```
import torch
import torchvision
import numpy as np
import matplotlib.pyplot as plt
import PIL


# From local helper files
from helper_evaluation import set_all_seeds, set_deterministic, compute_confusion_matrix
from helper_train import train_model
from helper_plotting import plot_training_loss, plot_accuracy, show_examples, plot_confusion_matrix
from helper_dataset import get_dataloaders_cifar10, UnNormalize


RANDOM_SEED = 123
BATCH_SIZE = 256
NUM_EPOCHS = 40
DEVICE = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')


train_transforms = torchvision.transforms.Compose([
    torchvision.transforms.Resize((16, 16)),
    torchvision.transforms.ToTensor(),
    torchvision.transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
                                  ])


test_transforms = torchvision.transforms.Compose([
    torchvision.transforms.Resize((16, 16)),
    torchvision.transforms.ToTensor(),
    torchvision.transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])


train_loader, valid_loader, test_loader = get_dataloaders_cifar10(
    batch_size=BATCH_SIZE,
    validation_fraction=0.1,
    train_transforms=train_transforms,
    test_transforms=test_transforms,
    num_workers=2)

# Checking the dataset
for images, labels in train_loader:
    print('Image batch dimensions:', images.shape)
    print('Image label dimensions:', labels.shape)
    print('Class labels of 10 examples:', labels[:10])
    break
```

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to data/cifar-10-python.tar.gz
100%                                          170498071/170498071 [00:02<00:00, 84300527.28it/s]
Extracting data/cifar-10-python.tar.gz to data
Image batch dimensions: torch.Size([256, 3, 16, 16])
Image label dimensions: torch.Size([256])
```

```
class CNN1(torch.nn.Module):
  def __init__(self, num_classes):
    super().__init__()
    self.features = torch.nn.Sequential(
            # Conv 1
            torch.nn.Conv2d(3, 16, kernel_size=3, padding="same"), # output 16 - 3 + 1 => 16
                            # , stride=4, padding=2),
            torch.nn.ReLU(inplace=True),
            torch.nn.MaxPool2d(kernel_size=2), # 16 / 2 => output 8

            # Conv 2
            torch.nn.Conv2d(16, 32, kernel_size=2, padding="same"), # output 7 - 2 + 1 => 8
                            # , padding=2),
            torch.nn.ReLU(inplace=True),
            torch.nn.MaxPool2d(kernel_size=2) #output 8 / 2 => output 4
    )

    self.classifier = torch.nn.Sequential(
         torch.nn.Linear(32*4*4, 100),
          torch.nn.ReLU(inplace=True),
          torch.nn.Linear(100, num_classes),
    )
    def forward(self, x):
        x = self.features(x)
        x = torch.flatten(x, 1)
        # print(x.size())
        logits = self.classifier(x)
        return logits


model1 = CNN1(num_classes=10)


model1 = model1.to(DEVICE)


optimizer = torch.optim.SGD(model1.parameters(), lr=0.1)
scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer,
                                                       factor=0.1,
                                                       mode='max',
                                                       verbose=True)



