

```
In [ ]: # Adapted from: Chapter 7 and 8 of Deep Learning with Pytorch by Eli Stevens
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 instr")
    print("ACG users: Please follow instructions here: https://vikasdhiman.i")

    raise

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

DTYPE = t.get_default_dtype()
```

```
In [ ]: ## 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
)

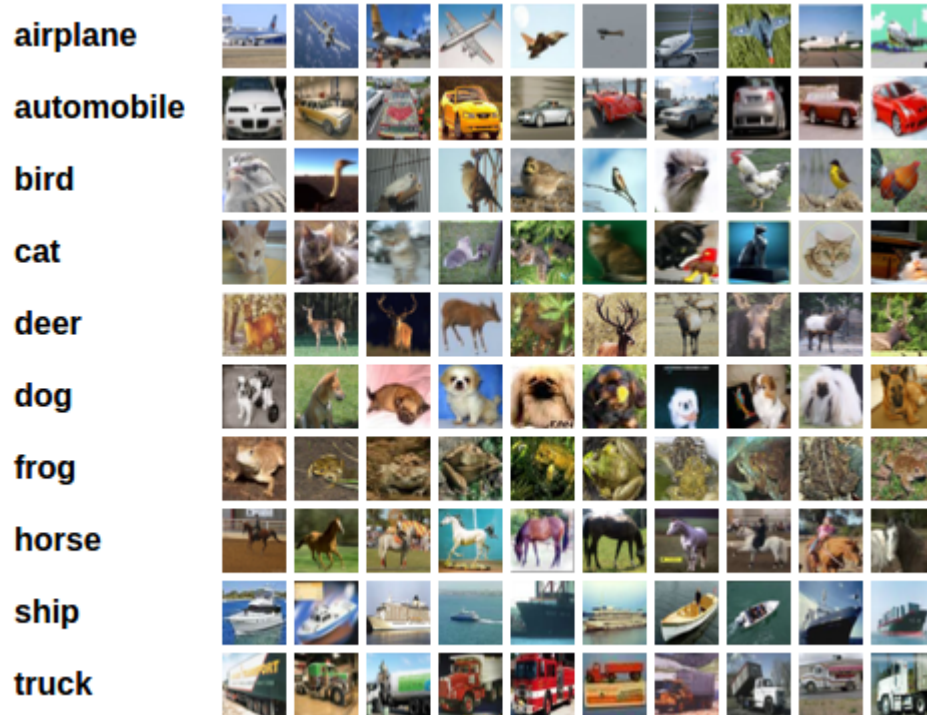
test_data = torchvision.datasets.CIFAR10(
    root="data",
    train=False,
    download=True,
    transform=Compose([ToTensor(),
                        Normalize(DATASET_MEAN, # dataset mean
                                DATASET_STD)]) # dataset std
)
```

Downloading <https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz> to data/
cifar-10-python.tar.gz

100%|██████████| 170498071/170498071 [00:06<00:00, 27535441.22it/s]

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

```
In [ ]: training_data, validation_data = torch.utils.data.random_split(all_training_
```



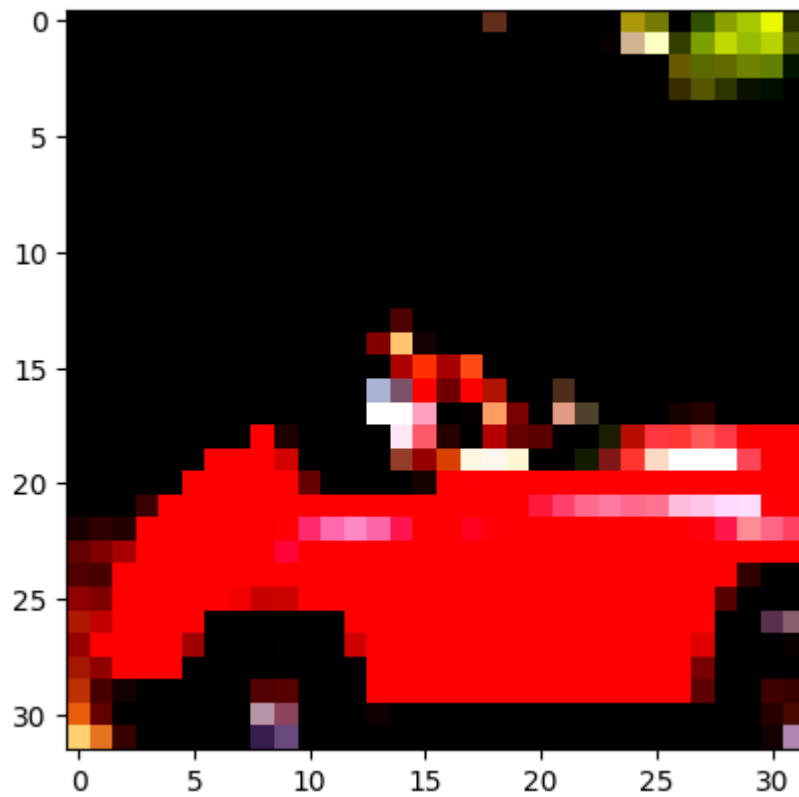
```
In [ ]: img, label = all_training_data[99]  
img.shape, label
```

