AI ASSIGNMENT 3

```
import os
import random
import numpy as np
import pandas as pd
from tqdm import tqdm
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import random_split
from torch.utils.data import DataLoader, Dataset, Subset
from torch.utils.data import random_split, SubsetRandomSampler
from torchvision import datasets, transforms, models
from torchvision.datasets import ImageFolder
from torchvision.transforms import ToTensor
from torchvision.utils import make_grid
from pytorch_lightning import LightningModule
from pytorch_lightning import Trainer
import pytorch_lightning as pl
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from PIL import Image
```

In [2]:

```
transform=transforms.Compose([
        transforms.RandomRotation(10),      # rotate +/- 10 degrees
        transforms.RandomHorizontalFlip(),  # reverse 50% of images
        transforms.Resize(224),             # resize shortest side to 224 pixels
        transforms.CenterCrop(224),         # crop longest side to 224 pixels at cent
er
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
])
```

In [3]:

```
dataset0=datasets.ImageFolder(root="/kaggle/input/animal-image-dataset-90-different-a
nimals/animals/animals",transform=None)

class_names=dataset0.classes
print(class_names)
print(len(class_names))
```

```
['antelope', 'badger', 'bat', 'bear', 'bee', 'beetle', 'bison', 'boar', 'butterfl
y', 'cat', 'caterpillar', 'chimpanzee', 'cockroach', 'cow', 'coyote', 'crab', 'cr
ow', 'deer', 'dog', 'dolphin', 'donkey', 'dragonfly', 'duck', 'eagle', 'elephant',
 'flamingo', 'fly', 'fox', 'goat', 'goldfish', 'goose', 'gorilla', 'grasshopper',
 'hamster', 'hare', 'hedgehog', 'hippopotamus', 'hornbill', 'horse', 'hummingbird
', 'hyena', 'jellyfish', 'kangaroo', 'koala', 'ladybugs', 'leopard', 'lion', 'liz
```

```
ard', 'lobster', 'mosquito', 'moth', 'mouse', 'octopus', 'okapi', 'orangutan', 'o
tter', 'owl', 'ox', 'oyster', 'panda', 'parrot', 'pelecaniformes', 'penguin', 'pi
g', 'pigeon', 'porcupine', 'possum', 'raccoon', 'rat', 'reindeer', 'rhinoceros',
'sandpiper', 'seahorse', 'seal', 'shark', 'sheep', 'snake', 'sparrow', 'squid', '
squirrel', 'starfish', 'swan', 'tiger', 'turkey', 'turtle', 'whale', 'wolf', 'wom
bat', 'woodpecker', 'zebra']
90
```

In [4]:

```
class DataModule(pl.LightningDataModule):

    def __init__(self, transform=transform, batch_size=32):
        super().__init__()
        self.root_dir = "/kaggle/input/animal-image-dataset-90-different-animals/anim
als/animals"
        self.transform = transform
        self.batch_size = batch_size

    def setup(self, stage=None):
        dataset = datasets.ImageFolder(root=self.root_dir, transform=self.transform)
        n_data = len(dataset)
        n_train = int(0.8 * n_data)
        n_test = n_data - n_train

        train_dataset, test_dataset = torch.utils.data.random_split(dataset, [n_train,
 n_test])

        self.train_dataset = DataLoader(train_dataset, batch_size=self.batch_size, sh
uffle=True)
        self.test_dataset = DataLoader(test_dataset, batch_size=self.batch_size)

    def train_dataloader(self):
        return self.train_dataset

    def test_dataloader(self):
        return self.test_dataset
```

In [5]:

```
class ConvolutionalNetwork(LightningModule):

    def __init__(self):
        super(ConvolutionalNetwork, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 3, 1)
        self.conv2 = nn.Conv2d(6, 16, 3, 1)
        self.fc1 = nn.Linear(16 * 54 * 54, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 20)
        self.fc4 = nn.Linear(20, len(class_names))

    def forward(self, X):
        X = F.relu(self.conv1(X))
        X = F.max_pool2d(X, 2, 2)
        X = F.relu(self.conv2(X))
        X = F.max_pool2d(X, 2, 2)
        X = X.view(-1, 16 * 54 * 54)
```

```
        X = F.relu(self.fc1(X))
        X = F.relu(self.fc2(X))
        X = F.relu(self.fc3(X))
        X = self.fc4(X)
        return F.log_softmax(X, dim=1)

    def configure_optimizers(self):
        optimizer = torch.optim.Adam(self.parameters(), lr=0.001)
        return optimizer

    def training_step(self, train_batch, batch_idx):
        X, y = train_batch
        y_hat = self(X)
        loss = F.cross_entropy(y_hat, y)
        pred = y_hat.argmax(dim=1, keepdim=True)
        acc = pred.eq(y.view_as(pred)).sum().item() / y.shape[0]
        self.log("train_loss", loss)
        self.log("train_acc", acc)
        return loss

    def validation_step(self, val_batch, batch_idx):
        X, y = val_batch
        y_hat = self(X)
        loss = F.cross_entropy(y_hat, y)
        pred = y_hat.argmax(dim=1, keepdim=True)
        acc = pred.eq(y.view_as(pred)).sum().item() / y.shape[0]
        self.log("val_loss", loss)
        self.log("val_acc", acc)

    def test_step(self, test_batch, batch_idx):
        X, y = test_batch
        y_hat = self(X)
        loss = F.cross_entropy(y_hat, y)
        pred = y_hat.argmax(dim=1, keepdim=True)
        acc = pred.eq(y.view_as(pred)).sum().item() / y.shape[0]
        self.log("test_loss", loss)
        self.log("test_acc", acc)
```

