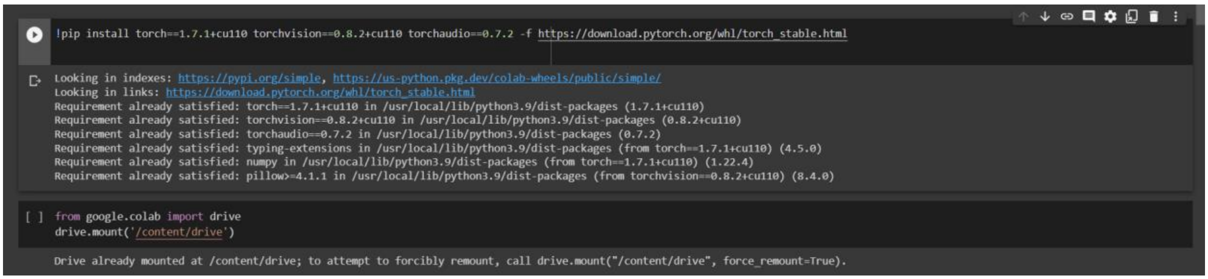
Name: Vontela Sanjay Kumar

ID: 1002090448

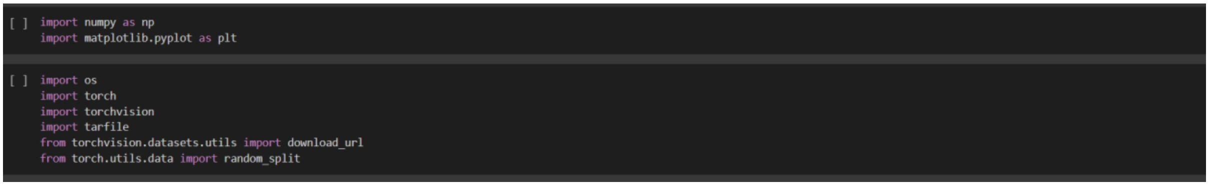
Code Screen Shots:



```
!pip install torch==1.7.1+cu110 torchvision==0.8.2+cu110 torchaudio==0.7.2 -f https://download.pytorch.org/whl/torch_stable.html
```

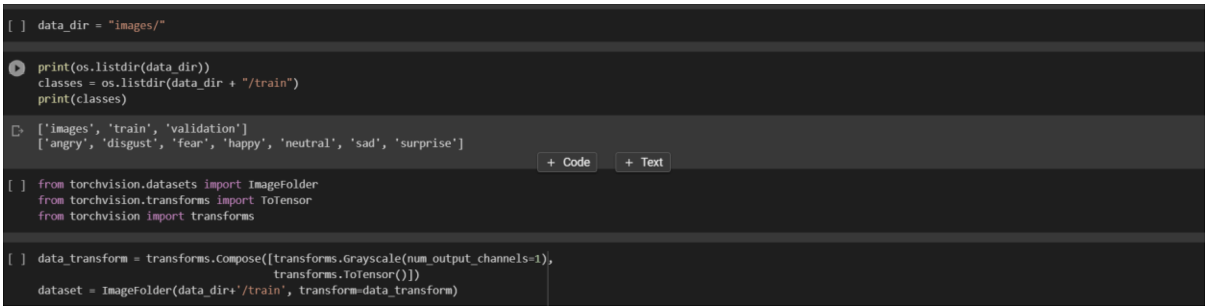
```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Looking in links: https://download.pytorch.org/whl/torch_stable.html
Requirement already satisfied: torch==1.7.1+cu110 in /usr/local/lib/python3.9/dist-packages (1.7.1+cu110)
Requirement already satisfied: torchvision==0.8.2+cu110 in /usr/local/lib/python3.9/dist-packages (0.8.2+cu110)
Requirement already satisfied: torchaudio==0.7.2 in /usr/local/lib/python3.9/dist-packages (0.7.2)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.9/dist-packages (from torch==1.7.1+cu110) (4.5.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (from torch==1.7.1+cu110) (1.22.4)
Requirement already satisfied: pillow>=4.1.1 in /usr/local/lib/python3.9/dist-packages (from torchvision==0.8.2+cu110) (8.4.0)
```

```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```