minibatch_loss_list, train_acc_list, valid_acc_list = train_model(
    model=model1,
    num_epochs=NUM_EPOCHS,
    train_loader=train_loader,
    valid_loader=valid_loader,
    test_loader=test_loader,
    optimizer=optimizer,
    device=DEVICE,
    scheduler=None,
    scheduler_on='valid_acc',
    logging_interval=100)
```

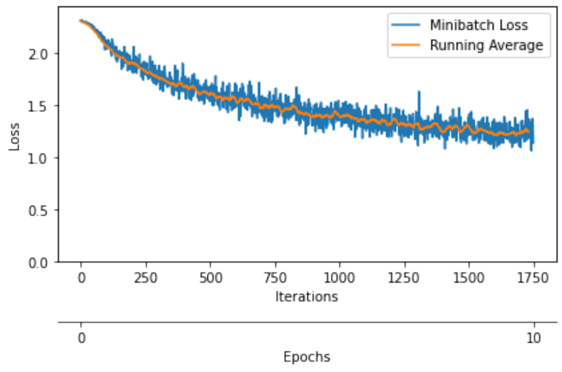
```
/usr/local/lib/python3.7/dist-packages/torch/nn/modules/conv.py:454: UserWarning: Using padding='same' with even kernel leng
  self.padding, self.dilation, self.groups)
Epoch: 001/010 | Batch 0000/0175 | Loss: 2.3107
Epoch: 001/010 | Batch 0100/0175 | Loss: 2.1200
Epoch: 001/010 | Train: 30.04% | Validation: 29.90%
Time elapsed: 1.16 min
Epoch: 002/010 | Batch 0000/0175 | Loss: 1.9083
Epoch: 002/010 | Batch 0100/0175 | Loss: 1.8180
Epoch: 002/010 | Train: 38.68% | Validation: 38.36%
Time elapsed: 2.22 min
Epoch: 003/010 | Batch 0000/0175 | Loss: 1.7657
Epoch: 003/010 | Batch 0100/0175 | Loss: 1.6283
Epoch: 003/010 | Train: 43.52% | Validation: 42.60%
Time elapsed: 3.27 min
Epoch: 004/010 | Batch 0000/0175 | Loss: 1.6700
Epoch: 004/010 | Batch 0100/0175 | Loss: 1.7042
Epoch: 004/010 | Train: 46.99% | Validation: 46.14%
Time elapsed: 4.32 min
Epoch: 005/010 | Batch 0000/0175 | Loss: 1.4037
Epoch: 005/010 | Batch 0100/0175 | Loss: 1.4912
Epoch: 005/010 | Train: 50.75% | Validation: 49.74%
Time elapsed: 5.35 min
Epoch: 006/010 | Batch 0000/0175 | Loss: 1.4252
Epoch: 006/010 | Batch 0100/0175 | Loss: 1.3006
Epoch: 006/010 | Train: 50.73% | Validation: 50.34%
Time elapsed: 6.39 min
Epoch: 007/010 | Batch 0000/0175 | Loss: 1.3887
Epoch: 007/010 | Batch 0100/0175 | Loss: 1.2501
Epoch: 007/010 | Train: 52.40% | Validation: 51.52%
Time elapsed: 7.43 min
Epoch: 008/010 | Batch 0000/0175 | Loss: 1.3049
Epoch: 008/010 | Batch 0100/0175 | Loss: 1.2453
Epoch: 008/010 | Train: 53.39% | Validation: 51.32%
Time elapsed: 8.44 min
Epoch: 009/010 | Batch 0000/0175 | Loss: 1.2285
Epoch: 009/010 | Batch 0100/0175 | Loss: 1.2858
Epoch: 009/010 | Train: 56.43% | Validation: 55.44%
Time elapsed: 9.48 min
Epoch: 010/010 | Batch 0000/0175 | Loss: 1.2854
Epoch: 010/010 | Batch 0100/0175 | Loss: 1.2725
Epoch: 010/010 | Train: 57.67% | Validation: 56.42%
Time elapsed: 10.51 min
Total Training Time: 10.51 min
Test accuracy 56.13%
```

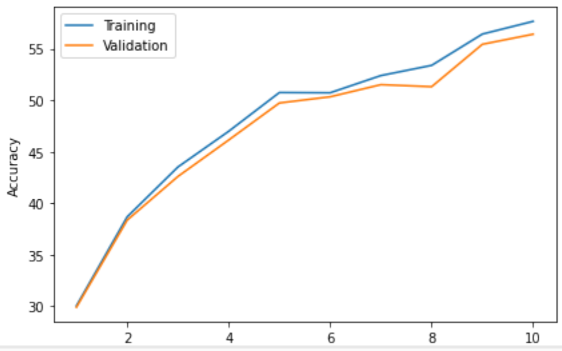
```
plot_training_loss(minibatch_loss_list=minibatch_loss_list,
                   num_epochs=NUM_EPOCHS,
                   iter_per_epoch=len(train_loader),
                   results_dir=None,
                   averaging_iterations=20)

plt.show()

plot_accuracy(train_acc_list=train_acc_list,
              valid_acc_list=valid_acc_list,
              results_dir=None)

# plt.ylim([80, 100])
plt.show()
```





```
class CNN2(torch.nn.Module):
  def __init__(self, num_classes):
    super().__init__()
    self.features = torch.nn.Sequential(
            # Conv 1
            torch.nn.Conv2d(3, 16, kernel_size=3, padding="same"), # output 16 - 3 + 1 => 16
                            # , stride=4, padding=2),
            torch.nn.ReLU(inplace=True),
            torch.nn.MaxPool2d(kernel_size=2), # 16 / 2 => output 8

            # Conv 2
            torch.nn.Conv2d(16, 32, kernel_size=2, padding="same"), # output 7 - 2 + 1 => 8
                            # , padding=2),
            torch.nn.ReLU(inplace=True),
            torch.nn.MaxPool2d(kernel_size=2), #output 8 / 2 => output 4

            # Conv 3
            torch.nn.Conv2d(32, 64, kernel_size=2, padding="same"), # output 7 - 2 + 1 => 4
                            # , padding=2),
            torch.nn.ReLU(inplace=True),
            torch.nn.MaxPool2d(kernel_size=2) #output 4 / 2 => output 2

    )

    self.classifier = torch.nn.Sequential(
         torch.nn.Linear(64*2*2, 100),
          torch.nn.ReLU(inplace=True),
          torch.nn.Linear(100, num_classes),
    )

  def forward(self, x):
        x = self.features(x)
        x = torch.flatten(x, 1)
        # print(x.size())
        logits = self.classifier(x)
        return logits


model2 = CNN2(num_classes=10)


model2 = model2.to(DEVICE)


optimizer2 = torch.optim.SGD(model2.parameters(), lr=0.1)
scheduler2 = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer,
                                                       factor=0.1,
                                                       mode='max',
                                                       verbose=True)