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

```
In [ ]: import matplotlib.pyplot as plt  
plt.imshow(img.permute(1, 2, 0))
```

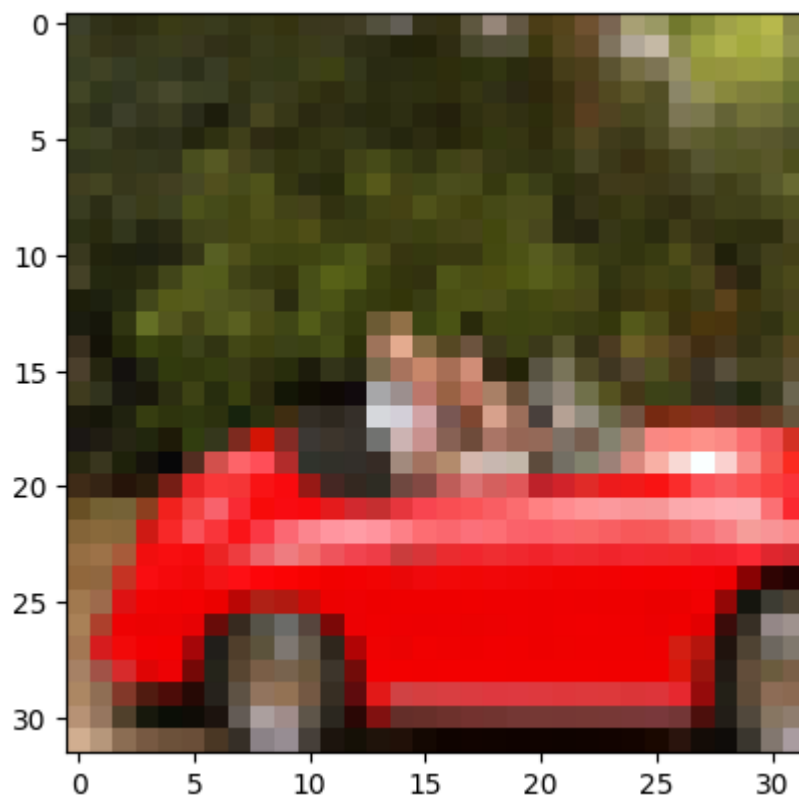
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

```
Out[ ]: <matplotlib.image.AxesImage at 0x7feacb580850>
```



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

```
Out[ ]: <matplotlib.image.AxesImage at 0x7feac19ae8b0>
```



```
In [ ]: 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)
```

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

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

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

```
In [ ]: # 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
```

```
Out[ ]: torch.Size([64, 3, 32, 32])
```

```
In [ ]: !pip install tensorboard
```

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/>

Requirement already satisfied: tensorboard in /usr/local/lib/python3.9/dist-packages (2.12.1)

Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.9/dist-packages (from tensorboard) (2.17.2)

Requirement already satisfied: protobuf<=3.19.6 in /usr/local/lib/python3.9/dist-packages (from tensorboard) (3.20.3)

Requirement already satisfied: numpy<=1.12.0 in /usr/local/lib/python3.9/dist-packages (from tensorboard) (1.22.4)

Requirement already satisfied: tensorboard-plugin-wit<=1.6.0 in /usr/local/lib/python3.9/dist-packages (from tensorboard) (1.8.1)

Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.9/dist-packages (from tensorboard) (2.27.1)

Requirement already satisfied: markdown<=2.6.8 in /usr/local/lib/python3.9/dist-packages (from tensorboard) (3.4.3)

Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.9/dist-packages (from tensorboard) (0.7.0)

Requirement already satisfied: absl-py<=0.4 in /usr/local/lib/python3.9/dist-packages (from tensorboard) (1.4.0)

Requirement already satisfied: setuptools<=41.0.0 in /usr/local/lib/python3.9/dist-packages (from tensorboard) (67.6.1)

Requirement already satisfied: werkzeug<=1.0.1 in /usr/local/lib/python3.9/dist-packages (from tensorboard) (2.2.3)

Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in /usr/local/lib/python3.9/dist-packages (from tensorboard) (1.0.0)

Requirement already satisfied: wheel<=0.26 in /usr/local/lib/python3.9/dist-packages (from tensorboard) (0.40.0)

Requirement already satisfied: grpcio<=1.48.2 in /usr/local/lib/python3.9/dist-packages (from tensorboard) (1.53.0)

Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.9/dist-packages (from google-auth<3,>=1.6.3->tensorboard) (4.9)

Requirement already satisfied: pyasn1-modules<=0.2.1 in /usr/local/lib/python3.9/dist-packages (from google-auth<3,>=1.6.3->tensorboard) (0.2.8)

Requirement already satisfied: six<=1.9.0 in /usr/local/lib/python3.9/dist-packages (from google-auth<3,>=1.6.3->tensorboard) (1.16.0)

Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from google-auth<3,>=1.6.3->tensorboard) (5.3.0)

Requirement already satisfied: requests-oauthlib<=0.7.0 in /usr/local/lib/python3.9/dist-packages (from google-auth-oauthlib<1.1,>=0.5->tensorboard) (1.3.1)

Requirement already satisfied: importlib-metadata<=4.4 in /usr/local/lib/python3.9/dist-packages (from markdown<=2.6.8->tensorboard) (6.2.0)

Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests<3,>=2.21.0->tensorboard) (3.4)

Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests<3,>=2.21.0->tensorboard) (1.26.15)

Requirement already satisfied: certifi<=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests<3,>=2.21.0->tensorboard) (2022.12.7)

Requirement already satisfied: charset-normalizer<=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests<3,>=2.21.0->tensorboard) (2.0.12)

Requirement already satisfied: MarkupSafe<=2.1.1 in /usr/local/lib/python3.9/dist-packages (from werkzeug<=1.0.1->tensorboard) (2.1.2)

Requirement already satisfied: zipp<=0.5 in /usr/local/lib/python3.9/dist-packages (from importlib-metadata<=4.4->markdown<=2.6.8->tensorboard) (3.15.0)

Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python

3.9/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard) (0.4.8)

Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.9/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<1.1,>=0.5->tensorboard) (3.2.2)

```
In [ ]: %load_ext tensorboard
        %tensorboard --logdir=runs
```

```
In [44]: from torch.utils.tensorboard import SummaryWriter
import os
writer = SummaryWriter()

loss = torch.nn.CrossEntropyLoss()
# TODO:
# Define model = ?

model = tnn.Sequential(
    tnn.Flatten(),
    tnn.Linear(3*32*32, 100),
    tnn.ReLU(),
    tnn.Linear(100, 10))