```
[ ] import numpy as np
    import matplotlib.pyplot as plt
```

```
[ ] import os
    import torch
    import torchvision
    import tarfile
    from torchvision.datasets.utils import download_url
    from torch.utils.data import random_split
```

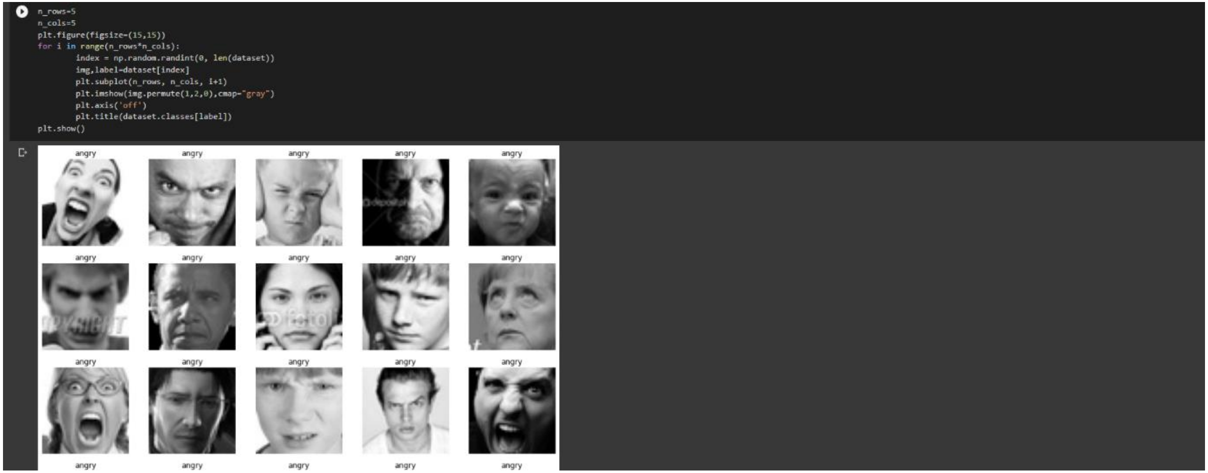


```
[ ] data_dir = "images/"
```

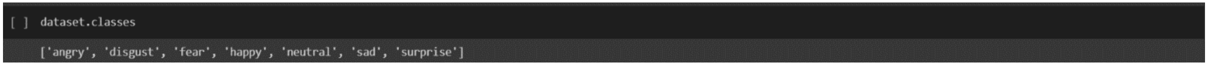
```
print(os.listdir(data_dir))
classes = os.listdir(data_dir + "/train")
print(classes)

['images', 'train', 'validation']
['angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'surprise']
```

