In [1]:
```
import torch
import torchvision
import torch.nn as nn
import torchvision.transforms as transforms
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
torch.manual_seed(4)
np.random.seed(4)
```

In [2]:
```
BATCH_SIZE = 128
NUM_ITERS = int(1e4)
CRITERION = nn.CrossEntropyLoss()
```

In [3]:
```
transform = transforms.Compose(
    [transforms.ToTensor(),
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])


trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                        download=True, transform=transform)


testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                       download=True,  transform=transform)


train_loader = torch.utils.data.DataLoader(trainset, batch_size=BATCH_SIZE,
                                        shuffle=True, num_workers=2)

test_loader = torch.utils.data.DataLoader(testset, batch_size=BATCH_SIZE,
                                        shuffle=False, num_workers=2)

EPOCHS = int(NUM_ITERS / (len(trainset) / BATCH_SIZE))

DEVICE = 'cuda' if torch.cuda.is_available() else 'cpu'

classes = ('plane', 'car', 'bird', 'cat',
'deer', 'dog', 'frog', 'horse', 'ship', 'truck')

print(DEVICE)
```

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./dat
a/cifar-10-python.tar.gz


Extracting ./data/cifar-10-python.tar.gz to ./data
Files already downloaded and verified
cuda
```

In [4]:
```
# Block

class ResidualBlock(nn.Module):

    def __init__(self, stride=1, padding=1, batch_norm=False):
        super().__init__()
        self.batch_norm = batch_norm

        self.conv1 = nn.Conv2d(in_channels=8, out_channels=8, kernel_size=3, s
tride=stride, padding=padding)

        if batch_norm:
            self.bn1 = nn.BatchNorm2d(8)

        self.relu1 = nn.ReLU()

        self.conv2 = nn.Conv2d(in_channels=8, out_channels=8, kernel_size=3, s
tride=stride, padding=padding)

        if batch_norm:
            self.bn2 = nn.BatchNorm2d(8)

        self.relu2 = nn.ReLU()




    def forward(self, x):

        residual = x

        out = self.conv1(x)

        if self.batch_norm:
            out = self.bn1(out)

        out = self.relu1(out)

        out = self.conv2(out)

        if self.batch_norm:
            out = self.bn2(out)

        out += residual

        out = self.relu2(out)

        return out

# Model
class CNNModel(nn.Module):
    def __init__(self, batch_norm=False, N=10):

        super().__init__()

        self.conv1 = nn.Conv2d(in_channels=3, out_channels=8, kernel_size=5, s
tride=1, padding=0)
        # Size: 28 x 28

        self.conv2 = nn.Conv2d(in_channels=8, out_channels=8, kernel_size=5, s
tride=2, padding=0)
        # Size: 12 x 12

        # Dynamic block num
        self.blocks = nn.Sequential(*[ResidualBlock(batch_norm=batch_norm) for
_ in range(N)])
        # Size: 12 x 12

        self.conv3 = nn.Conv2d(in_channels=8, out_channels=4, kernel_size=3, s
tride=2, padding=0)
        # Size: 5 X 5

        self.fc1 = nn.Linear(5 * 5 * 4, 100, bias=True)
        self.relu1 = nn.ReLU()
        self.fc2 = nn.Linear(100, 10, bias=False)


    def forward(self, x):

        out = self.conv1(x)
        out = self.conv2(out)

        out = self.blocks(out)         # How is the shape the same after thi
s???
        out = self.conv3(out)
        out = out.view(out.size(0), -1)
        out = self.fc1(out)
        out = self.relu1(out)
        out = self.fc2(out)

