

```
In [1]: # Pseudocode using Pytorch :
# 1. Import the necessary libraries
# 2. Load the image from the folder
# 3. Data preprocessing using various data augmentation pipeline and Image generation
# 4. Split the train data with validation data.
# 4. Analysis of data such as no of images in each class and no of classes
# 5. Define Forward and Backprop functions
# 6. Build the CNN model
# 7. Perform the training with the hyperparameters like epoch size, batch size, learning rate
# 8. Plot the loss and accuracy curves
# 9. Use the test data and estimate the confidence score.
```

```
In [2]: #import libraries
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
import seaborn as sns
import skimage
from skimage import io, transform
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
from torchvision import datasets, models, transforms
from PIL import Image
```

```
In [3]: from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
In [4]: #Load data
batch_size=20
data_dir = "/content/drive/MyDrive/Projects/Pet classification using CNN/1577957295"
TEST = 'test'
TRAIN = 'train'
VALID = 'val'
```

```
In [5]: #Data preprocessing
transform = transforms.Compose([
    transforms.RandomRotation(30),
    transforms.Resize((256,256)),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])
```

```
In [6]: device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(device)
```

cuda:0

```
In [7]: image_datasets={x: datasets.ImageFolder(os.path.join(data_dir,x),transform=transform)
num_validation = int(np.floor(0.2* len(image_datasets[TRAIN])))
num_train= len(image_datasets[TRAIN]) - num_validation
#Split the data
train_ds,val_ds=torch.utils.data.random_split(image_datasets[TRAIN],[num_train,num_validation])
test_ds=torch.utils.data.DataLoader(image_datasets[TEST])
```

```
# num_train = len(train_data)
# indices = list(range(num_train))
# np.random.shuffle(indices)
# split = int(np.floor(valid_size * num_train))
# train_idx, valid_idx = indices[split:], indices[:split]
# define samplers for obtaining training and validation batches
# train_sampler = SubsetRandomSampler(train_idx)
# valid_sampler = SubsetRandomSampler(valid_idx)
```

```
In [8]: dataloaders = {TRAIN: torch.utils.data.DataLoader(train_ds, batch_size=batch_size, shuffle=True),
                      TEST: torch.utils.data.DataLoader(image_datasets[TEST], batch_size=batch_size, shuffle=False),
                      VALID: torch.utils.data.DataLoader(val_ds, batch_size=batch_size, shuffle=False)}
```

```
In [8]:
```

```
In [9]: len(dataloaders[TRAIN])
len(dataloaders[TEST])
```

```
Out[9]: 1
```

```
In [10]: dataset_sizes=len(image_datasets[TRAIN])
class_names=image_datasets[TRAIN].classes
num_classes=len(class_names)
print(class_names)
```

```
['cats', 'dogs']
```

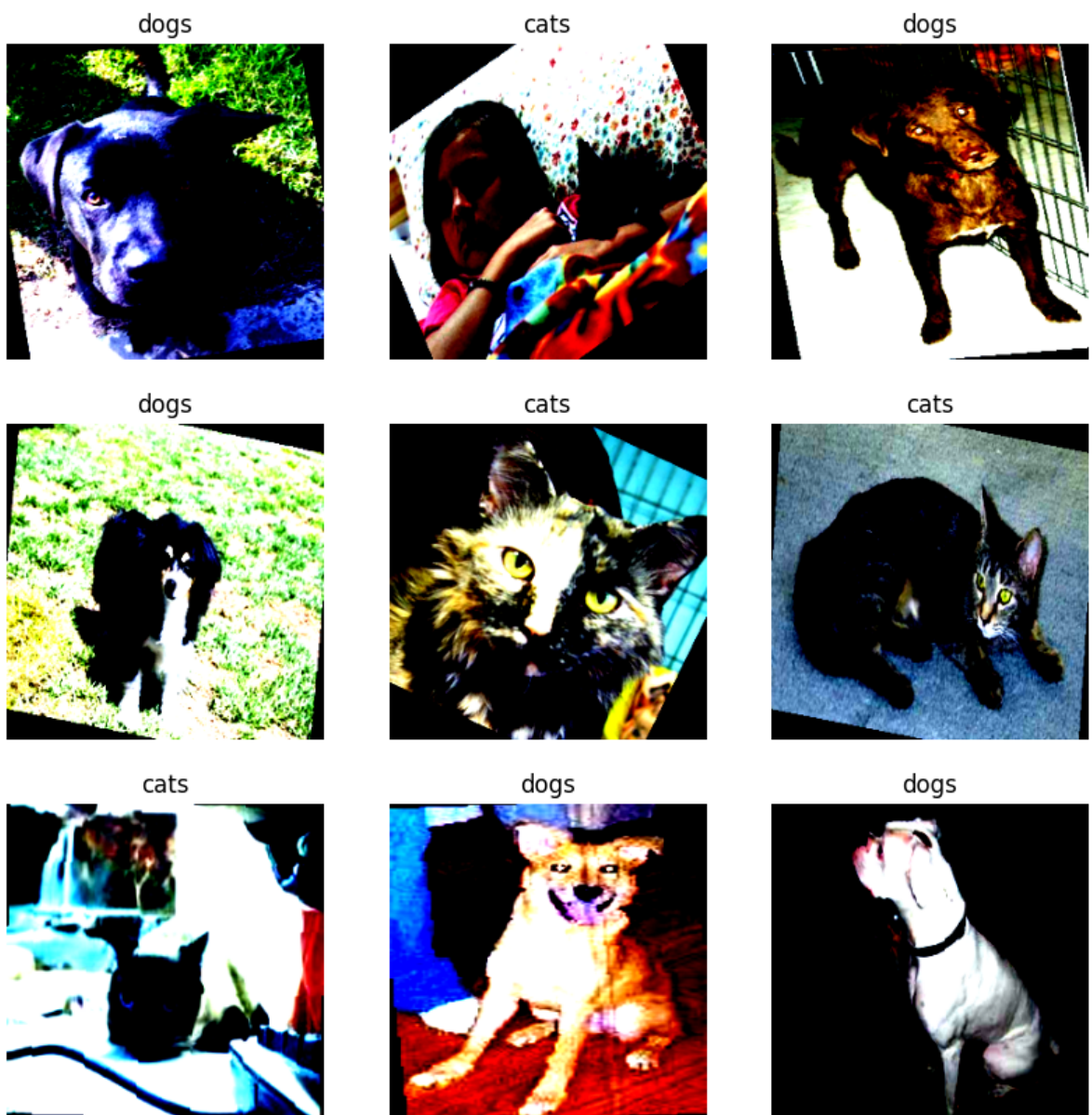
```
In [11]: #Visualize the data
#Plot sample images for all the classes
plt.figure(figsize=(10,10))
for images, labels in dataloaders[TRAIN]:
    print(images.shape, labels.shape)
    for i in range(9):
        plt.subplot(3,3,i+1)
        plt.imshow(images[i].numpy().transpose(1,2,0))
        plt.title(class_names[labels[i]])
        plt.axis("off")
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
torch.Size([20, 3, 256, 256]) torch.Size([20])
```

```

WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
torch.Size([12, 3, 256, 256]) torch.Size([12])

```



```

In [12]: #Visualize the test data
          #Plot sample images for all the classes
          plt.figure(figsize=(10,10))
          for images,labels in dataloaders[TEST]:
              print(images.shape,labels)
              for i in range(9):

```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
torch.Size([20, 3, 256, 256]) tensor([1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1])
```

A small, fluffy white dog, possibly a West Highland White Terrier, is sitting on a wooden surface. The dog is looking upwards and to the right, holding a small, dark object in its mouth. The background is dark and indistinct.

A black and white photograph of a cat with white fur and black patches, lying down. The cat has a white face with black markings around its eyes and ears. It is looking directly at the camera. The background is dark and indistinct.