minibatch_loss_list2, train_acc_list2, valid_acc_list2 = train_model(
    model=model2,
    num_epochs=NUM_EPOCHS,
    train_loader=train_loader,
    valid_loader=valid_loader,
    test_loader=test_loader,
    optimizer=optimizer2,
    device=DEVICE,
    scheduler=None,
    scheduler_on='valid_acc',
    logging_interval=100)
```

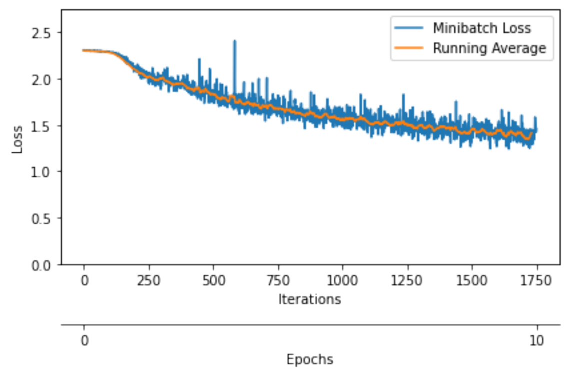
```
Epoch: 001/010 | Batch 0000/0175 | Loss: 2.3064
Epoch: 001/010 | Batch 0100/0175 | Loss: 2.2864
Epoch: 001/010 | Train: 22.01% | Validation: 21.94%
Time elapsed: 1.05 min
Epoch: 002/010 | Batch 0000/0175 | Loss: 2.1762
Epoch: 002/010 | Batch 0100/0175 | Loss: 2.0107
Epoch: 002/010 | Train: 28.13% | Validation: 27.68%
Time elapsed: 2.08 min
Epoch: 003/010 | Batch 0000/0175 | Loss: 1.9441
Epoch: 003/010 | Batch 0100/0175 | Loss: 1.7988
Epoch: 003/010 | Train: 34.48% | Validation: 34.52%
Time elapsed: 3.12 min
Epoch: 004/010 | Batch 0000/0175 | Loss: 1.7538
Epoch: 004/010 | Batch 0100/0175 | Loss: 1.6146
Epoch: 004/010 | Train: 38.63% | Validation: 38.98%
Time elapsed: 4.15 min
Epoch: 005/010 | Batch 0000/0175 | Loss: 1.6855
Epoch: 005/010 | Batch 0100/0175 | Loss: 1.8705
Epoch: 005/010 | Train: 41.14% | Validation: 41.40%
Time elapsed: 5.17 min
Epoch: 006/010 | Batch 0000/0175 | Loss: 1.5370
Epoch: 006/010 | Batch 0100/0175 | Loss: 1.6231
Epoch: 006/010 | Train: 40.54% | Validation: 41.22%
Time elapsed: 6.20 min
Epoch: 007/010 | Batch 0000/0175 | Loss: 1.6076
Epoch: 007/010 | Batch 0100/0175 | Loss: 1.4583
Epoch: 007/010 | Train: 45.01% | Validation: 44.86%
Time elapsed: 7.22 min
Epoch: 008/010 | Batch 0000/0175 | Loss: 1.5329
Epoch: 008/010 | Batch 0100/0175 | Loss: 1.4953
Epoch: 008/010 | Train: 47.04% | Validation: 46.62%
Time elapsed: 8.24 min
Epoch: 009/010 | Batch 0000/0175 | Loss: 1.4519
Epoch: 009/010 | Batch 0100/0175 | Loss: 1.3998
Epoch: 009/010 | Train: 45.86% | Validation: 46.26%
Time elapsed: 9.25 min
Epoch: 010/010 | Batch 0000/0175 | Loss: 1.4021
Epoch: 010/010 | Batch 0100/0175 | Loss: 1.3101
Epoch: 010/010 | Train: 50.69% | Validation: 50.16%
Time elapsed: 10.25 min
Total Training Time: 10.25 min
Test accuracy 50.06%
```

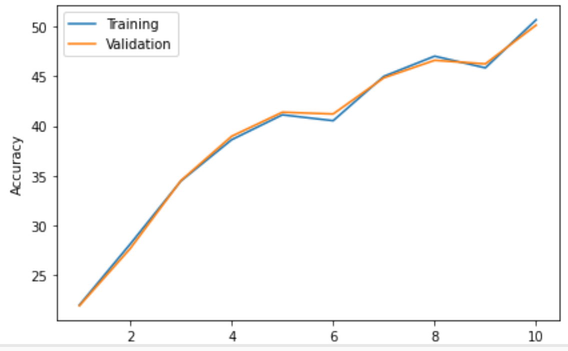
```
plot_training_loss(minibatch_loss_list=minibatch_loss_list2,
                   num_epochs=NUM_EPOCHS,
                   iter_per_epoch=len(train_loader),
                   results_dir=None,
                   averaging_iterations=20)
```

```
plt.show()

plot_accuracy(train_acc_list=train_acc_list2,
              valid_acc_list=valid_acc_list2,
              results_dir=None)

# plt.ylim([80, 100])
plt.show()
```





Data Augmentation

```
training_transforms = torchvision.transforms.Compose([
    torchvision.transforms.Resize((16, 16)),
    torchvision.transforms.ToTensor(),
    torchvision.transforms.RandomRotation(degrees=30, interpolation=PIL.Image.BILINEAR),
    torchvision.transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

test_transforms = torchvision.transforms.Compose([
      torchvision.transforms.Resize((16, 16)),
    torchvision.transforms.ToTensor(),
    torchvision.transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
```

```
train_loader_augmented, valid_loader_augmented, test_loader_augmented = get_dataloaders_cifar10(
    batch_size=BATCH_SIZE,
    validation_fraction=0.1,
    train_transforms=train_transforms,
    test_transforms=test_transforms,
    num_workers=2)
```