        return out
```

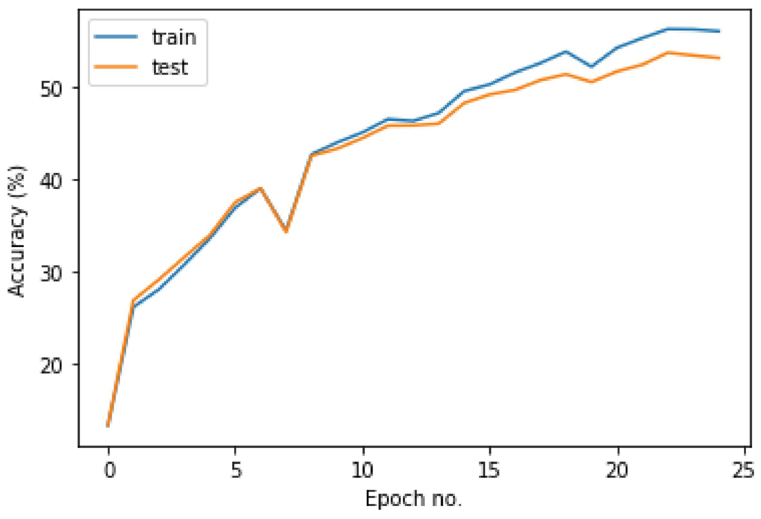
In [5]:
```
def calc_accuracy(model, train=False): # add train param to calculate accuracy
on both train and test
    # Calculate Accuracy
    correct = 0
    total = 0

    d_loader = train_loader if train else test_loader
    # Iterate through test dataset
    for images, labels in d_loader:
        # Load images
        images, labels = images.to(DEVICE), labels.to(DEVICE)

        # Forward pass only to get logits/output
        outputs = model(images)

        # Get predictions from the maximum value
        _, predicted = torch.max(outputs.data, 1)

        # Total number of labels
        total += labels.size(0)

        # Total correct predictions
        correct += (predicted == labels).sum()

    return 100 * correct / total

def train(model):
    print(f'Training for {EPOCHS} epochs')
    optimizer = torch.optim.SGD(model.parameters(), lr=.01)
    accuracy = {'train': [], 'test': []}

    for epoch in range(EPOCHS):
        for i, (images, labels) in enumerate(train_loader):
            # This will load batch_size amount of samples
            images, labels = images.requires_grad_().to(DEVICE), labels.to(DEV
ICE)

            # Clear gradients w.r.t. parameters
            optimizer.zero_grad()

            # Forward pass to get output/logits
            outputs = model(images)

            # Calculate Loss: softmax --> cross entropy loss
            loss = CRITERION(outputs, labels)

            # Getting gradients w.r.t. parameters
            loss.backward()

            # Updating parameters
            optimizer.step()

        train_accuracy = calc_accuracy(model, train=True) # abstract accuracy
 function away
        test_accuracy = calc_accuracy(model) # abstract accuracy function away
        # Print Loss
        print('Epoch: {} Loss: {}. Train Accuracy: {}, Test Accuracy: {}'.form
at(epoch, loss.item(), train_accuracy, test_accuracy))

        accuracy['train'].append(train_accuracy.item())
        accuracy['test'].append(test_accuracy.item())

    return pd.DataFrame(accuracy)
```

a) Plot Train and Test accuracy

In [6]:
```
resdiual_network = CNNModel().to(DEVICE)
resdiual_network_history = train(resdiual_network)
resdiual_network_history.plot(y=['train', 'test'], use_index=True, xlabel='Epo
ch no.', ylabel='Accuracy (%)')
```