```
In [13]: class PetNN(nn.Module):
def __init__(self, num_classes, fc_size=20, dropout_prob=0.4):
    super(PetNN, self).__init__()
    self.layer1 = nn.Sequential(
        nn.Conv2d(in_channels=3, out_channels=32, kernel_size=5),
        nn.BatchNorm2d(32),
        nn.ReLU(),
        nn.MaxPool2d(kernel_size=2, stride=2)
    )
    self.layer2 = nn.Sequential(
        nn.Conv2d(in_channels=32, out_channels=64, kernel_size=5),
        nn.BatchNorm2d(64),
        nn.ReLU(),
        nn.MaxPool2d(kernel_size=2, stride=2)
    )

    self.fc1 = nn.Linear(in_features=64 * 61 * 61, out_features=fc_size)
    #self.bn1 = nn.BatchNorm2d(fc_size)
    self.dropout = nn.Dropout(p=dropout_prob)
    self.fc2 = nn.Linear(in_features=fc_size, out_features=num_classes)

    def forward(self, x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = out.view(out.size(0), -1) #flatten to 2D tensor (batch_size, num_channels *
        out = torch.relu(self.fc1(out))
        #out = self.bn1(out)
        out = self.dropout(out)
        out = torch.softmax(self.fc2(out), dim=1)
        return(out)
```

```
In [14]: #define hyperparameters, loss function and optimiser
model = PetNN(num_classes)
model.to(device)
error = nn.CrossEntropyLoss()
learning_rate = 0.01
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
print(model)
```

```
PetNN(
  (layer1): Sequential(
    (0): Conv2d(3, 32, kernel_size=(5, 5), stride=(1, 1))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc1): Linear(in_features=238144, out_features=20, bias=True)
  (dropout): Dropout(p=0.4, inplace=False)
  (fc2): Linear(in_features=20, out_features=2, bias=True)
)
```

```

In [15]: #Initialisation of training variables
epochs = 300
count=0
train_accuracy_list = [] # Store train accuracy values
train_loss_list = [] # Store train loss values
valid_accuracy_list = [] # Store validation accuracy values
valid_loss_list = [] # Store validation loss values
test_accuracy_list = [] # Store test accuracy values
labels_list=[]
prediction_list=[]
#iteration_list=list(range(epochs))
iteration_list=[]
#print(iteration_list)
#Training

for epoch in range(epochs):
    for images,labels in dataloaders[TRAIN]:
        images,labels=images.to(device),labels.to(device)
        train_outputs=model(images)
        #print(outputs.shape)
        loss=error(train_outputs,labels)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        count += 1
        iteration_list.append(count)

    with torch.no_grad():
        #train_outputs=model(images)
        train_predictions=torch.max(train_outputs,1)[1]
        train_correct=(train_predictions==labels).sum().item()
        train_total=len(labels)
        train_accuracy=train_correct*100/train_total
        train_accuracy_list.append(train_accuracy)
        train_loss_list.append(loss.item())

    total=0
    correct=0
    for images,labels in dataloaders[VALID]:
        images,labels=images.to(device),labels.to(device)
        labels_list.append(labels)
        outputs=model(images)
        loss=error(outputs,labels)
        predictions=torch.max(outputs,1)[1].to(device)
        prediction_list.append(predictions)
        correct += (predictions==labels).sum()
        total+=len(labels)
    with torch.no_grad():
        valid_correct = (predictions == labels).sum().item()
        valid_total = len(labels)
        valid_accuracy = valid_correct * 100 / valid_total
        valid_accuracy_list.append(valid_accuracy)
        # accuracy_list.append(accuracy)
        # loss_list.append(loss.data)pend(valid_accuracy)
        valid_loss_list.append(loss.item())

        # accuracy=correct*100/total
        # iteration_list.append(count)

    print("epochs:{},loss:{},train_accuracy:{}%".format(epoch,loss.data,train_accuracy))

```

