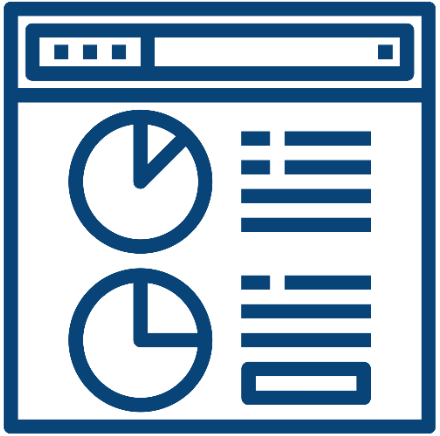
Assignment 4
Introduction

Sebastian Dörrich (MSc)

Deep Learning Exercise

Chair of Explainable Machine Learning (xAI)

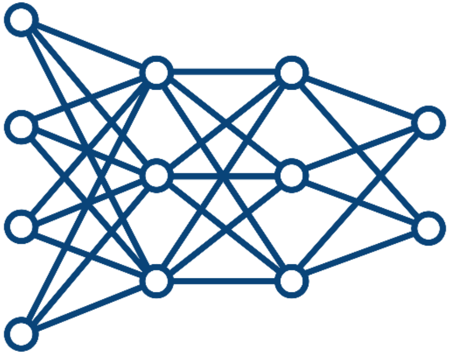
Overview



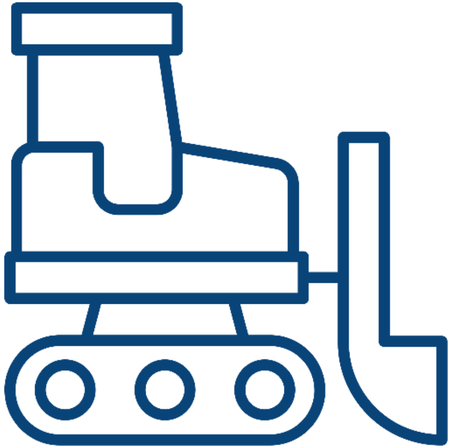
Overview

1. Convolutional Neural Networks
   a) CNN from scratch
   b) CNN with PyTorch

Convolutional Neural Networks



CNN from Scratch

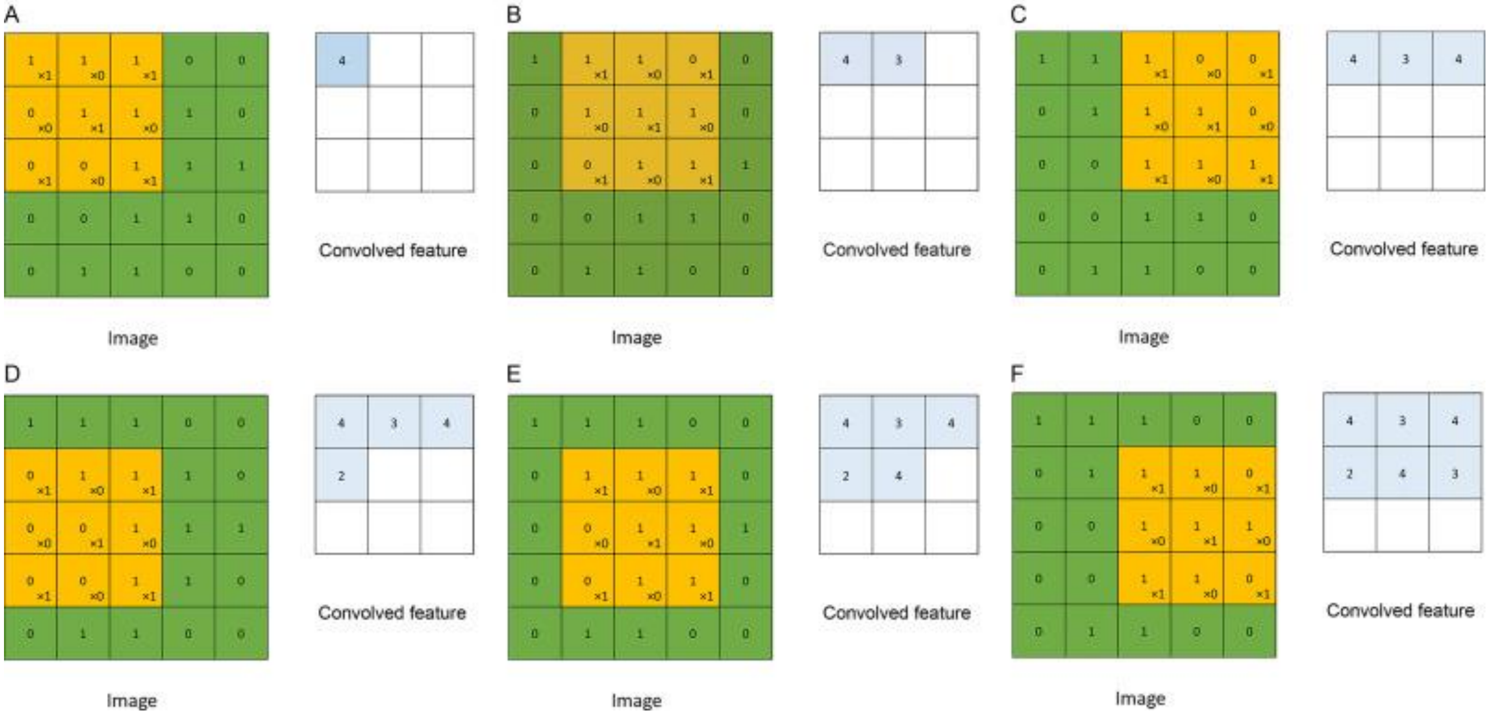


ToDo

- Implement modules/layers:
  - Convolution
  - Max Pooling
  - ReLU
  - Linear

- Implement optimizer
  - SGD with momentum

Convolution (or cross-correlation)



https://www.sciencedirect.com/topics/mathematics/convolutional-layer

Vectorization

- It’s okay to use loops for this assignment

- Vectorization is not necessary for this assignment but in practice, it is very important to speed up your code

- Note Convolutional layers are just linear transforms of kernels (weights) with different “views” of the image (i.e. toeplitz matrix)

- Use this notion to vectorize.

Vectorization – Tricks (1/2)

Goal:

Create views of an image the same size as the filters we’re applying:

```
(c, h, w) → (h_new, w_new, c, k, k)
```

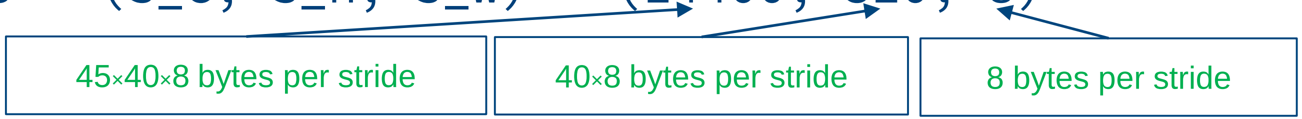
Given an image A as a NumPy array:

```
A.shape = (c, h, w) = (3, 45, 40)
```

Assuming A is of type float64, its strides from NumPy are:

```