```
Training for 25 epochs
Epoch: 0 Loss: 2.2685112953186035. Train Accuracy: 13.191999435424805, Test A
ccuracy: 13.239999771118164
Epoch: 1 Loss: 2.0367963314056396. Train Accuracy: 26.0679988861084, Test Acc
uracy: 26.809999465942383
Epoch: 2 Loss: 1.990281343460083. Train Accuracy: 27.983999252319336, Test Ac
curacy: 29.049999237060547
Epoch: 3 Loss: 1.8291699886322021. Train Accuracy: 30.673999786376953, Test A
ccuracy: 31.479999542236328
Epoch: 4 Loss: 1.9209339618682861. Train Accuracy: 33.49599838256836, Test Ac
curacy: 33.86000061035156
Epoch: 5 Loss: 1.7385276556015015. Train Accuracy: 36.84199905395508, Test Ac
curacy: 37.439998626708984
Epoch: 6 Loss: 1.7754242420196533. Train Accuracy: 38.947998046875, Test Accu
racy: 38.96999740600586
Epoch: 7 Loss: 1.5626598596572876. Train Accuracy: 34.369998931884766, Test A
ccuracy: 34.20000076293945
Epoch: 8 Loss: 1.592860460281372. Train Accuracy: 42.657997131347656, Test Ac
curacy: 42.5
Epoch: 9 Loss: 1.654094934463501. Train Accuracy: 43.93199920654297, Test Acc
uracy: 43.2599983215332
Epoch: 10 Loss: 1.6850175857543945. Train Accuracy: 45.025997161865234, Test
Accuracy: 44.40999984741211
Epoch: 11 Loss: 1.3844594955444336. Train Accuracy: 46.46999740600586, Test A
ccuracy: 45.7599983215332
Epoch: 12 Loss: 1.5455591678619385. Train Accuracy: 46.27199935913086, Test A
ccuracy: 45.78999710083008
Epoch: 13 Loss: 1.5366289615631104. Train Accuracy: 47.145999908447266, Test
Accuracy: 45.97999954223633
Epoch: 14 Loss: 1.4361560344696045. Train Accuracy: 49.51799774169922, Test A
ccuracy: 48.21999740600586
Epoch: 15 Loss: 1.4035873413085938. Train Accuracy: 50.24599838256836, Test A
ccuracy: 49.14999771118164
Epoch: 16 Loss: 1.440170407295227. Train Accuracy: 51.54199981689453, Test Ac
curacy: 49.64999771118164
Epoch: 17 Loss: 1.485811471939087. Train Accuracy: 52.56999969482422, Test Ac
curacy: 50.71999740600586
Epoch: 18 Loss: 1.4248459339141846. Train Accuracy: 53.8120002746582, Test Ac
curacy: 51.349998474121094
Epoch: 19 Loss: 1.3520452976226807. Train Accuracy: 52.17599868774414, Test A
ccuracy: 50.5099983215332
Epoch: 20 Loss: 1.233424425125122. Train Accuracy: 54.21799850463867, Test Ac
curacy: 51.64999771118164
Epoch: 21 Loss: 1.1922738552093506. Train Accuracy: 55.284000396728516, Test
Accuracy: 52.38999938964844
Epoch: 22 Loss: 1.3271561861038208. Train Accuracy: 56.25599670410156, Test A
ccuracy: 53.689998626708984
Epoch: 23 Loss: 1.1563096046447754. Train Accuracy: 56.22200012207031, Test A
ccuracy: 53.39999771118164
Epoch: 24 Loss: 1.2647123336791992. Train Accuracy: 56.02799987792969, Test A
ccuracy: 53.12999725341797
```

```
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7f439255aa50>
```

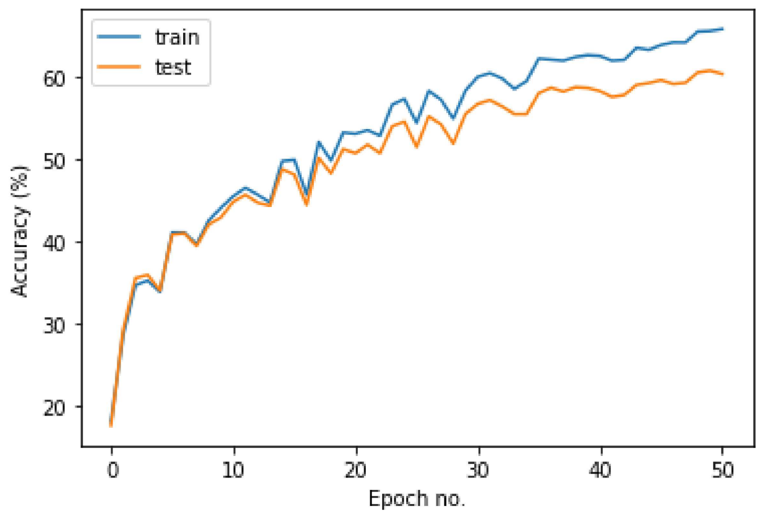

train
test
Accuracy (%)
Epoch no.
20
30
40
50
0
5
10
15
20
25

In [7]:
```
# Just for fun, as a straight comparison to last week
EPOCHS = 51
resdiual_network = CNNModel().to(DEVICE)
resdiual_network_history = train(resdiual_network)
resdiual_network_history.plot(y=['train', 'test'], use_index=True, xlabel='Epo
ch no.', ylabel='Accuracy (%)')
```