```
print("epochs:{},loss:{},valid_accuracy:{}%".format(epoch,loss.data,valid_accuracy))  
#print(len(train_accuracy_list))
```

epochs:0,loss:0.938261866569519,train_accuracy:50.0%
epochs:0,loss:0.938261866569519,valid_accuracy:37.5%
epochs:1,loss:0.8132617473602295,train_accuracy:25.0%
epochs:1,loss:0.8132617473602295,valid_accuracy:50.0%
epochs:2,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:2,loss:0.938261866569519,valid_accuracy:37.5%
epochs:3,loss:0.938261866569519,train_accuracy:41.666666666666664%
epochs:3,loss:0.938261866569519,valid_accuracy:37.5%
epochs:4,loss:0.938261866569519,train_accuracy:50.0%
epochs:4,loss:0.938261866569519,valid_accuracy:37.5%
epochs:5,loss:0.938261866569519,train_accuracy:33.333333333333336%
epochs:5,loss:0.938261866569519,valid_accuracy:37.5%
epochs:6,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:6,loss:0.938261866569519,valid_accuracy:37.5%
epochs:7,loss:0.938261866569519,train_accuracy:50.0%
epochs:7,loss:0.938261866569519,valid_accuracy:37.5%
epochs:8,loss:0.9382617473602295,train_accuracy:50.0%
epochs:8,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:9,loss:0.938261866569519,train_accuracy:41.666666666666664%
epochs:9,loss:0.938261866569519,valid_accuracy:37.5%
epochs:10,loss:0.938261866569519,train_accuracy:41.666666666666664%
epochs:10,loss:0.938261866569519,valid_accuracy:37.5%
epochs:11,loss:0.938261866569519,train_accuracy:58.333333333333336%
epochs:11,loss:0.938261866569519,valid_accuracy:37.5%
epochs:12,loss:0.938261866569519,train_accuracy:33.333333333333336%
epochs:12,loss:0.938261866569519,valid_accuracy:37.5%
epochs:13,loss:0.938261866569519,train_accuracy:50.0%
epochs:13,loss:0.938261866569519,valid_accuracy:37.5%
epochs:14,loss:0.938261866569519,train_accuracy:75.0%
epochs:14,loss:0.938261866569519,valid_accuracy:37.5%
epochs:15,loss:0.9382617473602295,train_accuracy:33.333333333333336%
epochs:15,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:16,loss:0.9382617473602295,train_accuracy:25.0%
epochs:16,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:17,loss:0.9382617473602295,train_accuracy:50.0%
epochs:17,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:18,loss:0.938261866569519,train_accuracy:58.333333333333336%
epochs:18,loss:0.938261866569519,valid_accuracy:37.5%
epochs:19,loss:0.938261866569519,train_accuracy:41.666666666666664%
epochs:19,loss:0.938261866569519,valid_accuracy:37.5%
epochs:20,loss:0.938261866569519,train_accuracy:58.333333333333336%
epochs:20,loss:0.938261866569519,valid_accuracy:37.5%
epochs:21,loss:0.938261866569519,train_accuracy:50.0%
epochs:21,loss:0.938261866569519,valid_accuracy:37.5%
epochs:22,loss:0.938261866569519,train_accuracy:50.0%
epochs:22,loss:0.938261866569519,valid_accuracy:37.5%
epochs:23,loss:0.938261866569519,train_accuracy:41.666666666666664%
epochs:23,loss:0.938261866569519,valid_accuracy:37.5%
epochs:24,loss:0.938261866569519,train_accuracy:50.0%
epochs:24,loss:0.938261866569519,valid_accuracy:37.5%
epochs:25,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:25,loss:0.938261866569519,valid_accuracy:37.5%
epochs:26,loss:0.938261866569519,train_accuracy:58.333333333333336%
epochs:26,loss:0.938261866569519,valid_accuracy:37.5%
epochs:27,loss:0.938261866569519,train_accuracy:75.0%
epochs:27,loss:0.938261866569519,valid_accuracy:37.5%
epochs:28,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:28,loss:0.938261866569519,valid_accuracy:37.5%
epochs:29,loss:0.9382617473602295,train_accuracy:58.333333333333336%
epochs:29,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:30,loss:0.938261866569519,train_accuracy:41.666666666666664%
epochs:30,loss:0.938261866569519,valid_accuracy:37.5%
epochs:31,loss:0.938261866569519,train_accuracy:50.0%
epochs:31,loss:0.938261866569519,valid_accuracy:37.5%

epochs:32,loss:0.938261866569519,train_accuracy:50.0%
epochs:32,loss:0.938261866569519,valid_accuracy:37.5%
epochs:33,loss:0.938261866569519,train_accuracy:66.6666666666667%
epochs:33,loss:0.938261866569519,valid_accuracy:37.5%
epochs:34,loss:0.938261866569519,train_accuracy:66.6666666666667%
epochs:34,loss:0.938261866569519,valid_accuracy:37.5%
epochs:35,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:35,loss:0.938261866569519,valid_accuracy:37.5%
epochs:36,loss:0.9382617473602295,train_accuracy:50.0%
epochs:36,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:37,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:37,loss:0.938261866569519,valid_accuracy:37.5%
epochs:38,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:38,loss:0.938261866569519,valid_accuracy:37.5%
epochs:39,loss:0.938261866569519,train_accuracy:50.0%
epochs:39,loss:0.938261866569519,valid_accuracy:37.5%
epochs:40,loss:0.938261866569519,train_accuracy:50.0%
epochs:40,loss:0.938261866569519,valid_accuracy:37.5%
epochs:41,loss:0.9382617473602295,train_accuracy:58.33333333333336%
epochs:41,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:42,loss:0.9382617473602295,train_accuracy:41.66666666666664%
epochs:42,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:43,loss:0.938261866569519,train_accuracy:83.33333333333333%
epochs:43,loss:0.938261866569519,valid_accuracy:37.5%
epochs:44,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:44,loss:0.938261866569519,valid_accuracy:37.5%
epochs:45,loss:0.938261866569519,train_accuracy:50.0%
epochs:45,loss:0.938261866569519,valid_accuracy:37.5%
epochs:46,loss:0.938261866569519,train_accuracy:66.6666666666667%
epochs:46,loss:0.938261866569519,valid_accuracy:37.5%
epochs:47,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:47,loss:0.938261866569519,valid_accuracy:37.5%
epochs:48,loss:0.938261866569519,train_accuracy:75.0%
epochs:48,loss:0.938261866569519,valid_accuracy:37.5%
epochs:49,loss:0.938261866569519,train_accuracy:50.0%
epochs:49,loss:0.938261866569519,valid_accuracy:37.5%
epochs:50,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:50,loss:0.938261866569519,valid_accuracy:37.5%
epochs:51,loss:0.938261866569519,train_accuracy:50.0%
epochs:51,loss:0.938261866569519,valid_accuracy:37.5%
epochs:52,loss:0.938261866569519,train_accuracy:75.0%
epochs:52,loss:0.938261866569519,valid_accuracy:37.5%
epochs:53,loss:0.9382617473602295,train_accuracy:83.33333333333333%
epochs:53,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:54,loss:0.938261866569519,train_accuracy:66.6666666666667%
epochs:54,loss:0.938261866569519,valid_accuracy:37.5%
epochs:55,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:55,loss:0.938261866569519,valid_accuracy:37.5%
epochs:56,loss:0.9382617473602295,train_accuracy:50.0%
epochs:56,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:57,loss:0.938261866569519,train_accuracy:83.33333333333333%
epochs:57,loss:0.938261866569519,valid_accuracy:37.5%
epochs:58,loss:0.938261866569519,train_accuracy:50.0%
epochs:58,loss:0.938261866569519,valid_accuracy:37.5%
epochs:59,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:59,loss:0.938261866569519,valid_accuracy:37.5%
epochs:60,loss:0.938261866569519,train_accuracy:66.6666666666667%
epochs:60,loss:0.938261866569519,valid_accuracy:37.5%
epochs:61,loss:0.938261866569519,train_accuracy:66.6666666666667%
epochs:61,loss:0.938261866569519,valid_accuracy:37.5%
epochs:62,loss:0.938261866569519,train_accuracy:50.0%
epochs:62,loss:0.938261866569519,valid_accuracy:37.5%
epochs:63,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:63,loss:0.938261866569519,valid_accuracy:37.5%

epochs:64,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:64,loss:0.938261866569519,valid_accuracy:37.5%
epochs:65,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:65,loss:0.938261866569519,valid_accuracy:37.5%
epochs:66,loss:0.938261866569519,train_accuracy:50.0%
epochs:66,loss:0.938261866569519,valid_accuracy:37.5%
epochs:67,loss:0.9382617473602295,train_accuracy:58.33333333333336%
epochs:67,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:68,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:68,loss:0.938261866569519,valid_accuracy:37.5%
epochs:69,loss:0.938261866569519,train_accuracy:50.0%
epochs:69,loss:0.938261866569519,valid_accuracy:37.5%
epochs:70,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:70,loss:0.938261866569519,valid_accuracy:37.5%
epochs:71,loss:0.938261866569519,train_accuracy:75.0%
epochs:71,loss:0.938261866569519,valid_accuracy:37.5%
