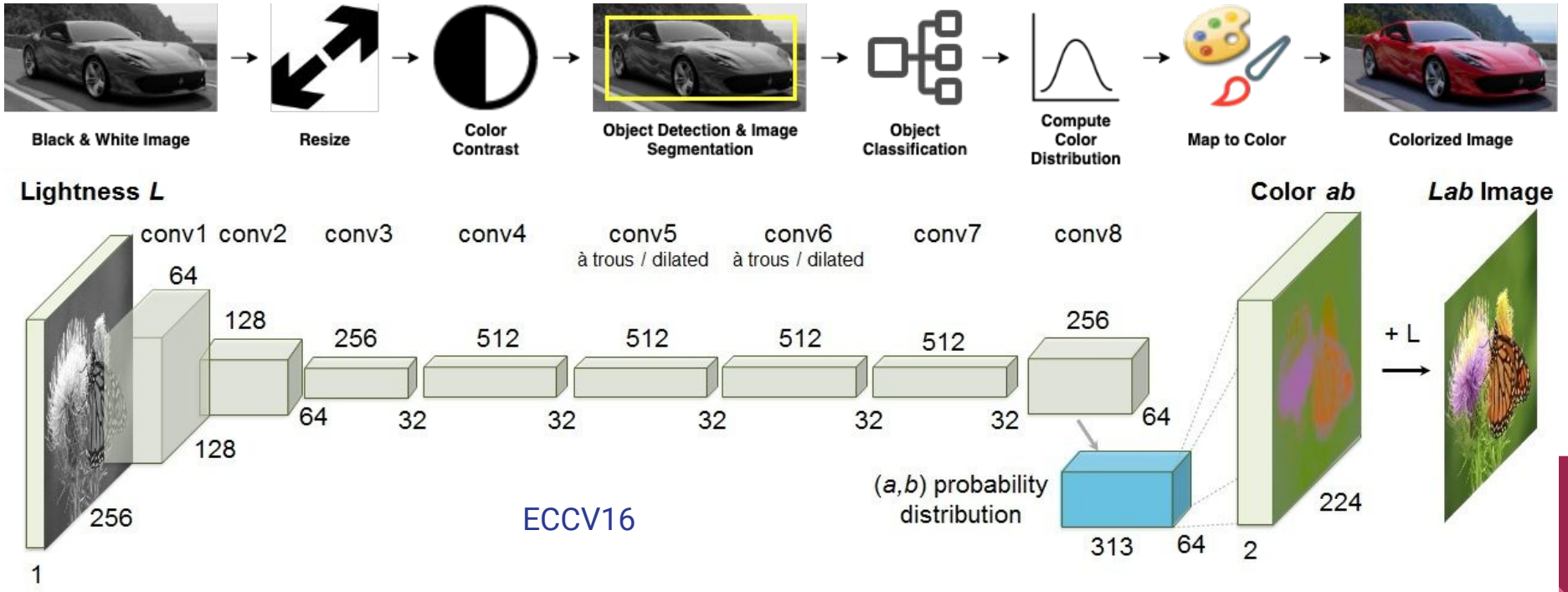
System Architecture

Karthik



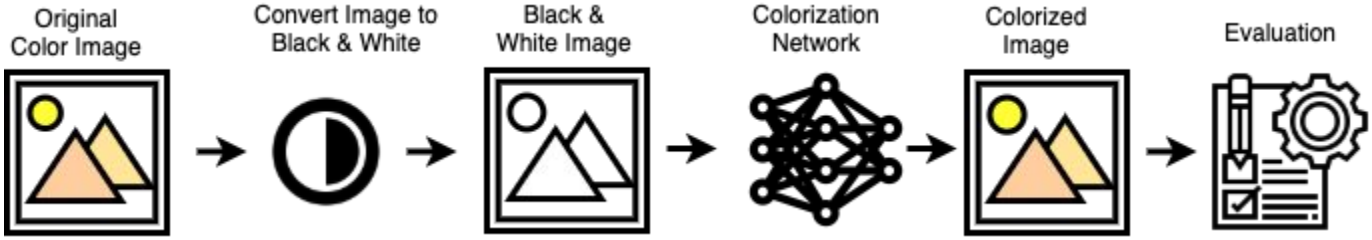
Colorisation Architecture

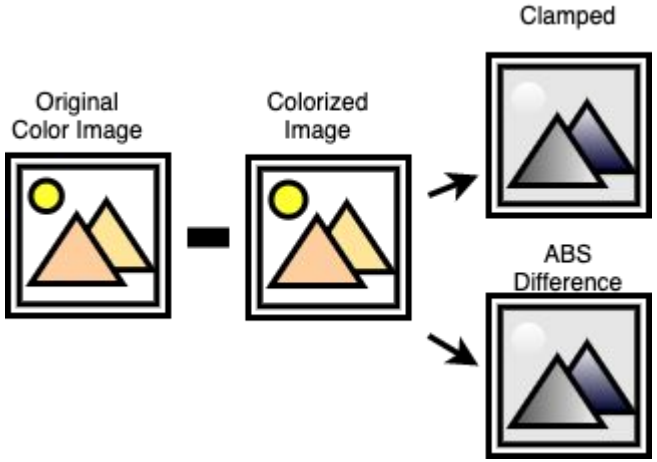
Karthik



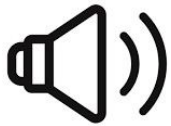
Colorization Processing Pipeline

Karthik





Colorization Code



Karthik

In [6]:
```
def colorize_image(image_filename=None, cv2_frame=None):
    """
    Where all the magic happens.  Colorizes the image provided. Can colorize either
    a filename OR a cv2 frame (read from a web cam most likely)
    :param image_filename: (str) full filename to colorize
    :param cv2_frame: (cv2 frame)
    :return: Tuple[cv2 frame, cv2 frame] both non-colorized and colorized images in cv2 format as a tuple
    """
    # load the input image from disk, scale the pixel intensities to the range [0, 1], and then convert the image from the BGR
to Lab color space
    image = cv2.imread(image_filename) if image_filename else cv2_frame
    scaled = image.astype("float32") / 255.0
    lab = cv2.cvtColor(scaled, cv2.COLOR_BGR2LAB)

    # resize the Lab image to 224x224 (the dimensions the colorization network accepts), split channels, extract the 'L' chann
el, and then perform mean centering
    resized = cv2.resize(lab, (224, 224))
    L = cv2.split(resized)[0]
    L -= 50

    # pass the L channel through the network which will *predict* the 'a' and 'b' channel values
    print("[INFO] colorizing image...")
    net.setInput(cv2.dnn.blobFromImage(L))
    ab = net.forward()[0, :, :, :].transpose((1, 2, 0))

    # resize the predicted 'ab' volume to the same dimensions as our input image
    ab = cv2.resize(ab, (image.shape[1], image.shape[0]))

    # grab the 'L' channel from the *original* input image (not the resized one) and concatenate the original 'L' channel with
the predicted 'ab' channels
    L = cv2.split(lab)[0]
    colorized = np.concatenate((L[:, :, np.newaxis], ab), axis=2)

    # convert the output image from the Lab color space to RGB, then clip any values that fall outside the range [0, 1]
    colorized = cv2.cvtColor(colorized, cv2.COLOR_LAB2BGR)
    colorized = np.clip(colorized, 0, 1)

    # the current colorized image is represented as a floating point data type in the range [0, 1] -- let's convert to an unsi
gned 8-bit integer representation in the range [0, 255]
    colorized = (255 * colorized).astype("uint8")
    return image, colorized


def convert_to_grayscale(frame):
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  # Convert webcam frame to grayscale
    gray_3_channels = np.zeros_like(frame)  # Convert grayscale frame (single channel) to 3 channels
    gray_3_channels[:, :, 0] = gray
    gray_3_channels[:, :, 1] = gray
    gray_3_channels[:, :, 2] = gray
    return gray_3_channels
```

