nityash-gautam-assignment-3

May 21, 2023

1 Motivation:

This exercise will explore the benefit of data augmentation on the learning performance. You will work with the CIFAR10 dataset. CIFAR10 is a standard image classification dataset frequently used as a benchmark for machine learning algorithms. The dataset is available at https://www.cs.toronto.edu/ ~kriz/cifar.html.

1. NAME: NITYASH GAUTAM
2. SID: 862395403
3. UCR MAIL ID: ngaut006@ucr.edu

1.1 IMPORTING ESSENTIALS

[1]:
```
import torch
print(torch.__version__)
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torchvision
import torchvision.models as models
import torchvision.transforms as transforms
from torch.utils.data import Subset
import matplotlib.pyplot as plt
import numpy as np
```

```
1.13.1
```

1.2 FORMING THE TRAINING DATA

[2]:
```
transform = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.
↪5,0.5,0.5), (0.5,0.5,0.5))])
```

[3]:
```
trainset = torchvision.datasets.CIFAR10(root = './data', train = True,␣
↪transform=transform, download = True)
testset = torchvision.datasets.CIFAR10(root = './data', train = False,␣
↪transform=transform, download = True)

X_train = trainset.data
```

```
y_train = trainset.targets

X_test = testset.data
y_test = testset.targets
```

```
Files already downloaded and verified
Files already downloaded and verified
```

SELECTING 1000 EXAMPLES UNIFORMLY AT RANDOM FROM EVERY CLASS

[4]:
```
classes = 10

train_indices = []

for label in range(classes):
  class_ind = torch.where(torch.tensor(y_train) == label)
  sampled_idx = torch.randperm(len(class_ind))[:1000]
  train_indices.extend(class_ind[sampled_idx].tolist())

training_subset = Subset(trainset, train_indices)

test_indices = []

for label in range(classes):
  class_ind = torch.where(torch.tensor(y_test) == label)
  sampled_idx = torch.randperm(len(class_ind))[:1000]
  test_indices.extend(class_ind[sampled_idx].tolist())

testing_subset = Subset(testset, test_indices)
```

[5]:
```
batch_size = 500
train_loader = torch.utils.data.DataLoader(training_subset,␣
 ↪batch_size=batch_size, shuffle=True)
test_loader = torch.utils.data.DataLoader(testing_subset,␣
 ↪batch_size=batch_size, shuffle=True)
```

[6]:
```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

[7]:
```
model = models.resnet18(pretrained=False).to(device)
lr = 0.001
optim_criter = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=lr)
epochs = 100
```

```
C:\Users\nitya\anaconda3\lib\site-packages\torchvision\models\_utils.py:208:
UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be
removed in the future, please use 'weights' instead.
  warnings.warn(
```

```
C:\Users\nitya\anaconda3\lib\site-packages\torchvision\models\_utils.py:223:
UserWarning: Arguments other than a weight enum or `None` for 'weights' are
deprecated since 0.13 and may be removed in the future. The current behavior is
equivalent to passing `weights=None`.
  warnings.warn(msg)
```

1.3 MAIN ASSIGNMENT TASKS BEGIN

1.3.1 TASK 1 - (3 pts) Train your Resnet model without augmentation and report the results.

[8]:
```
def plot_accuracies(train_acc, test_acc):
    plt.figure(figsize=(10, 5))
    plt.plot(train_acc, label='Training Accuracy')
    plt.plot(test_acc, label='Testing Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.title('Training vs Testing Accuracy')
    plt.legend()
    plt.show()

def plot_loss(train_loss):
    plt.figure(figsize=(10, 5))
    plt.plot(train_loss, label='Training Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training Loss')
    plt.legend()
    plt.show()
```

Main Function

[9]:
```
def train_model(model, train_loader, test_loader, epochs, optimizer, criterion,␣
 ↪device):
    train_loss_list = []
    train_acc_list = []
    test_acc_list = []

    for epoch in range(epochs):
        model.train()
        running_loss = 0.0
        correct = 0
        total = 0

        for i, (inputs, labels) in enumerate(train_loader):
            inputs, labels = inputs.to(device), labels.to(device)
            optimizer.zero_grad()
```

```
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            running_loss += loss.item()

            _, predicted = outputs.max(1)
            total += labels.size(0)
            correct += predicted.eq(labels).sum().item()

        train_loss = running_loss / len(train_loader)
        train_loss_list.append(train_loss)

        train_acc = 100 * correct / total
        train_acc_list.append(train_acc)

        # Calculate test accuracy for each epoch
        model.eval()
        correct = 0
        total = 0

        with torch.no_grad():
            for i, (inputs, labels) in enumerate(test_loader):
                inputs, labels = inputs.to(device), labels.to(device)
                outputs = model(inputs)
                _, predicted = outputs.max(1)
                total += labels.size(0)
                correct += predicted.eq(labels).sum().item()

        test_acc = 100 * correct / total
        test_acc_list.append(test_acc)

        print(f"Epoch [{epoch+1}/{epochs}] - Train Loss: {train_loss:.4f} -␣
↪Train Acc: {train_acc:.2f}% - Test Acc: {test_acc:.2f}%")

    return train_loss_list, train_acc_list, test_acc_list
```

Testing the Main Function

[10]:
```
t1_loss, t1_train_acc, t1_test_acc = train_model(model, train_loader,␣
↪test_loader, epochs, optimizer, optim_criter, device)
```

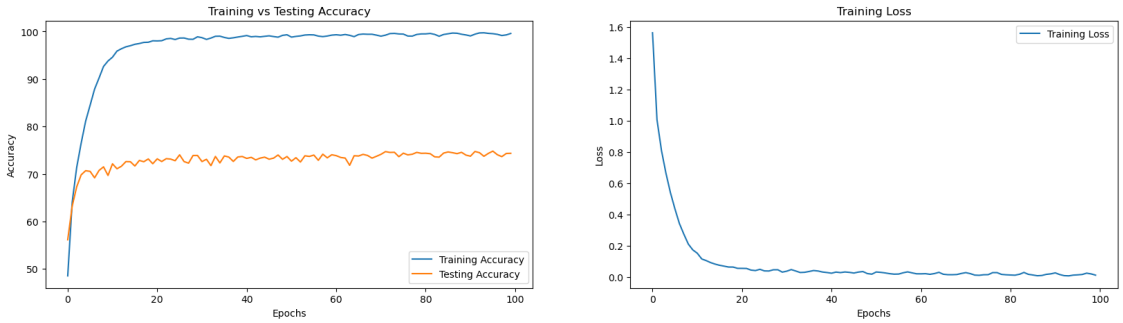
```
Epoch [1/100] - Train Loss: 1.5617 - Train Acc: 48.56% - Test Acc: 56.16%
Epoch [2/100] - Train Loss: 1.0053 - Train Acc: 64.12% - Test Acc: 63.25%
Epoch [3/100] - Train Loss: 0.8090 - Train Acc: 71.38% - Test Acc: 67.30%
Epoch [4/100] - Train Loss: 0.6639 - Train Acc: 76.51% - Test Acc: 69.83%
Epoch [5/100] - Train Loss: 0.5398 - Train Acc: 81.12% - Test Acc: 70.70%
```

```
Epoch [6/100] - Train Loss: 0.4368 - Train Acc: 84.49% - Test Acc: 70.56%
Epoch [7/100] - Train Loss: 0.3428 - Train Acc: 87.88% - Test Acc: 69.20%
Epoch [8/100] - Train Loss: 0.2744 - Train Acc: 90.23% - Test Acc: 70.76%
Epoch [9/100] - Train Loss: 0.2110 - Train Acc: 92.65% - Test Acc: 71.51%
Epoch [10/100] - Train Loss: 0.1740 - Train Acc: 93.80% - Test Acc: 69.70%
Epoch [11/100] - Train Loss: 0.1527 - Train Acc: 94.60% - Test Acc: 72.15%
Epoch [12/100] - Train Loss: 0.1161 - Train Acc: 95.88% - Test Acc: 71.11%
Epoch [13/100] - Train Loss: 0.1053 - Train Acc: 96.38% - Test Acc: 71.64%
Epoch [14/100] - Train Loss: 0.0926 - Train Acc: 96.79% - Test Acc: 72.62%
Epoch [15/100] - Train Loss: 0.0830 - Train Acc: 97.02% - Test Acc: 72.59%
Epoch [16/100] - Train Loss: 0.0754 - Train Acc: 97.32% - Test Acc: 71.69%
Epoch [17/100] - Train Loss: 0.0701 - Train Acc: 97.49% - Test Acc: 72.86%
Epoch [18/100] - Train Loss: 0.0644 - Train Acc: 97.71% - Test Acc: 72.59%
Epoch [19/100] - Train Loss: 0.0642 - Train Acc: 97.75% - Test Acc: 73.18%
Epoch [20/100] - Train Loss: 0.0562 - Train Acc: 98.05% - Test Acc: 72.17%
Epoch [21/100] - Train Loss: 0.0560 - Train Acc: 98.03% - Test Acc: 73.20%
Epoch [22/100] - Train Loss: 0.0554 - Train Acc: 98.07% - Test Acc: 72.63%
Epoch [23/100] - Train Loss: 0.0449 - Train Acc: 98.46% - Test Acc: 73.23%
Epoch [24/100] - Train Loss: 0.0418 - Train Acc: 98.55% - Test Acc: 73.15%
Epoch [25/100] - Train Loss: 0.0498 - Train Acc: 98.33% - Test Acc: 72.80%
Epoch [26/100] - Train Loss: 0.0397 - Train Acc: 98.63% - Test Acc: 74.04%
Epoch [27/100] - Train Loss: 0.0391 - Train Acc: 98.64% - Test Acc: 72.62%
Epoch [28/100] - Train Loss: 0.0469 - Train Acc: 98.40% - Test Acc: 72.29%
Epoch [29/100] - Train Loss: 0.0470 - Train Acc: 98.37% - Test Acc: 73.90%
Epoch [30/100] - Train Loss: 0.0319 - Train Acc: 98.90% - Test Acc: 73.91%
Epoch [31/100] - Train Loss: 0.0379 - Train Acc: 98.71% - Test Acc: 72.62%
Epoch [32/100] - Train Loss: 0.0481 - Train Acc: 98.35% - Test Acc: 73.11%
Epoch [33/100] - Train Loss: 0.0391 - Train Acc: 98.63% - Test Acc: 71.77%
Epoch [34/100] - Train Loss: 0.0293 - Train Acc: 99.01% - Test Acc: 73.70%
Epoch [35/100] - Train Loss: 0.0307 - Train Acc: 99.03% - Test Acc: 72.34%
Epoch [36/100] - Train Loss: 0.0362 - Train Acc: 98.76% - Test Acc: 73.81%
Epoch [37/100] - Train Loss: 0.0420 - Train Acc: 98.56% - Test Acc: 73.57%
Epoch [38/100] - Train Loss: 0.0390 - Train Acc: 98.69% - Test Acc: 72.65%
Epoch [39/100] - Train Loss: 0.0325 - Train Acc: 98.85% - Test Acc: 73.59%
Epoch [40/100] - Train Loss: 0.0292 - Train Acc: 99.00% - Test Acc: 73.69%
Epoch [41/100] - Train Loss: 0.0250 - Train Acc: 99.16% - Test Acc: 73.29%
Epoch [42/100] - Train Loss: 0.0321 - Train Acc: 98.90% - Test Acc: 73.52%
Epoch [43/100] - Train Loss: 0.0288 - Train Acc: 98.97% - Test Acc: 72.98%
Epoch [44/100] - Train Loss: 0.0328 - Train Acc: 98.89% - Test Acc: 73.35%
Epoch [45/100] - Train Loss: 0.0302 - Train Acc: 99.00% - Test Acc: 73.56%
Epoch [46/100] - Train Loss: 0.0260 - Train Acc: 99.10% - Test Acc: 73.12%
Epoch [47/100] - Train Loss: 0.0322 - Train Acc: 98.95% - Test Acc: 73.35%
Epoch [48/100] - Train Loss: 0.0356 - Train Acc: 98.82% - Test Acc: 74.01%
Epoch [49/100] - Train Loss: 0.0228 - Train Acc: 99.20% - Test Acc: 73.10%
Epoch [50/100] - Train Loss: 0.0189 - Train Acc: 99.32% - Test Acc: 73.69%
Epoch [51/100] - Train Loss: 0.0328 - Train Acc: 98.83% - Test Acc: 72.71%
Epoch [52/100] - Train Loss: 0.0303 - Train Acc: 98.98% - Test Acc: 73.44%
Epoch [53/100] - Train Loss: 0.0267 - Train Acc: 99.08% - Test Acc: 72.53%
```

