Exercise 8: Solution

Disclaimer

- In this solution, we just present a minimal model that passes the threshold
- Feel free to play around with different architectures and come up with your own model!
- But also note that you don't always need hundreds of millions parameters to solve a simple problem like this one ☺

Define your Hyperparameters

```
from exercise_code.MyPytorchModel import MyPytorchModel

hparams = {}
########################################################################
# TODO: Define your hyper parameters here!                             #
########################################################################
hparams = {
    "batch_size": 64,
    "learning_rate": 3e-4
}
########################################################################
#                               END OF YOUR CODE                       #
########################################################################
model = MyPytorchModel(hparams)
model.prepare_data()
```

Define your Trainer

```
trainer = None

################################################################################
# TODO: Define your trainer!                                                   #
################################################################################

trainer = pl.Trainer(
    max_epochs = 10,
    gpus = 1 if torch.cuda.is_available() else None
)

################################################################################
#                                END OF YOUR CODE                              #
################################################################################

trainer.fit(model)
```

Initialize your Model

```
def __init__(self, hparams, input_size=3 * 32 * 32, num_classes=10):
    super().__init__()

    # set hyperparams
    self.hparams = hparams
    self.model = None

    ############################################################################
    # TODO: Initialize your model!                                             #
    ############################################################################

    self.model = nn.Sequential(
        nn.Linear(input_size, 500),
        nn.BatchNorm1d(500),
        nn.ReLU(),
        nn.Dropout(p=0.5),
        nn.Linear(500, 100),
        nn.BatchNorm1d(100),
        nn.ReLU(),
        nn.Dropout(p=0.5),
        nn.Linear(100, 10)
    )

    ############################################################################
    #                           END OF YOUR CODE                               #
    ############################################################################
```

Just an example.
Be creative here and come up with your own architecture ☺

Prepare Data

```
################################################################################
# TODO: Define your transforms (convert to tensors, normalize).                #
# If you want, you can also perform data augmentation!                         #
################################################################################
mean = [0.485, 0.456, 0.406]
std = [0.229, 0.224, 0.225]
my_transform = transforms.Compose([
    transforms.RandomApply((transforms.RandomHorizontalFlip(p=0.8),
                            transforms.RandomResizedCrop((32,32))), p=0.1),
    transforms.ToTensor(),
    transforms.Normalize(mean, std)
])
################################################################################
#                               END OF YOUR CODE                               #
################################################################################
cifar_complete = torchvision.datasets.ImageFolder(root=CIFAR_ROOT, transform=my_transform)
```

Define Optimizers

```
def configure_optimizers(self):

    optim = None
    ##########################################################################
    # TODO: Define your optimizer.                                           #
    ##########################################################################

    optim = torch.optim.Adam(self.model.parameters(), self.hparams["learning_rate"])

    ##########################################################################
    #                              END OF YOUR CODE                          #
    ##########################################################################
    return optim
```

Questions? Moodle