Colorization Loss Function

Karthik

Cross Entropy Loss for Object Classification

$$L(\hat{\mathbf{Z}}, \mathbf{Z}) = -\frac{1}{HW} \sum_{h,w} v(\mathbf{Z}_{h,w}) \sum_{q} \mathbf{Z}_{h,w,q} \log(\hat{\mathbf{Z}}_{h,w,q})$$

Rarity weighting — Target distribution — Predicted distribution

Class Rebalancing to Encourage Rare Colors

$$v(\mathbf{Z}_{h,w}) = \mathbf{w}_{q^*}, \text{ where } q^* = \arg\max_q \mathbf{Z}_{h,w,q}$$

$$\mathbf{w} \propto \left((1-\lambda)\tilde{\mathbf{p}} + \frac{\lambda}{Q}\right)^{-1}, \quad \mathbb{E}[\mathbf{w}] = \sum_q \tilde{\mathbf{p}}_q \mathbf{w}_q = 1$$

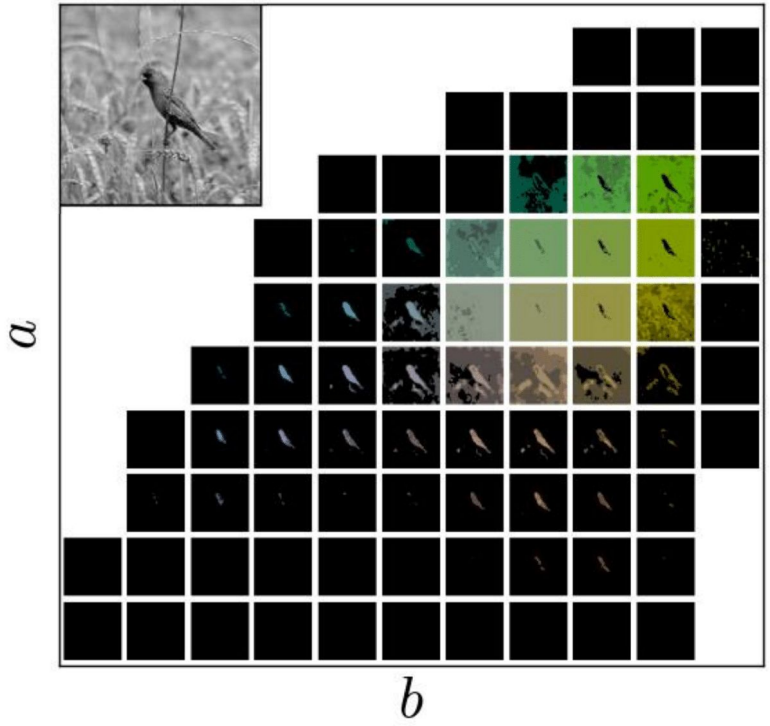
reweighting — empirical distribution — combine with uniform

Equations from Zhang R, Isola P, Efros AA. Colorful image colorization. European conference on computer vision 2016 Oct 8

Output of Colorization ab Layer

Karthik

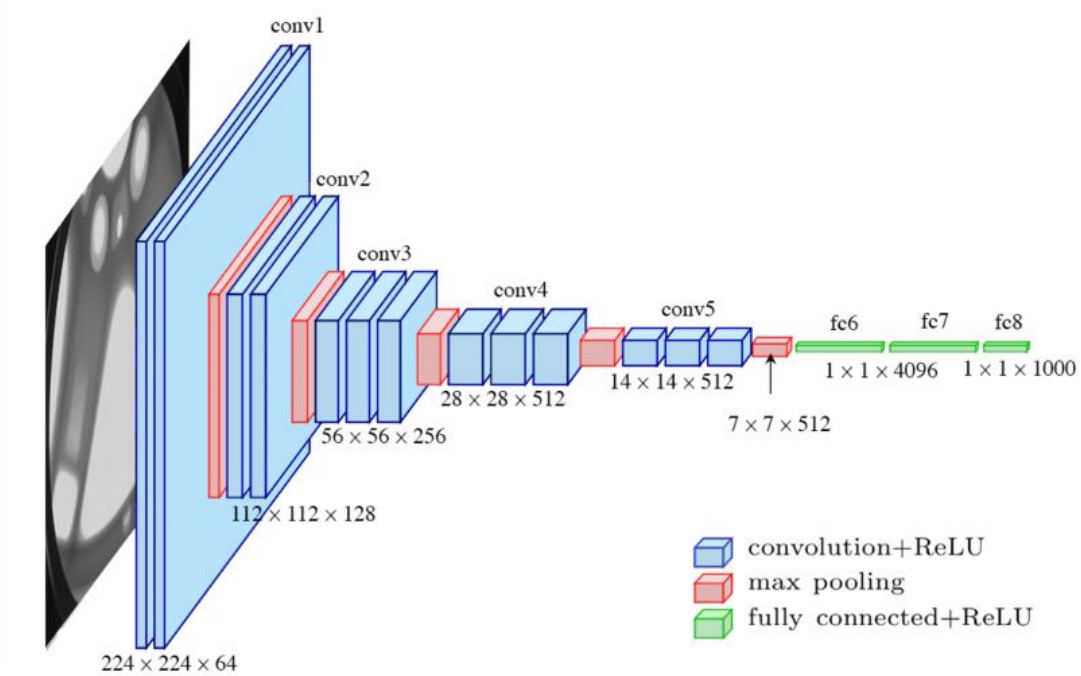
Best Colorization Determination in ab Layer



* Code to plot these graphs from Zhang R, Isola P, Efros AA. Colorful image colorization. European conference on computer vision 2016 Oct 8

Style Transfer Architecture

Sakshi



VGG16

- 3 x 3 convolution layers
- 2 x 2 size max pooling layers
- fully connected layers
- total 16 layers