```
Epoch [54/100] - Train Loss: 0.0221 - Train Acc: 99.27% - Test Acc: 73.84%
Epoch [55/100] - Train Loss: 0.0189 - Train Acc: 99.32% - Test Acc: 73.71%
Epoch [56/100] - Train Loss: 0.0196 - Train Acc: 99.29% - Test Acc: 73.99%
Epoch [57/100] - Train Loss: 0.0276 - Train Acc: 99.04% - Test Acc: 72.89%
Epoch [58/100] - Train Loss: 0.0335 - Train Acc: 98.94% - Test Acc: 74.18%
Epoch [59/100] - Train Loss: 0.0271 - Train Acc: 99.06% - Test Acc: 73.41%
Epoch [60/100] - Train Loss: 0.0213 - Train Acc: 99.25% - Test Acc: 74.06%
Epoch [61/100] - Train Loss: 0.0210 - Train Acc: 99.33% - Test Acc: 73.87%
Epoch [62/100] - Train Loss: 0.0220 - Train Acc: 99.23% - Test Acc: 73.49%
Epoch [63/100] - Train Loss: 0.0180 - Train Acc: 99.38% - Test Acc: 73.36%
Epoch [64/100] - Train Loss: 0.0228 - Train Acc: 99.20% - Test Acc: 71.84%
Epoch [65/100] - Train Loss: 0.0305 - Train Acc: 98.93% - Test Acc: 73.85%
Epoch [66/100] - Train Loss: 0.0182 - Train Acc: 99.38% - Test Acc: 73.80%
Epoch [67/100] - Train Loss: 0.0153 - Train Acc: 99.48% - Test Acc: 74.14%
Epoch [68/100] - Train Loss: 0.0152 - Train Acc: 99.43% - Test Acc: 73.91%
Epoch [69/100] - Train Loss: 0.0164 - Train Acc: 99.44% - Test Acc: 73.35%
Epoch [70/100] - Train Loss: 0.0234 - Train Acc: 99.24% - Test Acc: 73.74%
Epoch [71/100] - Train Loss: 0.0286 - Train Acc: 99.03% - Test Acc: 74.15%
Epoch [72/100] - Train Loss: 0.0224 - Train Acc: 99.24% - Test Acc: 74.72%
Epoch [73/100] - Train Loss: 0.0129 - Train Acc: 99.57% - Test Acc: 74.55%
Epoch [74/100] - Train Loss: 0.0118 - Train Acc: 99.60% - Test Acc: 74.58%
Epoch [75/100] - Train Loss: 0.0150 - Train Acc: 99.50% - Test Acc: 73.65%
Epoch [76/100] - Train Loss: 0.0155 - Train Acc: 99.47% - Test Acc: 74.39%
Epoch [77/100] - Train Loss: 0.0286 - Train Acc: 99.05% - Test Acc: 74.03%
Epoch [78/100] - Train Loss: 0.0286 - Train Acc: 99.04% - Test Acc: 74.17%
Epoch [79/100] - Train Loss: 0.0172 - Train Acc: 99.40% - Test Acc: 74.56%
Epoch [80/100] - Train Loss: 0.0148 - Train Acc: 99.50% - Test Acc: 74.36%
Epoch [81/100] - Train Loss: 0.0135 - Train Acc: 99.51% - Test Acc: 74.38%
Epoch [82/100] - Train Loss: 0.0124 - Train Acc: 99.60% - Test Acc: 74.28%
Epoch [83/100] - Train Loss: 0.0181 - Train Acc: 99.40% - Test Acc: 73.62%
Epoch [84/100] - Train Loss: 0.0299 - Train Acc: 99.01% - Test Acc: 73.56%
Epoch [85/100] - Train Loss: 0.0176 - Train Acc: 99.39% - Test Acc: 74.41%
Epoch [86/100] - Train Loss: 0.0135 - Train Acc: 99.53% - Test Acc: 74.65%
Epoch [87/100] - Train Loss: 0.0089 - Train Acc: 99.68% - Test Acc: 74.49%
Epoch [88/100] - Train Loss: 0.0106 - Train Acc: 99.64% - Test Acc: 74.29%
Epoch [89/100] - Train Loss: 0.0176 - Train Acc: 99.44% - Test Acc: 74.57%
Epoch [90/100] - Train Loss: 0.0209 - Train Acc: 99.28% - Test Acc: 73.98%
Epoch [91/100] - Train Loss: 0.0272 - Train Acc: 99.08% - Test Acc: 73.76%
Epoch [92/100] - Train Loss: 0.0162 - Train Acc: 99.45% - Test Acc: 74.75%
Epoch [93/100] - Train Loss: 0.0095 - Train Acc: 99.71% - Test Acc: 74.50%
Epoch [94/100] - Train Loss: 0.0081 - Train Acc: 99.74% - Test Acc: 73.74%
Epoch [95/100] - Train Loss: 0.0128 - Train Acc: 99.61% - Test Acc: 74.35%
Epoch [96/100] - Train Loss: 0.0144 - Train Acc: 99.55% - Test Acc: 74.82%
Epoch [97/100] - Train Loss: 0.0166 - Train Acc: 99.42% - Test Acc: 74.07%
Epoch [98/100] - Train Loss: 0.0254 - Train Acc: 99.16% - Test Acc: 73.66%
Epoch [99/100] - Train Loss: 0.0206 - Train Acc: 99.30% - Test Acc: 74.33%
Epoch [100/100] - Train Loss: 0.0125 - Train Acc: 99.59% - Test Acc: 74.36%
```

[11]:
```
print('*'*10, 'LOSS AND ACCURACY PLOTS WITHOUT AUGMENTATION', '*'*10)

print()

plt.figure(figsize=(20, 5))

plt.subplot(1, 2, 1)
plt.plot(t1_train_acc, label='Training Accuracy')
plt.plot(t1_test_acc, label='Testing Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training vs Testing Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(t1_loss, label='Training Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training Loss')
plt.legend()
plt.show()

print()

print('*'*100)
print('Final Test Accuracy:', t1_test_acc[-1], '%')
print('*'*100)
```

```
********** LOSS AND ACCURACY PLOTS WITHOUT AUGMENTATION **********
```




```
****************************************************************************************************
********************
Final Test Accuracy: 74.36 %
****************************************************************************************************
```

****

1.3.2 TASK 2 - (4 pts) *Mixup Augmentation* is based on the paper https://arxiv.org/pdf/1710.09412.pdf.

As the name suggests, it mixes a pair of training examples (both inputs and labels). Given a pair of training example (x1, y1), (x2, y2), we obtain the augmented training example (x, y) via

$$x = \lambda x_1 + (1 - \lambda) x_2$$
$$y = \lambda y_1 + (1 - \lambda) y_2$$

where mixing parameter has distribution1 with parameter .

TODO: Implement mixup and report the results for = 0.2 and = 0.4. Note that, in each minibatch, all training examples should have mixup transformation before gradient calculation (e.g. from original minibatch obtain a new minibatch by mixing random pairs of training examples).

Helper Functions

[12]:
```
# Function to calculate accuracies
def get_accuracy(outputs, labels):
    _, predicted = torch.max(outputs, 1)
    total = labels.size(0)
    labels = torch.argmax(labels, dim=1)  # Convert mixed labels to class␣
 ↪indices
    correct = (predicted == labels).sum().item()
    accuracy = correct / total * 100
    return accuracy
```

[13]:
```
# Function to train the model
def train(model, dataloader, criterion, optimizer, device):
    model.train()
    train_loss = 0.0
    train_acc = 0.0
    total_samples = 0

    for images, labels in dataloader:
        images, labels = images.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        train_loss += loss.item() * images.size(0)
        train_acc += get_accuracy(outputs, labels) * images.size(0)
        total_samples += images.size(0)
```

```
    train_loss /= total_samples
    train_acc /= total_samples
    return train_loss, train_acc
```

[14]:
```
# Function to test the model
def test(model, dataloader, criterion, device):
    model.eval()
    test_loss = 0.0
    test_acc = 0.0
    total_samples = 0

    with torch.no_grad():
        for images, labels in dataloader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            loss = criterion(outputs, labels)

            test_loss += loss.item() * images.size(0)
            test_acc += get_accuracy(outputs, labels) * images.size(0)
            total_samples += images.size(0)

    test_loss /= total_samples
    test_acc /= total_samples
    return test_loss, test_acc
```

[15]:
```
def mixup_data(x, y, alpha):
    lam = np.random.beta(alpha, alpha)
    batch_size = x.size()[0]
    index = torch.randperm(batch_size).to(x.device)

    mixed_x = lam * x + (1 - lam) * x[index, :]
    y_a, y_b = y, y[index]
    return mixed_x, y_a, y_b, lam


def get_accuracy(outputs, labels):
    _, pred = outputs.max(1)
    correct = pred.eq(labels).sum().item()
    return correct / outputs.size(0) * 100
```

Main Function

[16]:
```
train_indices = []
for class_idx in range(10):
  class_indices = np.where(np.array(y_train) == class_idx)[0]
  sampled_indices = np.random.choice(class_indices, size=1000, replace=False)
```

```
  train_indices.extend(sampled_indices.tolist())

train_sampler = torch.utils.data.sampler.SubsetRandomSampler(train_indices)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=128,␣
 ↪sampler=train_sampler, num_workers=2)

# Convert labels to one-hot encoding
trainset.targets = torch.tensor(trainset.targets)
trainset.targets = nn.functional.one_hot(trainset.targets, num_classes=10).
 ↪float()
```

[17]:
```
def train_mixup(model, train_loader, test_loader, epochs, optimizer, criterion,␣
 ↪device, alpha):
    train_loss_list = []
    train_acc_list = []
    test_acc_list = []

    for epoch in range(epochs):
        model.train()
        train_loss = 0.0
        train_acc = 0.0
        total_samples = 0

        for images, labels in train_loader:
            images, labels = images.to(device), labels.to(device)

            mixed_images, labels_a, labels_b, lam = mixup_data(images, labels,␣
 ↪alpha)
            outputs = model(mixed_images)

            # labels_a and labels_b should be 1-D tensors
            labels_a = torch.max(labels_a, 1)[1]
            labels_b = torch.max(labels_b, 1)[1]

            loss = criterion(outputs, labels_a) * lam + criterion(outputs,␣
 ↪labels_b) * (1 - lam)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            train_loss += loss.item() * images.size(0)

            _, preds = torch.max(outputs, 1)
            train_acc += ((preds == labels_a).float().cpu().sum() * lam +␣
 ↪(preds == labels_b).float().cpu().sum() * (1 - lam)).item()

            total_samples += images.size(0)
```

```
        train_loss /= total_samples
        train_acc = 100. * train_acc / total_samples

        # Test model
        model.eval()
        correct = 0
        total = 0
        with torch.no_grad():
            for images, labels in test_loader:
                images, labels = images.to(device), labels.to(device)
                outputs = model(images)

                _, predicted = outputs.max(1)
                total += labels.size(0)
                correct += predicted.eq(labels).sum().item()

        test_acc = 100. * correct / total

        train_loss_list.append(train_loss)
        train_acc_list.append(train_acc)
        test_acc_list.append(test_acc)

        print(f"Epoch [{epoch+1}/{epochs}] - Train Loss: {train_loss:.4f} -␣
↪Train Acc: {train_acc:.2f}% - Test Acc: {test_acc:.2f}%")

    return train_loss_list, train_acc_list, test_acc_list
```

Testing the Train Function

[18]:
```
for alpha in [0.2, 0.4]:
  print()
  print('*'*100)
  print('Running for ALPHA =', alpha)
  print('*'*23)

  mixup_loss = []
  mixup_train_acc = []
  mixup_test_acc = []

  t2_loss, t2_train_acc, t2_test_acc = train_mixup(model, train_loader,␣
↪test_loader, epochs, optimizer, optim_criter, device, alpha)

  print()

  print('LOSS AND ACCURACY PLOTS FOR ALPHA:', alpha)
```

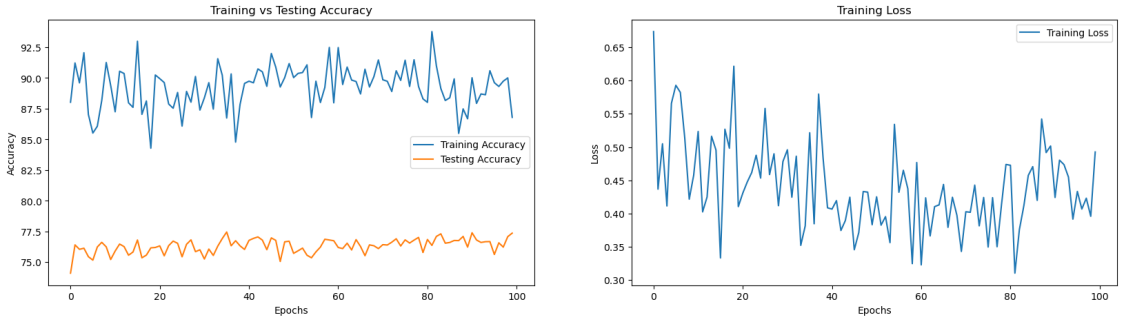
```
  print()

  plt.figure(figsize=(20, 5))

  plt.subplot(1, 2, 1)
  plt.plot(t2_train_acc, label='Training Accuracy')
  plt.plot(t2_test_acc, label='Testing Accuracy')
  plt.xlabel('Epochs')
  plt.ylabel('Accuracy')
  plt.title('Training vs Testing Accuracy')
  plt.legend()

  plt.subplot(1, 2, 2)
  plt.plot(t2_loss, label='Training Loss')
  plt.xlabel('Epochs')
  plt.ylabel('Loss')
  plt.title('Training Loss')
  plt.legend()
  plt.show()

  mixup_loss.append(t2_loss)
  mixup_train_acc.append(t2_train_acc)
  mixup_test_acc.append(t2_test_acc)
```