```
/usr/local/lib/python3.7/dist-packages/torchvision/transforms/transforms.py:1306: UserWarning: Argument 'interpolation' of t
  "Argument 'interpolation' of type int is deprecated since 0.13 and will be removed in 0.15. "
Files already downloaded and verified
```

```
model1_augmented = CNN1(num_classes=10)
```

```
model1_augmented = model1_augmented.to(DEVICE)


optimizer_augmented = torch.optim.SGD(model1_augmented.parameters(), lr=0.1)
scheduler_augmented = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer,
                                                       factor=0.1,
                                                       mode='max',
                                                       verbose=True)



minibatch_loss_list_augmented1, train_acc_list_augmented1, valid_acc_list_augmented1 = train_model(
    model=model1_augmented,
    num_epochs=NUM_EPOCHS,
    train_loader=train_loader_augmented,
    valid_loader=valid_loader_augmented,
    test_loader=test_loader_augmented,
    optimizer=optimizer_augmented,
    device=DEVICE,
    scheduler=None,
    scheduler_on='valid_acc',
    logging_interval=100)
```

```
Epoch: 001/200 | Batch 0000/0175 | Loss: 2.3148
Epoch: 001/200 | Batch 0100/0175 | Loss: 2.1903
Epoch: 001/200 | Train: 29.50% | Validation: 29.52%
Time elapsed: 1.03 min
Epoch: 002/200 | Batch 0000/0175 | Loss: 1.8754
Epoch: 002/200 | Batch 0100/0175 | Loss: 1.8607
Epoch: 002/200 | Train: 37.80% | Validation: 38.42%
Time elapsed: 2.04 min
Epoch: 003/200 | Batch 0000/0175 | Loss: 1.6420
Epoch: 003/200 | Batch 0100/0175 | Loss: 1.6492
Epoch: 003/200 | Train: 44.00% | Validation: 42.74%
Time elapsed: 3.03 min
Epoch: 004/200 | Batch 0000/0175 | Loss: 1.5914
Epoch: 004/200 | Batch 0100/0175 | Loss: 1.6133
Epoch: 004/200 | Train: 45.89% | Validation: 45.44%
Time elapsed: 4.05 min
Epoch: 005/200 | Batch 0000/0175 | Loss: 1.4920
Epoch: 005/200 | Batch 0100/0175 | Loss: 1.4084
Epoch: 005/200 | Train: 49.60% | Validation: 49.00%
Time elapsed: 5.05 min
Epoch: 006/200 | Batch 0000/0175 | Loss: 1.3994
Epoch: 006/200 | Batch 0100/0175 | Loss: 1.3412
Epoch: 006/200 | Train: 51.70% | Validation: 51.18%
Time elapsed: 6.05 min
Epoch: 007/200 | Batch 0000/0175 | Loss: 1.3779
Epoch: 007/200 | Batch 0100/0175 | Loss: 1.3370
Epoch: 007/200 | Train: 52.32% | Validation: 51.82%
Time elapsed: 7.05 min
Epoch: 008/200 | Batch 0000/0175 | Loss: 1.2955
Epoch: 008/200 | Batch 0100/0175 | Loss: 1.2618
Epoch: 008/200 | Train: 53.98% | Validation: 53.10%
Time elapsed: 8.06 min
Epoch: 009/200 | Batch 0000/0175 | Loss: 1.2390
Epoch: 009/200 | Batch 0100/0175 | Loss: 1.2337
Epoch: 009/200 | Train: 56.24% | Validation: 55.82%
Time elapsed: 9.06 min
Epoch: 010/200 | Batch 0000/0175 | Loss: 1.2523
Epoch: 010/200 | Batch 0100/0175 | Loss: 1.2770
Epoch: 010/200 | Train: 56.65% | Validation: 55.92%
Time elapsed: 10.06 min
Epoch: 011/200 | Batch 0000/0175 | Loss: 1.2074
Epoch: 011/200 | Batch 0100/0175 | Loss: 1.3200
Epoch: 011/200 | Train: 57.09% | Validation: 56.72%
Time elapsed: 11.05 min
Epoch: 012/200 | Batch 0000/0175 | Loss: 1.2549
Epoch: 012/200 | Batch 0100/0175 | Loss: 1.0894
Epoch: 012/200 | Train: 58.74% | Validation: 57.36%
Time elapsed: 12.06 min
Epoch: 013/200 | Batch 0000/0175 | Loss: 1.1947
Epoch: 013/200 | Batch 0100/0175 | Loss: 1.1765
Epoch: 013/200 | Train: 60.63% | Validation: 58.64%
Time elapsed: 13.08 min
Epoch: 014/200 | Batch 0000/0175 | Loss: 1.1961
Epoch: 014/200 | Batch 0100/0175 | Loss: 1.1816
Epoch: 014/200 | Train: 61.75% | Validation: 58.66%
Time elapsed: 14.10 min
Epoch: 015/200 | Batch 0000/0175 | Loss: 1.0895
Epoch: 015/200 | Batch 0100/0175 | Loss: 1.0552
Epoch: 015/200 | Train: 62.52% | Validation: 60.24%
Time elapsed: 15.12 min
Epoch: 016/200 | Batch 0000/0175 | Loss: 1.0076
Epoch: 016/200 | Batch 0100/0175 | Loss: 1.0254
Epoch: 016/200 | Train: 61.48% | Validation: 58.54%
Time elapsed: 16.13 min
Epoch: 017/200 | Batch 0000/0175 | Loss: 1.0743
Epoch: 017/200 | Batch 0100/0175 | Loss: 1.1721
Epoch: 017/200 | Train: 63.75% | Validation: 60.68%
Time elapsed: 17.13 min
Epoch: 018/200 | Batch 0000/0175 | Loss: 1.0992
Epoch: 018/200 | Batch 0100/0175 | Loss: 0.9466
Epoch: 018/200 | Train: 64.87% | Validation: 61.92%
Time elapsed: 18.16 min
Epoch: 019/200 | Batch 0000/0175 | Loss: 1.0100
Epoch: 019/200 | Batch 0100/0175 | Loss: 1.1097
Epoch: 019/200 | Train: 64.94% | Validation: 61.60%
Time elapsed: 19.17 min
Epoch: 020/200 | Batch 0000/0175 | Loss: 1.0213
Epoch: 020/200 | Batch 0100/0175 | Loss: 1.1055
Epoch: 020/200 | Train: 66.13% | Validation: 62.72%
Time elapsed: 20.18 min
Epoch: 021/200 | Batch 0000/0175 | Loss: 1.0420
Epoch: 021/200 | Batch 0100/0175 | Loss: 0.8931
Epoch: 021/200 | Train: 66.16% | Validation: 61.74%
Time elapsed: 21.19 min
Epoch: 022/200 | Batch 0000/0175 | Loss: 1.0587
```

