

```
# Adapted from: Chapter 7 and 8 of Deep Learning with Pytorch by Eli Stevens (2020)
# References
# 1. 2010-glorot.pdf from milestone papers
# 2. 2015-HeInitialization.pdf from milestone papers
# 3. 2015-BatchNorm.pdf from milestone papers
# 4. Section 11.4 of UDLBook
# 5. Chapter 7 of UDLBook
try:
    import torch as t
    import torch.nn as tnn
except ImportError:
    print("Colab users: pytorch comes preinstalled. Select Change Ru")
    print("Local users: Please install pytorch for your hardware using instructions:
    print("ACG users: Please follow instructions here: https://vikasdhiman.info/ECF

    raise

if t.cuda.is_available():
    DEVICE="cuda"
elif t.mps.is_available():
    DEVICE="mps"
else:
    DEVICE="cpu"

DTYPE = t.get_default_dtype()

## Doing it the Pytorch way without using our custom feature extraction

import torch
import torch.nn
import torch.optim
import torchvision
from torchvision.transforms import ToTensor, Compose, Normalize
from torch.utils.data import DataLoader

torch.manual_seed(17)
DATASET_MEAN = [0.4914, 0.4822, 0.4465]
DATASET_STD = [0.2470, 0.2435, 0.2616]
# Getting the dataset, the Pytorch way
all_training_data = torchvision.datasets.CIFAR10(
    root="data",
    train=True,
    download=True,
    transform=Compose([ToTensor(),
                        Normalize(DATASET_MEAN, # dataset mean
                                DATASET_STD)]) # dataset std
)

test_data = torchvision.datasets.CIFAR10(
```

```

▶ Executing (54s) <cell li... > tr... > _n... > _next... > f... > <list... > __geti... > __geti... > _c... > _c... > to_te... ... ✕
    train=False,
    download=True,
    transform=Compose([ToTensor(),
                        Normalize(DATASET_MEAN, # dataset mean
                                DATASET_STD)]) # dataset std
)

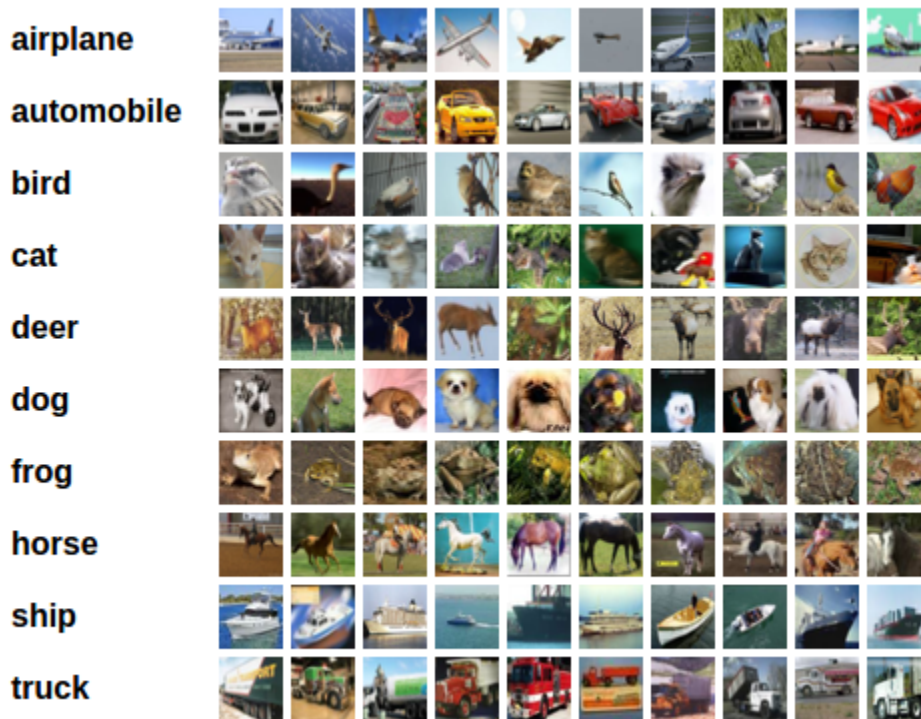
```

```

Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to data/c:
100%|██████████| 170498071/170498071 [00:02<00:00, 73180943.58it/s]
Extracting data/cifar-10-python.tar.gz to data
Files already downloaded and verified