```
**************************************************************************************
********************
Running for ALPHA = 0.2
************************
Epoch [1/100] - Train Loss: 0.6736 - Train Acc: 88.02% - Test Acc: 74.11%
Epoch [2/100] - Train Loss: 0.4368 - Train Acc: 91.22% - Test Acc: 76.41%
Epoch [3/100] - Train Loss: 0.5050 - Train Acc: 89.60% - Test Acc: 76.05%
Epoch [4/100] - Train Loss: 0.4115 - Train Acc: 92.05% - Test Acc: 76.14%
Epoch [5/100] - Train Loss: 0.5657 - Train Acc: 87.02% - Test Acc: 75.45%
Epoch [6/100] - Train Loss: 0.5929 - Train Acc: 85.51% - Test Acc: 75.17%
Epoch [7/100] - Train Loss: 0.5823 - Train Acc: 86.07% - Test Acc: 76.24%
Epoch [8/100] - Train Loss: 0.5134 - Train Acc: 88.17% - Test Acc: 76.62%
Epoch [9/100] - Train Loss: 0.4217 - Train Acc: 91.26% - Test Acc: 76.25%
Epoch [10/100] - Train Loss: 0.4578 - Train Acc: 89.39% - Test Acc: 75.22%
Epoch [11/100] - Train Loss: 0.5235 - Train Acc: 87.25% - Test Acc: 75.91%
Epoch [12/100] - Train Loss: 0.4026 - Train Acc: 90.55% - Test Acc: 76.48%
Epoch [13/100] - Train Loss: 0.4250 - Train Acc: 90.36% - Test Acc: 76.27%
Epoch [14/100] - Train Loss: 0.5163 - Train Acc: 87.98% - Test Acc: 75.57%
Epoch [15/100] - Train Loss: 0.4955 - Train Acc: 87.61% - Test Acc: 75.83%
Epoch [16/100] - Train Loss: 0.3334 - Train Acc: 92.99% - Test Acc: 76.81%
Epoch [17/100] - Train Loss: 0.5268 - Train Acc: 87.04% - Test Acc: 75.36%
Epoch [18/100] - Train Loss: 0.4983 - Train Acc: 88.13% - Test Acc: 75.58%
```

```
Epoch [19/100] - Train Loss: 0.6215 - Train Acc: 84.28% - Test Acc: 76.17%
Epoch [20/100] - Train Loss: 0.4105 - Train Acc: 90.24% - Test Acc: 76.20%
Epoch [21/100] - Train Loss: 0.4307 - Train Acc: 89.95% - Test Acc: 76.33%
Epoch [22/100] - Train Loss: 0.4474 - Train Acc: 89.63% - Test Acc: 75.52%
Epoch [23/100] - Train Loss: 0.4617 - Train Acc: 87.88% - Test Acc: 76.35%
Epoch [24/100] - Train Loss: 0.4878 - Train Acc: 87.54% - Test Acc: 76.72%
Epoch [25/100] - Train Loss: 0.4535 - Train Acc: 88.82% - Test Acc: 76.54%
Epoch [26/100] - Train Loss: 0.5581 - Train Acc: 86.08% - Test Acc: 75.44%
Epoch [27/100] - Train Loss: 0.4587 - Train Acc: 88.91% - Test Acc: 76.46%
Epoch [28/100] - Train Loss: 0.4899 - Train Acc: 88.03% - Test Acc: 76.82%
Epoch [29/100] - Train Loss: 0.4119 - Train Acc: 90.12% - Test Acc: 75.86%
Epoch [30/100] - Train Loss: 0.4782 - Train Acc: 87.39% - Test Acc: 76.01%
Epoch [31/100] - Train Loss: 0.4960 - Train Acc: 88.38% - Test Acc: 75.26%
Epoch [32/100] - Train Loss: 0.4247 - Train Acc: 89.62% - Test Acc: 76.07%
Epoch [33/100] - Train Loss: 0.4863 - Train Acc: 87.47% - Test Acc: 75.55%
Epoch [34/100] - Train Loss: 0.3523 - Train Acc: 91.57% - Test Acc: 76.31%
Epoch [35/100] - Train Loss: 0.3816 - Train Acc: 90.24% - Test Acc: 76.92%
Epoch [36/100] - Train Loss: 0.5216 - Train Acc: 86.74% - Test Acc: 77.46%
Epoch [37/100] - Train Loss: 0.3847 - Train Acc: 90.33% - Test Acc: 76.34%
Epoch [38/100] - Train Loss: 0.5797 - Train Acc: 84.78% - Test Acc: 76.75%
Epoch [39/100] - Train Loss: 0.4838 - Train Acc: 87.86% - Test Acc: 76.33%
Epoch [40/100] - Train Loss: 0.4087 - Train Acc: 89.56% - Test Acc: 76.04%
Epoch [41/100] - Train Loss: 0.4068 - Train Acc: 89.74% - Test Acc: 76.78%
Epoch [42/100] - Train Loss: 0.4196 - Train Acc: 89.61% - Test Acc: 76.94%
Epoch [43/100] - Train Loss: 0.3747 - Train Acc: 90.74% - Test Acc: 77.06%
Epoch [44/100] - Train Loss: 0.3897 - Train Acc: 90.50% - Test Acc: 76.80%
Epoch [45/100] - Train Loss: 0.4247 - Train Acc: 89.32% - Test Acc: 76.02%
Epoch [46/100] - Train Loss: 0.3458 - Train Acc: 91.99% - Test Acc: 76.98%
Epoch [47/100] - Train Loss: 0.3715 - Train Acc: 90.88% - Test Acc: 76.78%
Epoch [48/100] - Train Loss: 0.4332 - Train Acc: 89.27% - Test Acc: 75.06%
Epoch [49/100] - Train Loss: 0.4323 - Train Acc: 90.02% - Test Acc: 76.66%
Epoch [50/100] - Train Loss: 0.3833 - Train Acc: 91.17% - Test Acc: 76.71%
Epoch [51/100] - Train Loss: 0.4252 - Train Acc: 90.03% - Test Acc: 75.72%
Epoch [52/100] - Train Loss: 0.3827 - Train Acc: 90.37% - Test Acc: 75.93%
Epoch [53/100] - Train Loss: 0.3955 - Train Acc: 90.44% - Test Acc: 76.15%
Epoch [54/100] - Train Loss: 0.3564 - Train Acc: 91.07% - Test Acc: 75.57%
Epoch [55/100] - Train Loss: 0.5342 - Train Acc: 86.78% - Test Acc: 75.36%
Epoch [56/100] - Train Loss: 0.4321 - Train Acc: 89.74% - Test Acc: 75.84%
Epoch [57/100] - Train Loss: 0.4651 - Train Acc: 88.00% - Test Acc: 76.23%
Epoch [58/100] - Train Loss: 0.4375 - Train Acc: 89.23% - Test Acc: 76.87%
Epoch [59/100] - Train Loss: 0.3248 - Train Acc: 92.49% - Test Acc: 76.81%
Epoch [60/100] - Train Loss: 0.4768 - Train Acc: 87.98% - Test Acc: 76.75%
Epoch [61/100] - Train Loss: 0.3230 - Train Acc: 92.47% - Test Acc: 76.20%
Epoch [62/100] - Train Loss: 0.4237 - Train Acc: 89.47% - Test Acc: 76.11%
Epoch [63/100] - Train Loss: 0.3665 - Train Acc: 90.89% - Test Acc: 76.55%
Epoch [64/100] - Train Loss: 0.4105 - Train Acc: 89.83% - Test Acc: 76.00%
Epoch [65/100] - Train Loss: 0.4132 - Train Acc: 89.71% - Test Acc: 76.84%
Epoch [66/100] - Train Loss: 0.4439 - Train Acc: 88.70% - Test Acc: 76.28%
```

```
Epoch [67/100] - Train Loss: 0.3794 - Train Acc: 90.71% - Test Acc: 75.53%
Epoch [68/100] - Train Loss: 0.4246 - Train Acc: 89.27% - Test Acc: 76.41%
Epoch [69/100] - Train Loss: 0.3976 - Train Acc: 90.11% - Test Acc: 76.34%
Epoch [70/100] - Train Loss: 0.3432 - Train Acc: 91.47% - Test Acc: 76.11%
Epoch [71/100] - Train Loss: 0.4029 - Train Acc: 89.85% - Test Acc: 76.43%
Epoch [72/100] - Train Loss: 0.4020 - Train Acc: 89.73% - Test Acc: 76.40%
Epoch [73/100] - Train Loss: 0.4428 - Train Acc: 88.90% - Test Acc: 76.63%
Epoch [74/100] - Train Loss: 0.3816 - Train Acc: 90.58% - Test Acc: 76.91%
Epoch [75/100] - Train Loss: 0.4242 - Train Acc: 89.80% - Test Acc: 76.32%
Epoch [76/100] - Train Loss: 0.3498 - Train Acc: 91.44% - Test Acc: 76.82%
Epoch [77/100] - Train Loss: 0.4240 - Train Acc: 89.31% - Test Acc: 76.56%
Epoch [78/100] - Train Loss: 0.3502 - Train Acc: 91.48% - Test Acc: 76.80%
Epoch [79/100] - Train Loss: 0.4137 - Train Acc: 89.31% - Test Acc: 77.02%
Epoch [80/100] - Train Loss: 0.4740 - Train Acc: 88.30% - Test Acc: 75.79%
Epoch [81/100] - Train Loss: 0.4727 - Train Acc: 88.01% - Test Acc: 76.85%
Epoch [82/100] - Train Loss: 0.3105 - Train Acc: 93.78% - Test Acc: 76.37%
Epoch [83/100] - Train Loss: 0.3759 - Train Acc: 90.99% - Test Acc: 77.10%
Epoch [84/100] - Train Loss: 0.4116 - Train Acc: 89.14% - Test Acc: 77.31%
Epoch [85/100] - Train Loss: 0.4575 - Train Acc: 88.16% - Test Acc: 76.55%
Epoch [86/100] - Train Loss: 0.4706 - Train Acc: 88.39% - Test Acc: 76.61%
Epoch [87/100] - Train Loss: 0.4199 - Train Acc: 89.93% - Test Acc: 76.77%
Epoch [88/100] - Train Loss: 0.5422 - Train Acc: 85.49% - Test Acc: 76.76%
Epoch [89/100] - Train Loss: 0.4917 - Train Acc: 87.48% - Test Acc: 77.10%
Epoch [90/100] - Train Loss: 0.5017 - Train Acc: 86.67% - Test Acc: 76.23%
Epoch [91/100] - Train Loss: 0.4243 - Train Acc: 90.01% - Test Acc: 77.40%
Epoch [92/100] - Train Loss: 0.4804 - Train Acc: 87.93% - Test Acc: 76.81%
Epoch [93/100] - Train Loss: 0.4737 - Train Acc: 88.70% - Test Acc: 76.61%
Epoch [94/100] - Train Loss: 0.4550 - Train Acc: 88.63% - Test Acc: 76.67%
Epoch [95/100] - Train Loss: 0.3917 - Train Acc: 90.59% - Test Acc: 76.68%
Epoch [96/100] - Train Loss: 0.4333 - Train Acc: 89.62% - Test Acc: 75.63%
Epoch [97/100] - Train Loss: 0.4069 - Train Acc: 89.31% - Test Acc: 76.58%
Epoch [98/100] - Train Loss: 0.4232 - Train Acc: 89.74% - Test Acc: 76.23%
Epoch [99/100] - Train Loss: 0.3960 - Train Acc: 90.00% - Test Acc: 77.08%
Epoch [100/100] - Train Loss: 0.4927 - Train Acc: 86.79% - Test Acc: 77.37%
```

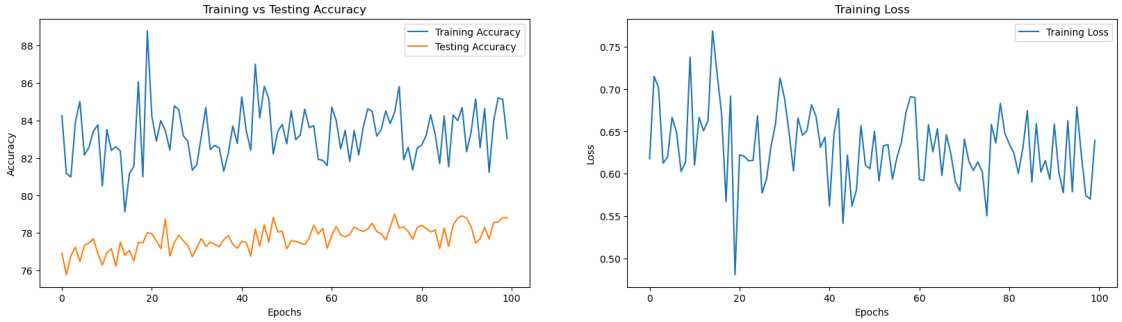
LOSS AND ACCURACY PLOTS FOR ALPHA: 0.2