A.strides = (s_c, s_h, s_w) = (14400, 320, 8)
```

“Strides represent how many bytes of offset to get to the next value in that dimension.”



45×40×8 bytes per stride

40×8 bytes per stride

8 bytes per stride

Vectorization – Tricks (2/2)

In preparation for a convolution layer, we want to create “views” of the image with the same size as the filter. Assuming filter size `k=3` and stride `stride=2`, we have:

```
    h_new = (45 - 3) // 2 + 1 = 22
    w_new = (40 - 3) // 2 + 1 = 19

np.lib.stride_tricks.as_strided(
    A,
    shape=(22, 19, 3, k, k),
    strides=(s_h*2, s_w*2, s_c, s_h, s_w),
    writeable=False)
```

Vectorization – Multiplying Tensors

As an example, given matrices:

A of shape `(n, c, h, w)` → containing n images

B of shape `(c, h, w, k)` → containing filters the same size as the image

How to apply the filters to all the images at once?

Method 1

```
np.tensordot(A, B, axes=3)
```

Method 2

```
np.einsum(‘nchw,chwk->nk’, A, B)
```

Vectorization – Broadcasting

Given two matrices:

`A` is of size `(a, b, c)` and `B` is of size `(b,)`

adding them together along a specific dimension can be done via:

`A + B[:, np.newaxis]` or `A + B[:, None]`

This tells NumPy to broadcast across the 1st and 3rd dimensions before performing addition.

Vectorization – Unravel Index

How to retrieve the location of a point given the argmax?