```

```
training_data, validation_data = torch.utils.data.random_split(all_training_data, [
```



```

img, label = all_training_data[99]
img.shape, label

```

```
(torch.Size([3, 32, 32]), 1)
```

```

import matplotlib.pyplot as plt
plt.imshow(img.permute(1, 2, 0))

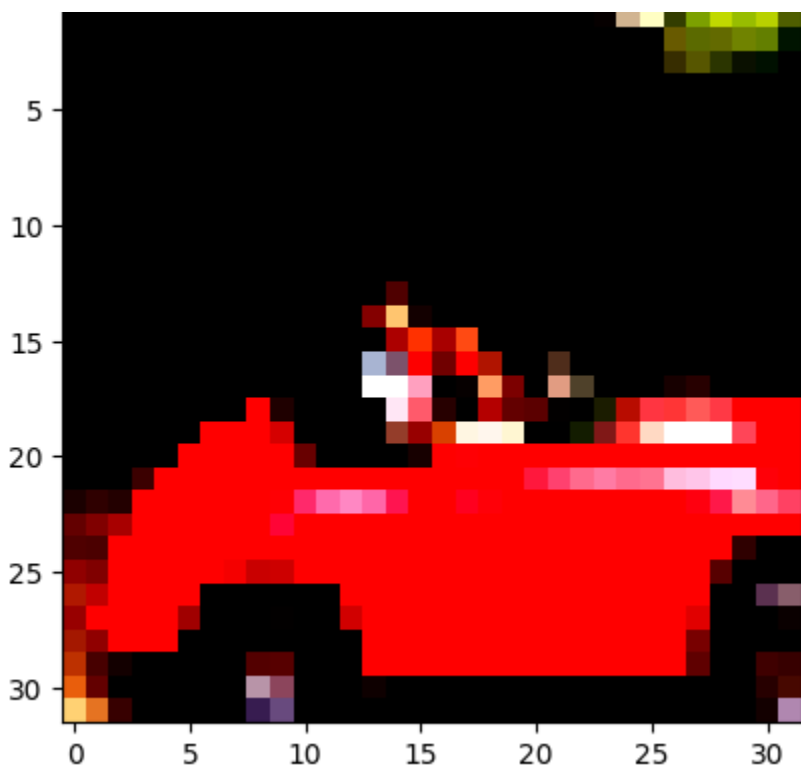
```

```

WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
<matplotlib.image.AxesImage at 0x7f3fd043b430>

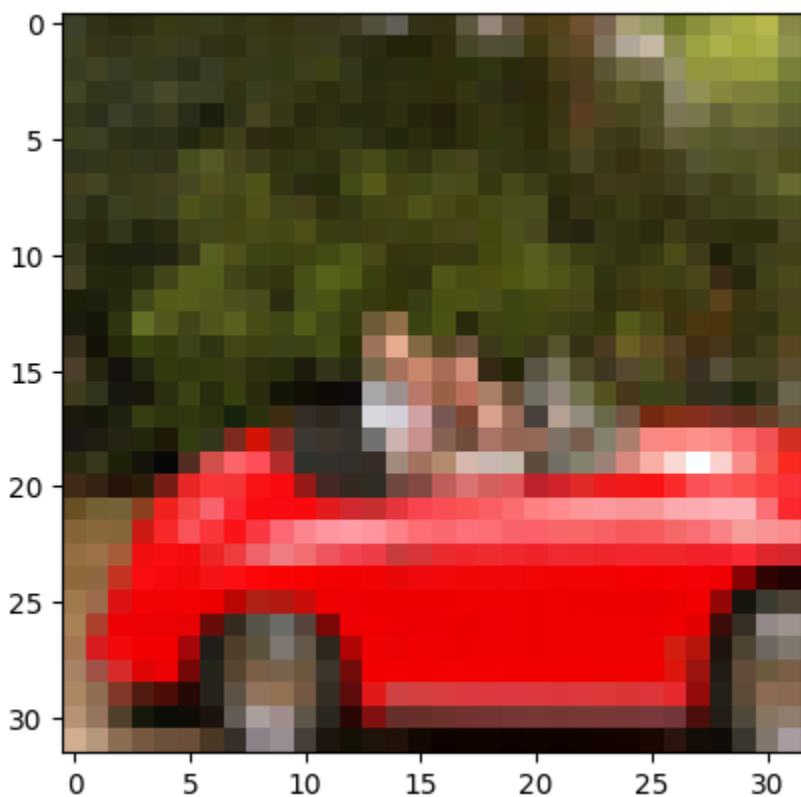
```





```
plt.imshow((img.permute(1, 2, 0) * torch.Tensor(DATASET_STD)
            + torch.Tensor(DATASET_MEAN)))
```