epochs:72,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:72,loss:0.938261866569519,valid_accuracy:37.5%
epochs:73,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:73,loss:0.938261866569519,valid_accuracy:37.5%
epochs:74,loss:0.9382617473602295,train_accuracy:58.33333333333336%
epochs:74,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:75,loss:0.938261866569519,train_accuracy:50.0%
epochs:75,loss:0.938261866569519,valid_accuracy:37.5%
epochs:76,loss:0.9382617473602295,train_accuracy:75.0%
epochs:76,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:77,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:77,loss:0.938261866569519,valid_accuracy:37.5%
epochs:78,loss:0.9382617473602295,train_accuracy:58.33333333333336%
epochs:78,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:79,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:79,loss:0.938261866569519,valid_accuracy:37.5%
epochs:80,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:80,loss:0.938261866569519,valid_accuracy:37.5%
epochs:81,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:81,loss:0.938261866569519,valid_accuracy:37.5%
epochs:82,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:82,loss:0.938261866569519,valid_accuracy:37.5%
epochs:83,loss:0.938261866569519,train_accuracy:33.33333333333336%
epochs:83,loss:0.938261866569519,valid_accuracy:37.5%
epochs:84,loss:0.9382617473602295,train_accuracy:66.66666666666667%
epochs:84,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:85,loss:0.938261866569519,train_accuracy:50.0%
epochs:85,loss:0.938261866569519,valid_accuracy:37.5%
epochs:86,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:86,loss:0.938261866569519,valid_accuracy:37.5%
epochs:87,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:87,loss:0.938261866569519,valid_accuracy:37.5%
epochs:88,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:88,loss:0.938261866569519,valid_accuracy:37.5%
epochs:89,loss:0.813261866569519,train_accuracy:58.33333333333336%
epochs:89,loss:0.813261866569519,valid_accuracy:50.0%
epochs:90,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:90,loss:0.938261866569519,valid_accuracy:37.5%
epochs:91,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:91,loss:0.938261866569519,valid_accuracy:37.5%
epochs:92,loss:0.938261866569519,train_accuracy:50.0%
epochs:92,loss:0.938261866569519,valid_accuracy:37.5%
epochs:93,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:93,loss:0.938261866569519,valid_accuracy:37.5%
epochs:94,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:94,loss:0.938261866569519,valid_accuracy:37.5%
epochs:95,loss:0.9382617473602295,train_accuracy:58.33333333333336%
epochs:95,loss:0.9382617473602295,valid_accuracy:37.5%

epochs:96,loss:0.938261866569519,train_accuracy:50.0%
epochs:96,loss:0.938261866569519,valid_accuracy:37.5%
epochs:97,loss:0.9382617473602295,train_accuracy:50.0%
epochs:97,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:98,loss:0.938261866569519,train_accuracy:50.0%
epochs:98,loss:0.938261866569519,valid_accuracy:37.5%
epochs:99,loss:0.938261866569519,train_accuracy:50.0%
epochs:99,loss:0.938261866569519,valid_accuracy:37.5%
epochs:100,loss:0.938261866569519,train_accuracy:66.6666666666667%
epochs:100,loss:0.938261866569519,valid_accuracy:37.5%
epochs:101,loss:0.938261866569519,train_accuracy:50.0%
epochs:101,loss:0.938261866569519,valid_accuracy:37.5%
epochs:102,loss:0.9382617473602295,train_accuracy:58.33333333333336%
epochs:102,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:103,loss:0.938261866569519,train_accuracy:33.33333333333336%
epochs:103,loss:0.938261866569519,valid_accuracy:37.5%
epochs:104,loss:0.938261866569519,train_accuracy:50.0%
epochs:104,loss:0.938261866569519,valid_accuracy:37.5%
epochs:105,loss:0.938261866569519,train_accuracy:66.6666666666667%
epochs:105,loss:0.938261866569519,valid_accuracy:37.5%
epochs:106,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:106,loss:0.938261866569519,valid_accuracy:37.5%
epochs:107,loss:0.938261866569519,train_accuracy:50.0%
epochs:107,loss:0.938261866569519,valid_accuracy:37.5%
epochs:108,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:108,loss:0.938261866569519,valid_accuracy:37.5%
epochs:109,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:109,loss:0.938261866569519,valid_accuracy:37.5%
epochs:110,loss:0.938261866569519,train_accuracy:25.0%
epochs:110,loss:0.938261866569519,valid_accuracy:37.5%
epochs:111,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:111,loss:0.938261866569519,valid_accuracy:37.5%
epochs:112,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:112,loss:0.938261866569519,valid_accuracy:37.5%
epochs:113,loss:0.9382617473602295,train_accuracy:50.0%
epochs:113,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:114,loss:0.938261866569519,train_accuracy:50.0%
epochs:114,loss:0.938261866569519,valid_accuracy:37.5%
epochs:115,loss:0.938261866569519,train_accuracy:66.6666666666667%
epochs:115,loss:0.938261866569519,valid_accuracy:37.5%
epochs:116,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:116,loss:0.938261866569519,valid_accuracy:37.5%
epochs:117,loss:0.938261866569519,train_accuracy:50.0%
epochs:117,loss:0.938261866569519,valid_accuracy:37.5%
epochs:118,loss:0.9382617473602295,train_accuracy:41.66666666666664%
epochs:118,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:119,loss:0.938261866569519,train_accuracy:50.0%
epochs:119,loss:0.938261866569519,valid_accuracy:37.5%
epochs:120,loss:0.938261866569519,train_accuracy:33.33333333333336%
epochs:120,loss:0.938261866569519,valid_accuracy:37.5%
epochs:121,loss:0.8715721368789673,train_accuracy:58.33333333333336%
epochs:121,loss:0.8715721368789673,valid_accuracy:37.5%
epochs:122,loss:0.938261866569519,train_accuracy:50.0%
epochs:122,loss:0.938261866569519,valid_accuracy:37.5%
epochs:123,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:123,loss:0.938261866569519,valid_accuracy:37.5%
epochs:124,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:124,loss:0.938261866569519,valid_accuracy:37.5%
epochs:125,loss:0.938261866569519,train_accuracy:50.0%
epochs:125,loss:0.938261866569519,valid_accuracy:37.5%
epochs:126,loss:0.9382617473602295,train_accuracy:66.6666666666667%
epochs:126,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:127,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:127,loss:0.938261866569519,valid_accuracy:37.5%

epochs:128,loss:0.938261866569519,train_accuracy:33.33333333333336%
epochs:128,loss:0.938261866569519,valid_accuracy:37.5%
epochs:129,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:129,loss:0.938261866569519,valid_accuracy:37.5%
epochs:130,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:130,loss:0.938261866569519,valid_accuracy:37.5%
epochs:131,loss:0.938261866569519,train_accuracy:50.0%
epochs:131,loss:0.938261866569519,valid_accuracy:37.5%
epochs:132,loss:0.938261866569519,train_accuracy:50.0%
epochs:132,loss:0.938261866569519,valid_accuracy:37.5%
epochs:133,loss:0.9382617473602295,train_accuracy:66.6666666666667%
epochs:133,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:134,loss:0.938261866569519,train_accuracy:50.0%
epochs:134,loss:0.938261866569519,valid_accuracy:37.5%
epochs:135,loss:0.938261866569519,train_accuracy:33.33333333333336%
epochs:135,loss:0.938261866569519,valid_accuracy:37.5%
epochs:136,loss:0.938261866569519,train_accuracy:50.0%
epochs:136,loss:0.938261866569519,valid_accuracy:37.5%
epochs:137,loss:0.938261866569519,train_accuracy:50.0%
epochs:137,loss:0.938261866569519,valid_accuracy:37.5%
epochs:138,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:138,loss:0.938261866569519,valid_accuracy:37.5%
epochs:139,loss:0.9382617473602295,train_accuracy:66.6666666666667%
epochs:139,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:140,loss:0.938261866569519,train_accuracy:66.6666666666667%
epochs:140,loss:0.938261866569519,valid_accuracy:37.5%
epochs:141,loss:0.938261866569519,train_accuracy:50.0%
epochs:141,loss:0.938261866569519,valid_accuracy:37.5%
epochs:142,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:142,loss:0.938261866569519,valid_accuracy:37.5%
epochs:143,loss:0.938261866569519,train_accuracy:50.0%
epochs:143,loss:0.938261866569519,valid_accuracy:37.5%
epochs:144,loss:0.938261866569519,train_accuracy:66.6666666666667%