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
        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=DE
    # Define optimizer
    optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate, moment
    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
    for batch, (X, y) in enumerate(training_dataloader):
        X = X.to(device)
        y = y.to(device)
        # Compute prediction and loss
        pred = model(X)
        loss_t = loss(pred, y)

        # Backpropagation
        optimizer.zero_grad()
        loss_t.backward()
        optimizer.step()

        if batch % 100 == 0:
            writer.add_scalar("Train/loss_batch", loss_t, t*nbatches +
                             batch)
            loss_t, current = loss_t.item(), (batch + 1) * len(X)
            print(f"loss: {loss_t:>7f}  [{current:>5d}/{training_size:>5d}]")

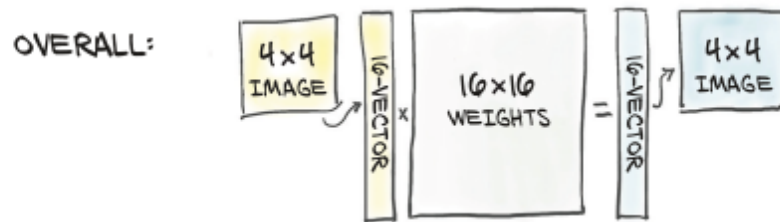
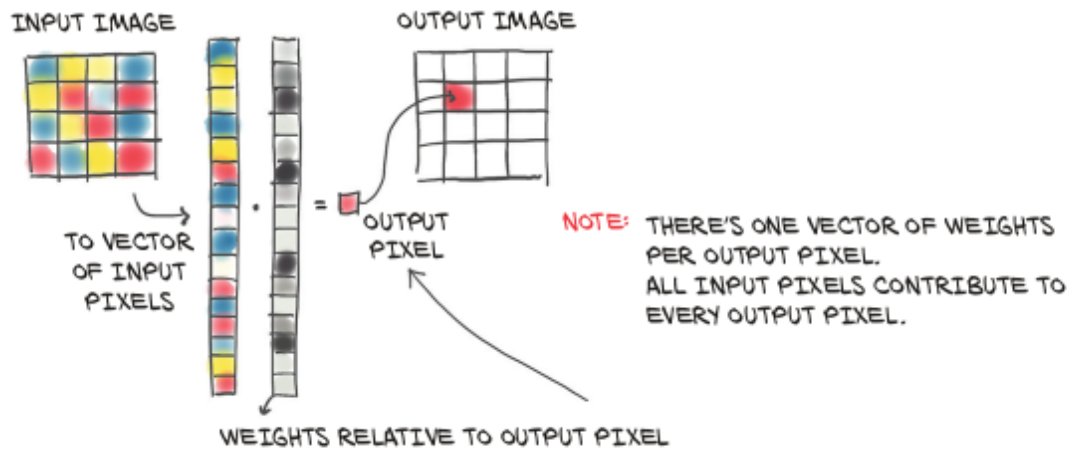
    writer.add_scalar("Train/loss", loss_t, t)
    valid_loss, correct = loss_and_accuracy(model, loss, validation_dataloader)
    writer.add_scalar("Valid/loss", valid_loss, t)
    writer.add_scalar("Valid/accuracy", correct, t)
    print(f"Validation Error: \n Accuracy: {(100*correct):>0.1f}%, Avg loss: {valid_loss:>0.1f}")
    if t % 3 == 0:
        torch.save({
            'epoch': t,
            'model_state_dict': model.state_dict(),
            'optimizer_state_dict': optimizer.state_dict(),
        }, f"runs/{chkpt_name}")
    return model

# trained_model = train(model, loss, training_dataloader, validation_dataloader)

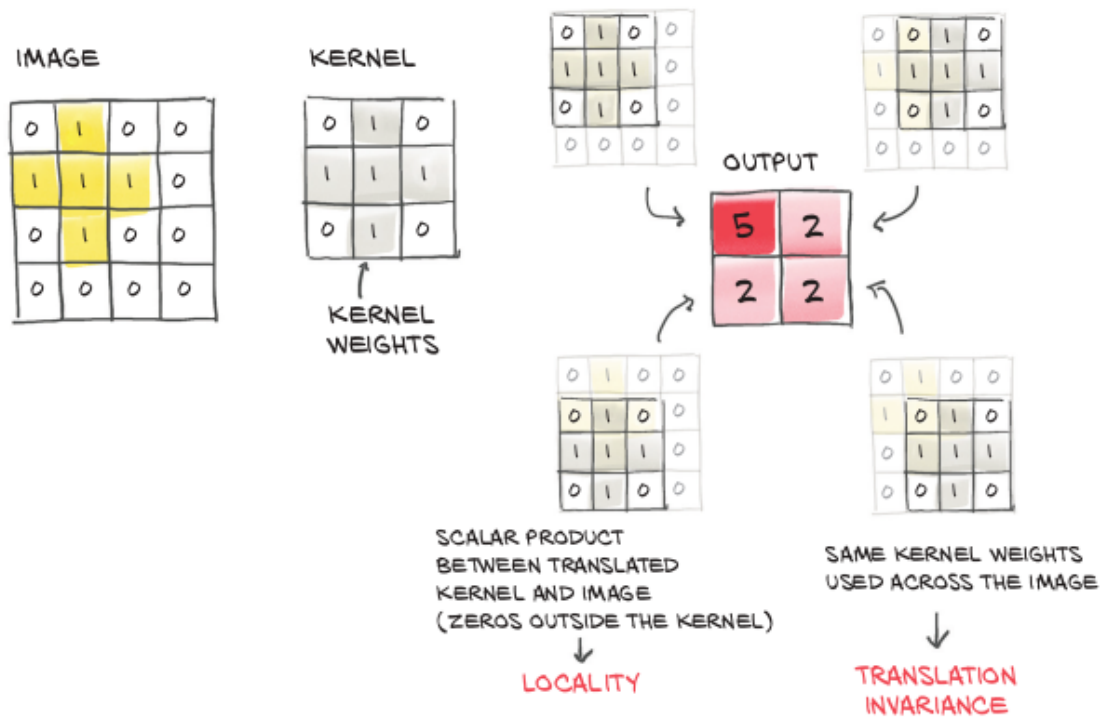
# test_loss, correct = loss_and_accuracy(model, loss, test_dataloader)
# print(f"Test Error: \n Accuracy: {(100*correct):>0.1f}%, Avg loss: {test_loss:>0.1f}")