```
**************************************************************************************
**********
Running for ALPHA = 0.4
**********
Epoch [1/100] - Train Loss: 0.6177 - Train Acc: 84.25% - Test Acc: 76.91%
Epoch [2/100] - Train Loss: 0.7150 - Train Acc: 81.17% - Test Acc: 75.77%
Epoch [3/100] - Train Loss: 0.7013 - Train Acc: 80.99% - Test Acc: 76.78%
Epoch [4/100] - Train Loss: 0.6123 - Train Acc: 83.92% - Test Acc: 77.24%
Epoch [5/100] - Train Loss: 0.6197 - Train Acc: 85.02% - Test Acc: 76.47%
Epoch [6/100] - Train Loss: 0.6666 - Train Acc: 82.15% - Test Acc: 77.33%
Epoch [7/100] - Train Loss: 0.6487 - Train Acc: 82.56% - Test Acc: 77.46%
Epoch [8/100] - Train Loss: 0.6027 - Train Acc: 83.43% - Test Acc: 77.69%
Epoch [9/100] - Train Loss: 0.6135 - Train Acc: 83.77% - Test Acc: 76.89%
Epoch [10/100] - Train Loss: 0.7376 - Train Acc: 80.51% - Test Acc: 76.28%
Epoch [11/100] - Train Loss: 0.6102 - Train Acc: 83.51% - Test Acc: 76.95%
Epoch [12/100] - Train Loss: 0.6664 - Train Acc: 82.40% - Test Acc: 77.16%
Epoch [13/100] - Train Loss: 0.6508 - Train Acc: 82.60% - Test Acc: 76.22%
Epoch [14/100] - Train Loss: 0.6627 - Train Acc: 82.37% - Test Acc: 77.51%
Epoch [15/100] - Train Loss: 0.7686 - Train Acc: 79.13% - Test Acc: 76.80%
Epoch [16/100] - Train Loss: 0.7188 - Train Acc: 81.16% - Test Acc: 77.07%
Epoch [17/100] - Train Loss: 0.6712 - Train Acc: 81.55% - Test Acc: 76.50%
Epoch [18/100] - Train Loss: 0.5674 - Train Acc: 86.07% - Test Acc: 77.49%
Epoch [19/100] - Train Loss: 0.6917 - Train Acc: 80.98% - Test Acc: 77.47%
Epoch [20/100] - Train Loss: 0.4808 - Train Acc: 88.78% - Test Acc: 78.01%
Epoch [21/100] - Train Loss: 0.6221 - Train Acc: 84.22% - Test Acc: 77.96%
Epoch [22/100] - Train Loss: 0.6207 - Train Acc: 82.91% - Test Acc: 77.57%
Epoch [23/100] - Train Loss: 0.6152 - Train Acc: 83.99% - Test Acc: 77.16%
Epoch [24/100] - Train Loss: 0.6156 - Train Acc: 83.45% - Test Acc: 78.73%
Epoch [25/100] - Train Loss: 0.6683 - Train Acc: 82.42% - Test Acc: 76.77%
Epoch [26/100] - Train Loss: 0.5772 - Train Acc: 84.78% - Test Acc: 77.50%
Epoch [27/100] - Train Loss: 0.5936 - Train Acc: 84.58% - Test Acc: 77.88%
Epoch [28/100] - Train Loss: 0.6321 - Train Acc: 83.18% - Test Acc: 77.57%
Epoch [29/100] - Train Loss: 0.6587 - Train Acc: 82.89% - Test Acc: 77.34%
Epoch [30/100] - Train Loss: 0.7131 - Train Acc: 81.35% - Test Acc: 76.73%
Epoch [31/100] - Train Loss: 0.6894 - Train Acc: 81.65% - Test Acc: 77.22%
Epoch [32/100] - Train Loss: 0.6493 - Train Acc: 83.16% - Test Acc: 77.70%
Epoch [33/100] - Train Loss: 0.6032 - Train Acc: 84.68% - Test Acc: 77.28%
Epoch [34/100] - Train Loss: 0.6659 - Train Acc: 82.45% - Test Acc: 77.51%
Epoch [35/100] - Train Loss: 0.6454 - Train Acc: 82.67% - Test Acc: 77.39%
Epoch [36/100] - Train Loss: 0.6508 - Train Acc: 82.53% - Test Acc: 77.26%
Epoch [37/100] - Train Loss: 0.6813 - Train Acc: 81.29% - Test Acc: 77.64%
Epoch [38/100] - Train Loss: 0.6679 - Train Acc: 82.25% - Test Acc: 77.86%
Epoch [39/100] - Train Loss: 0.6313 - Train Acc: 83.71% - Test Acc: 77.39%
Epoch [40/100] - Train Loss: 0.6431 - Train Acc: 82.77% - Test Acc: 77.18%
```

```
Epoch [41/100] - Train Loss: 0.5618 - Train Acc: 85.25% - Test Acc: 77.56%
Epoch [42/100] - Train Loss: 0.6467 - Train Acc: 83.44% - Test Acc: 77.49%
Epoch [43/100] - Train Loss: 0.6766 - Train Acc: 82.41% - Test Acc: 76.77%
Epoch [44/100] - Train Loss: 0.5413 - Train Acc: 87.00% - Test Acc: 78.20%
Epoch [45/100] - Train Loss: 0.6219 - Train Acc: 84.13% - Test Acc: 77.29%
Epoch [46/100] - Train Loss: 0.5617 - Train Acc: 85.82% - Test Acc: 78.44%
Epoch [47/100] - Train Loss: 0.5812 - Train Acc: 85.17% - Test Acc: 77.51%
Epoch [48/100] - Train Loss: 0.6567 - Train Acc: 82.21% - Test Acc: 78.84%
Epoch [49/100] - Train Loss: 0.6102 - Train Acc: 83.42% - Test Acc: 78.05%
Epoch [50/100] - Train Loss: 0.6058 - Train Acc: 83.79% - Test Acc: 78.09%
Epoch [51/100] - Train Loss: 0.6502 - Train Acc: 82.75% - Test Acc: 77.15%
Epoch [52/100] - Train Loss: 0.5915 - Train Acc: 84.51% - Test Acc: 77.59%
Epoch [53/100] - Train Loss: 0.6330 - Train Acc: 82.97% - Test Acc: 77.55%
Epoch [54/100] - Train Loss: 0.6343 - Train Acc: 83.22% - Test Acc: 77.46%
Epoch [55/100] - Train Loss: 0.5937 - Train Acc: 84.61% - Test Acc: 77.37%
Epoch [56/100] - Train Loss: 0.6190 - Train Acc: 83.61% - Test Acc: 77.74%
Epoch [57/100] - Train Loss: 0.6371 - Train Acc: 83.72% - Test Acc: 78.41%
Epoch [58/100] - Train Loss: 0.6719 - Train Acc: 81.93% - Test Acc: 77.96%
Epoch [59/100] - Train Loss: 0.6908 - Train Acc: 81.87% - Test Acc: 78.23%
Epoch [60/100] - Train Loss: 0.6901 - Train Acc: 81.60% - Test Acc: 77.18%
Epoch [61/100] - Train Loss: 0.5931 - Train Acc: 84.72% - Test Acc: 77.88%
Epoch [62/100] - Train Loss: 0.5919 - Train Acc: 83.99% - Test Acc: 78.34%
Epoch [63/100] - Train Loss: 0.6577 - Train Acc: 82.49% - Test Acc: 77.90%
Epoch [64/100] - Train Loss: 0.6258 - Train Acc: 83.47% - Test Acc: 77.79%
Epoch [65/100] - Train Loss: 0.6531 - Train Acc: 81.81% - Test Acc: 77.91%
Epoch [66/100] - Train Loss: 0.5979 - Train Acc: 83.46% - Test Acc: 78.31%
Epoch [67/100] - Train Loss: 0.6461 - Train Acc: 82.16% - Test Acc: 78.17%
Epoch [68/100] - Train Loss: 0.6231 - Train Acc: 83.63% - Test Acc: 78.08%
Epoch [69/100] - Train Loss: 0.5906 - Train Acc: 84.62% - Test Acc: 78.20%
Epoch [70/100] - Train Loss: 0.5797 - Train Acc: 84.49% - Test Acc: 78.52%
Epoch [71/100] - Train Loss: 0.6408 - Train Acc: 83.17% - Test Acc: 78.08%
Epoch [72/100] - Train Loss: 0.6145 - Train Acc: 83.52% - Test Acc: 77.95%
Epoch [73/100] - Train Loss: 0.6038 - Train Acc: 84.51% - Test Acc: 77.62%
Epoch [74/100] - Train Loss: 0.6140 - Train Acc: 83.83% - Test Acc: 78.31%
Epoch [75/100] - Train Loss: 0.6018 - Train Acc: 84.46% - Test Acc: 79.00%
Epoch [76/100] - Train Loss: 0.5502 - Train Acc: 85.82% - Test Acc: 78.26%
Epoch [77/100] - Train Loss: 0.6582 - Train Acc: 81.90% - Test Acc: 78.32%
Epoch [78/100] - Train Loss: 0.6362 - Train Acc: 82.57% - Test Acc: 78.07%
Epoch [79/100] - Train Loss: 0.6829 - Train Acc: 81.36% - Test Acc: 77.67%
Epoch [80/100] - Train Loss: 0.6475 - Train Acc: 82.53% - Test Acc: 78.29%
Epoch [81/100] - Train Loss: 0.6349 - Train Acc: 82.69% - Test Acc: 78.41%
Epoch [82/100] - Train Loss: 0.6245 - Train Acc: 83.20% - Test Acc: 78.23%
Epoch [83/100] - Train Loss: 0.6003 - Train Acc: 84.30% - Test Acc: 78.05%
Epoch [84/100] - Train Loss: 0.6302 - Train Acc: 83.27% - Test Acc: 78.16%
Epoch [85/100] - Train Loss: 0.6745 - Train Acc: 81.70% - Test Acc: 77.16%
Epoch [86/100] - Train Loss: 0.5899 - Train Acc: 84.24% - Test Acc: 78.27%
Epoch [87/100] - Train Loss: 0.6590 - Train Acc: 81.53% - Test Acc: 77.28%
Epoch [88/100] - Train Loss: 0.6020 - Train Acc: 84.30% - Test Acc: 78.42%
```

```
Epoch [89/100] - Train Loss: 0.6155 - Train Acc: 83.99% - Test Acc: 78.79%
Epoch [90/100] - Train Loss: 0.5932 - Train Acc: 84.69% - Test Acc: 78.91%
Epoch [91/100] - Train Loss: 0.6584 - Train Acc: 82.33% - Test Acc: 78.79%
Epoch [92/100] - Train Loss: 0.6013 - Train Acc: 83.41% - Test Acc: 78.33%
Epoch [93/100] - Train Loss: 0.5776 - Train Acc: 85.15% - Test Acc: 77.46%
Epoch [94/100] - Train Loss: 0.6624 - Train Acc: 82.55% - Test Acc: 77.69%
Epoch [95/100] - Train Loss: 0.5787 - Train Acc: 84.63% - Test Acc: 78.30%
Epoch [96/100] - Train Loss: 0.6789 - Train Acc: 81.23% - Test Acc: 77.67%
Epoch [97/100] - Train Loss: 0.6227 - Train Acc: 83.97% - Test Acc: 78.55%
Epoch [98/100] - Train Loss: 0.5742 - Train Acc: 85.20% - Test Acc: 78.58%
Epoch [99/100] - Train Loss: 0.5701 - Train Acc: 85.13% - Test Acc: 78.81%
Epoch [100/100] - Train Loss: 0.6393 - Train Acc: 83.04% - Test Acc: 78.80%

LOSS AND ACCURACY PLOTS FOR ALPHA: 0.4
```



[36]:
```
print('*'*100)
print('Final Test Accuracy:', np.mean(mixup_test_acc), '%')
print('*'*100)
print()
```