```
Epoch: 022/200 | Batch 0000/0175 | Loss: 1.0587
Epoch: 022/200 | Batch 0100/0175 | Loss: 0.9567
Epoch: 022/200 | Train: 67.09% | Validation: 63.02%
Time elapsed: 22.19 min
Epoch: 023/200 | Batch 0000/0175 | Loss: 1.0669
Epoch: 023/200 | Batch 0100/0175 | Loss: 0.8600
Epoch: 023/200 | Train: 67.84% | Validation: 62.86%
Time elapsed: 23.19 min
Epoch: 024/200 | Batch 0000/0175 | Loss: 0.9439
Epoch: 024/200 | Batch 0100/0175 | Loss: 0.9063
Epoch: 024/200 | Train: 68.36% | Validation: 63.48%
Time elapsed: 24.19 min
Epoch: 025/200 | Batch 0000/0175 | Loss: 1.0403
Epoch: 025/200 | Batch 0100/0175 | Loss: 1.0171
Epoch: 025/200 | Train: 69.99% | Validation: 64.46%
Time elapsed: 25.17 min
Epoch: 026/200 | Batch 0000/0175 | Loss: 0.8498
Epoch: 026/200 | Batch 0100/0175 | Loss: 0.8427
Epoch: 026/200 | Train: 70.53% | Validation: 64.70%
Time elapsed: 26.17 min
Epoch: 027/200 | Batch 0000/0175 | Loss: 0.7758
Epoch: 027/200 | Batch 0100/0175 | Loss: 0.9058
Epoch: 027/200 | Train: 70.68% | Validation: 63.84%
Time elapsed: 27.16 min
Epoch: 028/200 | Batch 0000/0175 | Loss: 0.8855
Epoch: 028/200 | Batch 0100/0175 | Loss: 0.8881
Epoch: 028/200 | Train: 71.53% | Validation: 65.12%
Time elapsed: 28.15 min
Epoch: 029/200 | Batch 0000/0175 | Loss: 0.8177
Epoch: 029/200 | Batch 0100/0175 | Loss: 0.9540
Epoch: 029/200 | Train: 69.85% | Validation: 63.66%
Time elapsed: 29.15 min
Epoch: 030/200 | Batch 0000/0175 | Loss: 0.9553
Epoch: 030/200 | Batch 0100/0175 | Loss: 0.8446
Epoch: 030/200 | Train: 71.78% | Validation: 64.54%
Time elapsed: 30.16 min
Epoch: 031/200 | Batch 0000/0175 | Loss: 0.8080
Epoch: 031/200 | Batch 0100/0175 | Loss: 0.8003
Epoch: 031/200 | Train: 72.52% | Validation: 64.44%
Time elapsed: 31.18 min
Epoch: 032/200 | Batch 0000/0175 | Loss: 0.8102
Epoch: 032/200 | Batch 0100/0175 | Loss: 0.7658
Epoch: 032/200 | Train: 73.18% | Validation: 65.02%
Time elapsed: 32.17 min
Epoch: 033/200 | Batch 0000/0175 | Loss: 0.7072
Epoch: 033/200 | Batch 0100/0175 | Loss: 0.8225
Epoch: 033/200 | Train: 73.41% | Validation: 64.70%
Time elapsed: 33.15 min
Epoch: 034/200 | Batch 0000/0175 | Loss: 0.6741
Epoch: 034/200 | Batch 0100/0175 | Loss: 0.7405
Epoch: 034/200 | Train: 73.09% | Validation: 64.40%
Time elapsed: 34.16 min
Epoch: 035/200 | Batch 0000/0175 | Loss: 0.8302
Epoch: 035/200 | Batch 0100/0175 | Loss: 0.8017
Epoch: 035/200 | Train: 72.76% | Validation: 63.46%
Time elapsed: 35.15 min
Epoch: 036/200 | Batch 0000/0175 | Loss: 0.8319
Epoch: 036/200 | Batch 0100/0175 | Loss: 0.8461
Epoch: 036/200 | Train: 73.73% | Validation: 63.84%
Time elapsed: 36.16 min
Epoch: 037/200 | Batch 0000/0175 | Loss: 0.6183
Epoch: 037/200 | Batch 0100/0175 | Loss: 0.8088
Epoch: 037/200 | Train: 73.84% | Validation: 64.08%
Time elapsed: 37.16 min
Epoch: 038/200 | Batch 0000/0175 | Loss: 0.6809
Epoch: 038/200 | Batch 0100/0175 | Loss: 0.7810
Epoch: 038/200 | Train: 75.88% | Validation: 64.90%
Time elapsed: 38.14 min
Epoch: 039/200 | Batch 0000/0175 | Loss: 0.6267
Epoch: 039/200 | Batch 0100/0175 | Loss: 0.6177
Epoch: 039/200 | Train: 76.50% | Validation: 65.46%
Time elapsed: 39.12 min
Epoch: 040/200 | Batch 0000/0175 | Loss: 0.6199
Epoch: 040/200 | Batch 0100/0175 | Loss: 0.7197
Epoch: 040/200 | Train: 76.58% | Validation: 65.16%
Time elapsed: 40.11 min
Epoch: 041/200 | Batch 0000/0175 | Loss: 0.6925
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-23-4b24166132f3> in <module>
      9     scheduler=None,
     10     scheduler_on='valid_acc',
---> 11     logging_interval=100)
```

9 frames