```

Limits of fully connected layers

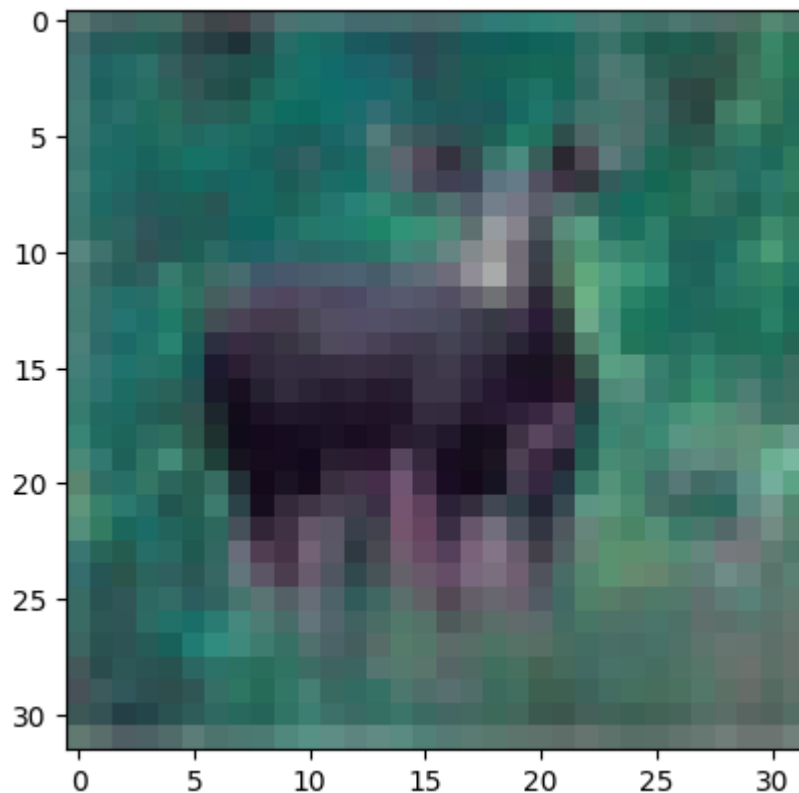


Convolutions bring translation invariance



```
In [ ]: img, label = training_data[99]
plt.imshow(img.permute(1, 2, 0) * t.Tensor(DATASET_STD) + t.Tensor(DATASET_M
```

```
Out[ ]: <matplotlib.image.AxesImage at 0x7feac18e0d00>
```

```
In [ ]: !mkdir -p imgs
!wget "http://cliparts.co/cliparts/Aib/KoG/AibKoGRzT.png" -O imgs/sobel.png

--2023-04-11 18:15:05-- http://cliparts.co/cliparts/Aib/KoG/AibKoGRzT.png
Resolving cliparts.co (cliparts.co)... 173.208.212.194
Connecting to cliparts.co (cliparts.co)|173.208.212.194|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 248899 (243K) [image/png]
Saving to: 'imgs/sobel.png'

imgs/sobel.png      100%[=====>] 243.07K   501KB/s   in 0.5s

2023-04-11 18:15:06 (501 KB/s) - 'imgs/sobel.png' saved [248899/248899]
```

```
In [ ]: conv = torch.nn.Conv2d(1, 1, kernel_size=3).to(DEVICE)

print(list(conv.named_parameters()))
with torch.no_grad():
    conv.weight[:] = torch.Tensor([[-1, 0, 1],
                                   [-1, 0, 1],
                                   [-1, 0, 1]])

    conv.bias[:] = 0
sobelimg = torchvision.io.read_image('imgs/sobel.png').to(DEVICE)
sobelimg = torchvision.transforms.functional.resize(sobelimg, (400, 400))
sobelimg = torchvision.transforms.functional.rgb_to_grayscale(sobelimg[:3, :])
sobelimg = sobelimg.to(DTYPE) / sobelimg.max()

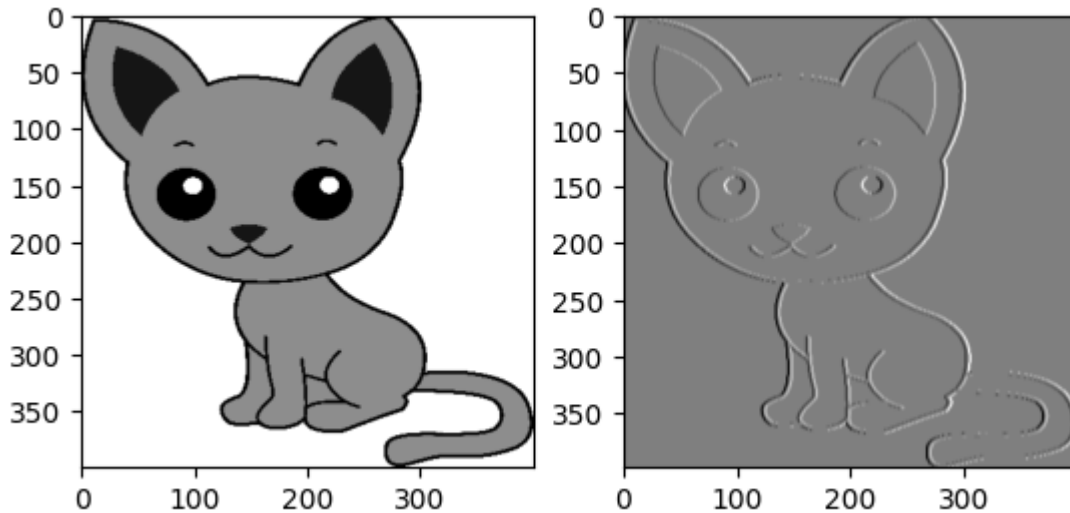
convded = conv(sobelimg)

fig, axes = plt.subplots(1, 2)
```

```
axes[0].imshow(sobelimg[0].detach().cpu(), cmap='gray')
axes[1].imshow(conved[0].detach().cpu(), cmap='gray')
```

```
[('weight', Parameter containing:
  tensor([[[[-0.2667, -0.0684,  0.0015],
             [ 0.0508,  0.2152,  0.2220],
             [ 0.2655,  0.0926, -0.1547]]]], device='cuda:0', requires_grad=True)), ('bias', Parameter containing:
  tensor([0.1529], device='cuda:0', requires_grad=True))]
```