```
Training for 51 epochs
Epoch: 0 Loss: 2.184399127960205. Train Accuracy: 18.091999053955078, Test Ac
curacy: 17.579999923706055
Epoch: 1 Loss: 1.9781843423843384. Train Accuracy: 28.599998474121094, Test A
ccuracy: 29.219999313354492
Epoch: 2 Loss: 2.1509909629821777. Train Accuracy: 34.64999771118164, Test Ac
curacy: 35.52000045776367
Epoch: 3 Loss: 1.7879705429077148. Train Accuracy: 35.22800064086914, Test Ac
curacy: 35.90999984741211
Epoch: 4 Loss: 1.6254431009292603. Train Accuracy: 33.83799743652344, Test Ac
curacy: 34.029998779296875
Epoch: 5 Loss: 1.4392951726913452. Train Accuracy: 41.06599807739258, Test Ac
curacy: 40.80999755859375
Epoch: 6 Loss: 1.7015838623046875. Train Accuracy: 41.055999755859375, Test A
ccuracy: 40.959999084472656
Epoch: 7 Loss: 1.6818602085113525. Train Accuracy: 39.641998291015625, Test A
ccuracy: 39.40999984741211
Epoch: 8 Loss: 1.6117048263549805. Train Accuracy: 42.53999710083008, Test Ac
curacy: 42.03999710083008
Epoch: 9 Loss: 1.4976998567581177. Train Accuracy: 44.06999969482422, Test Ac
curacy: 42.89999771118164
Epoch: 10 Loss: 1.4641451835632324. Train Accuracy: 45.47200012207031, Test A
ccuracy: 44.79999923706055
Epoch: 11 Loss: 1.328886866569519. Train Accuracy: 46.50199890136719, Test Ac
curacy: 45.64999771118164
Epoch: 12 Loss: 1.471566915512085. Train Accuracy: 45.65599822998047, Test Ac
curacy: 44.689998626708984
Epoch: 13 Loss: 1.3035829067230225. Train Accuracy: 44.70800018310547, Test A
ccuracy: 44.30999755859375
Epoch: 14 Loss: 1.3868215084075928. Train Accuracy: 49.74599838256836, Test A
ccuracy: 48.77000045776367
Epoch: 15 Loss: 1.1341962814331055. Train Accuracy: 49.90599822998047, Test A
ccuracy: 48.099998474121094
Epoch: 16 Loss: 1.5288037061691284. Train Accuracy: 45.67399978637695, Test A
ccuracy: 44.39999771118164
Epoch: 17 Loss: 1.5743404626846313. Train Accuracy: 52.06399917602539, Test A
ccuracy: 50.12999725341797
Epoch: 18 Loss: 1.5153672695159912. Train Accuracy: 49.80399703979492, Test A
ccuracy: 48.23999786376953
Epoch: 19 Loss: 1.272467851638794. Train Accuracy: 53.21999740600586, Test Ac
curacy: 51.19999694824219
Epoch: 20 Loss: 1.197765827178955. Train Accuracy: 53.067996978759766, Test A
ccuracy: 50.69999694824219
Epoch: 21 Loss: 1.2486612796783447. Train Accuracy: 53.51799774169922, Test A
ccuracy: 51.75
Epoch: 22 Loss: 1.304996132850647. Train Accuracy: 52.843997955322266, Test A
ccuracy: 50.709999084472656
Epoch: 23 Loss: 1.0269992351531982. Train Accuracy: 56.6099967956543, Test Ac
curacy: 53.97999954223633
Epoch: 24 Loss: 1.36045241355896. Train Accuracy: 57.30999755859375, Test Acc
uracy: 54.54999923706055
Epoch: 25 Loss: 1.4575788974761963. Train Accuracy: 54.38999938964844, Test A
ccuracy: 51.459999084472656
Epoch: 26 Loss: 1.1424319744110107. Train Accuracy: 58.28999710083008, Test A
ccuracy: 55.189998626708984
Epoch: 27 Loss: 1.1043063402175903. Train Accuracy: 57.21999740600586, Test A
ccuracy: 54.209999084472656
```

