In [1]:

```
# Shengping Jiang 12.5 capstone submission
```

In [49]:

```
# This Python 3 environment comes with many helpful analytics librar
# It is defined by the kaggle/python Docker image: https://github.co
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_cs

# Input data files are available in the read-only "../input/" direct
# For example, running this (by clicking run or pressing Shift+Enter

import os
for dirname, _, filenames in os.walk('./input'):
    for filename in filenames:
       # print(os.path.join(dirname,filename))

# You can write up to 5GB to the current directory (/kaggle/working/
# You can also write temporary files to /kaggle/temp/, but they won
```

```
  File "<ipython-input-49-5f37c95993f9>", line 17
    # You can also write temporary files to /kaggle/temp/, but they wo
n't be saved outside of the current session

^
SyntaxError: unexpected EOF while parsing
```

In [6]:

```
import os
import cv2
from PIL import Image
import time
import copy
import warnings
import random
import numpy as np
import pandas as pd
from tqdm import tqdm_notebook as tqdm
from torch.optim.lr_scheduler import ReduceLROnPlateau
from sklearn.model_selection import train_test_split
import torch
import torch.nn as nn
from torch.nn import functional as F
import torchvision
import torch.optim as optim
import torch.backends.cudnn as cudnn
from torch.utils.data import DataLoader, Dataset, sampler
from matplotlib import pyplot as plt
import torchvision.transforms as transforms
from albumentations import (HorizontalFlip,VerticalFlip, ShiftScaleR
from albumentations.pytorch import ToTensor
import albumentations as albu
import matplotlib.image as mpi
from sklearn.metrics import f1_score
warnings.filterwarnings("ignore")
seed = 69
random.seed(seed)
os.environ["PYTHONHASHSEED"] = str(seed)
np.random.seed(seed)
torch.cuda.manual_seed(seed)
torch.backends.cudnn.deterministic = True

```

In [7]:

```
path = "./input/British_Shorthair_95.jpg"
```

In [8]:

```python
img = plt.imread(path)
plt.imshow(img)
```

Out[8]:

```
<matplotlib.image.AxesImage at 0x7f0700b43510>
```

In [9]:

```python
import re
pat = r'/([^/]+)_\d+.jpg$'
breed = re.findall(pat,string="./input/British_Shorthair_95.jpg")
```

In [10]:

```python
path_list = []
breed_list = []
for dirname, _, filenames in os.walk('./input'):
    for filename in filenames:
        path = os.path.join(dirname,filename)
        breed = re.findall(pat,string = path)
        if len(breed)>0:
            img = plt.imread(path)
            if len(img.shape)==3:
                if img.shape[2]==3:
                    path_list.append(path)
                    breed_list.append(breed[0])

```

In [11]:

```python
labels = pd.DataFrame({'label_id':path_list,'breed':breed_list})
```

In [12]:

```python
labels.head()
```

Out[12]:

	label_id	breed
0	./input/great_pyrenees_86.jpg	great_pyrenees
1	./input/leonberger_54.jpg	leonberger
2	./input/japanese_chin_149.jpg	japanese_chin
3	./input/american_bulldog_8.jpg	american_bulldog
4	./input/Maine_Coon_23.jpg	Maine_Coon

In [13]:
```python
classes = labels['breed'].unique()
```

In [14]:
```python
classes
```

Out[14]:
```
array(['great_pyrenees', 'leonberger', 'japanese_chin',
       'american_bulldog', 'Maine_Coon', 'chihuahua', 'newfoundland',
       'Ragdoll', 'samoyed', 'Birman', 'basset_hound',
       'miniature_pinscher', 'saint_bernard', 'american_pit_bull_terri
er',
       'Russian_Blue', 'pug', 'British_Shorthair', 'Egyptian_Mau',
       'Siamese', 'Bengal', 'Abyssinian', 'wheaten_terrier',
       'staffordshire_bull_terrier', 'keeshond', 'Persian', 'boxer',
       'scottish_terrier', 'pomeranian', 'yorkshire_terrier',
       'english_cocker_spaniel', 'shiba_inu', 'Bombay',
       'german_shorthaired', 'english_setter', 'beagle', 'havanese',
       'Sphynx'], dtype=object)
```

In [15]:
```python
len(classes)
```

Out[15]:
```
37
```

In [21]:
```python
labels.shape
```

Out[21]:
```
(7378, 2)
```

In [17]:
```python
df1 = labels['breed']
df2 = labels["label_id"]
df1 = pd.get_dummies(df1)
df = pd.concat([df2,df1], axis=1)
df.head()
```

Out[17]:

	label_id	Abyssinian	Bengal	Birman	Bombay	British_Shorthair	Egyptia
0	./input/great_pyrenees_86.jpg	0	0	0	0	0	
1	./input/leonberger_54.jpg	0	0	0	0	0	
2	./input/japanese_chin_149.jpg	0	0	0	0	0	
3	./input/american_bulldog_8.jpg	0	0	0	0	0	
4	./input/Maine_Coon_23.jpg	0	0	0	0	0	