epochs:144,loss:0.938261866569519,valid_accuracy:37.5%
epochs:145,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:145,loss:0.938261866569519,valid_accuracy:37.5%
epochs:146,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:146,loss:0.938261866569519,valid_accuracy:37.5%
epochs:147,loss:0.938261866569519,train_accuracy:50.0%
epochs:147,loss:0.938261866569519,valid_accuracy:37.5%
epochs:148,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:148,loss:0.938261866569519,valid_accuracy:37.5%
epochs:149,loss:0.938261866569519,train_accuracy:50.0%
epochs:149,loss:0.938261866569519,valid_accuracy:37.5%
epochs:150,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:150,loss:0.938261866569519,valid_accuracy:37.5%
epochs:151,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:151,loss:0.938261866569519,valid_accuracy:37.5%
epochs:152,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:152,loss:0.938261866569519,valid_accuracy:37.5%
epochs:153,loss:0.9382617473602295,train_accuracy:41.66666666666664%
epochs:153,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:154,loss:0.938261866569519,train_accuracy:50.0%
epochs:154,loss:0.938261866569519,valid_accuracy:37.5%
epochs:155,loss:0.938261866569519,train_accuracy:50.0%
epochs:155,loss:0.938261866569519,valid_accuracy:37.5%
epochs:156,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:156,loss:0.938261866569519,valid_accuracy:37.5%
epochs:157,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:157,loss:0.938261866569519,valid_accuracy:37.5%
epochs:158,loss:0.938261866569519,train_accuracy:66.6666666666667%
epochs:158,loss:0.938261866569519,valid_accuracy:37.5%
epochs:159,loss:0.9382617473602295,train_accuracy:41.66666666666664%
epochs:159,loss:0.9382617473602295,valid_accuracy:37.5%

epochs:160,loss:0.938261866569519,train_accuracy:50.0%
epochs:160,loss:0.938261866569519,valid_accuracy:37.5%
epochs:161,loss:0.9382617473602295,train_accuracy:41.666666666666664%
epochs:161,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:162,loss:0.9382617473602295,train_accuracy:58.333333333333336%
epochs:162,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:163,loss:0.9382617473602295,train_accuracy:58.333333333333336%
epochs:163,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:164,loss:0.813261866569519,train_accuracy:66.66666666666667%
epochs:164,loss:0.813261866569519,valid_accuracy:50.0%
epochs:165,loss:0.938261866569519,train_accuracy:50.0%
epochs:165,loss:0.938261866569519,valid_accuracy:37.5%
epochs:166,loss:0.938261866569519,train_accuracy:50.0%
epochs:166,loss:0.938261866569519,valid_accuracy:37.5%
epochs:167,loss:0.938261866569519,train_accuracy:41.666666666666664%
epochs:167,loss:0.938261866569519,valid_accuracy:37.5%
epochs:168,loss:0.938261866569519,train_accuracy:50.0%
epochs:168,loss:0.938261866569519,valid_accuracy:37.5%
epochs:169,loss:0.869063138961792,train_accuracy:50.0%
epochs:169,loss:0.869063138961792,valid_accuracy:37.5%
epochs:170,loss:0.813261866569519,train_accuracy:58.333333333333336%
epochs:170,loss:0.813261866569519,valid_accuracy:50.0%
epochs:171,loss:0.938261866569519,train_accuracy:58.333333333333336%
epochs:171,loss:0.938261866569519,valid_accuracy:37.5%
epochs:172,loss:0.8529727458953857,train_accuracy:50.0%
epochs:172,loss:0.8529727458953857,valid_accuracy:50.0%
epochs:173,loss:0.938261866569519,train_accuracy:58.333333333333336%
epochs:173,loss:0.938261866569519,valid_accuracy:37.5%
epochs:174,loss:0.9382617473602295,train_accuracy:58.333333333333336%
epochs:174,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:175,loss:0.938261866569519,train_accuracy:58.333333333333336%
epochs:175,loss:0.938261866569519,valid_accuracy:37.5%
epochs:176,loss:0.9779872894287109,train_accuracy:50.0%
epochs:176,loss:0.9779872894287109,valid_accuracy:37.5%
epochs:177,loss:0.938261866569519,train_accuracy:41.666666666666664%
epochs:177,loss:0.938261866569519,valid_accuracy:37.5%
epochs:178,loss:0.938261866569519,train_accuracy:41.666666666666664%
epochs:178,loss:0.938261866569519,valid_accuracy:37.5%
epochs:179,loss:0.9382617473602295,train_accuracy:33.333333333333336%
epochs:179,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:180,loss:0.9212439060211182,train_accuracy:83.33333333333333%
epochs:180,loss:0.9212439060211182,valid_accuracy:37.5%
epochs:181,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:181,loss:0.938261866569519,valid_accuracy:37.5%
epochs:182,loss:0.938261866569519,train_accuracy:58.333333333333336%
epochs:182,loss:0.938261866569519,valid_accuracy:37.5%
epochs:183,loss:0.938261866569519,train_accuracy:58.333333333333336%
epochs:183,loss:0.938261866569519,valid_accuracy:37.5%
epochs:184,loss:0.813261866569519,train_accuracy:58.333333333333336%
epochs:184,loss:0.813261866569519,valid_accuracy:50.0%
epochs:185,loss:0.938261866569519,train_accuracy:58.333333333333336%
epochs:185,loss:0.938261866569519,valid_accuracy:37.5%
epochs:186,loss:0.9382617473602295,train_accuracy:50.0%
epochs:186,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:187,loss:0.938261866569519,train_accuracy:25.0%
epochs:187,loss:0.938261866569519,valid_accuracy:37.5%
epochs:188,loss:0.938261866569519,train_accuracy:50.0%
epochs:188,loss:0.938261866569519,valid_accuracy:37.5%
epochs:189,loss:0.938261866569519,train_accuracy:58.333333333333336%
epochs:189,loss:0.938261866569519,valid_accuracy:37.5%
epochs:190,loss:0.938261866569519,train_accuracy:50.0%
epochs:190,loss:0.938261866569519,valid_accuracy:37.5%
epochs:191,loss:0.813261866569519,train_accuracy:66.66666666666667%
epochs:191,loss:0.813261866569519,valid_accuracy:50.0%

epochs:192,loss:0.938261866569519,train_accuracy:50.0%
epochs:192,loss:0.938261866569519,valid_accuracy:37.5%
epochs:193,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:193,loss:0.938261866569519,valid_accuracy:37.5%
epochs:194,loss:0.8692269325256348,train_accuracy:58.33333333333336%
epochs:194,loss:0.8692269325256348,valid_accuracy:37.5%
epochs:195,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:195,loss:0.938261866569519,valid_accuracy:37.5%
epochs:196,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:196,loss:0.938261866569519,valid_accuracy:37.5%
epochs:197,loss:0.938261866569519,train_accuracy:50.0%
epochs:197,loss:0.938261866569519,valid_accuracy:37.5%
epochs:198,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:198,loss:0.938261866569519,valid_accuracy:37.5%
epochs:199,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:199,loss:0.938261866569519,valid_accuracy:37.5%
epochs:200,loss:0.9382617473602295,train_accuracy:50.0%
epochs:200,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:201,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:201,loss:0.938261866569519,valid_accuracy:37.5%
epochs:202,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:202,loss:0.938261866569519,valid_accuracy:37.5%
epochs:203,loss:0.9382617473602295,train_accuracy:58.33333333333336%
epochs:203,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:204,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:204,loss:0.938261866569519,valid_accuracy:37.5%
epochs:205,loss:0.9382617473602295,train_accuracy:41.66666666666664%
epochs:205,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:206,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:206,loss:0.938261866569519,valid_accuracy:37.5%
epochs:207,loss:0.938261866569519,train_accuracy:75.0%
epochs:207,loss:0.938261866569519,valid_accuracy:37.5%
epochs:208,loss:0.938261866569519,train_accuracy:50.0%
epochs:208,loss:0.938261866569519,valid_accuracy:37.5%
epochs:209,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:209,loss:0.938261866569519,valid_accuracy:37.5%
epochs:210,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:210,loss:0.938261866569519,valid_accuracy:37.5%
epochs:211,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:211,loss:0.938261866569519,valid_accuracy:37.5%
epochs:212,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:212,loss:0.938261866569519,valid_accuracy:37.5%
epochs:213,loss:0.9382617473602295,train_accuracy:58.33333333333336%
epochs:213,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:214,loss:0.9382617473602295,train_accuracy:50.0%
epochs:214,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:215,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:215,loss:0.938261866569519,valid_accuracy:37.5%