Style Transfer Code Snippet

Sakshi

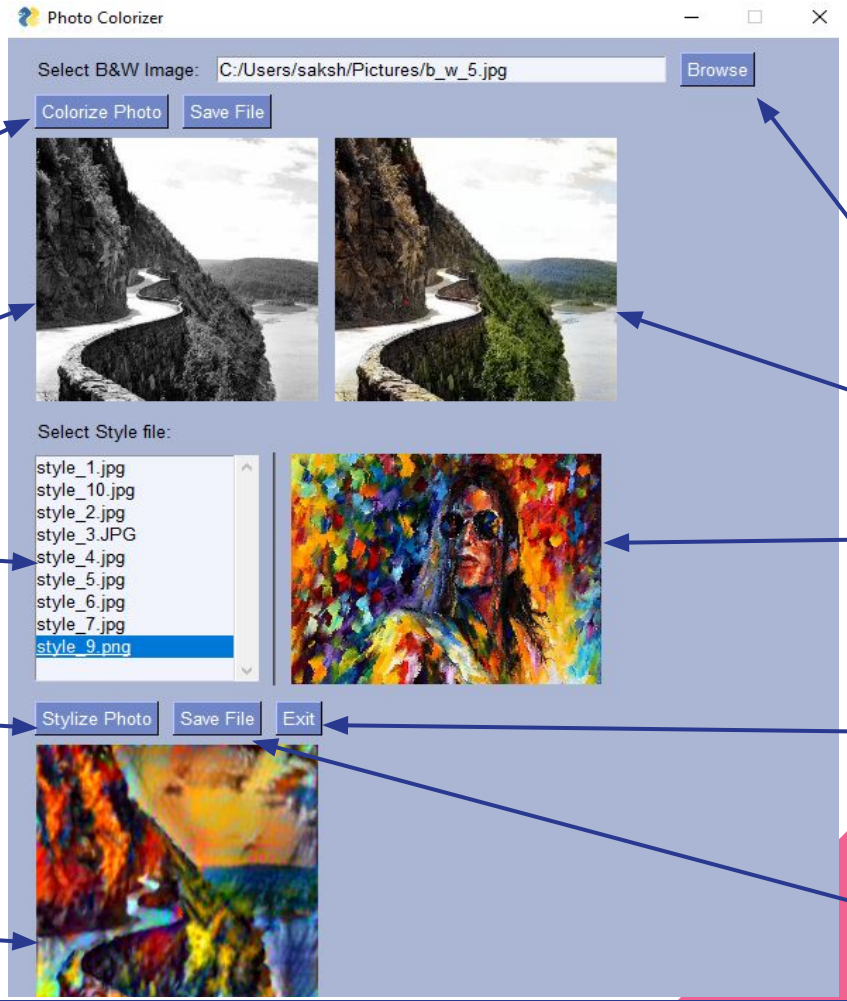
Number of epochs

```
def run_style_transfer(cnn, normalization_mean, normalization_std,
                       content_img, style_img, input_img, num_steps=250,
                       style_weight=1000000, content_weight=1):

    """Run the style transfer."""
    print('Building the style transfer model..')
    model, style_losses, content_losses = get_style_model_and_losses(cnn,
        normalization_mean, normalization_std, style_img, content_img)
    optimizer = get_input_optimizer(input_img)
    print('Optimizing..')
    run = [0]
    while run[0] <= num_steps:

        def closure():
            # correct the values of updated input image
            input_img.data.clamp_(0, 1)

            optimizer.zero_grad()
            model(input_img)
            style_score = 0
            content_score = 0

            for sl in style_losses:
                style_score += sl.loss
            for cl in content_losses:
                content_score += cl.loss

            style_score *= style_weight
            content_score *= content_weight

            loss = style_score + content_score
            loss.backward()

            run[0] += 1
            if run[0] % 50 == 0:
                print("run {}:".format(run))
                print('Style Loss : {:4f} Content Loss: {:4f}'.format(
                    style_score.item(), content_score.item()))
                print()

            return style_score + content_score

        optimizer.step(closure)

    # a last correction...
    input_img.data.clamp_(0, 1)
    return input_img
```

```
# losses
content_losses = []
style_losses = []

# assuming that cnn is a nn.Sequential, so we make a new nn.Sequential
# to put in modules that are supposed to be activated sequentially
model = nn.Sequential(normalization)

try:
    i = 0  # increment every time we see a conv
    for layer in cnn.children():
        if isinstance(layer, nn.Conv2d):
            i += 1
            name = 'conv_{}'.format(i)
        elif isinstance(layer, nn.ReLU):
            name = 'relu_{}'.format(i)
            # The in-place version doesn't play very nicely with the ContentLoss
            # and StyleLoss we insert below. So we replace with out-of-place
            # ones here.
            layer = nn.ReLU(inplace=False)
        elif isinstance(layer, nn.MaxPool2d):
            name = 'pool_{}'.format(i)
        elif isinstance(layer, nn.BatchNorm2d):
            name = 'bn_{}'.format(i)
        else:
            raise RuntimeError('Unrecognized layer: {}'.format(layer.__class__.__name__))

        model.add_module(name, layer)

        if name in content_layers:
            # add content loss:
            target = model(content_img).detach()
            content_loss = ContentLoss(target)
            model.add_module("content_loss_{}".format(i), content_loss)
            content_losses.append(content_loss)

        if name in style_layers:
            # add style loss:
            target_feature = model(style_img).detach()
            style_loss = StyleLoss(target_feature)
            model.add_module("style_loss_{}".format(i), style_loss)
            style_losses.append(style_loss)
except Exception as e:
    print(e)

# now we trim off the layers after the last content and style losses
for i in range(len(model) - 1, -1, -1):
    if isinstance(model[i], ContentLoss) or isinstance(model[i], StyleLoss):
        break

model = model[:(i + 1)]

return model, style_losses, content_losses
```

UI Design

Sakshi



Step 2: Click to Colorize

Selected Black & White Image

Step 3: Select from a set of Style Images

Step 4: Click to apply stylize the Colorized Image

Stylization Result : Stylized Image using our model

Step 1: Browse a Black & White Image

Colorization Result: Colorized Image from our model

Selected Style Image

Step 5: Click to Exit the GUI

Save the Image on system

UI Code (Implemented using PySimpleGUI)

Sakshi

```
# First the window layout...2 columns
images_col = [[sg.Text('Select B&W Image:'), sg.In(enable_events=True, key='-IN FILE-'), sg.FileBrowse()],
              [sg.Button('Colorize Photo', key='-PHOTO-'), sg.Button('Save File', key='-SAVE-')],
              [sg.Image(filename='', key='-IN-', size=(5,5)), sg.Image(filename='', key='-OUT-', size=(5,5))]]

style_col = [[sg.Button('Stylize Photo', key='-STYLEPHOTO-'), sg.Button('Save File', key='-SAVE STYLE-'), sg.Button('Exit')],
             [sg.Image(filename='', key='-STYLE OUT-', size=(20,20))]]

image_list_column = [
    [sg.Text('Select Style File:')],
        [
        sg.Listbox(
            values=[], enable_events=True, size=(20, 10), key="-FILE LIST-"
        ), sg.VSeperator(), sg.Text("Choose a style image from list on left:", key='-TEXT-'), sg.Image(key="-IMAGE-")
    ]
]


# ----- Full Layout -----
layout = [[sg.Column(images_col, key='-COLORIZE-')], [sg.Column(image_list_column, key="-IMAGE LIST-", visible=False)],
          [sg.Column(style_col, key='-STYLE-', visible=False)]]

# ----- Make the window -----
window = sg.Window('Image Colorization & Style Transfer', layout, grab_anywhere=True, location=(0,0), size=(600,800), keep_on_t

# ----- Run the Event Loop -----
prev_filename = colorized = cap = None
prev_stylefile = stylized = None
while True:
    event, values = window.read()
    if event in (None, 'Exit'):
        break
    if event == '-PHOTO-':          # Colorize photo button clicked
        print('colorize image')
        try:
            if values['-IN FILE-']:
                filename = values['-IN FILE-']
                print('Filename :'+filename)
            elif values['-FILE LIST-']:
                filename = os.path.join(values['-FOLDER-'], values['-FILE LIST-'][0])
            else:
                continue
            image, colorized = colorize_image(filename)
            cv2.imwrite("colorized.png", colorized)
            image = cv2.resize(image, (200, 200))
            colorized = cv2.resize(colorized, (200, 200))
            window['-IN-'].update(data=cv2.imencode('.png', image)[1].tobytes())
            window['-OUT-'].update(data=cv2.imencode('.png', colorized)[1].tobytes())
            try:
                # Get list of files in folder
                file_list = os.listdir(folder)
            except:
                file_list = []

            fnames = [
                f
                for f in file_list
                if os.path.isfile(os.path.join(folder, f))
                and f.lower().endswith((".jpg", ".jpeg", ".png"))
            ]

            window["-FILE LIST-"].update(fnames)
            window['-IMAGE LIST-'].update(visible=True)
            window['-STYLE-'].update(visible=True)
        except Exception as e:
            print(e)
            continue
    elif event == '-STYLEPHOTO-':          # Stylize photo button clicked
        try:
            if values["-FILE LIST-"]:
                filename = os.path.join(folder, values["-FILE LIST-"][0])
            else:
                continue
            print(filename)
```

```
    elif event == '-STYLEPHOTO-':          # Stylize photo button clicked
        try:
            if values["-FILE LIST-"]:
                filename = os.path.join(folder, values["-FILE LIST-"][0])
            else:
                continue
            print(filename)
            stylized = start_style_transfer(filename, "colorized.png")
            print("Image stylized")
            stylized = cv2.imread("stylized_image.png")
            stylized = cv2.resize(stylized, (200, 200))
            #window['-STYLE IN-'].update(data=cv2.imencode('.png', image)[1].tobytes())
            window["-STYLE OUT-"].update(data=cv2.imencode('.png', stylized)[1].tobytes())
        except Exception as e:
            print(e)
            continue
    elif event == '-IN FILE-':      # A single filename was chosen
        filename = values['-IN FILE-']
        if filename != prev_filename:
            prev_filename = filename
            try:
                image = cv2.imread(filename)
                image = cv2.resize(image, (200, 200))
                window['-IN-'].update(data=cv2.imencode('.png', image)[1].tobytes())
            except:
                continue
    elif event == '-IN STYLE FILE-':      # A single style filename was chosen
        filename = values['-IN STYLE FILE-']
        if filename != prev_stylefile:
            prev_stylefile = filename
            try:
                image = cv2.imread(filename)
                image = cv2.resize(image, (200, 200))
                window['-STYLE IN-'].update(data=cv2.imencode('.png', image)[1].tobytes())
            except:
                continue
    elif event == '-SAVE-' and colorized is not None:   # Clicked the Save File button
        filename = sg.popup_get_file('Save colorized image.\nColorized image be saved in format matching the extension you enter
        try:
            if filename:
                cv2.imwrite(filename, colorized)
                sg.popup_quick_message('Image save complete', background_color='red', text_color='white', font='Any 16')
        except:
            sg.popup_quick_message('ERROR - Image NOT saved!', background_color='red', text_color='white', font='Any 16')
    elif event == '-SAVE STYLE-' and stylized is not None:   # Clicked the Save File button
        filename = sg.popup_get_file('Save Stylized image.\Stylized image be saved in format matching the extension you enter.',
        try:
            if filename:
                cv2.imwrite(filename, stylized)
                sg.popup_quick_message('Image save complete', background_color='red', text_color='white', font='Any 16')
        except:
            sg.popup_quick_message('ERROR - Image NOT saved!', background_color='red', text_color='white', font='Any 16')
    elif event == "-FOLDER-":
        folder = values["-FOLDER-"]
        try:
            # Get list of files in folder
            file_list = os.listdir(folder)
        except:
            file_list = []

        fnames = [
            f
            for f in file_list
            if os.path.isfile(os.path.join(folder, f))
            and f.lower().endswith((".jpg", ".jpeg", ".png"))
        ]

        window["-FILE LIST-"].update(fnames)
    elif event == "-FILE LIST-":  # A file was chosen from the listbox
        try:
            filename = os.path.join(
                folder, values["-FILE LIST-"][0]
            )
            window["-TEXT-"].update(visible=False)
            image = cv2.imread(filename)
```