<matplotlib.image.AxesImage at 0x7f3fd02df4c0>



```
imgs = torch.stack([img_t for img_t, _ in all_training_data], dim=3)
imgs.reshape(3, -1).mean(dim=-1), imgs.reshape(3, -1).std(dim=-1)

(tensor([-1.2762e-06, -1.7074e-04,  1.1819e-04]),
 tensor([1.0001, 0.9999, 1.0000]))

import pickle
cifar_meta = pickle.load(open("data/cifar-10-batches-py/batches.meta", "rb"), encoding='utf-8')
class_names = [c.decode('utf-8') for c in cifar_meta[b'label_names']]
class_names

['airplane',
 'automobile',
 'bird',
 'cat',
 'deer',
 'dog',
 'frog',
 'horse',
 'ship',
 'truck']

# Hyper parameters
learning_rate = 1e-3 # controls how fast the gradient descent goes
batch_size = 64
epochs = 5
momentum = 0.9

training_dataloader = DataLoader(training_data, shuffle=True, batch_size=batch_size)
validation_dataloader = DataLoader(validation_data, batch_size=batch_size)
test_dataloader = DataLoader(test_data, batch_size=batch_size)
X, y = next(iter(training_dataloader))
X.shape

torch.Size([64, 3, 32, 32])

!pip install tensorboard

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-apt
Requirement already satisfied: tensorboard in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: protobuf>=3.19.6 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: numpy>=1.12.0 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: absl-py>=0.4 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: grpcio>=1.48.2 in /usr/local/lib/python3.9/dist-packages
```

```
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.9/
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.9/dist-pa
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.9/dist-
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/pytho
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.9/dist-pa
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/pytl
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/pytho
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/py
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-p
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9,
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.9/c
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.9/dist-pac
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.9/di
```

```
%load_ext tensorboard
%tensorboard --logdir=runs
```

TensorBoard

INACTIVE

No dashboards are active for the current data set.

Probable causes:

- You haven't written any data to your event files.
- TensorBoard can't find your event files.

If you're new to using TensorBoard, and want to find out how to add data and set up your event files, check out the [README](#) and perhaps the [TensorBoard tutorial](#).

If you think TensorBoard is configured properly, please see [the section of the README devoted to missing data problems](#) and consider filing an issue on GitHub.