Out[]: <matplotlib.image.AxesImage at 0x7feald2ccbe0>



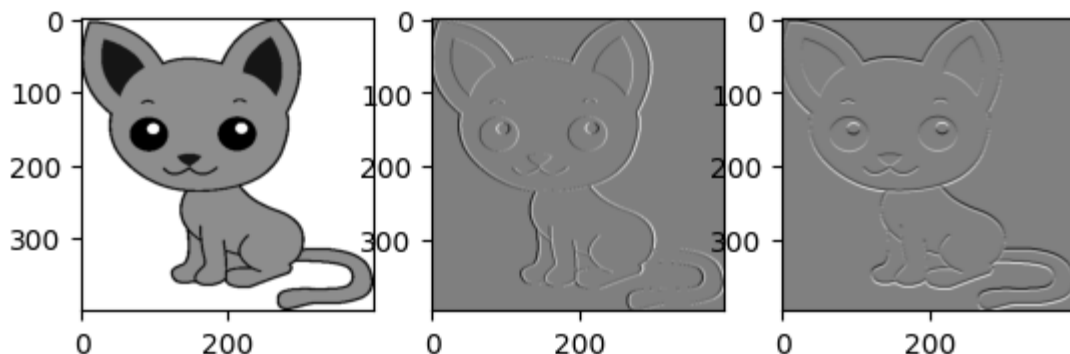
```
In [ ]: with torch.no_grad():
        conv.weight[:] = torch.Tensor([[-1, -1, -1],
                                         [ 0,  0,  0],
                                         [ 1,  1,  1]])

        conv.bias[:] = 0

        conved2 = conv(sobelimg)

        fig, axes = plt.subplots(1, 3)
        axes[0].imshow(sobelimg[0].detach().cpu(), cmap='gray')
        axes[1].imshow(conved[0].detach().cpu(), cmap='gray')
        axes[2].imshow(conved2[0].detach().cpu(), cmap='gray')
```

Out[]: <matplotlib.image.AxesImage at 0x7fea240aef40>



```

In [ ]: import math
ks=20
conv = torch.nn.Conv2d(1, 1, kernel_size=ks).to(DEVICE)

with torch.no_grad():
    imin = -ks//2
    imax = imin + ks
    i = torch.arange(imin, imax).to(DTYPE)
    j = torch.arange(imin, imax).to(DTYPE)
    sigma = torch.Tensor([ks//6])

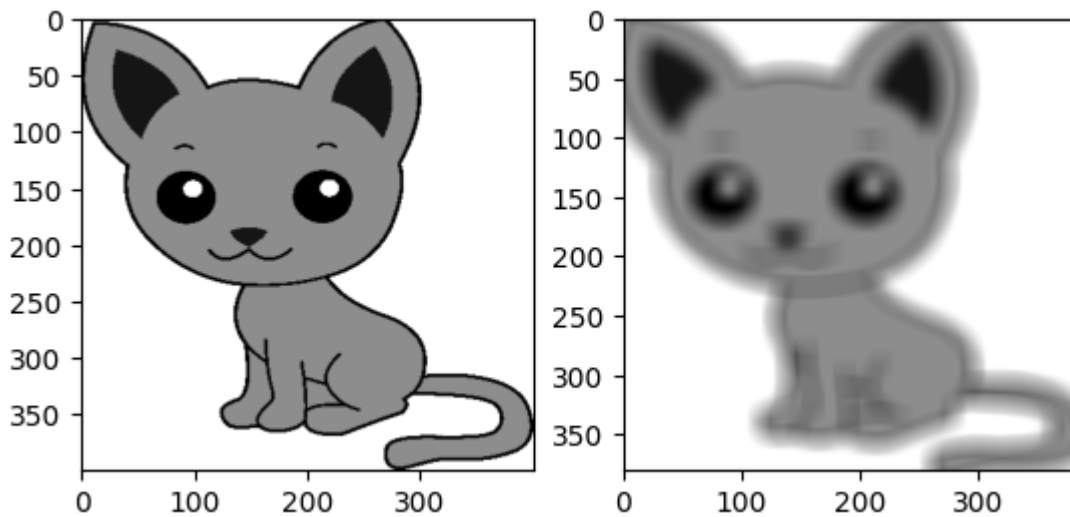
    grid_x, grid_y = torch.meshgrid(i, j, indexing='ij')
    #conv.weight[:] = torch.exp(- (grid_x**2 + grid_y**2) / (2*sigma**2))
    conv.weight[:] = torch.ones((ks, ks)) / ks**2
    conv.bias[:] = 0

convolved2 = conv(sobeling)

fig, axes = plt.subplots(1, 2)
axes[0].imshow(sobeling[0].detach().cpu(), cmap='gray')
axes[1].imshow(convolved2[0].detach().cpu(), cmap='gray')

```

Out[]: <matplotlib.image.AxesImage at 0x7fealc5e1af0>



```

In [ ]: !wget https://github.com/rajatsaxena/OpenCV/raw/master/mario.jpg -O imgs/mar
!wget https://github.com/rajatsaxena/OpenCV/raw/master/mario_template.jpg -O