```
Epoch: 28 Loss: 1.2542470693588257. Train Accuracy: 54.9219970703125, Test Ac
curacy: 51.88999938964844
Epoch: 29 Loss: 1.1667451858520508. Train Accuracy: 58.321998596191406, Test
Accuracy: 55.47999954223633
Epoch: 30 Loss: 1.0937412977218628. Train Accuracy: 59.99199676513672, Test A
ccuracy: 56.69999694824219
Epoch: 31 Loss: 1.1019313335418701. Train Accuracy: 60.45399856567383, Test A
ccuracy: 57.16999816894531
Epoch: 32 Loss: 1.0874099731445312. Train Accuracy: 59.805999755859375, Test
Accuracy: 56.40999984741211
Epoch: 33 Loss: 1.173365831375122. Train Accuracy: 58.5359992980957, Test Acc
uracy: 55.46999740600586
Epoch: 34 Loss: 1.118118405342102. Train Accuracy: 59.49199676513672, Test Ac
curacy: 55.46999740600586
Epoch: 35 Loss: 0.9556946754455566. Train Accuracy: 62.227996826171875, Test
Accuracy: 58.019996643066406
Epoch: 36 Loss: 0.9454149007797241. Train Accuracy: 62.09599685668945, Test A
ccuracy: 58.68000030517578
Epoch: 37 Loss: 1.088111162185669. Train Accuracy: 61.96399688720703, Test Ac
curacy: 58.209999084472656
Epoch: 38 Loss: 1.1753406524658203. Train Accuracy: 62.4119987487793, Test Ac
curacy: 58.73999786376953
Epoch: 39 Loss: 1.1861950159072876. Train Accuracy: 62.637996673583984, Test
Accuracy: 58.64999771118164
Epoch: 40 Loss: 1.1898328065872192. Train Accuracy: 62.53999710083008, Test A
ccuracy: 58.269996643066406
Epoch: 41 Loss: 0.9710673093795776. Train Accuracy: 61.961997985839844, Test
Accuracy: 57.55999755859375
Epoch: 42 Loss: 1.1339343786239624. Train Accuracy: 62.06399917602539, Test A
ccuracy: 57.79999923706055
Epoch: 43 Loss: 0.8726213574409485. Train Accuracy: 63.5359992980957, Test Ac
curacy: 59.019996643066406
Epoch: 44 Loss: 1.1357409954071045. Train Accuracy: 63.28799819946289, Test A
ccuracy: 59.25
Epoch: 45 Loss: 0.9507195353507996. Train Accuracy: 63.87999725341797, Test A
ccuracy: 59.599998474121094
Epoch: 46 Loss: 0.9577592611312866. Train Accuracy: 64.19999694824219, Test A
ccuracy: 59.13999938964844
Epoch: 47 Loss: 0.85992830991745. Train Accuracy: 64.18799591064453, Test Acc
uracy: 59.279998779296875
Epoch: 48 Loss: 0.9802007675170898. Train Accuracy: 65.51200103759766, Test A
ccuracy: 60.529998779296875
Epoch: 49 Loss: 1.0233908891677856. Train Accuracy: 65.57799530029297, Test A
ccuracy: 60.7599983215332
Epoch: 50 Loss: 0.8567731976509094. Train Accuracy: 65.80799865722656, Test A
ccuracy: 60.3599967956543
```

```
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4391448410>
```



b) Accuracy and convergence comparison to ConvBlock with Batch Norm

From these experiments, we can see that the residual network is able to outperform last week's model by a significant margin. The CNN with batch normalization was only able to achieve 55.6% train accuracy, 52.5% test accuracy and a loss of 1.36 by epoch 25, or around 10k iterations (these are numbers are from my pdf last week). This is in comparison to the residual network which achieved 56% train accuracy, 53% test, and a loss of 1.26. However, one could still say that this was just due to the randomness inherent in the initialization. So I decided to train for 51 epochs or around 20k iterations to have a one to one comparison with last week. Here we see a drastic difference; the residual network performs around 5% better on both train and test, as well as a loss that is almost 15% lower (.85 vs .99).