```
/usr/lib/python3.7/selectors.py in select(self, timeout)
    413         ready = []
    414         try:
--> 415             fd_event_list = self._selector.poll(timeout)
    416         except InterruptedError:
    417             return ready

KeyboardInterrupt:
```

```
plot_training_loss(minibatch_loss_list=minibatch_loss_list_augmented1,
                   num_epochs=NUM_EPOCHS,
                   iter_per_epoch=len(train_loader),
                   results_dir=None,
                   averaging_iterations=20)

plt.show()

plot_accuracy(train_acc_list=train_acc_list_augmented1,
              valid_acc_list=valid_acc_list_augmented1,
              results_dir=None)

plt.ylim([80, 100])
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-24-b733a797cf57> in <module>
----> 1 plot_training_loss(minibatch_loss_list=minibatch_loss_list_augmented1,
      2                    num_epochs=NUM_EPOCHS,
      3                    iter_per_epoch=len(train_loader),
      4                    results_dir=None,
      5                    averaging_iterations=20)

NameError: name 'minibatch_loss_list_augmented1' is not defined
```

Dropout

```
class CNN1Dropout(torch.nn.Module):
  def __init__(self, num_classes, drop_probas=[]):
    super().__init__()
    self.features = torch.nn.Sequential(
            # Conv 1
            torch.nn.Conv2d(3, 16, kernel_size=3, padding="same"), # output 16 - 3 + 1 => 16
                            # , stride=4, padding=2),
            torch.nn.Dropout2d(p=drop_probas[0]),
            torch.nn.ReLU(inplace=True),
            torch.nn.MaxPool2d(kernel_size=2), # 16 / 2 => output 8

            # Conv 2
            torch.nn.Conv2d(16, 32, kernel_size=2, padding="same"), # output 7 - 2 + 1 => 8
                            # , padding=2),
            torch.nn.Dropout2d(p=drop_probas[1]),
            torch.nn.ReLU(inplace=True),
            torch.nn.MaxPool2d(kernel_size=2) #output 8 / 2 => output 4
    )

    self.classifier = torch.nn.Sequential(
         torch.nn.Linear(32*4*4, 100),
          torch.nn.ReLU(inplace=True),
          torch.nn.Linear(100, num_classes),
    )

  def forward(self, x):
    x = self.features(x)
    x = torch.flatten(x, 1)
    # print(x.size())
    logits = self.classifier(x)
    return logits


model_dropout = CNN1Dropout(num_classes=10, drop_probas=[0.5, 0.5])


model_dropout = model_dropout.to(DEVICE)


optimizer_dropout = torch.optim.SGD(model_dropout.parameters(), lr=0.1)
scheduler_dropout = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer,
                                                       factor=0.1,
                                                       mode='max',
                                                       verbose=True)
```

```
minibatch_loss_list_dropout, train_acc_list_dropout, valid_acc_list_dropout = train_model(
    model=model_dropout,
    num_epochs=NUM_EPOCHS,
    train_loader=train_loader,
    valid_loader=valid_loader,
    test_loader=test_loader,
    optimizer=optimizer_dropout,
    device=DEVICE,
    scheduler=None,
    scheduler_on='valid_acc',
    logging_interval=100)
```