5 rows × 38 columns

In [18]:
```python
df.shape
```

Out[18]:
```
(7378, 38)
```

In [19]:
```python
train_df,test_df = train_test_split(df,test_size=0.2,random_state=85
```

In [20]:

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

Out[20]:

```
device(type='cpu')
```

In [38]:

```
class DogandCat(Dataset):

    def __init__(self,df,phase):
        self.phase = phase
        self.df = df
        if phase == 'train':
            self.transforms = albu.Compose([
               albu.SmallestMaxSize(256),
               albu.RandomCrop(256,256),
                albu.Normalize((0.485, 0.456, 0.406),(0.229, 0.224,
                ToTensor()
            ])
        elif phase == 'val':
            self.transforms = albu.Compose([
                albu.Resize(256,256),
                albu.Normalize((0.485, 0.456, 0.406),(0.229, 0.224,
                ToTensor()
            ])

    def __len__(self):
        return len(self.df)

    def __getitem__(self,index):
        label = self.df.iloc[index,1:]
        label = label.to_numpy()
        label = np.argmax(label)
        path = self.df.iloc[index,0]
        img = plt.imread(path)
        img = self.transforms(image = np.array(img))
        img = img['image']

        return img,label
```

In [39]:

```
traindata = DogandCat(train_df,phase="train")
valdata = DogandCat(test_df,phase="val")
```

In [40]:

```
trainloader = DataLoader(traindata,batch_size=16)
valloader = DataLoader(valdata,batch_size = 16)
```

In [41]:

```
dataiter = iter(trainloader)
img,label = dataiter.next()
```

In [42]:

```
mean = np.array([0.485, 0.456, 0.406])
std = np.array([0.229, 0.224, 0.225])
def im_show(img):
    npimg = img.numpy().transpose((1,2,0))*std + mean
    npimg = np.clip(npimg, 0., 1.)
    plt.imshow(npimg)
fig = plt.figure(figsize=(18,5))

for i in np.arange(16):
    ax = fig.add_subplot(2,8,i+1,xticks=[],yticks=[])
    im_show(img[i])
    ax.set_title(classes[label[i]])

```

In [43]:

```
from torchvision import models
resnet = models.resnet18(pretrained=True,progress = True)
```

```
Downloading: "https://download.pytorch.org/models/resnet18-5c106cde.pt
h" to /home/simon/.cache/torch/checkpoints/resnet18-5c106cde.pth

HBox(children=(FloatProgress(value=0.0, max=46827520.0), HTML(value
=''))))
```

In [44]:

```
for param in resnet.parameters():
    param.requires_grad=False
fc_inputs = resnet.fc.in_features
resnet.fc = nn.Linear(fc_inputs,37)
```

In [45]:

```
from torch.optim import lr_scheduler
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(resnet.fc.parameters(), lr=0.001, betas
scheduler = ReduceLROnPlateau(optimizer,factor=0.33, mode="min", pat
```

In [46]:

```
def train_model(dataloaders,model, criterion, optimizer, scheduler,
    since = time.time()
    dataset_sizes = {'train': len(dataloaders['train'].dataset),
                     'val': len(dataloaders['val'].dataset)}
    best_model_wts = copy.deepcopy(model.state_dict())
    best_acc = 0.0
    number_of_iter = 0
    acc_train = []
    acc_val = []
    loss_train = []
    loss_val = []
    for epoch in range(num_epochs):
        print('Epoch {}/{}'.format(epoch, num_epochs - 1))

        for phase in ['train', 'val']:
            if phase == 'train':
                model.train()
            else:
                model.eval()

            current_loss = 0.0
            current_corrects = 0

            for inputs, labels in dataloaders[phase]:
                inputs = inputs.to(device)
                labels = labels.to(device)
                optimizer.zero_grad()
                with torch.set_grad_enabled(phase == 'train'):
                    outputs = model(inputs)
                    _, preds = torch.max(outputs, 1)
                    loss = criterion(outputs, labels)

                    if phase == 'train':
                        loss.backward()
                        optimizer.step()

                current_loss += loss.item() * inputs.size(0)
                current_corrects += torch.sum(preds == labels.data)

            epoch_loss = current_loss / dataset_sizes[phase]
            epoch_acc = current_corrects.double() / dataset_sizes[ph
            if phase=="train":
                acc_train.append(epoch_acc)
                loss_train.append(epoch_loss)
            else:
                acc_val.append(epoch_acc)
                loss_val.append(epoch_loss)

            print('{} Loss: {:.4f} Acc: {:.4f}'.format(
                phase, epoch_loss, epoch_acc))

            if phase == 'val' and epoch_acc > best_acc:
                best_acc = epoch_acc
                best_model_wts = copy.deepcopy(model.state_dict())
                torch.save(model.state_dict(),'best_weights.pth')
                torch.save(optimizer.state_dict(), 'optimizer.pth')

        print()

    time_since = time.time() - since
    print('Training complete in {:.0f}m {:.0f}s'.format(
        time_since // 60, time_since % 60))
    print('Best val Acc: {:4f}'.format(best_acc))
    model.load_state_dict(best_model_wts)


    return model,acc_val,acc_train,loss_train,loss_val
```