```

```
--2023-04-11 18:15:12-- https://github.com/rajatsaxena/OpenCV/raw/master/mario.jpg
Resolving github.com (github.com)... 140.82.121.4
Connecting to github.com (github.com)|140.82.121.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/rajatsaxena/OpenCV/master/mario.jpg [following]
--2023-04-11 18:15:12-- https://raw.githubusercontent.com/rajatsaxena/OpenCV/master/mario.jpg
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7301 (7.1K) [image/jpeg]
Saving to: 'imgs/mario.jpg'
```

```
imgs/mario.jpg      100%[=====>]    7.13K  --.-KB/s    in 0s
```

```
2023-04-11 18:15:12 (55.6 MB/s) - 'imgs/mario.jpg' saved [7301/7301]
```

```
--2023-04-11 18:15:12-- https://github.com/rajatsaxena/OpenCV/raw/master/mario_template.jpg
Resolving github.com (github.com)... 140.82.121.3
Connecting to github.com (github.com)|140.82.121.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/rajatsaxena/OpenCV/master/mario_template.jpg [following]
--2023-04-11 18:15:12-- https://raw.githubusercontent.com/rajatsaxena/OpenCV/master/mario_template.jpg
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 855 [image/jpeg]
Saving to: 'imgs/mario_template.jpg'
```

```
imgs/mario_template 100%[=====>]      855  --.-KB/s    in 0s
```

```
2023-04-11 18:15:13 (41.1 MB/s) - 'imgs/mario_template.jpg' saved [855/855]
```

```
In [ ]: with torch.no_grad():
        sobelimg = torchvision.io.read_image('imgs/mario.jpg').to(DEVICE)
        #sobelimg = torchvision.transforms.functional.rgb_to_grayscale(sobelimg)
        sobelimg = sobelimg.to(DTYPE) / sobelimg.max()

        patch = torchvision.io.read_image('imgs/mario_template.jpg').to(DEVICE)
        #patch = torchvision.transforms.functional.rgb_to_grayscale(patch[:3, :, :])
        patch = patch.to(DTYPE) / patch.max()
        fig, ax = plt.subplots(figsize=(0.5,0.5))
        ax.imshow(patch.permute(1, 2, 0).cpu())

        kr, kc = patch.shape[1:]
        conv = torch.nn.Conv2d(3, 1, kernel_size=(kr, kc)).to(DEVICE)
```

```

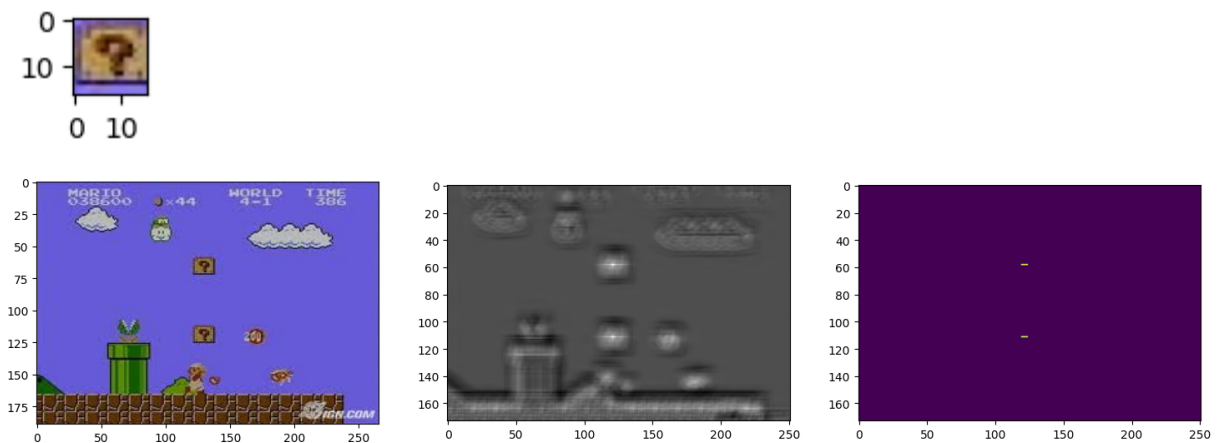
with torch.no_grad():
    patch = (patch - patch.mean()) / patch.std()
    conv.weight[:] = patch
    conv.bias[:] = 0

sobelimgnormed = (sobelimg - sobelimg.mean()) / sobelimg.std()
conved2 = conv(sobelimgnormed)

fig, axes = plt.subplots(1, 3, figsize=(18, 6))
axes[0].imshow(sobelimg.permute(1, 2, 0).detach().cpu())
axes[1].imshow(conved2[0].detach().cpu(), cmap='gray')
axes[2].imshow((conved2[0].abs() > 380).detach().cpu())