epochs:216,loss:0.9382617473602295,train_accuracy:58.33333333333336%
epochs:216,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:217,loss:0.938261866569519,train_accuracy:50.0%
epochs:217,loss:0.938261866569519,valid_accuracy:37.5%
epochs:218,loss:0.938261866569519,train_accuracy:75.0%
epochs:218,loss:0.938261866569519,valid_accuracy:37.5%
epochs:219,loss:0.938261866569519,train_accuracy:50.0%
epochs:219,loss:0.938261866569519,valid_accuracy:37.5%
epochs:220,loss:0.938261866569519,train_accuracy:33.33333333333336%
epochs:220,loss:0.938261866569519,valid_accuracy:37.5%
epochs:221,loss:0.938261866569519,train_accuracy:50.0%
epochs:221,loss:0.938261866569519,valid_accuracy:37.5%
epochs:222,loss:0.938261866569519,train_accuracy:50.0%
epochs:222,loss:0.938261866569519,valid_accuracy:37.5%
epochs:223,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:223,loss:0.938261866569519,valid_accuracy:37.5%

epochs:224,loss:0.938261866569519,train_accuracy:50.0%
epochs:224,loss:0.938261866569519,valid_accuracy:37.5%
epochs:225,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:225,loss:0.938261866569519,valid_accuracy:37.5%
epochs:226,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:226,loss:0.938261866569519,valid_accuracy:37.5%
epochs:227,loss:0.938261866569519,train_accuracy:50.0%
epochs:227,loss:0.938261866569519,valid_accuracy:37.5%
epochs:228,loss:0.938261866569519,train_accuracy:75.0%
epochs:228,loss:0.938261866569519,valid_accuracy:37.5%
epochs:229,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:229,loss:0.938261866569519,valid_accuracy:37.5%
epochs:230,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:230,loss:0.938261866569519,valid_accuracy:37.5%
epochs:231,loss:0.938261866569519,train_accuracy:41.666666666666664%
epochs:231,loss:0.938261866569519,valid_accuracy:37.5%
epochs:232,loss:0.938261866569519,train_accuracy:50.0%
epochs:232,loss:0.938261866569519,valid_accuracy:37.5%
epochs:233,loss:0.9382617473602295,train_accuracy:41.666666666666664%
epochs:233,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:234,loss:0.9382617473602295,train_accuracy:66.66666666666667%
epochs:234,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:235,loss:0.938261866569519,train_accuracy:33.33333333333336%
epochs:235,loss:0.938261866569519,valid_accuracy:37.5%
epochs:236,loss:0.938261866569519,train_accuracy:25.0%
epochs:236,loss:0.938261866569519,valid_accuracy:37.5%
epochs:237,loss:0.9382617473602295,train_accuracy:50.0%
epochs:237,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:238,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:238,loss:0.938261866569519,valid_accuracy:37.5%
epochs:239,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:239,loss:0.938261866569519,valid_accuracy:37.5%
epochs:240,loss:0.938261866569519,train_accuracy:33.33333333333336%
epochs:240,loss:0.938261866569519,valid_accuracy:37.5%
epochs:241,loss:0.938261866569519,train_accuracy:50.0%
epochs:241,loss:0.938261866569519,valid_accuracy:37.5%
epochs:242,loss:0.9778552055358887,train_accuracy:58.33333333333336%
epochs:242,loss:0.9778552055358887,valid_accuracy:37.5%
epochs:243,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:243,loss:0.938261866569519,valid_accuracy:37.5%
epochs:244,loss:0.938261866569519,train_accuracy:25.0%
epochs:244,loss:0.938261866569519,valid_accuracy:37.5%
epochs:245,loss:0.9382617473602295,train_accuracy:75.0%
epochs:245,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:246,loss:0.938261866569519,train_accuracy:50.0%
epochs:246,loss:0.938261866569519,valid_accuracy:37.5%
epochs:247,loss:0.8691732883453369,train_accuracy:50.0%
epochs:247,loss:0.8691732883453369,valid_accuracy:37.5%
epochs:248,loss:0.938261866569519,train_accuracy:25.0%
epochs:248,loss:0.938261866569519,valid_accuracy:37.5%
epochs:249,loss:0.938261866569519,train_accuracy:50.0%
epochs:249,loss:0.938261866569519,valid_accuracy:37.5%
epochs:250,loss:0.938261866569519,train_accuracy:50.0%
epochs:250,loss:0.938261866569519,valid_accuracy:37.5%
epochs:251,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:251,loss:0.938261866569519,valid_accuracy:37.5%
epochs:252,loss:0.938261866569519,train_accuracy:33.33333333333336%
epochs:252,loss:0.938261866569519,valid_accuracy:37.5%
epochs:253,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:253,loss:0.938261866569519,valid_accuracy:37.5%
epochs:254,loss:0.938261866569519,train_accuracy:50.0%
epochs:254,loss:0.938261866569519,valid_accuracy:37.5%
epochs:255,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:255,loss:0.938261866569519,valid_accuracy:37.5%

epochs:256,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:256,loss:0.938261866569519,valid_accuracy:37.5%
epochs:257,loss:0.938261866569519,train_accuracy:50.0%
epochs:257,loss:0.938261866569519,valid_accuracy:37.5%
epochs:258,loss:0.938261866569519,train_accuracy:50.0%
epochs:258,loss:0.938261866569519,valid_accuracy:37.5%
epochs:259,loss:0.813261866569519,train_accuracy:58.33333333333336%
epochs:259,loss:0.813261866569519,valid_accuracy:50.0%
epochs:260,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:260,loss:0.938261866569519,valid_accuracy:37.5%
epochs:261,loss:0.9382617473602295,train_accuracy:58.33333333333336%
epochs:261,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:262,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:262,loss:0.938261866569519,valid_accuracy:37.5%
epochs:263,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:263,loss:0.938261866569519,valid_accuracy:37.5%
epochs:264,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:264,loss:0.938261866569519,valid_accuracy:37.5%
epochs:265,loss:0.8132617473602295,train_accuracy:58.33333333333336%
epochs:265,loss:0.8132617473602295,valid_accuracy:50.0%
epochs:266,loss:0.9382617473602295,train_accuracy:58.33333333333336%
epochs:266,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:267,loss:0.938261866569519,train_accuracy:75.0%
epochs:267,loss:0.938261866569519,valid_accuracy:37.5%
epochs:268,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:268,loss:0.938261866569519,valid_accuracy:37.5%
epochs:269,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:269,loss:0.938261866569519,valid_accuracy:37.5%
epochs:270,loss:0.938261866569519,train_accuracy:50.0%
epochs:270,loss:0.938261866569519,valid_accuracy:37.5%
epochs:271,loss:0.9800688028335571,train_accuracy:75.0%
epochs:271,loss:0.9800688028335571,valid_accuracy:37.5%
epochs:272,loss:0.938261866569519,train_accuracy:75.0%
epochs:272,loss:0.938261866569519,valid_accuracy:37.5%
epochs:273,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:273,loss:0.938261866569519,valid_accuracy:37.5%
epochs:274,loss:0.938261866569519,train_accuracy:50.0%
epochs:274,loss:0.938261866569519,valid_accuracy:37.5%
epochs:275,loss:0.9382617473602295,train_accuracy:41.66666666666664%
epochs:275,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:276,loss:0.9382617473602295,train_accuracy:58.33333333333336%
epochs:276,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:277,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:277,loss:0.938261866569519,valid_accuracy:37.5%
epochs:278,loss:0.938261866569519,train_accuracy:33.33333333333336%
epochs:278,loss:0.938261866569519,valid_accuracy:37.5%
epochs:279,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:279,loss:0.938261866569519,valid_accuracy:37.5%
epochs:280,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:280,loss:0.938261866569519,valid_accuracy:37.5%
epochs:281,loss:0.938261866569519,train_accuracy:8.33333333333334%
epochs:281,loss:0.938261866569519,valid_accuracy:37.5%
epochs:282,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:282,loss:0.938261866569519,valid_accuracy:37.5%
epochs:283,loss:0.938261866569519,train_accuracy:75.0%
epochs:283,loss:0.938261866569519,valid_accuracy:37.5%
epochs:284,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:284,loss:0.938261866569519,valid_accuracy:37.5%
epochs:285,loss:0.938261866569519,train_accuracy:75.0%
epochs:285,loss:0.938261866569519,valid_accuracy:37.5%
epochs:286,loss:0.938261866569519,train_accuracy:33.33333333333336%
epochs:286,loss:0.938261866569519,valid_accuracy:37.5%
epochs:287,loss:0.938261866569519,train_accuracy:41.66666666666664%
epochs:287,loss:0.938261866569519,valid_accuracy:37.5%