In [47]:

```
resnet = resnet.to(device)
dataloaders = {"train":trainloader,"val":valloader}
num_epochs=10
start_time = time.time()
model,acc_val,acc_train,loss_train,loss_val = train_model(dataloade
```

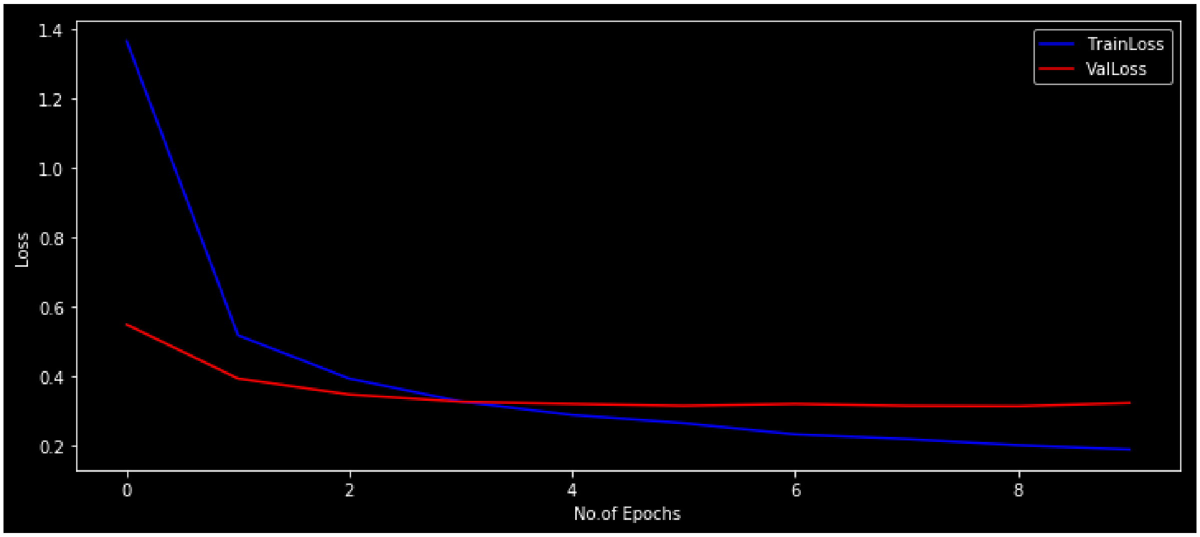
```
Epoch 0/9
train Loss: 1.3647 Acc: 0.6837
val Loss: 0.5491 Acc: 0.8523

Epoch 1/9
train Loss: 0.5181 Acc: 0.8694
val Loss: 0.3939 Acc: 0.8774

Epoch 2/9
train Loss: 0.3935 Acc: 0.8953
val Loss: 0.3478 Acc: 0.8855

Epoch 3/9
train Loss: 0.3277 Acc: 0.9122
val Loss: 0.3271 Acc: 0.8936

Epoch 4/9
train Loss: 0.2889 Acc: 0.9236
val Loss: 0.3209 Acc: 0.8909

Epoch 5/9
train Loss: 0.2655 Acc: 0.9268
val Loss: 0.3154 Acc: 0.8936

Epoch 6/9
train Loss: 0.2333 Acc: 0.9395
val Loss: 0.3209 Acc: 0.8882

Epoch 7/9
train Loss: 0.2197 Acc: 0.9410
val Loss: 0.3153 Acc: 0.8902

Epoch 8/9
train Loss: 0.2021 Acc: 0.9495
val Loss: 0.3142 Acc: 0.8916

Epoch 9/9
train Loss: 0.1899 Acc: 0.9487
val Loss: 0.3237 Acc: 0.8923

Training complete in 69m 20s
Best val Acc: 0.893631
```

In [48]:

```
epoch = []
for x in range(num_epochs):
    epoch.append(x)
plt.style.use('dark_background')
fig = plt.figure(figsize = (12,5))
plt.plot(epoch,loss_train,label = 'TrainLoss',color = 'blue')
plt.plot(epoch,loss_val,label = 'ValLoss',color = 'red')
plt.xlabel('No.of Epochs')
plt.ylabel('Loss')
plt.legend()
```

Out[48]:

```
<matplotlib.legend.Legend at 0x7f06f42b2310>
```



In [50]:

```
labels.to_csv('labels',index = False)
```