Colorization Examples

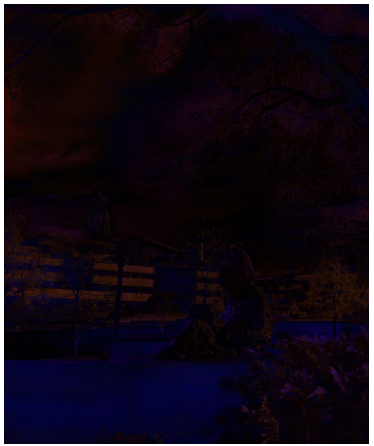
Vandana

Black & White Image

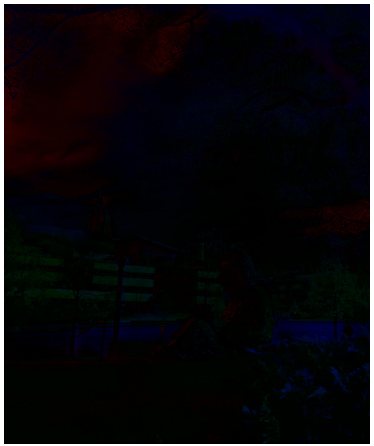
Generated Color Image

Original Color Image

Clamped

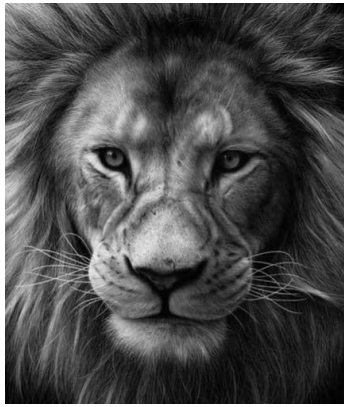
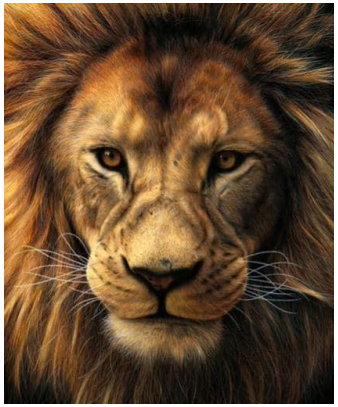
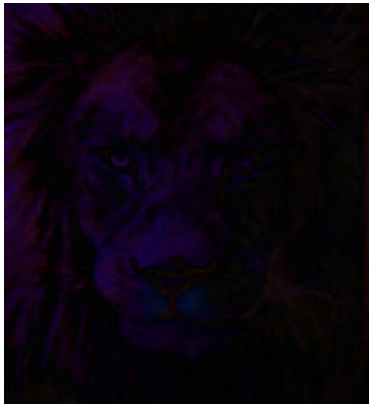
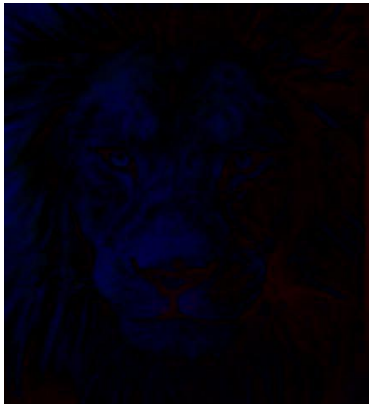