```

epochs:288,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:288,loss:0.938261866569519,valid_accuracy:37.5%
epochs:289,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:289,loss:0.938261866569519,valid_accuracy:37.5%
epochs:290,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:290,loss:0.938261866569519,valid_accuracy:37.5%
epochs:291,loss:0.9382617473602295,train_accuracy:41.666666666666664%
epochs:291,loss:0.9382617473602295,valid_accuracy:37.5%
epochs:292,loss:0.938261866569519,train_accuracy:50.0%
epochs:292,loss:0.938261866569519,valid_accuracy:37.5%
epochs:293,loss:0.938261866569519,train_accuracy:58.33333333333336%
epochs:293,loss:0.938261866569519,valid_accuracy:37.5%
epochs:294,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:294,loss:0.938261866569519,valid_accuracy:37.5%
epochs:295,loss:0.938261866569519,train_accuracy:41.666666666666664%
epochs:295,loss:0.938261866569519,valid_accuracy:37.5%
epochs:296,loss:0.938261866569519,train_accuracy:50.0%
epochs:296,loss:0.938261866569519,valid_accuracy:37.5%
epochs:297,loss:1.063261866569519,train_accuracy:66.66666666666667%
epochs:297,loss:1.063261866569519,valid_accuracy:25.0%
epochs:298,loss:0.938261866569519,train_accuracy:66.66666666666667%
epochs:298,loss:0.938261866569519,valid_accuracy:37.5%
epochs:299,loss:0.938261866569519,train_accuracy:50.0%
epochs:299,loss:0.938261866569519,valid_accuracy:37.5%

```

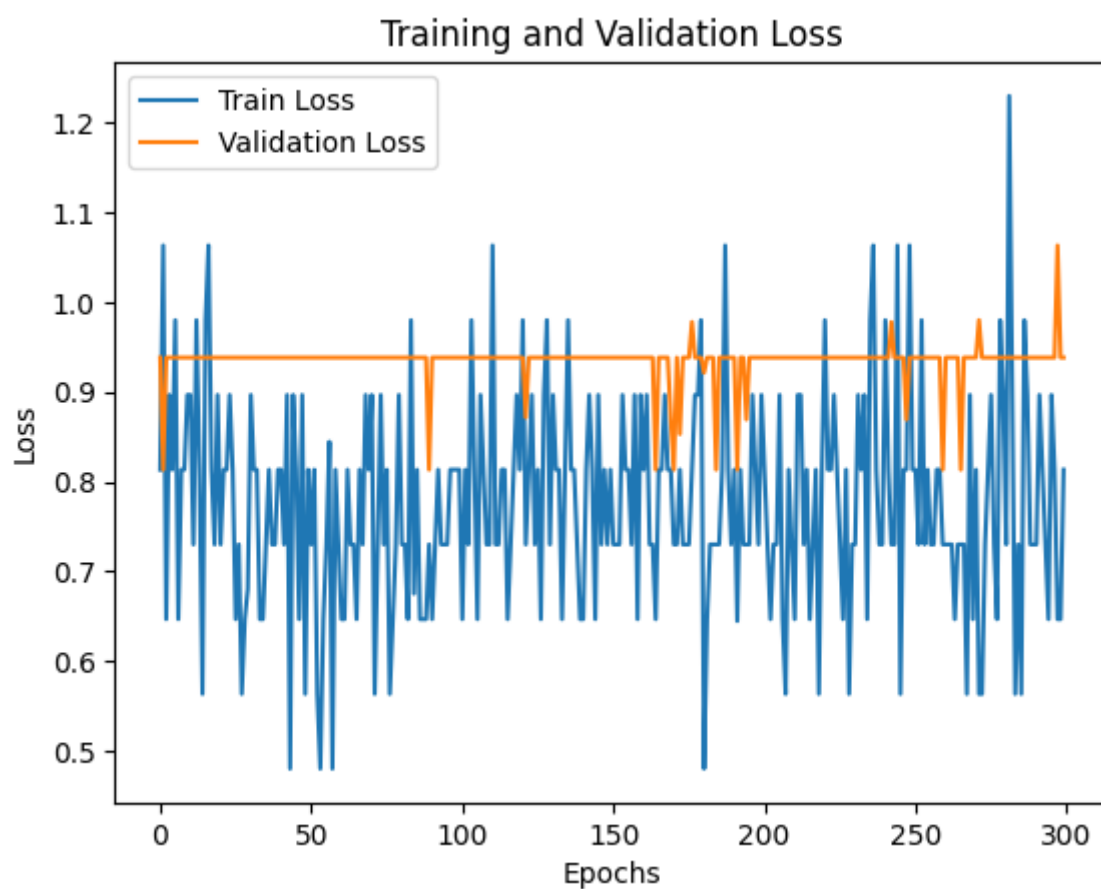
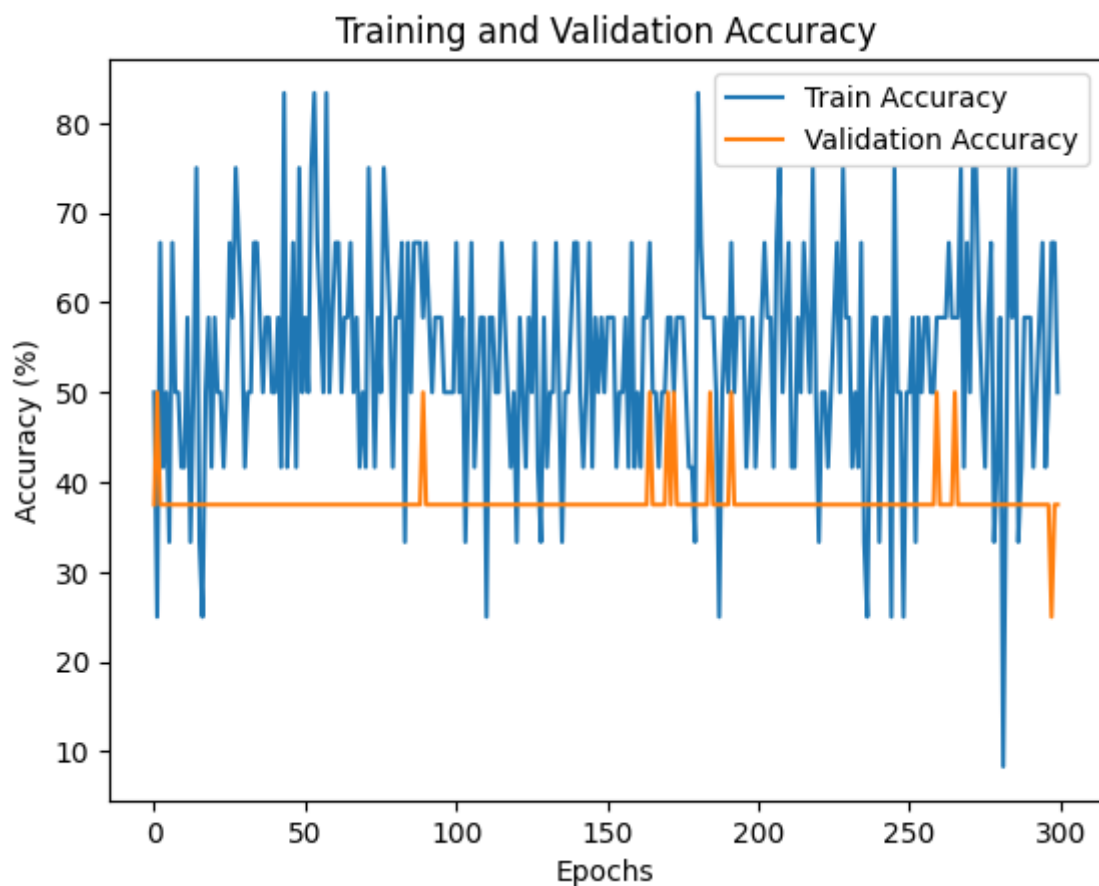
```
In [16]: torch.save(model.state_dict(), '/content/drive/MyDrive/Projects/Pet_classification
```

```
In [17]: # Plotting train and validation accuracy/loss together

plt.plot(list(range(epochs)), train_accuracy_list, label='Train Accuracy')
plt.plot(list(range(epochs)), valid_accuracy_list, label='Validation Accuracy')
plt.xlabel("Epochs")
plt.ylabel("Accuracy (%)")
plt.title("Training and Validation Accuracy")
plt.legend()
plt.show()

plt.plot(list(range(epochs)), train_loss_list, label='Train Loss')
plt.plot(list(range(epochs)), valid_loss_list, label='Validation Loss')
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("Training and Validation Loss")
plt.legend()
plt.show()

# # Plotting test accuracy in a separate plot
# plt.plot(iteration_list[4::5], test_accuracy_list, label='Test Accuracy', color=
# plt.xlabel("No. of Iteration")
# plt.ylabel("Accuracy (%)")
# plt.title("Test Accuracy")
# plt.legend()
# plt.show()
```



```
In [18]: test_path='/content/drive/MyDrive/Projects/Pet classification using CNN/1577957291_
print(os.listdir(test_path))

['cats', 'dogs']
```

```

In [19]: # #test the model
# model = PetNN(2).to(device)
# model.load_state_dict(torch.load('/content/drive/MyDrive/Projects/Pet_classification_using_CNN/157795729'))
# model.eval()
# test_path='/content/drive/MyDrive/Projects/Pet_classification_using_CNN/157795729'
# for class_name in os.listdir(test_path):
#     print(f"Class name: {class_name}")
#     class_path=os.path.join(test_path,class_name)
#     for image_name in os.listdir(class_path):
#         image_path=os.path.join(class_path,image_name)
#         image=Image.open(image_path)
#         # Define a transformation to convert PIL Image to PyTorch tensor
#         transform = transforms.ToTensor()

#         # Apply the transformation to the image
#         transform = transforms.Compose([
#             transforms.RandomRotation(30),
#             transforms.Resize((256, 256)),
#             transforms.RandomHorizontalFlip(),
#             transforms.ToTensor(),
#             transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
#         ])
#         image_tensor = transform(image)
#         image_tensor=image_tensor.unsqueeze(0)
#         image_tensor=image_tensor.to(device)
#         #print(image_tensor.shape)

#         with torch.no_grad():
#             outputs=model(image_tensor)

#         predicted_class_index=torch.max(outputs,1)[1].to(device)
#         #predicted_class_index=torch.argmax(outputs,dim=1).item()
#         print(predicted_class_index)
#         predicted_class=class_names[predicted_class_index]
#         confidence_scores = torch.softmax(outputs, dim=1)

#         print(f'Image : {image_name}, Predicted class : {predicted_class},confidence

```

```

In [22]: def predict(img_detect, model):
img = transform(img_detect).unsqueeze(0).cuda() # Apply transformation and add batch dimension
model.eval() # Set eval mode
output = model(img)
predicted = torch.argmax(output)
return predicted

```

```

In [23]: model = PetNN(2)
model = model.to(device)
model.load_state_dict(torch.load('/content/drive/MyDrive/Projects/Pet_classification_using_CNN/157795729'))

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(device)

test_path = '/content/drive/MyDrive/Projects/Pet_classification_using_CNN/157795729'
for class_name in os.listdir(test_path):
    class_path = os.path.join(test_path, class_name)
    for image_name in os.listdir(class_path):
        image_path = os.path.join(class_path, image_name)
        image = Image.open(image_path)
        predicted_class_index = predict(image, model)
        predicted_class = class_names[predicted_class_index.item()]
        print(f"Image: {image_name}, Predicted class: {predicted_class}")

```

```
cuda:0
Image: 110.jpg, Predicted class: dogs
Image: 104.jpg, Predicted class: dogs
Image: 109.jpg, Predicted class: dogs
Image: 101.jpg, Predicted class: dogs
Image: 106.jpg, Predicted class: dogs
Image: 107.jpg, Predicted class: dogs
Image: 105.jpg, Predicted class: dogs
Image: 108.jpg, Predicted class: dogs
Image: 103.jpg, Predicted class: dogs
Image: 102.jpg, Predicted class: dogs
Image: 106.jpg, Predicted class: dogs
Image: 105.jpg, Predicted class: dogs
Image: 103.jpg, Predicted class: dogs
Image: 110.jpg, Predicted class: dogs
Image: 109.jpg, Predicted class: dogs
Image: 101.jpg, Predicted class: dogs
Image: 107.jpg, Predicted class: dogs
Image: 104.jpg, Predicted class: dogs
Image: 102.jpg, Predicted class: dogs
Image: 108.jpg, Predicted class: dogs
```

```
In [24]: # Test Loop
test_correct = 0
test_total = 0
test_acc_list=[]
with torch.no_grad():
    for images, labels in dataloaders[TEST]:
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        predictions = torch.argmax(outputs, dim=1)
        test_correct += (predictions == labels).sum().item()
        test_total += len(labels)
    test_accuracy = test_correct * 100 / test_total
    print("Test Accuracy: {}".format(test_accuracy))
```

Test Accuracy: 50.0%