Min Hyeok Lee / Anton Liu / Vladislav Trukhin
How to run script:
code: Colab_code.py
script:

```
if __name__ == '__main__':
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    print(device)
    coco_path = untar_data(URLs.COCO_SAMPLE)
    coco_path = str(coco_path) + "\\train_sample"
    #coco_path = "C:\\Users\\MinHyeok\\.fastai\\data\\coco_sample\\train_sample"
    SIZE = 256

    paths = glob.glob(coco_path + "/*.jpg") # Grabbing all the image file names
    np.random.seed(123)
    paths_subset = np.random.choice(paths, 10_000, replace=False) # choosing 1000 images randomly
    rand_idxs = np.random.permutation(10_000)
    train_idxs = rand_idxs[:8000] # choosing the first 8000 as training set
    val_idxs = rand_idxs[8000:] # choosing last 2000 as validation set
    train_paths = paths_subset[train_idxs]
    val_paths = paths_subset[val_idxs]
    print(len(train_paths), len(val_paths))

    train_dl = make_dataloaders(paths=train_paths, split='train')
    val_dl = make_dataloaders(paths=val_paths, split='val')

    data = next(iter(train_dl))
    Ls, abs_ = data['L'], data['ab']
    print(len(train_dl), len(val_dl))

    model = MainModel()
    train_model(model, train_dl, 100)
```

Running script will automatically download and save the path into variable “coco_path”
Once you download you can comment the line “coco_path = untar_data(URLs.COCO_SAMPLE)” and the line below and change the variable coco_path to the actual path on your local.

Changing Model

```
if net_G is None:
    self.net_G = init_model(Unet(input_c=1, output_c=2, n_down=8, num_filters=64), self.device)
    #self.net_G = init_model(ECCVGenerator(), self.device)
else:
    self.net_G = net_G.to(self.device)
```

comment and uncomment line to init_model with desired network architecture (U-Net, ECCVGen).

Hyper parameters
1. Learning rate change

```
class MainModel(nn.Module):
    def __init__(self, net_G=None, lr_G=5e-4, lr_D=5e-4,
                 beta1=0.5, beta2=0.999, lambda_L1=100.):
        super().__init__()

        self.device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
        self.lambda_L1 = lambda_L1
```

learning rate can be changed by changing “lr_G” and “lr_D” in MainModel initialization. Default is 5e-4

2. Batch Size

```
def make_dataloaders(batch_size=16, n_workers=2, pin_memory=True, **kwargs): # A handy function to make our
    dataset = ColorizationDataset(**kwargs)
    dataloader = DataLoader(dataset, batch_size=batch_size, num_workers=n_workers, pin_memory=pin_memory)
    return dataloader
```

Batch size tests are done by changing batch_size in the make_dataloaders
Default is 16.

3. weight initialization

```
def init_weights(net, init='norm', gain=0.02):

    def init_func(m):
        classname = m.__class__.__name__
        if hasattr(m, 'weight') and 'Conv' in classname:
            if init == 'norm':
                nn.init.normal_(m.weight.data, mean=0.0, std=gain)
            elif init == 'xavier':
                nn.init.xavier_normal_(m.weight.data, gain=gain)
            elif init == 'kaiming':
                nn.init.kaiming_normal_(m.weight.data, a=0, mode='fan_in')

            if hasattr(m, 'bias') and m.bias is not None:
                nn.init.constant_(m.bias.data, 0.0)
        elif 'BatchNorm2d' in classname:
            nn.init.normal_(m.weight.data, 1., gain)
            nn.init.constant_(m.bias.data, 0.)

    net.apply(init_func)
    print(f"model initialized with {init} initialization")
    return net
```

Change weight initialization by changing init=”norm” to appropriate weight in “init_weights” function

4. Number of Layer
4.1 Zhang et al model [2016]

```
def __init__(self, norm_layer=nn.BatchNorm2d):
    super(ECCVGenerator, self).__init__()

    model1=[nn.Conv2d(1, 64, kernel_size=3, stride=1, padding=1, bias=True),]
    model1+=[nn.ReLU(True),]
    model1+=[nn.Conv2d(64, 64, kernel_size=3, stride=2, padding=1, bias=True),]
    model1+=[nn.ReLU(True),]
    model1+=[norm_layer(64),]

    model2=[nn.Conv2d(64, 128, kernel_size=3, stride=1, padding=1, bias=True),]
    model2+=[nn.ReLU(True),]
    model2+=[nn.Conv2d(128, 128, kernel_size=3, stride=2, padding=1, bias=True),]
    model2+=[nn.ReLU(True),]
    model2+=[norm_layer(128),]
```

Under “__init__” function of ECCVGenerator() you can commend and uncomment the model block to add more layers or less layers. For example, the code has model number up to 8 which represents full layer.

4.2 U-Net model

```
class Unet(nn.Module):
    def __init__(self, input_c=1, output_c=2, n_down=8, num_filters=64):
        super().__init__()
        unet_block = UnetBlock(num_filters * 8, num_filters * 8, innermost=True)
```

default value is set to n_down=8. This is changed to 9 and 7 for each layer test.