```
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__ at 0x7f6381a9e5f0>
Epoch: 001/010 | Batch 0000/0175 | Loss: 2.3020
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py", line 1510, in __del__
    self._shutdown_workers()
  File "/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py", line 1493, in _shutdown_workers
    if w.is_alive():
  File "/usr/lib/python3.7/multiprocessing/process.py", line 151, in is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child process'
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__ at 0x7f6381a9e5f0>
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py", line 1510, in __del__
    self._shutdown_workers()
  File "/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py", line 1493, in _shutdown_workers
    if w.is_alive():
  File "/usr/lib/python3.7/multiprocessing/process.py", line 151, in is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child process'
AssertionError: can only test a child process
Epoch: 001/010 | Batch 0100/0175 | Loss: 2.3109
Epoch: 001/010 | Train: 6.17% | Validation: 6.30%
Time elapsed: 1.04 min
Epoch: 002/010 | Batch 0000/0175 | Loss: 2.3064
Epoch: 002/010 | Batch 0100/0175 | Loss: 2.3031
Epoch: 002/010 | Train: 6.17% | Validation: 6.30%
Time elapsed: 2.09 min
Epoch: 003/010 | Batch 0000/0175 | Loss: 2.3146
Epoch: 003/010 | Batch 0100/0175 | Loss: 2.3152
Epoch: 003/010 | Train: 6.17% | Validation: 6.30%
Time elapsed: 3.21 min
Epoch: 004/010 | Batch 0000/0175 | Loss: 2.3025
Epoch: 004/010 | Batch 0100/0175 | Loss: 2.3094
Epoch: 004/010 | Train: 6.15% | Validation: 6.30%
Time elapsed: 4.27 min
Epoch: 005/010 | Batch 0000/0175 | Loss: 2.2981
Epoch: 005/010 | Batch 0100/0175 | Loss: 2.3052
Epoch: 005/010 | Train: 6.15% | Validation: 6.30%
Time elapsed: 5.38 min
Epoch: 006/010 | Batch 0000/0175 | Loss: 2.3051
Epoch: 006/010 | Batch 0100/0175 | Loss: 2.3033
Epoch: 006/010 | Train: 6.16% | Validation: 6.30%
Time elapsed: 6.43 min
Epoch: 007/010 | Batch 0000/0175 | Loss: 2.3076
Epoch: 007/010 | Batch 0100/0175 | Loss: 2.3137
Epoch: 007/010 | Train: 6.17% | Validation: 6.30%
Time elapsed: 7.46 min
Epoch: 008/010 | Batch 0000/0175 | Loss: 2.3142
Epoch: 008/010 | Batch 0100/0175 | Loss: 2.3178
Epoch: 008/010 | Train: 6.16% | Validation: 6.30%
Time elapsed: 8.48 min
Epoch: 009/010 | Batch 0000/0175 | Loss: 2.3098
Epoch: 009/010 | Batch 0100/0175 | Loss: 2.3091
Epoch: 009/010 | Train: 6.16% | Validation: 6.30%
Time elapsed: 9.51 min
Epoch: 010/010 | Batch 0000/0175 | Loss: 2.3042
Epoch: 010/010 | Batch 0100/0175 | Loss: 2.3015
Epoch: 010/010 | Train: 6.16% | Validation: 6.30%
Time elapsed: 10.52 min
```

```
plot_training_loss(minibatch_loss_list=minibatch_loss_list_dropout,
                   num_epochs=NUM_EPOCHS,
                   iter_per_epoch=len(train_loader),
                   results_dir=None,
                   averaging_iterations=20)

plt.show()

plot_accuracy(train_acc_list=train_acc_list_dropout,
              valid_acc_list=valid_acc_list_dropout,
              results_dir=None)
```

```
plt.ylim([80, 100])
plt.show()


class CNN2Dropout(torch.nn.Module):
  def __init__(self, num_classes, drop_probas=[]):
    super().__init__()
    self.features = torch.nn.Sequential(
            # Conv 1
            torch.nn.Conv2d(3, 16, kernel_size=3, padding="same"), # output 16 - 3 + 1 => 16
                            # , stride=4, padding=2),
            torch.nn.Dropout2d(p=drop_probas[0]),
            torch.nn.ReLU(inplace=True),
            torch.nn.MaxPool2d(kernel_size=2), # 16 / 2 => output 8

            # Conv 2
            torch.nn.Conv2d(16, 32, kernel_size=2, padding="same"), # output 7 - 2 + 1 => 8
                            # , padding=2),
            torch.nn.Dropout2d(p=drop_probas[1]),
            torch.nn.ReLU(inplace=True),
            torch.nn.MaxPool2d(kernel_size=2), #output 8 / 2 => output 4

            # Conv 3
            torch.nn.Conv2d(32, 64, kernel_size=2, padding="same"), # output 7 - 2 + 1 => 4
                            # , padding=2),
            torch.nn.Dropout2d(p=drop_probas[2]),
            torch.nn.ReLU(inplace=True),
            torch.nn.MaxPool2d(kernel_size=2) #output 4 / 2 => output 2

    )

    self.classifier = torch.nn.Sequential(
         torch.nn.Linear(64*2*2, 100),
          torch.nn.ReLU(inplace=True),
          torch.nn.Linear(100, num_classes),
    )

  def forward(self, x):
    x = self.features(x)
    x = torch.flatten(x, 1)
    # print(x.size())
    logits = self.classifier(x)
    return logits


model2_dropout = CNN2Dropout(num_classes=10, drop_probas=[0.2, 0.4, 0.8])


model2_dropout = model2_dropout.to(DEVICE)


optimizer2_dropout = torch.optim.SGD(model2_dropout.parameters(), lr=0.1)
scheduler2_dropout = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer,
                                                       factor=0.1,
                                                       mode='max',
                                                       verbose=True)



minibatch_loss_list_dropout2, train_acc_list_dropout2, valid_acc_list_dropout2 = train_model(
    model=model2_dropout,
    num_epochs=NUM_EPOCHS,
    train_loader=train_loader,
    valid_loader=valid_loader,
    test_loader=test_loader,
    optimizer=optimizer2_dropout,
    device=DEVICE,
    scheduler=None,
    scheduler_on='valid_acc',
    logging_interval=100)
```

```
Epoch: 001/010 | Batch 0000/0175 | Loss: 2.3010
Epoch: 001/010 | Batch 0100/0175 | Loss: 2.3011
Epoch: 001/010 | Train: 11.75% | Validation: 12.00%
Time elapsed: 1.02 min
Epoch: 002/010 | Batch 0000/0175 | Loss: 2.3077
Epoch: 002/010 | Batch 0100/0175 | Loss: 2.2993
Epoch: 002/010 | Train: 11.76% | Validation: 12.00%
```

```
Time elapsed: 2.04 min
Epoch: 003/010 | Batch 0000/0175 | Loss: 2.3054
Epoch: 003/010 | Batch 0100/0175 | Loss: 2.3080
Epoch: 003/010 | Train: 11.73% | Validation: 12.00%
Time elapsed: 3.08 min
Epoch: 004/010 | Batch 0000/0175 | Loss: 2.3050
Epoch: 004/010 | Batch 0100/0175 | Loss: 2.2970
Epoch: 004/010 | Train: 11.74% | Validation: 12.00%
Time elapsed: 4.11 min
Epoch: 005/010 | Batch 0000/0175 | Loss: 2.3085
Epoch: 005/010 | Batch 0100/0175 | Loss: 2.3123
Epoch: 005/010 | Train: 11.75% | Validation: 12.00%
Time elapsed: 5.13 min
Epoch: 006/010 | Batch 0000/0175 | Loss: 2.3067
Epoch: 006/010 | Batch 0100/0175 | Loss: 2.2995
Epoch: 006/010 | Train: 11.73% | Validation: 12.00%
Time elapsed: 6.14 min
Epoch: 007/010 | Batch 0000/0175 | Loss: 2.3009
Epoch: 007/010 | Batch 0100/0175 | Loss: 2.3082
Epoch: 007/010 | Train: 11.75% | Validation: 12.00%
Time elapsed: 7.15 min
Epoch: 008/010 | Batch 0000/0175 | Loss: 2.3010
Epoch: 008/010 | Batch 0100/0175 | Loss: 2.3058
Epoch: 008/010 | Train: 11.75% | Validation: 12.00%
Time elapsed: 8.17 min
Epoch: 009/010 | Batch 0000/0175 | Loss: 2.3018
Epoch: 009/010 | Batch 0100/0175 | Loss: 2.3009
Epoch: 009/010 | Train: 11.72% | Validation: 12.00%
Time elapsed: 9.18 min
Epoch: 010/010 | Batch 0000/0175 | Loss: 2.2975
Epoch: 010/010 | Batch 0100/0175 | Loss: 2.2963
Epoch: 010/010 | Train: 11.75% | Validation: 12.00%
Time elapsed: 10.21 min
Total Training Time: 10.21 min
Test accuracy 11.58%
```

```
plot_training_loss(minibatch_loss_list=minibatch_loss_list_dropout2,
                   num_epochs=NUM_EPOCHS,
                   iter_per_epoch=len(train_loader),
                   results_dir=None,
                   averaging_iterations=20)

plt.show()

plot_accuracy(train_acc_list=train_acc_list_dropout2,
              valid_acc_list=valid_acc_list_dropout2,
              results_dir=None)

plt.ylim([80, 100])
plt.show()
```

Optimser changes

```
class CNN1RMS(torch.nn.Module):
  def __init__(self, num_classes):
    super().__init__()
    self.features = torch.nn.Sequential(
            # Conv 1
            torch.nn.Conv2d(3, 16, kernel_size=3, padding="same"), # output 16 - 3 + 1 => 16
                            # , stride=4, padding=2),
            torch.nn.ReLU(inplace=True),
            torch.nn.MaxPool2d(kernel_size=2), # 16 / 2 => output 8

            # Conv 2
            torch.nn.Conv2d(16, 32, kernel_size=2, padding="same"), # output 7 - 2 + 1 => 8
                            # , padding=2),
            torch.nn.ReLU(inplace=True),
            torch.nn.MaxPool2d(kernel_size=2) #output 8 / 2 => output 4
    )

    self.classifier = torch.nn.Sequential(
         torch.nn.Linear(32*4*4, 100),
          torch.nn.ReLU(inplace=True),
          torch.nn.Linear(100, num_classes),
    )

  def forward(self, x):
```