```
****************************************************************************************************
Final Test Accuracy: 77.7185 %
****************************************************************************************************
```

1.3.3 TASK 3 - (4 pts) *Cutout Augmentation* is based on the paper https://arxiv.org/pdf/1708.04552.pdf.

For each training image with 50% probability you keep the image intact. With 50% probability, select a random pixel which serves as the center of your cutout mask. Then, set the square mask of size K × K pixels around this center pixel to be zero. Note that part of the mask is allowed to be outside of the image. For visualization, see Figure 1 of the paper.

TODO: Implement and use cutout augmentation with K = 16 and report the results.

Helper Functions

[20]:
```
# Function to implement CUTOUT AUGMENTATION
def cutout(imgs, mask_size):
    if mask_size <= 0:
        return imgs
    h, w = imgs.shape[2], imgs.shape[3]
    mask_value = imgs.min()

    cutout_imgs = imgs.clone()  # Create a copy of the original images

    for idx in range(cutout_imgs.shape[0]):
        img = cutout_imgs[idx]
        for _ in range(1):
            top = np.random.randint(0 - mask_size // 2, h - mask_size)
            left = np.random.randint(0 - mask_size // 2, w - mask_size)
            bottom = top + mask_size
            right = left + mask_size

            if top < 0:
                top = 0
            if left < 0:
                left = 0

            img[:, top:bottom, left:right] = mask_value
    return cutout_imgs
```

Main Function

[21]:
```
def train_cutout(model, train_loader, test_loader, epochs, optimizer,␣
 ↪criterion, device, mask_size):
    train_loss_list = []
    train_acc_list = []
    test_acc_list = []

    for epoch in range(epochs):
        model.train()
        train_loss = 0.0
        train_acc = 0.0
        total_samples = 0

        for images, labels in train_loader:
          images, labels = images.to(device), labels.to(device)

          # If labels are one-hot encoded, convert them to class indices
          if labels.dim() > 1:
              labels = torch.argmax(labels, dim=1)
```

```
            # Apply cutout augmentation
            images = cutout(images, mask_size=mask_size)

            outputs = model(images)
            loss = criterion(outputs, labels)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            _, predicted = torch.max(outputs.data, 1)
            total_samples += labels.size(0)
            correct = (predicted == labels).sum().item()
            train_loss += loss.item() * images.size(0)
            train_acc += correct * 100

        train_loss /= total_samples
        train_acc /= total_samples

        # Evaluate on the test set
        model.eval()
        with torch.no_grad():
            correct = 0
            total = 0
            for images, labels in test_loader:
                images, labels = images.to(device), labels.to(device)
                outputs = model(images)
                _, predicted = torch.max(outputs.data, 1)
                total += labels.size(0)
                correct += (predicted == labels).sum().item()
        test_acc = 100 * correct / total

        train_loss_list.append(train_loss)
        train_acc_list.append(train_acc)
        test_acc_list.append(test_acc)

        print(f"Epoch [{epoch+1}/{epochs}] - Train Loss: {train_loss:.4f} -␣
↪Train Acc: {train_acc:.2f}% - Test Acc: {test_acc:.2f}%")

    return train_loss_list, train_acc_list, test_acc_list
```

Testing the Train Function

[22]:
```
mask_size = 16
t3_loss, t3_train_acc, t3_test_acc = train_cutout(model, train_loader,␣
↪test_loader, epochs, optimizer, optim_criter, device, mask_size)
```

```
Epoch [1/100] - Train Loss: 0.6413 - Train Acc: 78.84% - Test Acc: 77.49%
Epoch [2/100] - Train Loss: 0.4391 - Train Acc: 85.10% - Test Acc: 76.95%
```

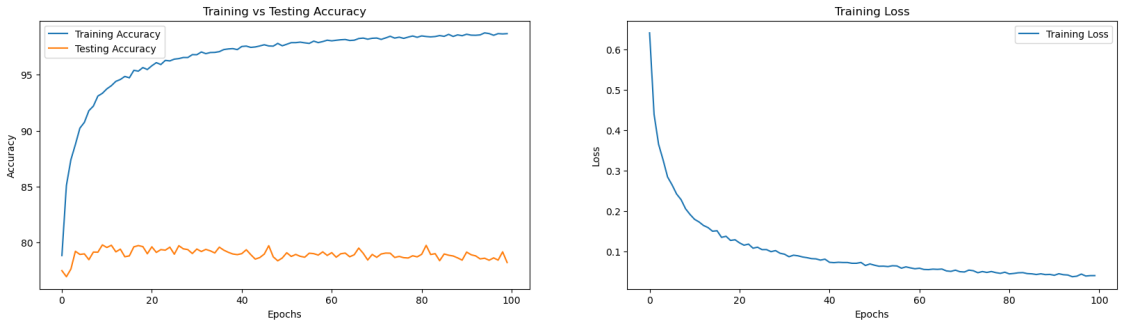
```
Epoch [3/100] - Train Loss: 0.3652 - Train Acc: 87.40% - Test Acc: 77.63%
Epoch [4/100] - Train Loss: 0.3265 - Train Acc: 88.76% - Test Acc: 79.22%
Epoch [5/100] - Train Loss: 0.2841 - Train Acc: 90.22% - Test Acc: 78.94%
Epoch [6/100] - Train Loss: 0.2645 - Train Acc: 90.74% - Test Acc: 78.99%
Epoch [7/100] - Train Loss: 0.2419 - Train Acc: 91.78% - Test Acc: 78.47%
Epoch [8/100] - Train Loss: 0.2278 - Train Acc: 92.20% - Test Acc: 79.15%
Epoch [9/100] - Train Loss: 0.2052 - Train Acc: 93.09% - Test Acc: 79.14%
Epoch [10/100] - Train Loss: 0.1910 - Train Acc: 93.34% - Test Acc: 79.79%
Epoch [11/100] - Train Loss: 0.1791 - Train Acc: 93.73% - Test Acc: 79.56%
Epoch [12/100] - Train Loss: 0.1725 - Train Acc: 94.02% - Test Acc: 79.76%
Epoch [13/100] - Train Loss: 0.1637 - Train Acc: 94.40% - Test Acc: 79.17%
Epoch [14/100] - Train Loss: 0.1585 - Train Acc: 94.57% - Test Acc: 79.41%
Epoch [15/100] - Train Loss: 0.1496 - Train Acc: 94.85% - Test Acc: 78.73%
Epoch [16/100] - Train Loss: 0.1507 - Train Acc: 94.72% - Test Acc: 78.81%
Epoch [17/100] - Train Loss: 0.1343 - Train Acc: 95.39% - Test Acc: 79.62%
Epoch [18/100] - Train Loss: 0.1370 - Train Acc: 95.32% - Test Acc: 79.73%
Epoch [19/100] - Train Loss: 0.1267 - Train Acc: 95.64% - Test Acc: 79.64%
Epoch [20/100] - Train Loss: 0.1284 - Train Acc: 95.46% - Test Acc: 79.00%
Epoch [21/100] - Train Loss: 0.1207 - Train Acc: 95.80% - Test Acc: 79.62%
Epoch [22/100] - Train Loss: 0.1149 - Train Acc: 96.07% - Test Acc: 79.12%
Epoch [23/100] - Train Loss: 0.1176 - Train Acc: 95.91% - Test Acc: 79.37%
Epoch [24/100] - Train Loss: 0.1075 - Train Acc: 96.28% - Test Acc: 79.32%
Epoch [25/100] - Train Loss: 0.1099 - Train Acc: 96.23% - Test Acc: 79.59%
Epoch [26/100] - Train Loss: 0.1039 - Train Acc: 96.39% - Test Acc: 78.96%
Epoch [27/100] - Train Loss: 0.1039 - Train Acc: 96.44% - Test Acc: 79.72%
Epoch [28/100] - Train Loss: 0.0989 - Train Acc: 96.54% - Test Acc: 79.44%
Epoch [29/100] - Train Loss: 0.1015 - Train Acc: 96.53% - Test Acc: 79.37%
Epoch [30/100] - Train Loss: 0.0950 - Train Acc: 96.79% - Test Acc: 79.02%
Epoch [31/100] - Train Loss: 0.0926 - Train Acc: 96.78% - Test Acc: 79.42%
Epoch [32/100] - Train Loss: 0.0863 - Train Acc: 97.03% - Test Acc: 79.20%
Epoch [33/100] - Train Loss: 0.0902 - Train Acc: 96.88% - Test Acc: 79.39%
Epoch [34/100] - Train Loss: 0.0886 - Train Acc: 96.98% - Test Acc: 79.26%
Epoch [35/100] - Train Loss: 0.0857 - Train Acc: 96.99% - Test Acc: 79.07%
Epoch [36/100] - Train Loss: 0.0840 - Train Acc: 97.06% - Test Acc: 79.59%
Epoch [37/100] - Train Loss: 0.0816 - Train Acc: 97.24% - Test Acc: 79.32%
Epoch [38/100] - Train Loss: 0.0810 - Train Acc: 97.30% - Test Acc: 79.13%
Epoch [39/100] - Train Loss: 0.0779 - Train Acc: 97.33% - Test Acc: 78.98%
Epoch [40/100] - Train Loss: 0.0803 - Train Acc: 97.24% - Test Acc: 78.92%
Epoch [41/100] - Train Loss: 0.0725 - Train Acc: 97.52% - Test Acc: 79.01%
Epoch [42/100] - Train Loss: 0.0717 - Train Acc: 97.56% - Test Acc: 79.36%
Epoch [43/100] - Train Loss: 0.0726 - Train Acc: 97.44% - Test Acc: 78.92%
Epoch [44/100] - Train Loss: 0.0720 - Train Acc: 97.48% - Test Acc: 78.52%
Epoch [45/100] - Train Loss: 0.0720 - Train Acc: 97.57% - Test Acc: 78.66%
Epoch [46/100] - Train Loss: 0.0701 - Train Acc: 97.67% - Test Acc: 78.97%
Epoch [47/100] - Train Loss: 0.0700 - Train Acc: 97.57% - Test Acc: 79.72%
Epoch [48/100] - Train Loss: 0.0722 - Train Acc: 97.55% - Test Acc: 78.72%
Epoch [49/100] - Train Loss: 0.0645 - Train Acc: 97.80% - Test Acc: 78.37%
Epoch [50/100] - Train Loss: 0.0685 - Train Acc: 97.58% - Test Acc: 78.62%
```