ABS Diff



Colorization Examples

Vandana

Black & White Image	Generated Color Image	Original Color Image	ABS Diff	Clamped
				

Colorization Examples

Vandana

Black & White Image

Generated Color Image

Original Color Image

ABS Diff

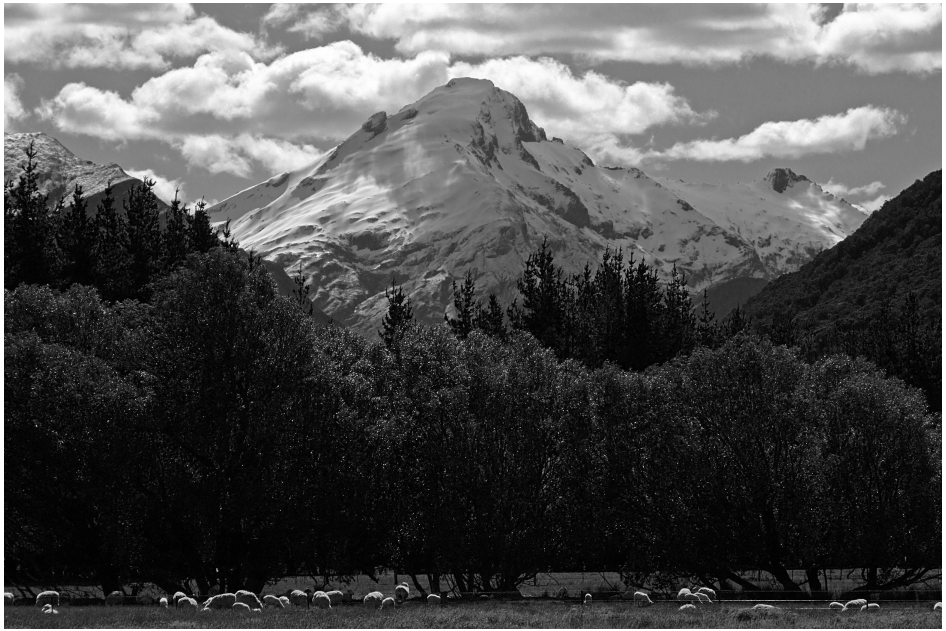
Clamped

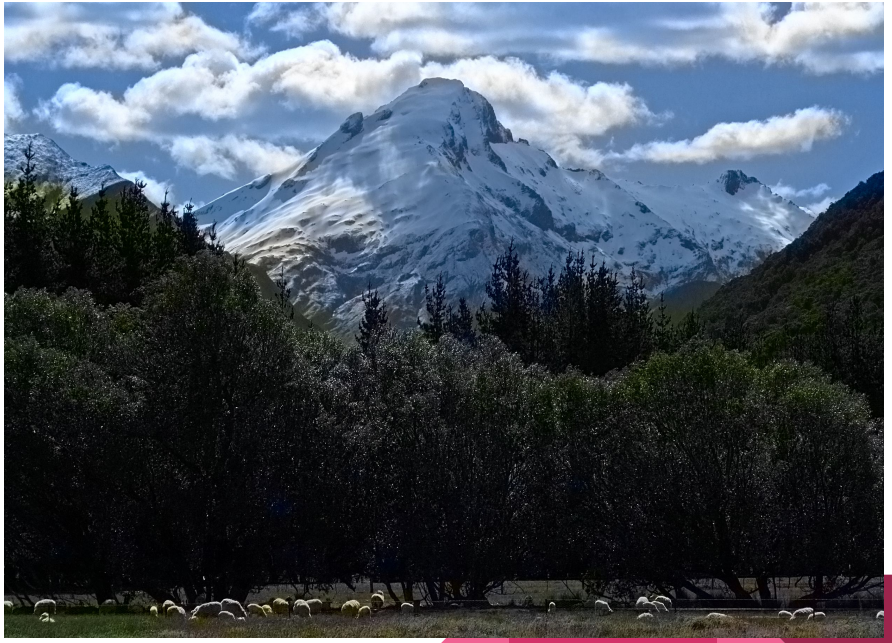
Example - Challenging Colorization

Vandana

Examples - Medium Colorization

Vandana





Examples - Good Colorization

Vandana

Examples - Good Colorization

Vandana

Colorization Models Comparison

Sakshi

Black & White Image

Our Model Image

SIGGRAPH17

Deoldify Image

Colorization Models Comparison

Sakshi

Black & White Image

Our Model Image

SIGGRAPH17

Deoldify Image

Colorization Models Comparison

Sakshi

Black & White Image

Our Model Image

SIGGRAPH17

Deoldify Image

Colorization Models Comparison

Sakshi

Black & White Image

Our Model Image

SIGGRAPH17

Deoldify Image

Colorization Models Comparison

Sakshi

Black & White Image

Our Model Image

SIGGRAPH17

Deoldify Image

Evaluation - Colorization Turing Test

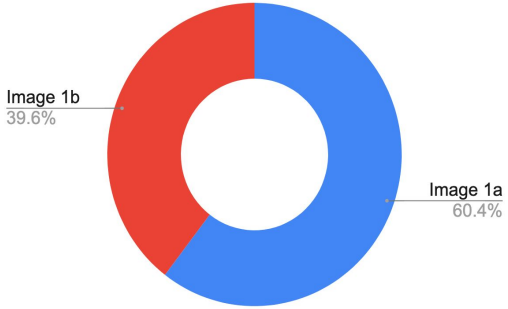
Vandana

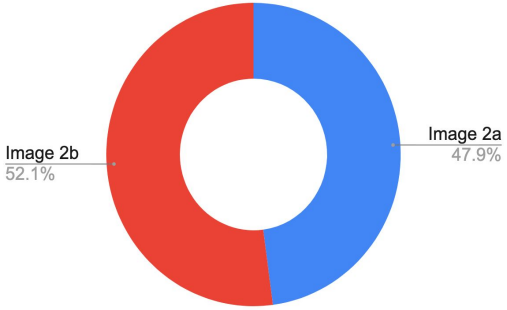
1a Colorized

1b Original

2a Colorized

2b Original





Colorization Turing Test

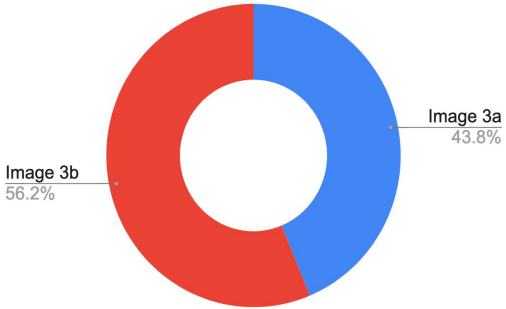
Vandana

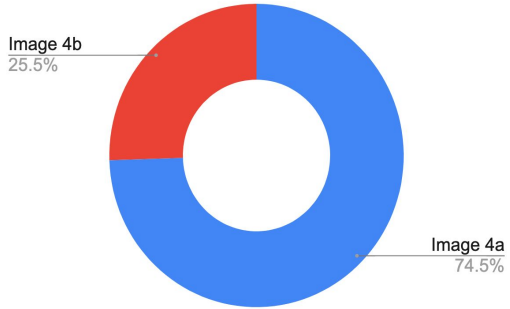
3a Colorized

3b Original

4a Colorized

4b Original





Colorization Turing Test

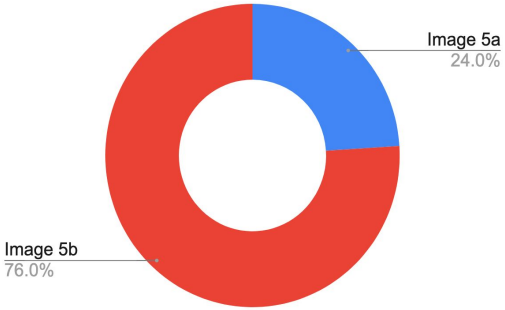
Vandana

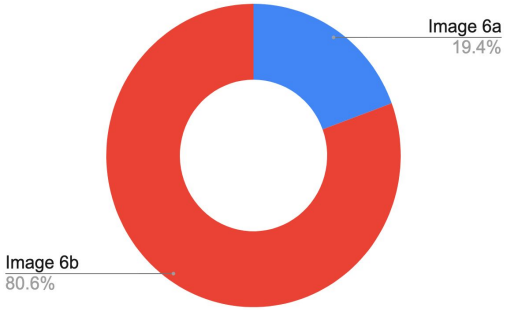
5a Original

5b Colorized

6a Original

6b Colorized





Style Transfer Examples

Prajwal

Original Image

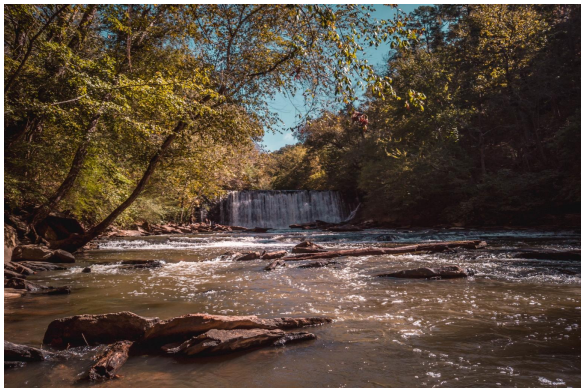
Style Image

Stylized Image

Style Transfer Examples

Prajwal

Original Image



Style Image

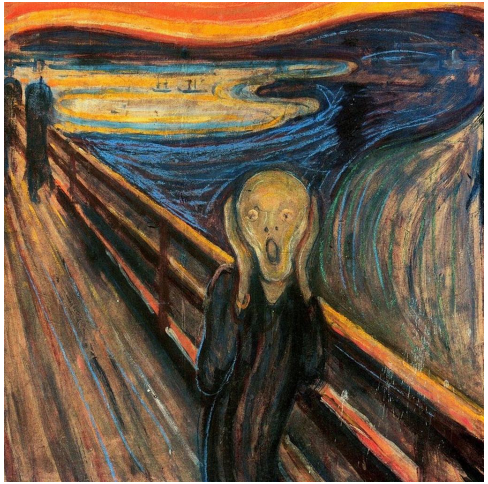
Stylized Image

Style Transfer Examples

Prajwal

Original Image

Style Image



Stylized Image

Style Transfer Analysis

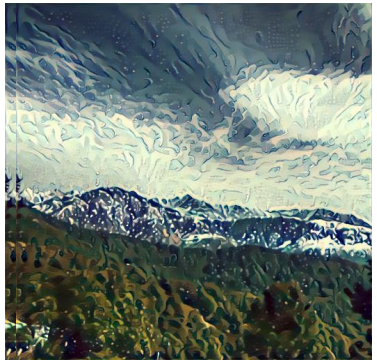
Prajwal

Original Image

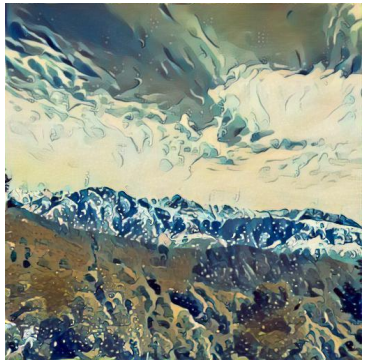
Style Image

Hyper parameter Optimization(VGG16)

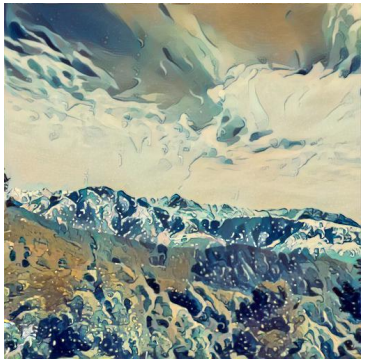
Prajwal



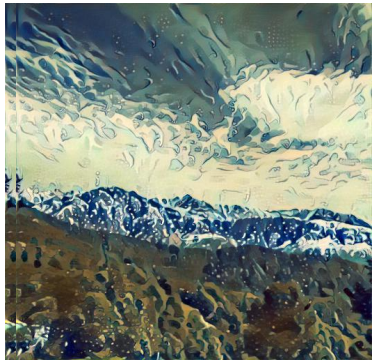
100 Epochs



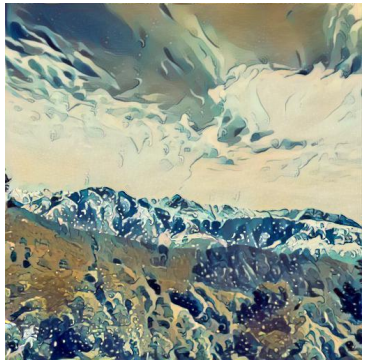
500 Epochs



900 Epochs



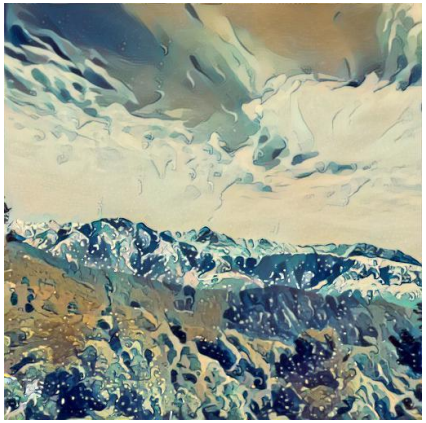
300 Epochs



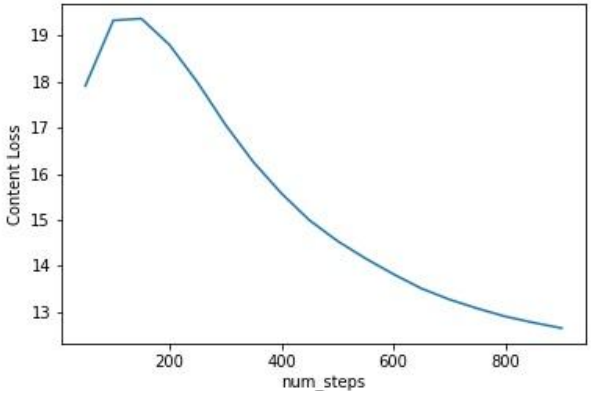
700 Epochs

Evaluation for Style Transfer - Losses(900 Epochs)

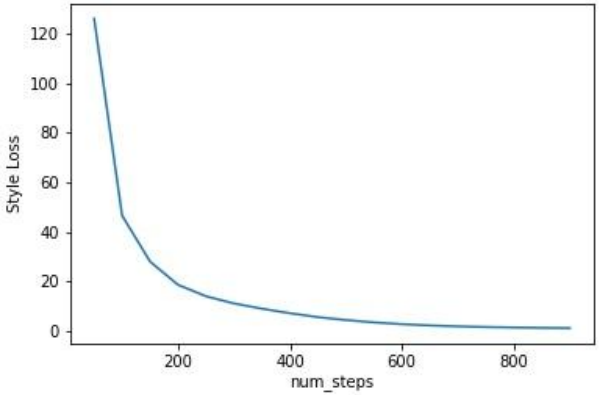
Prajwal



900 Epochs



Style Loss vs Epochs



Content Loss vs Epochs

VGG16

Style Transfer Analysis

Prajwal

Original Image

Style Image

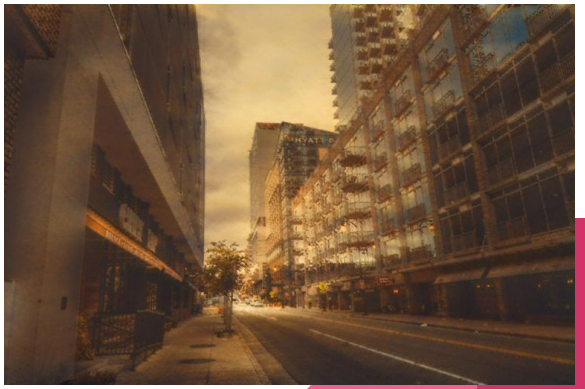
Style Transfer Comparison(900 Epochs)

Prajwal

AlexNet

VGG16

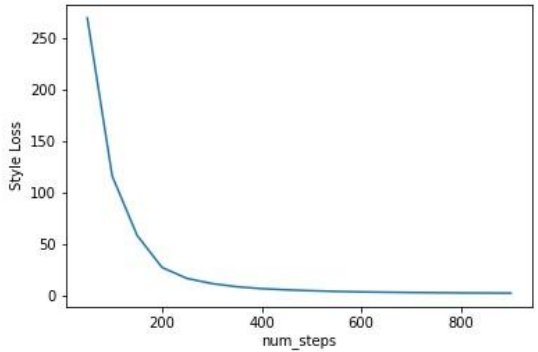
VGG11



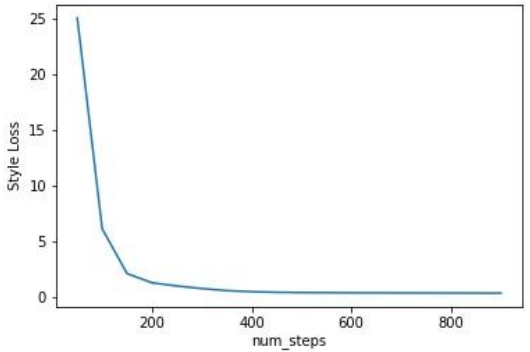
VGG19

Style Loss Comparison(900 Epochs)

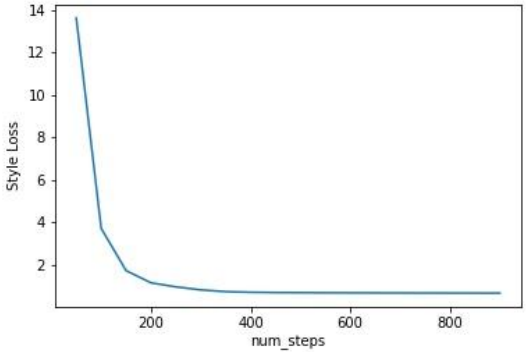
Prajwal



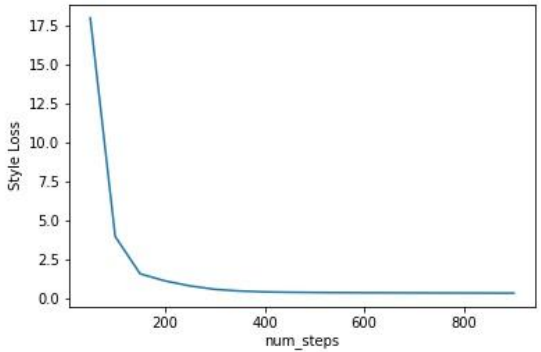
AlexNet



VGG16



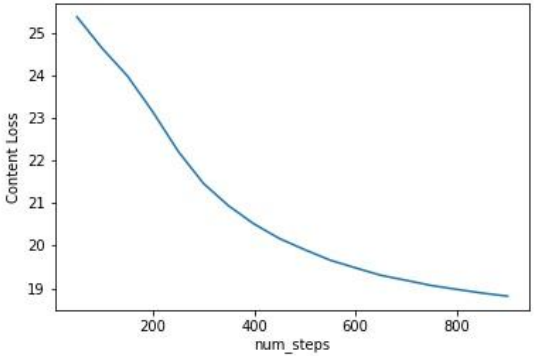
VGG11



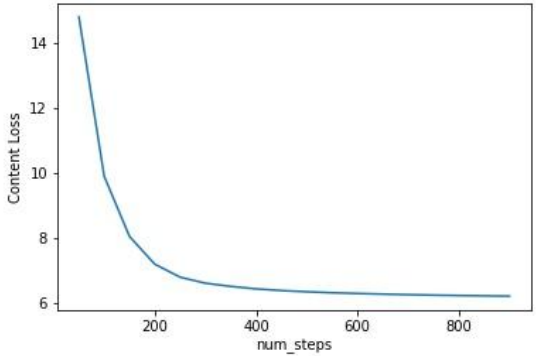
VGG19

Content Loss Comparison(900 Epochs)

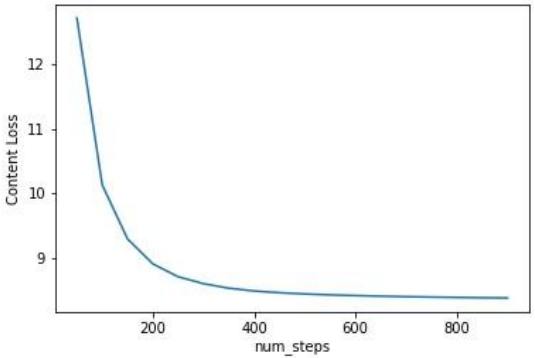
Prajwal



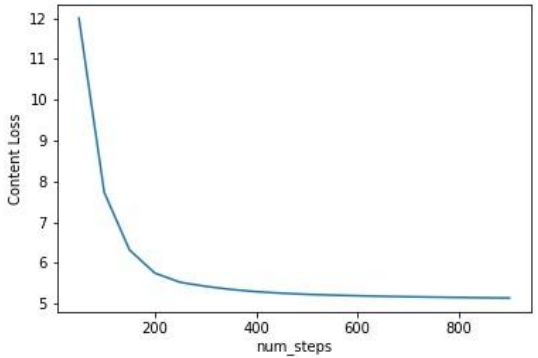
AlexNet



VGG16



VGG11



VGG19

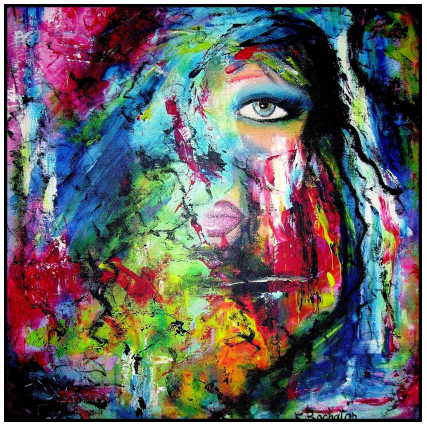
Colorization + Style Transfer Examples

Kalyani

Black & White image

Colorized Image

Style Template



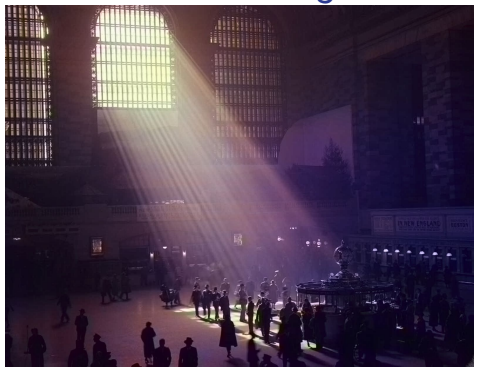
Stylized Result

Colorization + Style Transfer Examples

Kalyani

Black & White Image

Colorized Image



Style Template

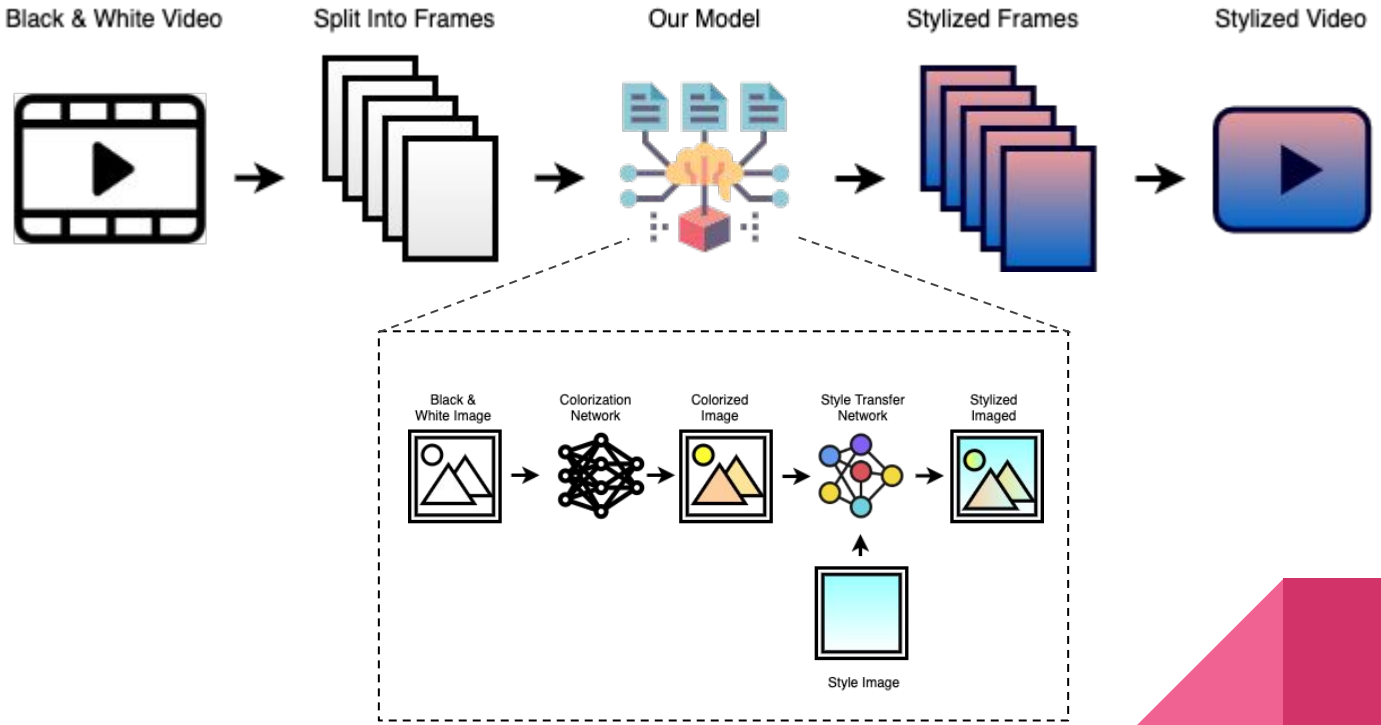
Stylized Result

Application: Digital Art

Kalyani

Video Stylization - Stretch Goal

Kalyani



Video Stylization Code Snippet

Kalyani

```
[ ]  from torchvision.utils import save_image
     import torch.nn.functional as nnf

     device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
     imsize = 512

     loader = transforms.Compose([
         transforms.Resize([imsize, imsize]),  # scale imported image
         transforms.ToTensor()])  # transform it into a torch tensor


     def image_loader(image_name):
         image = Image.open(image_name).convert('RGB')
         # fake batch dimension required to fit network's input dimensions
         image = loader(image).unsqueeze(0)
         # print(image.size())
         return image.to(device, torch.float)

     style_img = image_loader("/content/drive/My Drive/BigDataProject/style_image.jpg")

     count = 0
     while True:
       cnn = models.vgg19(pretrained=True).features.to(device).eval()
       content_layers_default = ['conv_4']
       style_layers_default = ['conv_1', 'conv_2', 'conv_3', 'conv_4', 'conv_5']
       normalization_mean = torch.tensor([0.485, 0.456, 0.406]).to(device)
       normalization_std = torch.tensor([0.229, 0.224, 0.225]).to(device)
       content_image = image_loader('/content/drive/MyDrive/BigDataProject/fox_frames/frame_%d.jpeg' % count)
       input_img = content_image.clone()
       image = run_style_transfer(cnn, normalization_mean, normalization_std,
                           content_image, style_img, input_img, num_steps=200,
                           style_weight=1000000, content_weight=1)

       save_image(image,"/content/drive/My Drive/BigDataProject/fox_try2/fox_style_%d.jpg" % count)     # save frame as JPEG file
       count += 1
       print("On image %d" % count)


     Building the style transfer model..
     Optimizing..
```

```
# converts video to image frames
import cv2
vidcap = cv2.VideoCapture('/content/drive/My Drive/BigDataProject/lion.mp4')
success,image = vidcap.read()
count = 1
while success:
  # save frame as JPG file
  cv2.imwrite("/content/drive/My Drive/BigDataProject/frames/image_frame%d.jpg" % count, image)
  success, image = vidcap.read()
  count+=1
```

```
import cv2
import numpy as np
from PIL import Image

# choose codec according to format needed
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
video=cv2.VideoWriter('stylized_fox.avi', fourcc, 24,(1024,1024))

for j in range(0,353):
   img = cv2.imread('/content/drive/My Drive/BigDataProject/fox_try2/fox_style_'+str(j)+'.jpg')
   frame = cv2.resize(img, (1024,1024))
   video.write(frame)

cv2.destroyAllWindows()
video.release()
```

Experiments with Video Stylization

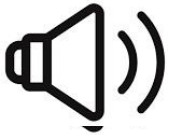
Kalyani

Original Video

Style Image

Stylized Video

Experiments with Video Stylization



Kalyani

- Model was run Google Colab with GPU enabled
- Stylization was run for 200 epochs for each frame
- Style Weight = 1M
- Content Weight = 10
- Video Length = 14 seconds

352

Frames

21

fps

3607

Seconds to process video

Frame Splitting = 1.1 seconds

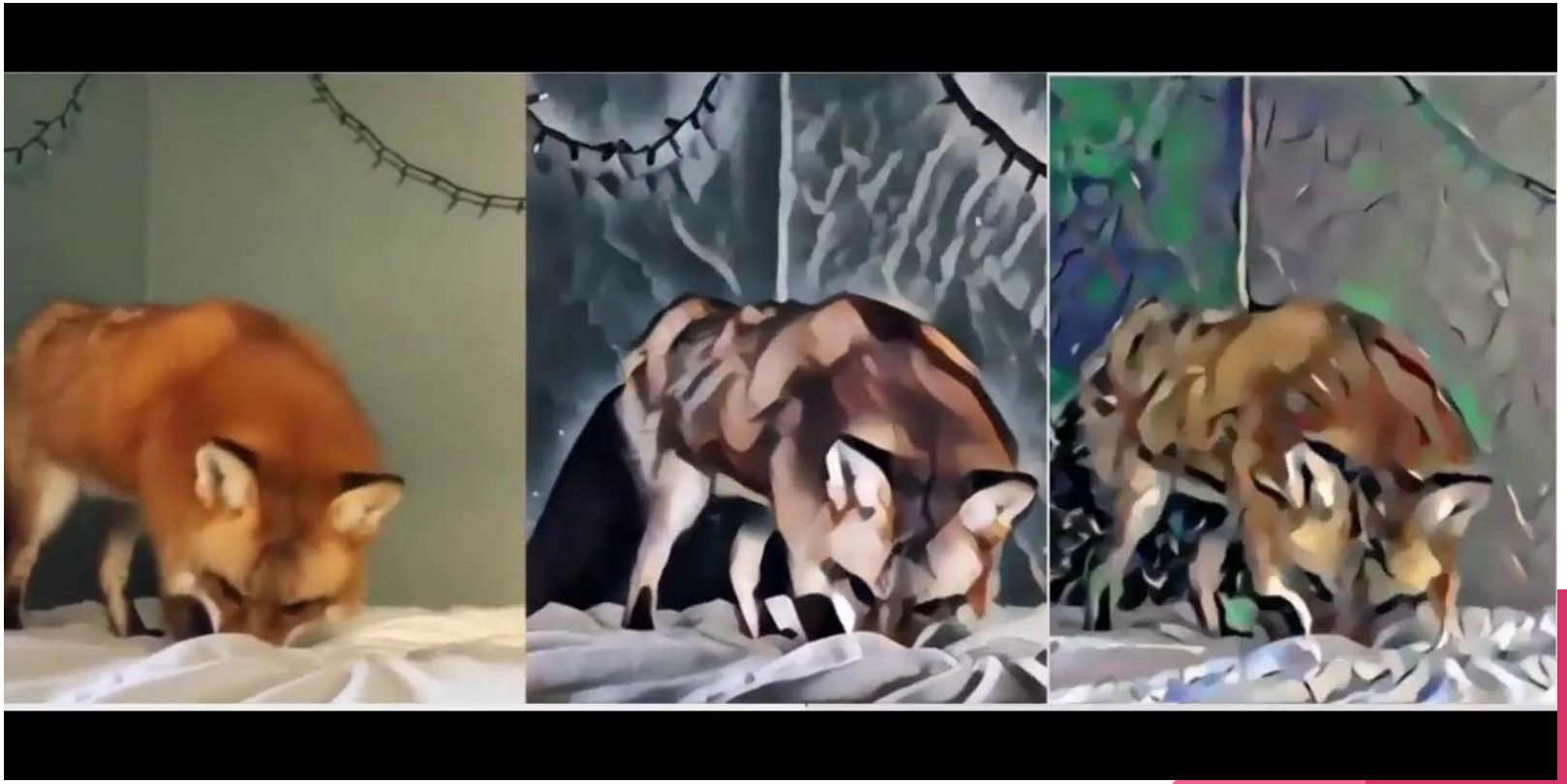
Frame Recombination = 1.3 seconds

Avg Per Frame Processing Time = 10.79 seconds

- Colorization = 1.4 seconds
- Stylization = 9.4 seconds

Comparison with Baseline

Kalyani

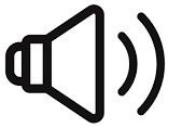


Ref: https://github.com/lengstrom/fast-style-transfer

Video Stylization Example

Kalyani

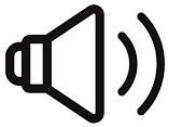
Areas of Improvement



Karthik

- Train the colorization model on larger dataset
- Introducing inter frame correlation in videos to minimise unwanted artifacts
- Content / Style trade off

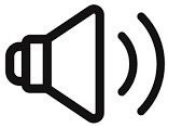
Conclusion and Future Work



Karthik

- Colorization network can be adapted to perform
  - Object detection
  - Object classification
  - Image segmentation
  - Image augmentation
- Compare the output of our model with the outputs of deeper networks employing conditional GANs and VAEs
- Modify the models to run on lower compute devices like mobiles and tablets

Thank you



Karthik