Last reload: Apr 13, 2023, 1:38:41 PM

Log directory: runs

```
from torch.utils.tensorboard import SummaryWriter
from torch.optim.lr_scheduler import ReduceLROnPlateau
import os
writer = SummaryWriter()

loss = torch.nn.CrossEntropyLoss()

# class Net(tnn.Module):
#     def __init__(self):
#         super().__init__()
#         # define input size, hidden layer size, output size
#         D_i, D_k, D_o = 3*32*32, 100, 10
#         self.f = tnn.Flatten()
#         self.l1 = tnn.Linear(D_i, D_k, bias=False)
#         self.b1 = tnn.BatchNorm1d(D_k)
#         self.a1 = tnn.ReLU()
#         self.l2 = tnn.Linear(D_k, D_o)

#     def forward(self, x):
#         self.f_out = self.f(x)
#         self.l1_out = self.l1(self.f_out)
#         self.b1_out = self.b1(self.l1_out)
#         self.a1_out = self.a1(self.b1_out)
#         self.l2_out = self.l2(self.a1_out)
#         return self.l2_out

# model = Net()

# define input size, hidden layer size, output size
D_i, D_k, D_o = 3*32*32, 100, 10
model = tnn.Sequential(
    tnn.Flatten(),
    tnn.Linear(D_i, D_k, bias=False),
    tnn.BatchNorm1d(D_k),
    tnn.ReLU(),
    tnn.Linear(D_k, D_o)
)
```

```

# print(list(model.named_parameters()))

# Glorot or Xavier initialization of weights
def init_weights(m):
    if isinstance(m, (tnn.Linear, tnn.Conv2d)):
        torch.nn.init.kaiming_uniform_(m.weight, nonlinearity='relu')
        # m.bias.data.fill_(0)

model.apply(init_weights)

def loss_and_accuracy(model, loss, validation_dataloader, device=DEVICE):
    # Validation loop
    validation_size = len(validation_dataloader.dataset)
    num_batches = len(validation_dataloader)
    test_loss, correct = 0, 0

    with torch.no_grad():
        model.eval() # Put model in eval mode, affects layers like dropout and
        for X, y in validation_dataloader:
            X = X.to(device)
            y = y.to(device)
            pred = model(X)
            test_loss += loss(pred, y)
            correct += (pred.argmax(dim=-1) == y).type(DTYPE).sum()

    test_loss /= num_batches
    correct /= validation_size
    return test_loss, correct

def train(model, loss, training_dataloader, validation_dataloader, device=DEVICE, c
    # Define optimizer
    optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate, momentum=mome
    scheduler = ReduceLROnPlateau(optimizer, 'min')
    model.to(device)
    t0 = 0
    if not ignore_chkpt and os.path.exists(f"runs/{chkpt_name}"):
        checkpoint = torch.load(f"runs/{chkpt_name}")
        model.load_state_dict(checkpoint['model_state_dict'])
        optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
        t0 = checkpoint['epoch']

    for t in range(t0, epochs):
        # Train loop
        training_size = len(training_dataloader.dataset)
        nbatches = len(training_dataloader)
        model.train() # Put model in train mode, affects layers like dropout and ba
        for batch, (X, y) in enumerate(training_dataloader):
            X = X.to(device)
            y = y.to(device)
            # Compute prediction and loss

```

```
trained_model = train(model, loss, training_dataloader, validation_dataloader, chkp
test_loss, correct = loss_and_accuracy(model, loss, test_dataloader)
print(f"Test Error: \n Accuracy: {(100*correct):>0.1f}%, Avg loss: {test_loss:>8f}
```