unfold_moreShow hidden markdown

In [6]:

```
if __name__ == '__main__':
    datamodule = DataModule()
    datamodule.setup()
    model = ConvolutionalNetwork()
    trainer = pl.Trainer(max_epochs=20)
    trainer.fit(model, datamodule)
    datamodule.setup(stage='test')
    test_loader = datamodule.test_dataloader()
    trainer.test(dataloaders=test_loader)
```

```
/opt/conda/lib/python3.7/site-packages/pytorch_lightning/trainer/configuration_va
lidator.py:110: PossibleUserWarning: You defined a `validation_step` but have no
`val_dataloader`. Skipping val loop.
  category=PossibleUserWarning,
```

```
/opt/conda/lib/python3.7/site-packages/pytorch_lightning/trainer/connectors/data_
connector.py:229: PossibleUserWarning: The dataloader, train_dataloader, does not
 have many workers which may be a bottleneck. Consider increasing the value of th
e `num_workers` argument` (try 4 which is the number of cpus on this machine) in
the `DataLoader` init to improve performance.
  category=PossibleUserWarning,
Epoch 19: 100%
135/135 [02:37<00:00, 1.17s/it, loss=1.7, v_num=0]
/opt/conda/lib/python3.7/site-packages/pytorch_lightning/trainer/connectors/check
point_connector.py:128: UserWarning: `.test(ckpt_path=None)` was called without a
 model. The best model of the previous `fit` call will be used. You can pass `.te
st(ckpt_path='best')` to use the best model or `.test(ckpt_path='last')` to use t
he last model. If you pass a value, this warning will be silenced.
  + f" You can pass `.{fn}(ckpt_path='best')` to use the best model or"
/opt/conda/lib/python3.7/site-packages/pytorch_lightning/trainer/connectors/data_
connector.py:229: PossibleUserWarning: The dataloader, test_dataloader 0, does no
t have many workers which may be a bottleneck. Consider increasing the value of t
he `num_workers` argument` (try 4 which is the number of cpus on this machine) in
 the `DataLoader` init to improve performance.
  category=PossibleUserWarning,
Testing DataLoader 0: 100%
34/34 [00:36<00:00, 1.06s/it]
```

Test metric	DataLoader 0
test_acc	0.5425925850868225
test_loss	2.0664007663726807

In [7]:

```
for images, labels in datamodule.train_dataloader():
    break
im=make_grid(images,nrow=16)

plt.figure(figsize=(12,12))
plt.imshow(np.transpose(im.numpy(),(1,2,0)))

inv_normalize=transforms.Normalize(mean=[-0.485/0.229,-0.456/0.224,-0.406/0.225],
                                   std=[1/0.229,1/0.224,1/0.225])
im=inv_normalize(im)

plt.figure(figsize=(12,12))
plt.imshow(np.transpose(im.numpy(),(1,2,0)))
```