```
Epoch [51/100] - Train Loss: 0.0653 - Train Acc: 97.72% - Test Acc: 79.09%
Epoch [52/100] - Train Loss: 0.0627 - Train Acc: 97.86% - Test Acc: 78.76%
Epoch [53/100] - Train Loss: 0.0630 - Train Acc: 97.86% - Test Acc: 78.94%
Epoch [54/100] - Train Loss: 0.0619 - Train Acc: 97.91% - Test Acc: 78.77%
Epoch [55/100] - Train Loss: 0.0640 - Train Acc: 97.85% - Test Acc: 78.69%
Epoch [56/100] - Train Loss: 0.0634 - Train Acc: 97.79% - Test Acc: 79.05%
Epoch [57/100] - Train Loss: 0.0579 - Train Acc: 98.00% - Test Acc: 79.01%
Epoch [58/100] - Train Loss: 0.0612 - Train Acc: 97.87% - Test Acc: 78.88%
Epoch [59/100] - Train Loss: 0.0586 - Train Acc: 97.96% - Test Acc: 79.17%
Epoch [60/100] - Train Loss: 0.0565 - Train Acc: 98.08% - Test Acc: 78.85%
Epoch [61/100] - Train Loss: 0.0578 - Train Acc: 98.02% - Test Acc: 79.10%
Epoch [62/100] - Train Loss: 0.0548 - Train Acc: 98.07% - Test Acc: 78.69%
Epoch [63/100] - Train Loss: 0.0544 - Train Acc: 98.12% - Test Acc: 79.00%
Epoch [64/100] - Train Loss: 0.0556 - Train Acc: 98.15% - Test Acc: 79.06%
Epoch [65/100] - Train Loss: 0.0550 - Train Acc: 98.05% - Test Acc: 78.73%
Epoch [66/100] - Train Loss: 0.0558 - Train Acc: 98.08% - Test Acc: 78.91%
Epoch [67/100] - Train Loss: 0.0511 - Train Acc: 98.23% - Test Acc: 79.51%
Epoch [68/100] - Train Loss: 0.0501 - Train Acc: 98.27% - Test Acc: 79.04%
Epoch [69/100] - Train Loss: 0.0530 - Train Acc: 98.17% - Test Acc: 78.44%
Epoch [70/100] - Train Loss: 0.0492 - Train Acc: 98.25% - Test Acc: 78.94%
Epoch [71/100] - Train Loss: 0.0485 - Train Acc: 98.27% - Test Acc: 78.68%
Epoch [72/100] - Train Loss: 0.0531 - Train Acc: 98.15% - Test Acc: 78.98%
Epoch [73/100] - Train Loss: 0.0515 - Train Acc: 98.29% - Test Acc: 79.06%
Epoch [74/100] - Train Loss: 0.0466 - Train Acc: 98.43% - Test Acc: 79.05%
Epoch [75/100] - Train Loss: 0.0496 - Train Acc: 98.27% - Test Acc: 78.67%
Epoch [76/100] - Train Loss: 0.0473 - Train Acc: 98.35% - Test Acc: 78.76%
Epoch [77/100] - Train Loss: 0.0497 - Train Acc: 98.24% - Test Acc: 78.65%
Epoch [78/100] - Train Loss: 0.0468 - Train Acc: 98.36% - Test Acc: 78.62%
Epoch [79/100] - Train Loss: 0.0451 - Train Acc: 98.46% - Test Acc: 78.82%
Epoch [80/100] - Train Loss: 0.0479 - Train Acc: 98.34% - Test Acc: 78.72%
Epoch [81/100] - Train Loss: 0.0438 - Train Acc: 98.46% - Test Acc: 78.95%
Epoch [82/100] - Train Loss: 0.0450 - Train Acc: 98.41% - Test Acc: 79.75%
Epoch [83/100] - Train Loss: 0.0465 - Train Acc: 98.38% - Test Acc: 78.93%
Epoch [84/100] - Train Loss: 0.0469 - Train Acc: 98.41% - Test Acc: 79.01%
Epoch [85/100] - Train Loss: 0.0445 - Train Acc: 98.50% - Test Acc: 78.38%
Epoch [86/100] - Train Loss: 0.0439 - Train Acc: 98.43% - Test Acc: 78.98%
Epoch [87/100] - Train Loss: 0.0422 - Train Acc: 98.60% - Test Acc: 78.87%
Epoch [88/100] - Train Loss: 0.0440 - Train Acc: 98.42% - Test Acc: 78.80%
Epoch [89/100] - Train Loss: 0.0420 - Train Acc: 98.56% - Test Acc: 78.63%
Epoch [90/100] - Train Loss: 0.0425 - Train Acc: 98.48% - Test Acc: 78.43%
Epoch [91/100] - Train Loss: 0.0403 - Train Acc: 98.61% - Test Acc: 79.15%
Epoch [92/100] - Train Loss: 0.0442 - Train Acc: 98.53% - Test Acc: 78.91%
Epoch [93/100] - Train Loss: 0.0417 - Train Acc: 98.52% - Test Acc: 78.81%
Epoch [94/100] - Train Loss: 0.0410 - Train Acc: 98.56% - Test Acc: 78.54%
Epoch [95/100] - Train Loss: 0.0368 - Train Acc: 98.73% - Test Acc: 78.60%
Epoch [96/100] - Train Loss: 0.0381 - Train Acc: 98.67% - Test Acc: 78.42%
Epoch [97/100] - Train Loss: 0.0433 - Train Acc: 98.52% - Test Acc: 78.63%
Epoch [98/100] - Train Loss: 0.0385 - Train Acc: 98.67% - Test Acc: 78.43%
```

```
Epoch [99/100] - Train Loss: 0.0395 - Train Acc: 98.64% - Test Acc: 79.17%
Epoch [100/100] - Train Loss: 0.0395 - Train Acc: 98.67% - Test Acc: 78.22%
```

[23]:
```python
print('*'*10, 'LOSS AND ACCURACY PLOTS FOR CUTOUT AUGMENTATION', '*'*10)

print()

plt.figure(figsize=(20, 5))

plt.subplot(1, 2, 1)
plt.plot(t3_train_acc, label='Training Accuracy')
plt.plot(t3_test_acc, label='Testing Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training vs Testing Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(t3_loss, label='Training Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training Loss')
plt.legend()
plt.show()

print()

print('*'*100)
print('Final Test Accuracy:', t3_test_acc[-1], '%')
print('*'*100)
```

```
********** LOSS AND ACCURACY PLOTS FOR CUTOUT AUGMENTATION **********
```




```
****************************************************************************************
```

```
********************
Final Test Accuracy: 78.22 %
********************************************************************************
********************
```

1.3.4 TASK 4 - (4 pts) *Standard Augmentation* applies horizontal flip and random shifts. See the website https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/

for illustrations.

Given an input image, first you shift it left-right and up-down as follows. Pick two independent integers k_1, k_2 uniformly between [−K,K] range. Move image upwards by k_1 and rightwards by k_2 pixels (negative value means downwards and leftwards). Zero pad the missing pixels. After this random shift, with 50% probability, apply a horizontal flip on the image.

TODO: Implement standard augmentation with K = 4 and report the results.

Main Functions

[24]:
```python
# Function to Implement Standard Augmentation
def standard_augmentation(imgs, K):
    h, w = imgs.shape[2], imgs.shape[3]
    new_imgs = torch.zeros_like(imgs)

    for idx in range(imgs.shape[0]):
        img = imgs[idx]

        k1 = np.random.randint(-K, K)
        k2 = np.random.randint(-K, K)
        flip = np.random.rand() > 0.5

        img = torch.roll(img, shifts=[k1, k2], dims=[1, 2])

        if flip:
            img = torch.flip(img, [2])

        new_imgs[idx] = img

    return new_imgs
```

[25]:
```python
def train_standard_aug(model, train_loader, test_loader, epochs, optimizer,␣
↪criterion, device):
    train_loss_list = []
    train_acc_list = []
    test_acc_list = []

    for epoch in range(epochs):
        model.train()
```

```
        train_loss = 0.0
        train_acc = 0.0
        total_samples = 0

        for images, labels in train_loader:
            images, labels = images.to(device), labels.to(device)

            # Apply standard augmentation
            images = standard_augmentation(images, K=4)

            outputs = model(images)

            # Convert one-hot encoded labels to class indices
            _, labels = torch.max(labels, 1)

            loss = criterion(outputs, labels)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            _, predicted = torch.max(outputs.data, 1)
            total_samples += labels.size(0)
            correct = (predicted == labels).sum().item()
            train_loss += loss.item() * images.size(0)
            train_acc += correct * 100

        train_loss /= total_samples
        train_acc /= total_samples

        # Evaluate on the test set
        model.eval()
        with torch.no_grad():
            correct = 0
            total = 0
            for images, labels in test_loader:
                images, labels = images.to(device), labels.to(device)
                outputs = model(images)
                _, predicted = torch.max(outputs.data, 1)
                total += labels.size(0)
                correct += (predicted == labels).sum().item()
        test_acc = 100 * correct / total

        train_loss_list.append(train_loss)
        train_acc_list.append(train_acc)
        test_acc_list.append(test_acc)
```

```
        print(f"Epoch [{epoch+1}/{epochs}] - Train Loss: {train_loss:.4f} -␣
↪Train Acc: {train_acc:.2f}% - Test Acc: {test_acc:.2f}%")

    return train_loss_list, train_acc_list, test_acc_list
```

Testing the Train Function

[26]:
```
t4_loss, t4_train_acc, t4_test_acc = train_standard_aug(model, train_loader,␣
↪test_loader, epochs, optimizer, optim_criter, device)
```

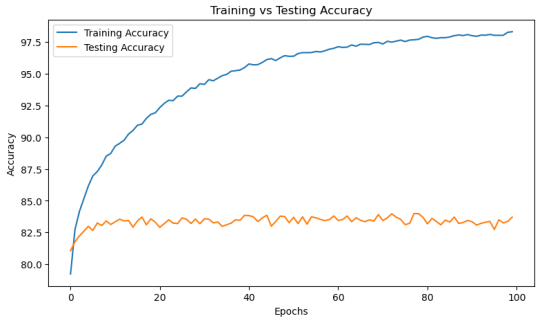
```
Epoch [1/100] - Train Loss: 0.6520 - Train Acc: 79.23% - Test Acc: 81.07%
Epoch [2/100] - Train Loss: 0.5044 - Train Acc: 82.75% - Test Acc: 81.78%
Epoch [3/100] - Train Loss: 0.4597 - Train Acc: 84.17% - Test Acc: 82.23%
Epoch [4/100] - Train Loss: 0.4319 - Train Acc: 85.17% - Test Acc: 82.61%
Epoch [5/100] - Train Loss: 0.4066 - Train Acc: 86.17% - Test Acc: 82.99%
Epoch [6/100] - Train Loss: 0.3826 - Train Acc: 86.95% - Test Acc: 82.66%
Epoch [7/100] - Train Loss: 0.3653 - Train Acc: 87.30% - Test Acc: 83.25%
Epoch [8/100] - Train Loss: 0.3551 - Train Acc: 87.82% - Test Acc: 83.05%
Epoch [9/100] - Train Loss: 0.3359 - Train Acc: 88.51% - Test Acc: 83.41%
Epoch [10/100] - Train Loss: 0.3289 - Train Acc: 88.72% - Test Acc: 83.13%
Epoch [11/100] - Train Loss: 0.3107 - Train Acc: 89.29% - Test Acc: 83.34%
Epoch [12/100] - Train Loss: 0.3010 - Train Acc: 89.51% - Test Acc: 83.55%
Epoch [13/100] - Train Loss: 0.2928 - Train Acc: 89.76% - Test Acc: 83.41%
Epoch [14/100] - Train Loss: 0.2835 - Train Acc: 90.24% - Test Acc: 83.44%
Epoch [15/100] - Train Loss: 0.2721 - Train Acc: 90.53% - Test Acc: 82.92%
Epoch [16/100] - Train Loss: 0.2602 - Train Acc: 90.94% - Test Acc: 83.41%
Epoch [17/100] - Train Loss: 0.2583 - Train Acc: 91.03% - Test Acc: 83.72%
Epoch [18/100] - Train Loss: 0.2428 - Train Acc: 91.48% - Test Acc: 83.10%
Epoch [19/100] - Train Loss: 0.2384 - Train Acc: 91.80% - Test Acc: 83.58%
Epoch [20/100] - Train Loss: 0.2338 - Train Acc: 91.92% - Test Acc: 83.31%
Epoch [21/100] - Train Loss: 0.2229 - Train Acc: 92.33% - Test Acc: 82.90%
Epoch [22/100] - Train Loss: 0.2133 - Train Acc: 92.66% - Test Acc: 83.20%
Epoch [23/100] - Train Loss: 0.2048 - Train Acc: 92.90% - Test Acc: 83.50%
Epoch [24/100] - Train Loss: 0.2045 - Train Acc: 92.88% - Test Acc: 83.24%
Epoch [25/100] - Train Loss: 0.1954 - Train Acc: 93.23% - Test Acc: 83.20%
Epoch [26/100] - Train Loss: 0.1928 - Train Acc: 93.23% - Test Acc: 83.65%
Epoch [27/100] - Train Loss: 0.1847 - Train Acc: 93.57% - Test Acc: 83.54%
Epoch [28/100] - Train Loss: 0.1760 - Train Acc: 93.88% - Test Acc: 83.21%
Epoch [29/100] - Train Loss: 0.1778 - Train Acc: 93.84% - Test Acc: 83.56%
Epoch [30/100] - Train Loss: 0.1678 - Train Acc: 94.21% - Test Acc: 83.19%
Epoch [31/100] - Train Loss: 0.1670 - Train Acc: 94.16% - Test Acc: 83.58%
Epoch [32/100] - Train Loss: 0.1563 - Train Acc: 94.53% - Test Acc: 83.55%
Epoch [33/100] - Train Loss: 0.1608 - Train Acc: 94.44% - Test Acc: 83.26%
Epoch [34/100] - Train Loss: 0.1542 - Train Acc: 94.65% - Test Acc: 83.33%
Epoch [35/100] - Train Loss: 0.1484 - Train Acc: 94.84% - Test Acc: 82.98%
Epoch [36/100] - Train Loss: 0.1436 - Train Acc: 94.94% - Test Acc: 83.11%
Epoch [37/100] - Train Loss: 0.1406 - Train Acc: 95.19% - Test Acc: 83.24%
Epoch [38/100] - Train Loss: 0.1348 - Train Acc: 95.23% - Test Acc: 83.50%
```

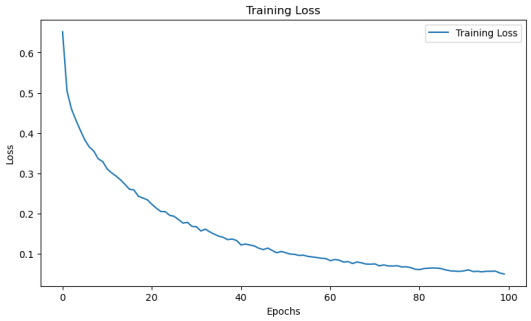
```
Epoch [39/100] - Train Loss: 0.1363 - Train Acc: 95.29% - Test Acc: 83.46%
Epoch [40/100] - Train Loss: 0.1327 - Train Acc: 95.48% - Test Acc: 83.85%
Epoch [41/100] - Train Loss: 0.1217 - Train Acc: 95.77% - Test Acc: 83.83%
Epoch [42/100] - Train Loss: 0.1239 - Train Acc: 95.69% - Test Acc: 83.73%
Epoch [43/100] - Train Loss: 0.1214 - Train Acc: 95.71% - Test Acc: 83.36%
Epoch [44/100] - Train Loss: 0.1190 - Train Acc: 95.89% - Test Acc: 83.65%
Epoch [45/100] - Train Loss: 0.1133 - Train Acc: 96.11% - Test Acc: 83.86%
Epoch [46/100] - Train Loss: 0.1101 - Train Acc: 96.18% - Test Acc: 83.00%
Epoch [47/100] - Train Loss: 0.1138 - Train Acc: 96.03% - Test Acc: 83.37%
Epoch [48/100] - Train Loss: 0.1079 - Train Acc: 96.25% - Test Acc: 83.79%
Epoch [49/100] - Train Loss: 0.1020 - Train Acc: 96.42% - Test Acc: 83.75%
Epoch [50/100] - Train Loss: 0.1055 - Train Acc: 96.36% - Test Acc: 83.27%
Epoch [51/100] - Train Loss: 0.1023 - Train Acc: 96.37% - Test Acc: 83.70%
Epoch [52/100] - Train Loss: 0.0990 - Train Acc: 96.58% - Test Acc: 83.20%
Epoch [53/100] - Train Loss: 0.0982 - Train Acc: 96.65% - Test Acc: 83.74%
Epoch [54/100] - Train Loss: 0.0955 - Train Acc: 96.65% - Test Acc: 83.15%
Epoch [55/100] - Train Loss: 0.0961 - Train Acc: 96.66% - Test Acc: 83.74%
Epoch [56/100] - Train Loss: 0.0931 - Train Acc: 96.75% - Test Acc: 83.66%
Epoch [57/100] - Train Loss: 0.0918 - Train Acc: 96.71% - Test Acc: 83.53%
Epoch [58/100] - Train Loss: 0.0902 - Train Acc: 96.80% - Test Acc: 83.44%
Epoch [59/100] - Train Loss: 0.0886 - Train Acc: 96.93% - Test Acc: 83.51%
Epoch [60/100] - Train Loss: 0.0877 - Train Acc: 97.00% - Test Acc: 83.80%
Epoch [61/100] - Train Loss: 0.0826 - Train Acc: 97.12% - Test Acc: 83.45%
Epoch [62/100] - Train Loss: 0.0853 - Train Acc: 97.07% - Test Acc: 83.52%
Epoch [63/100] - Train Loss: 0.0838 - Train Acc: 97.09% - Test Acc: 83.80%
Epoch [64/100] - Train Loss: 0.0790 - Train Acc: 97.26% - Test Acc: 83.35%
Epoch [65/100] - Train Loss: 0.0802 - Train Acc: 97.16% - Test Acc: 83.66%
Epoch [66/100] - Train Loss: 0.0754 - Train Acc: 97.32% - Test Acc: 83.46%
Epoch [67/100] - Train Loss: 0.0793 - Train Acc: 97.31% - Test Acc: 83.35%
Epoch [68/100] - Train Loss: 0.0771 - Train Acc: 97.30% - Test Acc: 83.50%
Epoch [69/100] - Train Loss: 0.0741 - Train Acc: 97.43% - Test Acc: 83.40%
Epoch [70/100] - Train Loss: 0.0738 - Train Acc: 97.46% - Test Acc: 83.91%
Epoch [71/100] - Train Loss: 0.0745 - Train Acc: 97.33% - Test Acc: 83.44%
Epoch [72/100] - Train Loss: 0.0697 - Train Acc: 97.55% - Test Acc: 83.65%
Epoch [73/100] - Train Loss: 0.0719 - Train Acc: 97.48% - Test Acc: 83.97%
Epoch [74/100] - Train Loss: 0.0695 - Train Acc: 97.56% - Test Acc: 83.71%
Epoch [75/100] - Train Loss: 0.0691 - Train Acc: 97.64% - Test Acc: 83.53%
Epoch [76/100] - Train Loss: 0.0699 - Train Acc: 97.53% - Test Acc: 83.12%
Epoch [77/100] - Train Loss: 0.0670 - Train Acc: 97.64% - Test Acc: 83.22%
Epoch [78/100] - Train Loss: 0.0672 - Train Acc: 97.67% - Test Acc: 83.99%
Epoch [79/100] - Train Loss: 0.0655 - Train Acc: 97.70% - Test Acc: 83.98%
Epoch [80/100] - Train Loss: 0.0612 - Train Acc: 97.88% - Test Acc: 83.70%
Epoch [81/100] - Train Loss: 0.0604 - Train Acc: 97.94% - Test Acc: 83.18%
Epoch [82/100] - Train Loss: 0.0628 - Train Acc: 97.83% - Test Acc: 83.61%
Epoch [83/100] - Train Loss: 0.0638 - Train Acc: 97.78% - Test Acc: 83.37%
Epoch [84/100] - Train Loss: 0.0644 - Train Acc: 97.83% - Test Acc: 83.10%
Epoch [85/100] - Train Loss: 0.0640 - Train Acc: 97.83% - Test Acc: 83.48%
Epoch [86/100] - Train Loss: 0.0624 - Train Acc: 97.88% - Test Acc: 83.32%
```