```
[ ] from torchvision.datasets import ImageFolder
    from torchvision.transforms import ToTensor
    from torchvision import transforms
```

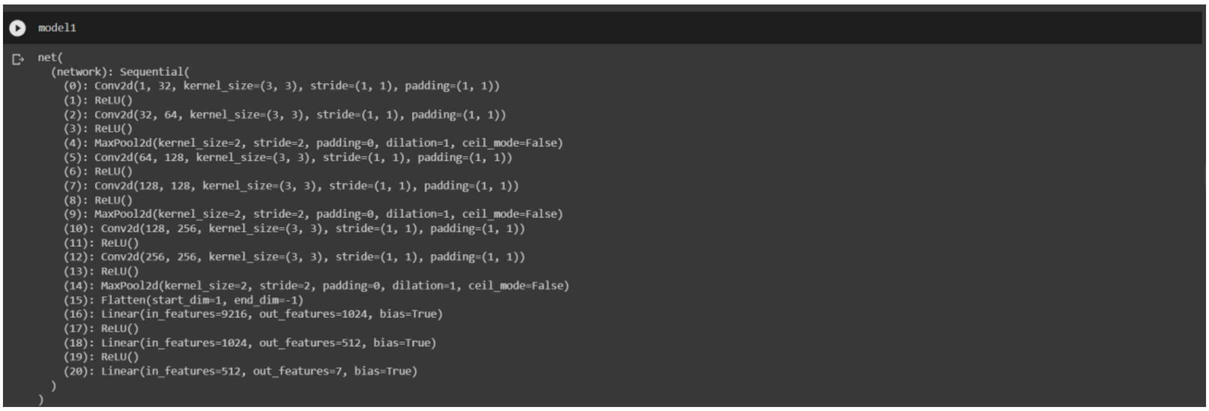
```
[ ] data_transform = transforms.Compose([transforms.Grayscale(num_output_channels=1),
                                         transforms.ToTensor()])
    dataset = ImageFolder(data_dir+'/train', transform=data_transform)
```



```
n_rows=5
n_cols=5
plt.figure(figsize=(15,15))
for i in range(n_rows*n_cols):
        index = np.random.randint(0, len(dataset))
        img,label=dataset[index]
        plt.subplot(n_rows, n_cols, i+1)
        plt.imshow(img.permute(1,2,0),cmap="gray")
        plt.axis('off')
        plt.title(dataset.classes[label])
plt.show()
```

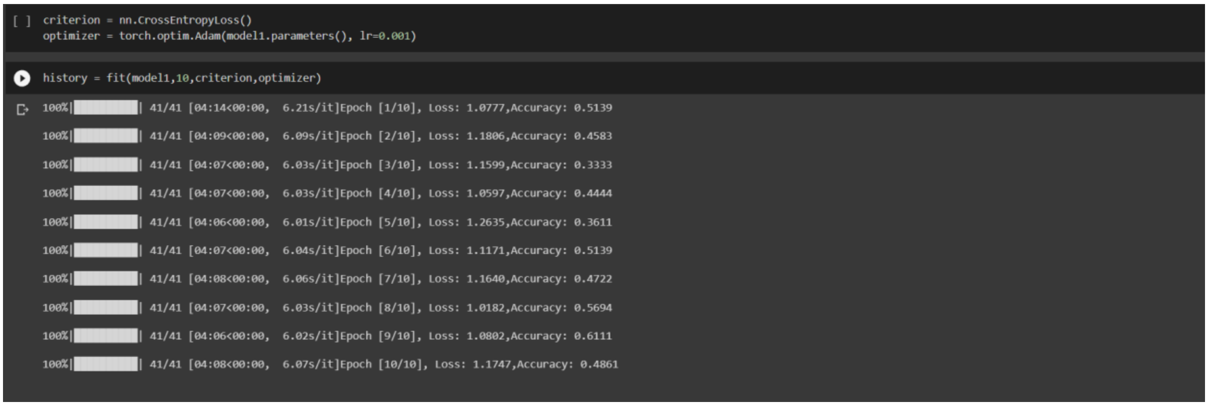