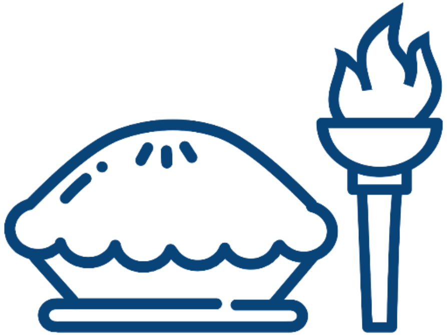
```
A = np.array([[ 3,  5],
              [17,  2],
              [16,  3]])

i = A.argmax() = 2
```

This can be done via:

```
np.unravel_index(i, A.shape) = (1, 0)
```

CNN with PyTorch



ToDo

- Complete the training loop of PyTorch
  - Notice the difference between training and validation

- Implement two specified models
  - Two fully connected layers with a sigmoid activation function in between
  - Vanilla CNN

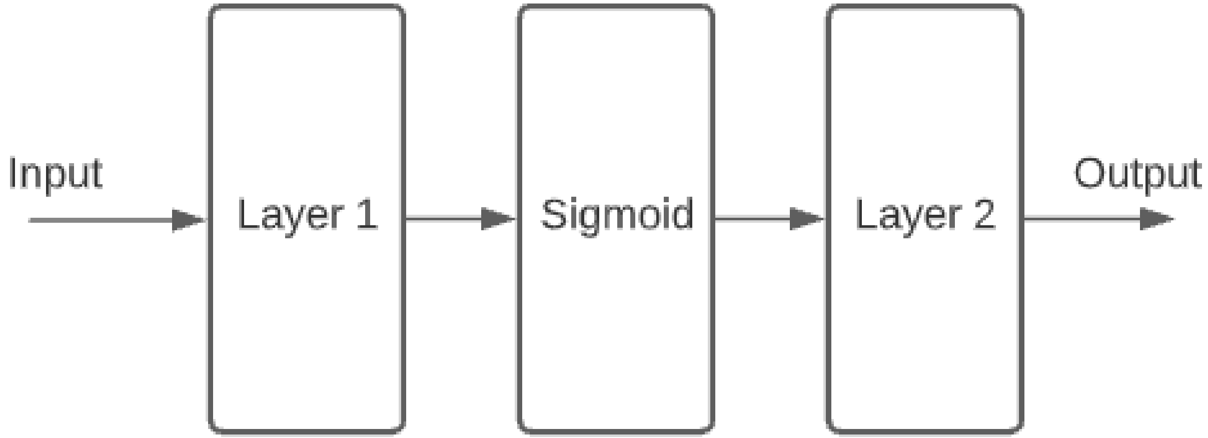
PyTorch – Complete the Training Loop

- Implement the training and validation step utilizing PyTorch

- Key concept is that in validation you won't be updating the gradients

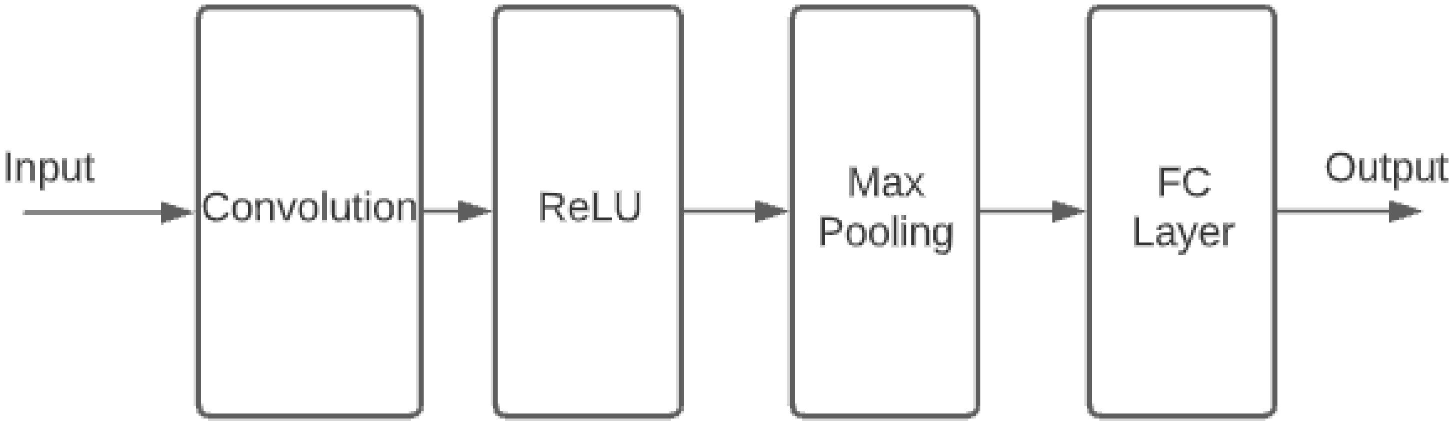
PyTorch – Two Layer Model

Two fully connected layers with a sigmoid activation function in between



PyTorch – CNN

Implement a basic CNN utilizing PyTorch


Input
Convolution
ReLU
Max Pooling
FC Layer
Output