```
Epoch [87/100] - Train Loss: 0.0593 - Train Acc: 98.00% - Test Acc: 83.72%
Epoch [88/100] - Train Loss: 0.0570 - Train Acc: 98.05% - Test Acc: 83.21%
Epoch [89/100] - Train Loss: 0.0565 - Train Acc: 98.00% - Test Acc: 83.29%
Epoch [90/100] - Train Loss: 0.0559 - Train Acc: 98.08% - Test Acc: 83.45%
Epoch [91/100] - Train Loss: 0.0571 - Train Acc: 97.99% - Test Acc: 83.35%
Epoch [92/100] - Train Loss: 0.0596 - Train Acc: 97.94% - Test Acc: 83.08%
Epoch [93/100] - Train Loss: 0.0555 - Train Acc: 98.04% - Test Acc: 83.22%
Epoch [94/100] - Train Loss: 0.0562 - Train Acc: 98.03% - Test Acc: 83.31%
Epoch [95/100] - Train Loss: 0.0547 - Train Acc: 98.08% - Test Acc: 83.37%
Epoch [96/100] - Train Loss: 0.0562 - Train Acc: 98.02% - Test Acc: 82.74%
Epoch [97/100] - Train Loss: 0.0561 - Train Acc: 98.02% - Test Acc: 83.50%
Epoch [98/100] - Train Loss: 0.0566 - Train Acc: 98.02% - Test Acc: 83.26%
Epoch [99/100] - Train Loss: 0.0519 - Train Acc: 98.26% - Test Acc: 83.37%
Epoch [100/100] - Train Loss: 0.0493 - Train Acc: 98.30% - Test Acc: 83.71%
```

[27]:
```python
print('*'*10, 'LOSS AND ACCURACY PLOTS FOR STANDARD AUGMENTATION', '*'*10)

print()

plt.figure(figsize=(20, 5))

plt.subplot(1, 2, 1)
plt.plot(t4_train_acc, label='Training Accuracy')
plt.plot(t4_test_acc, label='Testing Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training vs Testing Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(t4_loss, label='Training Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training Loss')
plt.legend()
plt.show()

print()

print('*'*100)
print('Final Test Accuracy:', t4_test_acc[-1], '%')
print('*'*100)
```

```
********** LOSS AND ACCURACY PLOTS FOR STANDARD AUGMENTATION **********
```






```
**********************************************************************************
********************
Final Test Accuracy: 83.71 %
**********************************************************************************
********************
```

1.3.5 TASK 5 - (3 pts) Combine all augmentations together.

First apply standard and cutout augmentations on the training images and then apply mixup to blend them. For mixup, use the parameter that has higher test accuracy. Report the results. Does combining improve things further?

Helper Functions

[28]:
```python
# Function to Calculate Accuracies
def get_acc(model, data_loader, device):
    model.eval()
    correct_predictions = 0
    total_predictions = 0

    for images, labels in data_loader:
        images, labels = images.to(device), labels.to(device)

        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)

        total_predictions += labels.size(0)
        correct_predictions += (predicted == labels).sum().item()

    model.train()
    return (correct_predictions / total_predictions) * 100
```

Main Functions

[29]:
```
# Explicitly defining the function to perform Mixup Augmentation
def updated_mixup_data(x, y, alpha=1.0):
    '''Compute the mixup data. Return mixed inputs, pairs of targets, and␣
 ↪lambda'''
    if alpha > 0.:
        lam = np.random.beta(alpha, alpha)
    else:
        lam = 1.

    batch_size = x.size()[0]
    index = torch.randperm(batch_size).to(x.device)

    mixed_x = lam * x + (1 - lam) * x[index, :]

    # Convert one-hot encoded labels to class indices
    _, y_a = torch.max(y, 1)
    _, y_b = torch.max(y[index], 1)

    return mixed_x, y_a, y_b, lam
```

[30]:
```
# Explicitly defining the function to perform Combined Augmentation
def combine_augmentations(images, labels, alpha, K, mask_size):
    # Apply standard and cutout augmentations
    images = standard_augmentation(images, K)
    images = cutout(images, mask_size)

    # Apply mixup augmentation
    mixed_images, labels_a, labels_b, lam = updated_mixup_data(images, labels,␣
 ↪alpha)

    return mixed_images, labels_a, labels_b, lam
```

[31]:
```
def mixup_criterion(criterion, pred, y_a, y_b, lam):
    return lam * criterion(pred, y_a) + (1 - lam) * criterion(pred, y_b)
```

[32]:
```
def train_combined_aug(model, train_loader, test_loader, epochs, optimizer,␣
 ↪criterion, device, alpha, K, mask_size):
    model = model.to(device)

    train_loss_arr = []
    train_acc_arr = []
    test_acc_arr = []

    for epoch in range(epochs):
        train_loss = 0.0
        train_acc = 0.0
        correct = 0.0
```

```
        total_samples = 0.0

        for i, data in enumerate(train_loader, 0):
            inputs, labels = data
            inputs, labels = inputs.to(device), labels.to(device)
            inputs, labels_a, labels_b, lam = combine_augmentations(inputs,␣
↪labels, alpha, K, mask_size)

            outputs = model(inputs)
            loss = mixup_criterion(criterion, outputs, labels_a, labels_b, lam)

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            _, predicted = torch.max(outputs.data, 1)
            total_samples += labels_a.size(0)
            correct += ((predicted == labels_a) * lam + (predicted == labels_b)␣
↪* (1 - lam)).sum().item()
            train_loss += loss.item() * inputs.size(0)
            train_acc += correct

        train_loss /= total_samples
        train_acc /= total_samples

        train_loss_arr.append(train_loss)
        train_acc_arr.append(train_acc)

        test_acc = get_acc(model, test_loader, device)
        test_acc_arr.append(test_acc)

        print(f"Epoch [{epoch+1}/{epochs}] - Train Loss: {train_loss:.4f} -␣
↪Train Acc: {train_acc:.2f}% - Test Acc: {test_acc:.2f}%")

    return train_loss_arr, train_acc_arr, test_acc_arr
```

Testing the Train Function

[33]:
```
K=4
t5_loss, t5_train_acc, t5_test_acc = train_combined_aug(model, train_loader,␣
↪test_loader, epochs, optimizer, optim_criter, device, alpha, K, mask_size)
```

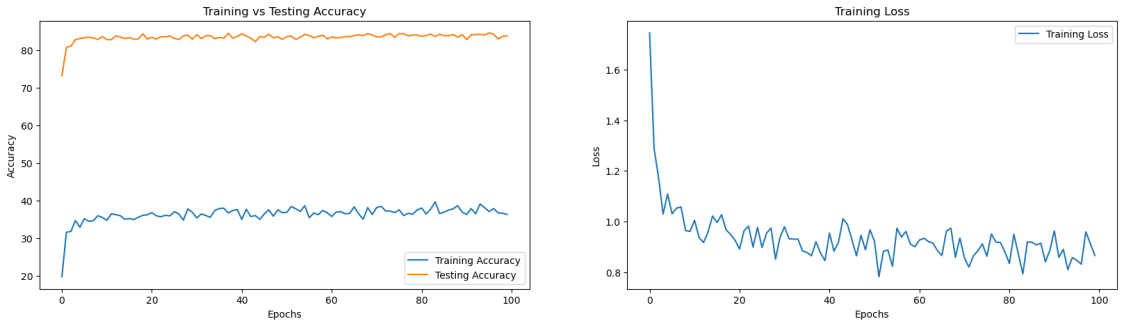
```
Epoch [1/100] - Train Loss: 1.7453 - Train Acc: 19.71% - Test Acc: 73.11%
Epoch [2/100] - Train Loss: 1.2878 - Train Acc: 31.54% - Test Acc: 80.69%
Epoch [3/100] - Train Loss: 1.1728 - Train Acc: 31.73% - Test Acc: 80.97%
Epoch [4/100] - Train Loss: 1.0292 - Train Acc: 34.63% - Test Acc: 82.72%
Epoch [5/100] - Train Loss: 1.1086 - Train Acc: 32.85% - Test Acc: 82.99%
```

```
Epoch [6/100] - Train Loss: 1.0306 - Train Acc: 35.14% - Test Acc: 83.23%
Epoch [7/100] - Train Loss: 1.0530 - Train Acc: 34.42% - Test Acc: 83.35%
Epoch [8/100] - Train Loss: 1.0575 - Train Acc: 34.63% - Test Acc: 83.19%
Epoch [9/100] - Train Loss: 0.9644 - Train Acc: 35.92% - Test Acc: 82.75%
Epoch [10/100] - Train Loss: 0.9610 - Train Acc: 35.44% - Test Acc: 83.53%
Epoch [11/100] - Train Loss: 1.0048 - Train Acc: 34.69% - Test Acc: 82.71%
Epoch [12/100] - Train Loss: 0.9369 - Train Acc: 36.43% - Test Acc: 82.71%
Epoch [13/100] - Train Loss: 0.9170 - Train Acc: 36.21% - Test Acc: 83.69%
Epoch [14/100] - Train Loss: 0.9583 - Train Acc: 35.90% - Test Acc: 83.38%
Epoch [15/100] - Train Loss: 1.0218 - Train Acc: 34.96% - Test Acc: 82.99%
Epoch [16/100] - Train Loss: 0.9960 - Train Acc: 35.17% - Test Acc: 83.26%
Epoch [17/100] - Train Loss: 1.0270 - Train Acc: 34.89% - Test Acc: 82.81%
Epoch [18/100] - Train Loss: 0.9680 - Train Acc: 35.50% - Test Acc: 82.90%
Epoch [19/100] - Train Loss: 0.9487 - Train Acc: 36.02% - Test Acc: 84.27%
Epoch [20/100] - Train Loss: 0.9261 - Train Acc: 36.18% - Test Acc: 82.87%
Epoch [21/100] - Train Loss: 0.8915 - Train Acc: 36.72% - Test Acc: 83.31%
Epoch [22/100] - Train Loss: 0.9636 - Train Acc: 35.88% - Test Acc: 82.86%
Epoch [23/100] - Train Loss: 0.9820 - Train Acc: 35.64% - Test Acc: 83.51%
Epoch [24/100] - Train Loss: 0.8991 - Train Acc: 36.02% - Test Acc: 83.46%
Epoch [25/100] - Train Loss: 0.9771 - Train Acc: 35.88% - Test Acc: 83.71%
Epoch [26/100] - Train Loss: 0.8975 - Train Acc: 36.98% - Test Acc: 83.03%
Epoch [27/100] - Train Loss: 0.9538 - Train Acc: 36.40% - Test Acc: 82.77%
Epoch [28/100] - Train Loss: 0.9743 - Train Acc: 34.74% - Test Acc: 83.70%
Epoch [29/100] - Train Loss: 0.8511 - Train Acc: 37.72% - Test Acc: 83.91%
Epoch [30/100] - Train Loss: 0.9378 - Train Acc: 36.83% - Test Acc: 82.85%
Epoch [31/100] - Train Loss: 0.9796 - Train Acc: 35.33% - Test Acc: 84.05%
Epoch [32/100] - Train Loss: 0.9318 - Train Acc: 36.39% - Test Acc: 82.97%
Epoch [33/100] - Train Loss: 0.9304 - Train Acc: 35.95% - Test Acc: 83.70%
Epoch [34/100] - Train Loss: 0.9308 - Train Acc: 35.50% - Test Acc: 83.81%
Epoch [35/100] - Train Loss: 0.8838 - Train Acc: 37.31% - Test Acc: 82.98%
Epoch [36/100] - Train Loss: 0.8777 - Train Acc: 37.82% - Test Acc: 83.30%
Epoch [37/100] - Train Loss: 0.8645 - Train Acc: 37.90% - Test Acc: 83.09%
Epoch [38/100] - Train Loss: 0.9205 - Train Acc: 36.67% - Test Acc: 84.41%
Epoch [39/100] - Train Loss: 0.8757 - Train Acc: 37.30% - Test Acc: 83.11%
Epoch [40/100] - Train Loss: 0.8454 - Train Acc: 37.55% - Test Acc: 83.56%
Epoch [41/100] - Train Loss: 0.9546 - Train Acc: 34.92% - Test Acc: 84.25%
Epoch [42/100] - Train Loss: 0.8830 - Train Acc: 37.65% - Test Acc: 83.71%
Epoch [43/100] - Train Loss: 0.9171 - Train Acc: 35.69% - Test Acc: 83.09%
Epoch [44/100] - Train Loss: 1.0109 - Train Acc: 35.97% - Test Acc: 82.16%
Epoch [45/100] - Train Loss: 0.9883 - Train Acc: 34.93% - Test Acc: 83.51%
Epoch [46/100] - Train Loss: 0.9276 - Train Acc: 36.38% - Test Acc: 83.38%
Epoch [47/100] - Train Loss: 0.8645 - Train Acc: 37.50% - Test Acc: 84.17%
Epoch [48/100] - Train Loss: 0.9451 - Train Acc: 35.78% - Test Acc: 83.20%
Epoch [49/100] - Train Loss: 0.8885 - Train Acc: 37.47% - Test Acc: 83.50%
Epoch [50/100] - Train Loss: 0.9672 - Train Acc: 36.74% - Test Acc: 82.76%
Epoch [51/100] - Train Loss: 0.9231 - Train Acc: 36.80% - Test Acc: 83.55%
Epoch [52/100] - Train Loss: 0.7820 - Train Acc: 38.35% - Test Acc: 83.65%
Epoch [53/100] - Train Loss: 0.8827 - Train Acc: 37.75% - Test Acc: 82.74%
```

```
Epoch [54/100] - Train Loss: 0.8874 - Train Acc: 37.04% - Test Acc: 83.34%
Epoch [55/100] - Train Loss: 0.8229 - Train Acc: 38.57% - Test Acc: 84.09%
Epoch [56/100] - Train Loss: 0.9736 - Train Acc: 35.37% - Test Acc: 83.82%
Epoch [57/100] - Train Loss: 0.9382 - Train Acc: 36.63% - Test Acc: 83.24%
Epoch [58/100] - Train Loss: 0.9610 - Train Acc: 36.20% - Test Acc: 83.57%
Epoch [59/100] - Train Loss: 0.9109 - Train Acc: 37.32% - Test Acc: 83.88%
Epoch [60/100] - Train Loss: 0.9005 - Train Acc: 36.72% - Test Acc: 82.90%
Epoch [61/100] - Train Loss: 0.9268 - Train Acc: 35.71% - Test Acc: 83.44%
Epoch [62/100] - Train Loss: 0.9339 - Train Acc: 36.90% - Test Acc: 83.19%
Epoch [63/100] - Train Loss: 0.9208 - Train Acc: 36.98% - Test Acc: 83.30%
Epoch [64/100] - Train Loss: 0.9147 - Train Acc: 36.42% - Test Acc: 83.52%
Epoch [65/100] - Train Loss: 0.8853 - Train Acc: 36.55% - Test Acc: 83.50%
Epoch [66/100] - Train Loss: 0.8658 - Train Acc: 38.27% - Test Acc: 83.77%
Epoch [67/100] - Train Loss: 0.9619 - Train Acc: 36.37% - Test Acc: 83.99%
Epoch [68/100] - Train Loss: 0.9738 - Train Acc: 34.99% - Test Acc: 83.79%
Epoch [69/100] - Train Loss: 0.8584 - Train Acc: 38.07% - Test Acc: 84.31%
Epoch [70/100] - Train Loss: 0.9351 - Train Acc: 36.23% - Test Acc: 83.99%
Epoch [71/100] - Train Loss: 0.8571 - Train Acc: 38.09% - Test Acc: 83.43%
Epoch [72/100] - Train Loss: 0.8198 - Train Acc: 38.42% - Test Acc: 83.44%
Epoch [73/100] - Train Loss: 0.8643 - Train Acc: 37.17% - Test Acc: 83.98%
Epoch [74/100] - Train Loss: 0.8843 - Train Acc: 37.14% - Test Acc: 84.36%
Epoch [75/100] - Train Loss: 0.9119 - Train Acc: 36.74% - Test Acc: 83.33%
Epoch [76/100] - Train Loss: 0.8626 - Train Acc: 37.47% - Test Acc: 84.28%
Epoch [77/100] - Train Loss: 0.9509 - Train Acc: 35.95% - Test Acc: 84.31%
Epoch [78/100] - Train Loss: 0.9188 - Train Acc: 36.55% - Test Acc: 83.70%
Epoch [79/100] - Train Loss: 0.9174 - Train Acc: 36.31% - Test Acc: 83.94%
Epoch [80/100] - Train Loss: 0.8804 - Train Acc: 37.45% - Test Acc: 83.94%
Epoch [81/100] - Train Loss: 0.8346 - Train Acc: 37.95% - Test Acc: 83.58%
Epoch [82/100] - Train Loss: 0.9499 - Train Acc: 36.37% - Test Acc: 83.83%
Epoch [83/100] - Train Loss: 0.8712 - Train Acc: 37.66% - Test Acc: 84.17%
Epoch [84/100] - Train Loss: 0.7934 - Train Acc: 39.60% - Test Acc: 83.54%
Epoch [85/100] - Train Loss: 0.9194 - Train Acc: 36.48% - Test Acc: 84.16%
Epoch [86/100] - Train Loss: 0.9181 - Train Acc: 36.88% - Test Acc: 83.74%
Epoch [87/100] - Train Loss: 0.9075 - Train Acc: 37.42% - Test Acc: 83.80%
Epoch [88/100] - Train Loss: 0.9145 - Train Acc: 37.75% - Test Acc: 84.04%
Epoch [89/100] - Train Loss: 0.8406 - Train Acc: 38.58% - Test Acc: 83.38%
Epoch [90/100] - Train Loss: 0.8861 - Train Acc: 36.88% - Test Acc: 84.05%
Epoch [91/100] - Train Loss: 0.9632 - Train Acc: 36.24% - Test Acc: 82.72%
Epoch [92/100] - Train Loss: 0.8585 - Train Acc: 37.82% - Test Acc: 83.98%
Epoch [93/100] - Train Loss: 0.8900 - Train Acc: 36.41% - Test Acc: 84.06%
Epoch [94/100] - Train Loss: 0.8101 - Train Acc: 39.05% - Test Acc: 84.10%
Epoch [95/100] - Train Loss: 0.8577 - Train Acc: 38.03% - Test Acc: 83.96%
Epoch [96/100] - Train Loss: 0.8469 - Train Acc: 37.03% - Test Acc: 84.50%
Epoch [97/100] - Train Loss: 0.8316 - Train Acc: 37.82% - Test Acc: 84.12%
Epoch [98/100] - Train Loss: 0.9593 - Train Acc: 36.66% - Test Acc: 82.92%
Epoch [99/100] - Train Loss: 0.9134 - Train Acc: 36.57% - Test Acc: 83.58%
Epoch [100/100] - Train Loss: 0.8659 - Train Acc: 36.27% - Test Acc: 83.75%
```

[34]:
```
print('*'*10, 'LOSS AND ACCURACY PLOTS FOR COMBINED AUGMENTATION', '*'*10)

print()

plt.figure(figsize=(20, 5))

plt.subplot(1, 2, 1)
plt.plot(t5_train_acc, label='Training Accuracy')
plt.plot(t5_test_acc, label='Testing Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training vs Testing Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(t5_loss, label='Training Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training Loss')
plt.legend()
plt.show()

print()

print('*'*100)
print('Final Test Accuracy:', t5_test_acc[-1], '%')
print('*'*100)
```

********** LOSS AND ACCURACY PLOTS FOR COMBINED AUGMENTATION **********




```
****************************************************************************************************
********************
Final Test Accuracy: 83.75 %
****************************************************************************************************
```

****

1.3.6 TASK 6 - (2 pts) Comment on the role of data augmentation. How does it affect test accuracy, train accuracy and the convergence of optimization? Is test accuracy higher? Does training loss converge faster?

Performing Data Augmentation has been beneficial. The trend in the train accuracy is that it has increased slightly (not much) after implementing each Augmentation Technique. The Test Accuracy on the other hand shows major improvement after the implementation of every augmentation technique. The loss also converges quickly as seen in the results.

In summary, Data augmentation helps generalize the model more and in result give out good results on unseen data.