```
[ ] dataset.classes

    ['angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'surprise']
```



```
model1
```

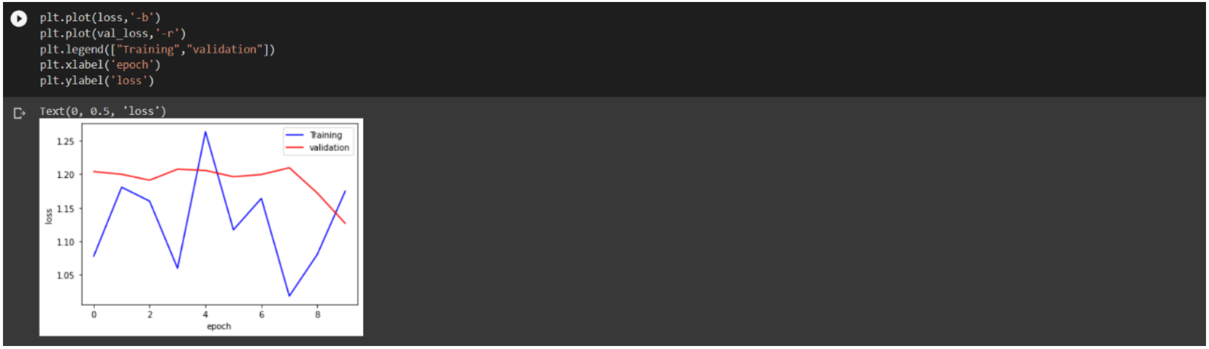
```
net(
  (network): Sequential(
    (0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU()
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU()
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU()
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU()
    (14): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (15): Flatten(start_dim=1, end_dim=-1)
    (16): Linear(in_features=9216, out_features=1024, bias=True)
    (17): ReLU()
    (18): Linear(in_features=1024, out_features=512, bias=True)
    (19): ReLU()
    (20): Linear(in_features=512, out_features=7, bias=True)
  )
)
```



```
[ ] criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(model1.parameters(), lr=0.001)
```

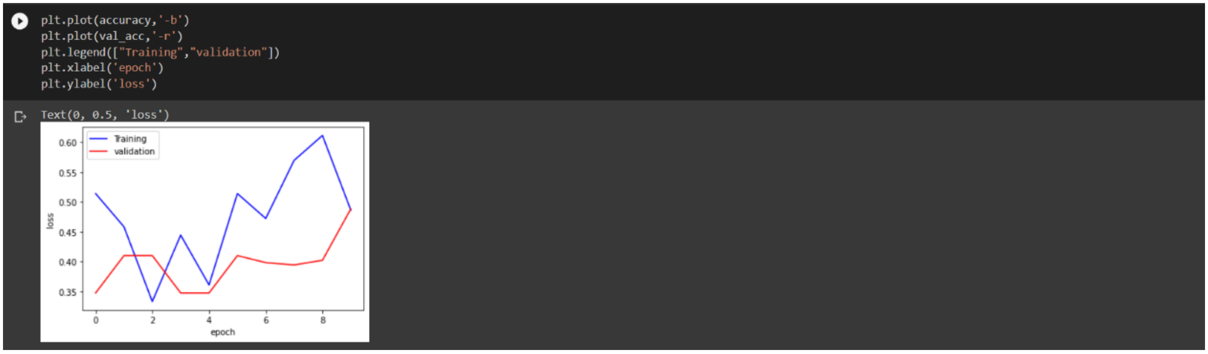
```
history = fit(model1,10,criterion,optimizer)
```

```
100%|██████████| 41/41 [04:14<00:00,  6.21s/it]Epoch [1/10], Loss: 1.0777,Accuracy: 0.5139

100%|██████████| 41/41 [04:09<00:00,  6.09s/it]Epoch [2/10], Loss: 1.1806,Accuracy: 0.4583

100%|██████████| 41/41 [04:07<00:00,  6.03s/it]Epoch [3/10], Loss: 1.1599,Accuracy: 0.3333

100%|██████████| 41/41 [04:07<00:00,  6.03s/it]Epoch [4/10], Loss: 1.0597,Accuracy: 0.4444

100%|██████████| 41/41 [04:06<00:00,  6.01s/it]Epoch [5/10], Loss: 1.2635,Accuracy: 0.3611

100%|██████████| 41/41 [04:07<00:00,  6.04s/it]Epoch [6/10], Loss: 1.1171,Accuracy: 0.5139

100%|██████████| 41/41 [04:08<00:00,  6.06s/it]Epoch [7/10], Loss: 1.1640,Accuracy: 0.4722

100%|██████████| 41/41 [04:07<00:00,  6.03s/it]Epoch [8/10], Loss: 1.0182,Accuracy: 0.5694