8 of 10


```
0., 0., 0., 0.], requires_grad=True)), ('4.weight', Parameter containing:
tensor([[ 0.0709,  0.0782,  0.0848, -0.0909, -0.0726,  0.0927,  0.0114, -0.0100,
         -0.0608, -0.0433,  0.0770, -0.0703, -0.0210, -0.0316, -0.0518,  0.0490,
          0.0136, -0.0489, -0.0238, -0.0347,  0.0809,  0.0455,  0.0984, -0.0400,
         -0.0562, -0.0729,  0.0985,  0.0218, -0.0347, -0.0804,  0.0060,  0.0190,
          0.0298, -0.0306,  0.0793,  0.0897,  0.0392, -0.0096,  0.0931,  0.0170,
         -0.0718, -0.0351, -0.0133,  0.0873, -0.0747, -0.0172, -0.0958,  0.0080,
         -0.0508, -0.0934,  0.0348, -0.0389,  0.0372, -0.0371,  0.0141, -0.0700,
         -0.0675,  0.0806, -0.0965, -0.0980,  0.0127,  0.0440, -0.0584,  0.0900,
          0.0964, -0.0403,  0.0963,  0.0796, -0.0636, -0.0133,  0.0358, -0.0100,
          0.0373, -0.0487,  0.0901,  0.0995,  0.0008,  0.0702,  0.0146,  0.0800,
          0.0094,  0.0963,  0.0146,  0.0245,  0.0065, -0.0438, -0.0614,  0.0740,
          0.0128, -0.0173, -0.0965, -0.0417, -0.0960, -0.0260,  0.0025, -0.0890,
          0.0284,  0.0480, -0.0144, -0.0521],
        [-0.0851,  0.0805,  0.0900, -0.0173, -0.0005, -0.0925,  0.0612, -0.0400,
         -0.0946,  0.0524,  0.0226, -0.0501,  0.0109,  0.0450,  0.0653,  0.0900,
          0.0857, -0.0151, -0.0560,  0.0294, -0.0166,  0.0335,  0.0782,  0.0000,
          0.0454, -0.0105, -0.0878,  0.0290, -0.0168, -0.0111,  0.0344, -0.0200,
          0.0367,  0.0931,  0.0323,  0.0160,  0.0651, -0.0514,  0.0038, -0.0190,
          0.0393,  0.0193,  0.0465, -0.0680, -0.0848,  0.0457, -0.0351,  0.0600,
          0.0859, -0.0309, -0.0798,  0.0473,  0.0099, -0.0528,  0.0280,  0.0600,
         -0.0579,  0.0152, -0.0115, -0.0596, -0.0682, -0.0161, -0.0451,  0.0900,
          0.0425,  0.0418,  0.0512, -0.0760, -0.0100, -0.0437,  0.0616, -0.0800,
         -0.0113, -0.0864,  0.0253, -0.0600, -0.0555, -0.0045, -0.0403, -0.0900,
         -0.0446, -0.0178, -0.0258, -0.0181,  0.0706, -0.0967, -0.0053, -0.0200,
          0.0818,  0.0196,  0.0578, -0.0638, -0.0309, -0.0526, -0.0533,  0.0000,
          0.0894, -0.0606,  0.0309,  0.0490],
        [ 0.0030,  0.0755,  0.0253,  0.0273, -0.0347, -0.0118,  0.0463, -0.0000,
          0.0704, -0.0103, -0.0588, -0.0779,  0.0074, -0.0607,  0.0695,  0.0700,
          0.0119, -0.0048, -0.0443, -0.0292, -0.0643, -0.0809,  0.0356,  0.0400,
          0.0235,  0.0922,  0.0020, -0.0316, -0.0575, -0.0695,  0.0368,  0.0100,
         -0.0237,  0.0079, -0.0558, -0.0597,  0.0246, -0.0686,  0.0042, -0.0900,
          0.0218,  0.0899,  0.0337, -0.0536,  0.0333,  0.0166,  0.0354, -0.0000,
          0.0558, -0.0413,  0.0592, -0.0987,  0.0463, -0.0291, -0.0359, -0.0700,
         -0.0924,  0.0776,  0.0422, -0.0312, -0.0908,  0.0573,  0.0588,  0.0900,
         -0.0928, -0.0131, -0.0091,  0.0777, -0.0899,  0.0634,  0.0807,  0.0100,
         -0.0523,  0.0134,  0.0306,  0.0451,  0.0528,  0.0679, -0.0788, -0.0000,
         -0.0083,  0.0255, -0.0343, -0.0378,  0.0787, -0.0797,  0.0828, -0.0200,
         -0.0939, -0.0582, -0.0016,  0.0378,  0.0073, -0.0678, -0.0449,  0.0300])
```

[Colab paid products](#) - [Cancel contracts here](#)