Out[7]:

```
<matplotlib.image.AxesImage at 0x71341b417390>
```

In [8]:

```
device = torch.device("cpu")   #"cuda:0"

model.eval()
y_true=[]
y_pred=[]
with torch.no_grad():
    for test_data in datamodule.test_dataloader():
        test_images, test_labels = test_data[0].to(device), test_data[1].to(device)
        pred = model(test_images).argmax(dim=1)
        for i in range(len(pred)):
            y_true.append(test_labels[i].item())
            y_pred.append(pred[i].item())

print(classification_report(y_true,y_pred,target_names=class_names,digits=4))
```

```
                precision    recall  f1-score   support

      antelope     0.4167    0.4167    0.4167        12
        badger     0.5000    0.4545    0.4762        11
           bat     0.2308    0.5455    0.3243        11
          bear     0.3333    0.3636    0.3478        11
           bee     0.6250    0.5000    0.5556        10
        beetle     0.6667    0.3636    0.4706        11
         bison     0.6000    0.6000    0.6000        10
          boar     0.4286    0.2500    0.3158        12
     butterfly     0.5333    0.7273    0.6154        11
           cat     0.3750    0.2000    0.2609        15
   caterpillar     0.8333    0.5882    0.6897        17
    chimpanzee     0.7500    0.7500    0.7500        12
     cockroach     0.7222    1.0000    0.8387        13
           cow     0.3571    0.4545    0.4000        11
        coyote     0.2857    0.4444    0.3478         9
          crab     0.7222    0.6500    0.6842        20
          crow     0.8125    0.7647    0.7879        17
          deer     0.5385    0.4667    0.5000        15
           dog     0.4000    0.5455    0.4615        11
       dolphin     0.8000    1.0000    0.8889         8
        donkey     0.2857    0.1333    0.1818        15
     dragonfly     0.6667    0.5714    0.6154         7
          duck     0.2000    0.1176    0.1481        17
         eagle     0.6429    0.8182    0.7200        11
      elephant     0.6429    0.6429    0.6429        14
      flamingo     0.7222    0.9286    0.8125        14
           fly     0.8667    0.8667    0.8667        15
           fox     0.2308    0.5000    0.3158         6
          goat     0.3636    0.6667    0.4706        12
     goldfish     0.5455    0.7500    0.6316         8
        goose     0.3333    0.0909    0.1429        11
      gorilla     0.8000    0.6667    0.7273        12
   grasshopper     0.5000    0.3333    0.4000        15
       hamster     0.6667    0.7692    0.7143        13
          hare     0.4444    0.3636    0.4000        11
```

hedgehog	0.3846	0.4545	0.4167	11
hippopotamus	0.5000	0.5556	0.5263	9
hornbill	0.5556	0.4167	0.4762	12
horse	0.6429	0.6429	0.6429	14
hummingbird	0.5333	0.7273	0.6154	11
hyena	0.2857	0.3333	0.3077	6
jellyfish	0.7000	0.7778	0.7368	9
kangaroo	0.0000	0.0000	0.0000	15
koala	0.3846	0.4167	0.4000	12
ladybugs	0.5556	0.9091	0.6897	11
leopard	0.7143	0.3846	0.5000	13
lion	0.3810	0.5714	0.4571	14
lizard	1.0000	0.2500	0.4000	16
lobster	0.5000	0.7500	0.6000	8
mosquito	0.8182	0.8182	0.8182	11
moth	0.5455	0.5455	0.5455	11
mouse	0.2500	0.6000	0.3529	5
octopus	0.2500	0.2500	0.2500	8
okapi	0.6923	0.7500	0.7200	12
orangutan	0.6000	0.8182	0.6923	11
otter	0.3158	0.5000	0.3871	12
owl	0.5833	0.6364	0.6087	11
ox	0.8333	0.2778	0.4167	18
oyster	0.5000	0.7273	0.5926	11
panda	0.3684	0.7778	0.5000	9
parrot	0.6111	0.7333	0.6667	15
pelecaniformes	0.3750	0.4000	0.3871	15
penguin	0.7000	0.7000	0.7000	10
pig	0.6250	0.6250	0.6250	16
pigeon	0.5000	0.3636	0.4211	11
porcupine	0.6667	1.0000	0.8000	8
possum	0.4444	0.3333	0.3810	12
raccoon	0.6250	0.3333	0.4348	15
rat	0.6154	0.8000	0.6957	10
reindeer	0.5000	0.4667	0.4828	15
rhinoceros	0.2727	0.4286	0.3333	7
sandpiper	0.7647	0.7647	0.7647	17
seahorse	0.5714	0.5333	0.5517	15
seal	0.5000	0.6154	0.5517	13
shark	0.6429	0.7500	0.6923	12
sheep	0.5556	0.2941	0.3846	17
snake	0.5714	0.3636	0.4444	11
sparrow	0.3333	0.2857	0.3077	7
squid	0.7500	0.5000	0.6000	18
squirrel	0.2857	0.2857	0.2857	14
starfish	0.7143	0.3333	0.4545	15
swan	0.5625	0.7500	0.6429	12
tiger	0.6667	0.4444	0.5333	9
turkey	0.5833	0.7778	0.6667	9
turtle	0.5000	0.6364	0.5600	11
whale	1.0000	0.4286	0.6000	14
wolf	0.4375	0.7778	0.5600	9

```
      wombat     0.6000    0.5455    0.5714        11
  woodpecker     0.5455    0.4286    0.4800        14
       zebra     0.9000    0.9000    0.9000        10

    accuracy                         0.5417      1080
   macro avg     0.5462    0.5533    0.5295      1080
weighted avg     0.5607    0.5417    0.5300      1080

/opt/conda/lib/python3.7/site-packages/sklearn/metrics/_classification.py:1318: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
 in labels with no predicted samples. Use `zero_division` parameter to control th
is behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.7/site-packages/sklearn/metrics/_classification.py:1318: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
 in labels with no predicted samples. Use `zero_division` parameter to control th
is behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.7/site-packages/sklearn/metrics/_classification.py:1318: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
 in labels with no predicted samples. Use `zero_division` parameter to control th
is behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```