```

Out[]: <matplotlib.image.AxesImage at 0x7feald62fa00>



Learning with Convolutions

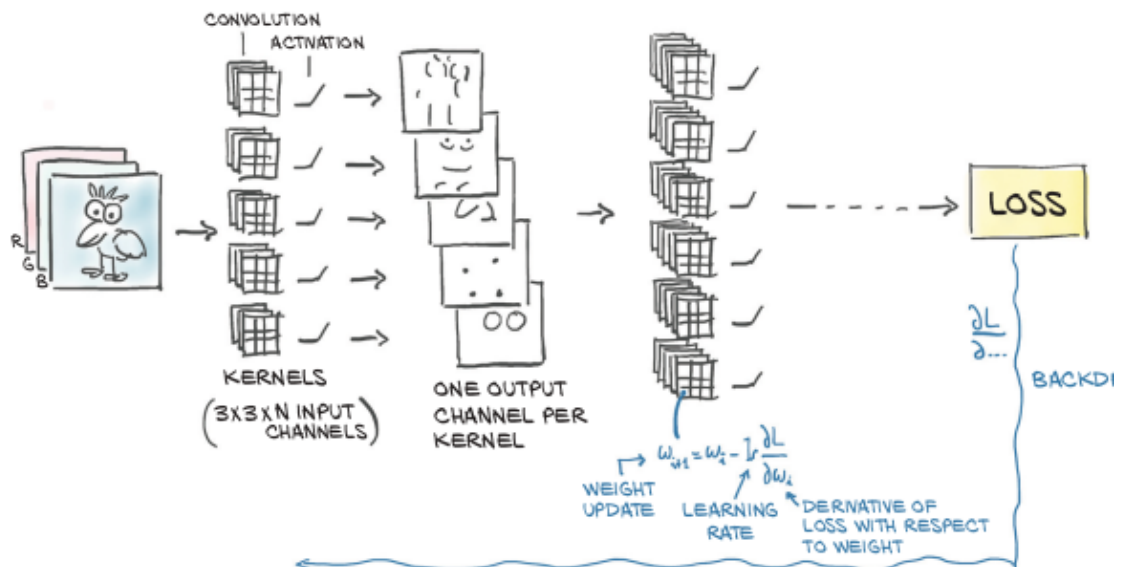


Figure 8.6 The process of learning with convolutions by estimating the gradient at the kernel weights and updating them individually in order to optimize for the loss

Maxpooling

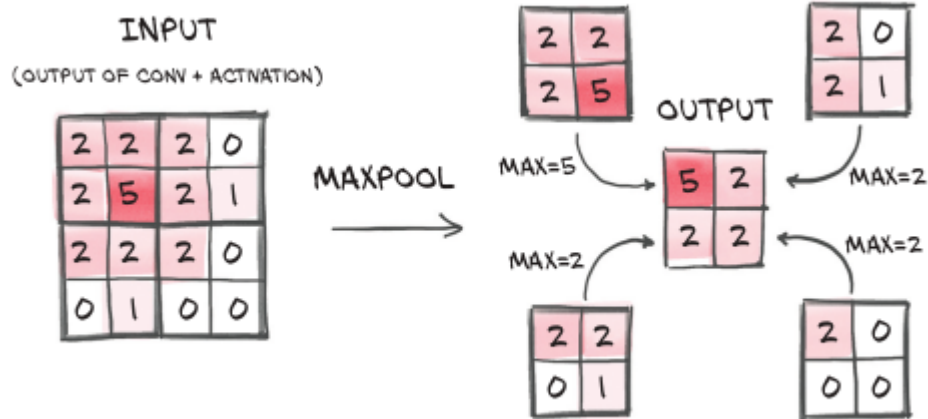


Figure 9.7 Max pooling in detail

```
In [ ]: conv = tnn.Conv2d(3, 16, 3, stride=2, padding=1) # number of weights = 3*3 *
img, label = training_data[99]
conv(img).shape
```

```
Out[ ]: torch.Size([16, 16, 16])
```

```
In [ ]: import os

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(
    root="data",
    train=False,
    download=True,
    transform=Compose([ToTensor(),
                        Normalize(DATASET_MEAN, # dataset mean
                                DATASET_STD)]) # dataset std
)

training_data, validation_data = torch.utils.data.random_split(all_training_data,
                                                                [10000, 1000])

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)
```

```

loss = torch.nn.CrossEntropyLoss()
model = tnn.Sequential(
    tnn.Conv2d(3, 16, 3, padding=1), # 32x32 image 3 channels
    tnn.ReLU(),
    tnn.MaxPool2d(2), # -> 16x16 image with 16 channels
    tnn.Conv2d(16, 16, 3, padding=1), # 16x16 image with 16 channels
    tnn.ReLU(),
    tnn.MaxPool2d(2), # -> 8x8 image with 16 channels
    tnn.Flatten(),
    tnn.Linear(16*8*8, 100),
    tnn.ReLU(),
    tnn.Linear(100, 10))

trained_model = train(model, loss, training_dataloader, validation_dataloader,
                      chkpt_name='conv_model_chkpt.pt', ignore_chkpt=True)

test_loss, correct = loss_and_accuracy(model, loss, test_dataloader)
print(f"Test Error: \n Accuracy: {(100*correct):>0.1f}%, Avg loss: {test_loss}")

```

Files already downloaded and verified

Files already downloaded and verified

Validation Error:

Accuracy: 35.3%, Avg loss: 1.825797

Validation Error:

Accuracy: 43.3%, Avg loss: 1.586233

In []: