Answers	Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Subject Teacher
5	5	5	5	20	

Start Date :..... Date of Completion:.....

Deep Learning Mini Project -3

Title - Implement Colorizing Old B&W Images: color old black and white images to colorful images

Project title: Colorizing Old B&W Images using CNN

Objective: To build a deep neural network model that can Colorizing Old B&W Images: color old black and white images to colorful images using CNN

Theory - Colorizing black and white images to colorful images involves a complex process that requires expertise and specialized software. However, here are some general steps involved in the process:

Scan the black and white image: The first step is to scan the black and white image and convert it into a digital format.

Preprocess the image: The image needs to be preprocessed to remove any scratches, dust, or other defects. This can be done using image editing software like Photoshop or GIMP.

Convert the image to grayscale: The black and white image needs to be converted to grayscale. This can be done using image editing software or programming languages like Python.

Collect training data: The next step is to collect training data for the colorization model. This can include a dataset of colorful images with their corresponding grayscale versions.

Train the colorization model: A deep learning model can be trained to colorize grayscale images using a dataset of colorful images. This model can be trained using software like TensorFlow, PyTorch, or Keras.

Apply the colorization model to the black and white image: Once the model is trained, it can be applied to the black and white image to generate a colorized version. This can be done using programming languages like Python.

Refine the colorized image: The colorized image may need some manual refinement to ensure that the colors are accurate and the image looks natural. This can be done using image editing software like Photoshop or GIMP.

Save the final image: Once the image has been colorized and refined, it can be saved in a digital format for printing or online use.

Note that the quality of the colorized image will depend on the quality of the original black and white image, the accuracy of the colorization model, and the manual refinement process.

Source Code-

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, UpSampling2D, InputLayer, BatchNormalization
from tensorflow.keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
import numpy as np
# Load the grayscale image
img_gray = load_img('bw_image.jpg', grayscale=True)
# Resize the image to the desired size
img_gray = img_gray.resize((256,256))
# Convert the image to a numpy array
img_gray = img_to_array(img_gray)
# Normalize the image
img_gray = img_gray / 255.0
```

```
# Add a new dimension to the array to make it compatible with the input shape of the model
img_gray = np.expand_dims(img_gray, axis=0)
# Load the pre-trained model
model = Sequential()
model.add(InputLayer(input_shape=(None, None, 1)))
model.add(Conv2D(64, (3,3), activation='relu', padding='same', strides=2))
model.add(Conv2D(128, (3,3), activation='relu', padding='same'))
model.add(Conv2D(128, (3,3), activation='relu', padding='same', strides=2))
model.add(Conv2D(256, (3,3), activation='relu', padding='same'))
model.add(Conv2D(256, (3,3), activation='relu', padding='same', strides=2))
model.add(Conv2D(512, (3,3), activation='relu', padding='same'))
model.add(Conv2D(512, (3,3), activation='relu', padding='same'))
model.add(Conv2D(256, (3,3), activation='relu', padding='same'))
model.add(UpSampling2D((2,2)))
model.add(Conv2D(128, (3,3), activation='relu', padding='same'))
model.add(UpSampling2D((2,2)))
model.add(Conv2D(64, (3,3), activation='relu', padding='same'))
model.add(Conv2D(32, (3,3), activation='relu', padding='same'))
model.add(Conv2D(2, (3, 3), activation='tanh', padding='same'))
model.add(UpSampling2D((2, 2)))
model.compile(optimizer='adam', loss='mse')
# Load the pre-trained weights
model.load_weights('colorization_weights.h5')
# Colorize the grayscale image
img_colorized = model.predict(img_gray)
```

```
# Save the colorized image

img_colorized = img_colorized * 128 + 128

img_colorized = np.clip(img_colorized, 0, 255).astype('uint8')

img_colorized = array_to_img(img_colorized[0])

img_colorized.save('colorized_image.jpg')
```

Conclusion- In this way coloring B & W images Implemented.

Assignment Question

1. Explain VGG CNN Model?
2. Explain Batch Normalization?
3. Explain UpSampling2D?