100%|██████████| 41/41 [04:06<00:00,  6.02s/it]Epoch [9/10], Loss: 1.0802,Accuracy: 0.6111

100%|██████████| 41/41 [04:08<00:00,  6.07s/it]Epoch [10/10], Loss: 1.1747,Accuracy: 0.4861
```



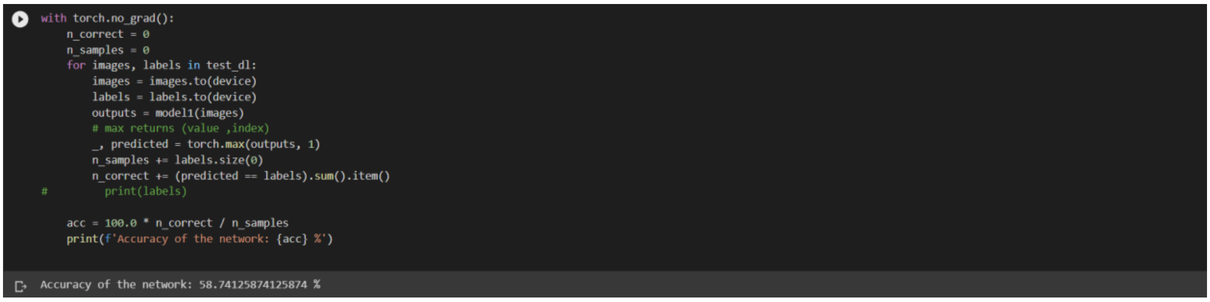
```
plt.plot(loss,'-b')
plt.plot(val_loss,'-r')
plt.legend(["Training","validation"])
plt.xlabel('epoch')
plt.ylabel('loss')
```

```
Text(0, 0.5, 'loss')
```



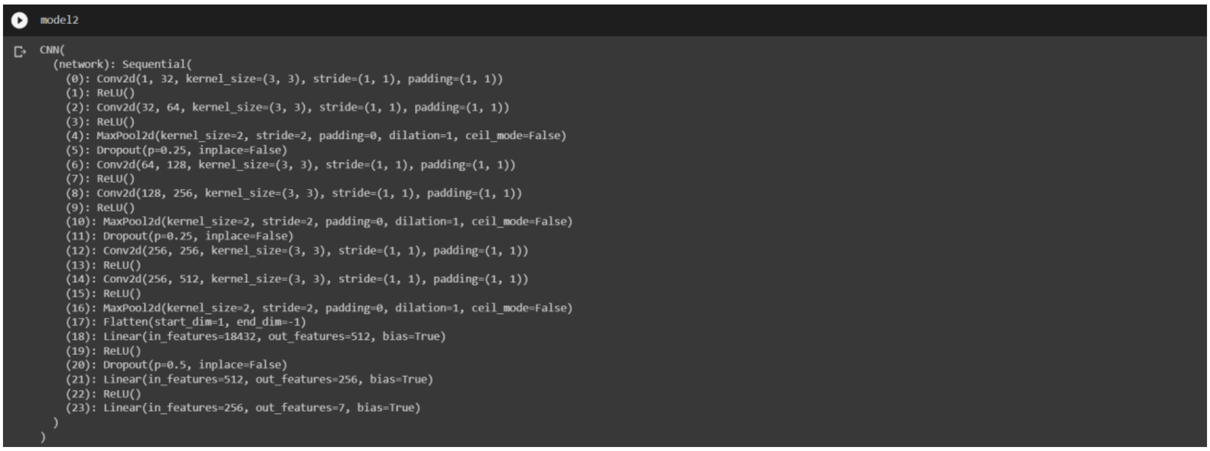
```
plt.plot(accuracy,'-b')
plt.plot(val_acc,'-r')
plt.legend(["Training","validation"])
plt.xlabel('epoch')
plt.ylabel('loss')
```

```
Text(0, 0.5, 'loss')
```



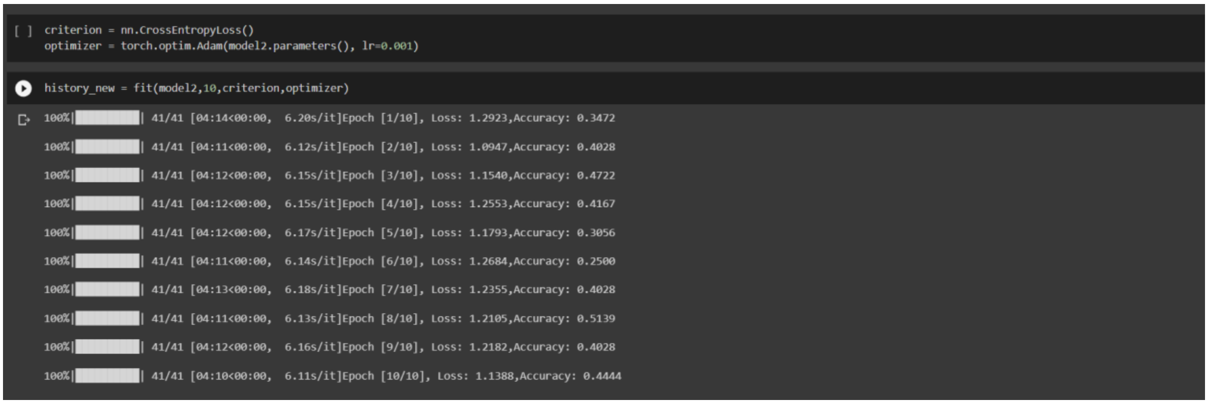
```
with torch.no_grad():
    n_correct = 0
    n_samples = 0
    for images, labels in test_dl:
        images = images.to(device)
        labels = labels.to(device)
        outputs = model1(images)
        # max returns (value ,index)
        _, predicted = torch.max(outputs, 1)
        n_samples += labels.size(0)
        n_correct += (predicted == labels).sum().item()
#          print(labels)

    acc = 100.0 * n_correct / n_samples
    print(f'Accuracy of the network: {acc} %')
```

```
Accuracy of the network: 58.74125874125874 %
```



```
model2
```

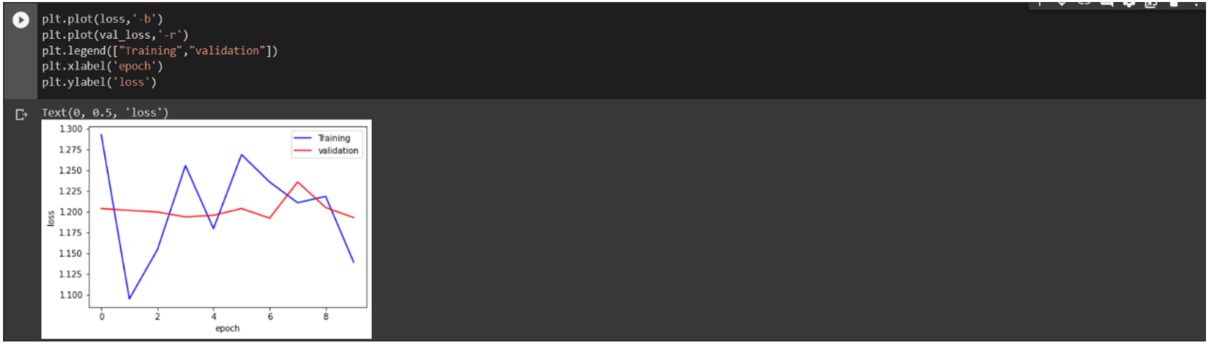
```
CNN(
  (network): Sequential(
    (0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Dropout(p=0.25, inplace=False)
    (6): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU()
    (8): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU()
    (10): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (11): Dropout(p=0.25, inplace=False)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU()
    (14): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU()
    (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (17): Flatten(start_dim=1, end_dim=-1)
    (18): Linear(in_features=18432, out_features=512, bias=True)
    (19): ReLU()
    (20): Dropout(p=0.5, inplace=False)
    (21): Linear(in_features=512, out_features=256, bias=True)
    (22): ReLU()
    (23): Linear(in_features=256, out_features=7, bias=True)
  )
)
```



```
[ ] criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(model2.parameters(), lr=0.001)
```

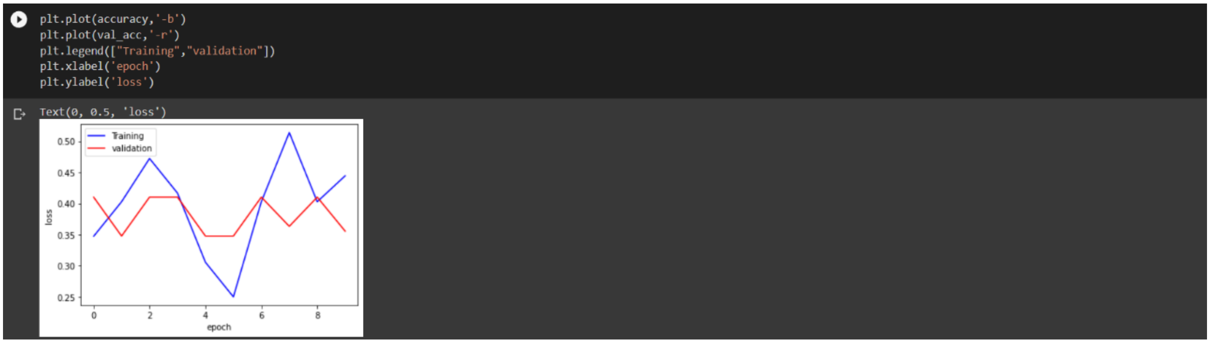
```
history_new = fit(model2,10,criterion,optimizer)
```

```
100%|██████████| 41/41 [04:14<00:00,  6.20s/it]Epoch [1/10], Loss: 1.2923,Accuracy: 0.3472

100%|██████████| 41/41 [04:11<00:00,  6.12s/it]Epoch [2/10], Loss: 1.0947,Accuracy: 0.4028

100%|██████████| 41/41 [04:12<00:00,  6.15s/it]Epoch [3/10], Loss: 1.1540,Accuracy: 0.4722

100%|██████████| 41/41 [04:12<00:00,  6.15s/it]Epoch [4/10], Loss: 1.2553,Accuracy: 0.4167

100%|██████████| 41/41 [04:12<00:00,  6.17s/it]Epoch [5/10], Loss: 1.1793,Accuracy: 0.3056

100%|██████████| 41/41 [04:11<00:00,  6.14s/it]Epoch [6/10], Loss: 1.2684,Accuracy: 0.2500

100%|██████████| 41/41 [04:13<00:00,  6.18s/it]Epoch [7/10], Loss: 1.2355,Accuracy: 0.4028

100%|██████████| 41/41 [04:11<00:00,  6.13s/it]Epoch [8/10], Loss: 1.2105,Accuracy: 0.5139

100%|██████████| 41/41 [04:12<00:00,  6.16s/it]Epoch [9/10], Loss: 1.2182,Accuracy: 0.4028

100%|██████████| 41/41 [04:10<00:00,  6.11s/it]Epoch [10/10], Loss: 1.1388,Accuracy: 0.4444
```



```
plt.plot(loss,'-b')
plt.plot(val_loss,'-r')
plt.legend(["Training","validation"])
plt.xlabel('epoch')
plt.ylabel('loss')
```

```
Text(0, 0.5, 'loss')
```



```
plt.plot(accuracy,'-b')
plt.plot(val_acc,'-r')
plt.legend(["Training","validation"])
plt.xlabel('epoch')
plt.ylabel('loss')
```

```
Text(0, 0.5, 'loss')
```