# Video Streaming and tracking

HW1 - Classification

Deadline: 2022/10/10

# Objective

- Train a neural network to do classification.
  - Implement a VGG19-like Network.
  - Environment: **Anaconda**
  - Framework: **pytorch**
- Goal: Top-1 Accuracy >= 55%
- Fewer parameters can get higher score!

**Not allow !!!**

**1. Use pretrained weight**

**2. Call torchvision build model.**

# Environment Version Constraint

You can use the following four package version (Suggestion).

- Python: 3.8
- Pytorch: 1.9.1
- Pandas: 1.4.4
- Matplotlib: 3.5.3
- Scikit-Image: 0.19.3

If you have other package, you need to descript them in your report.

You can provide your environment (Optional).

# Dataset

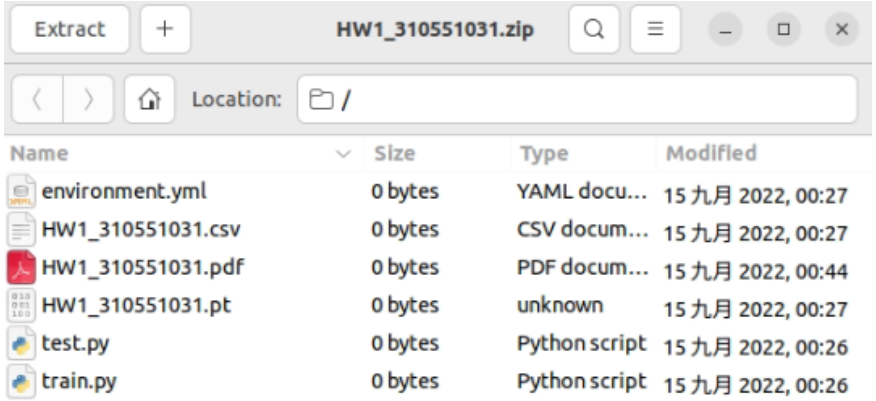
Dataset Link: https://reurl.cc/YXn934

- The dataset consists 10 classes of 224 × 224 RGB images.
- We will give you 850 images for training, 285 for validation, 287 for testing.
- For training and validation data, we will give you labels in **train.csv** and **val.csv**.
- For testing data, we will not give you labels.

# Submission

- **Your submission should contain:**

  - Training Code and Testing Code
  - Model Weight(HW1_StudentID.pt)
  - Testing Result(HW1_StudentID.csv)
  - Report (HW1_StudentID.pdf)

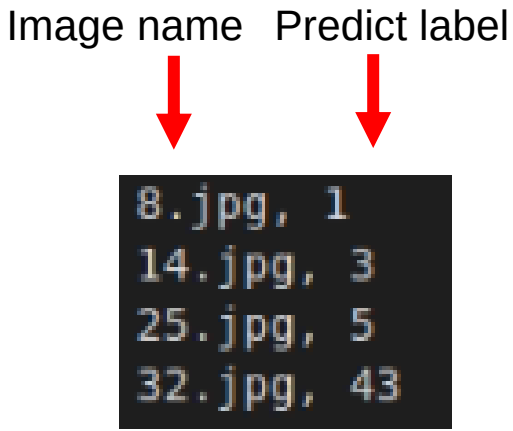



Compress them into **one zip** file name HW1_StudentID.zip.

(Do **not** contain dataset in your submission)

# Submission - Testing Result

- Please predict the label for testing data(in order) into a txt file named HW1_StudentID.csv (ex: HW1_310551031.csv)
- The format description is as follows.
- Put it in the HW1_StudentID.csv.
  (if the path or format is wrong, you will get -10 points)
- TA will test the csv file.



# Submission - Report

1. How to reproduce your result. (Including package, environment, reproduced method) (4 points)

   ex: python test.py

2. Number of Model parameters (4 points)

   ex: `number_of_params = sum(p.numel() for p in model.parameters() if p.requires_grad)`

3. Explain model structure (as detail as possible) (4 points)

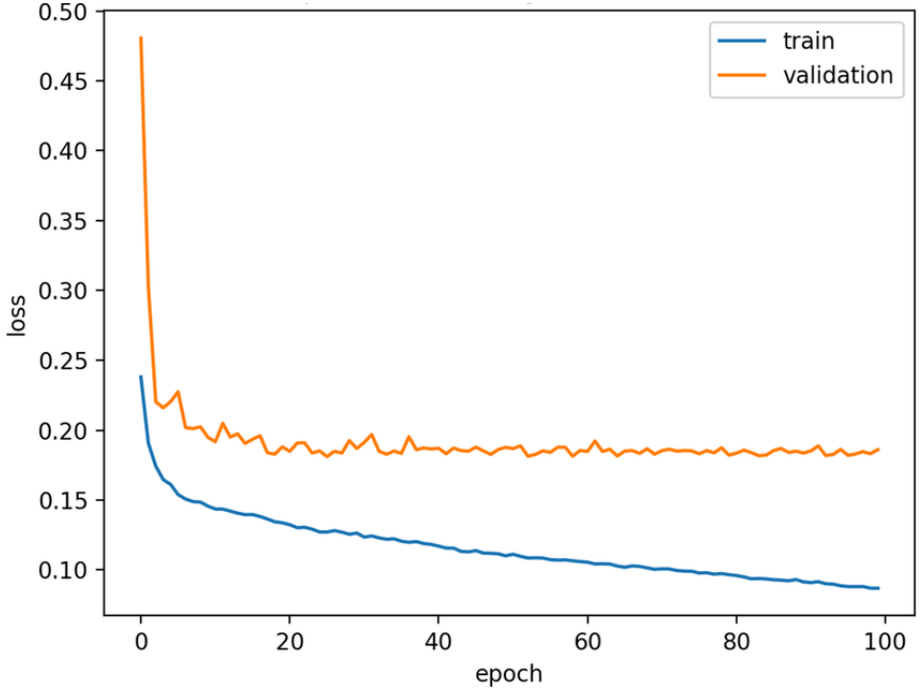
   ex: The layer number, convolution layer, channel number, loss function… revised from VGGnet19.

4. Results (training and validation loss curve, training and validation accuracy curve) (4 points)

5. Problems encountered and discussion (4 points)

# Submission - Report

## Loss Curve



# Grading

- Top-1 Accuracy >= 55% (65 points)
  - 50% <= Accuracy < 55% (55 points)
  - 35% <= Accuracy < 50% (50 points)
  - 35% > Accuracy (0 points)

- Number of parameters (15 points)
  - Compare with your classmates(fewer parameters gets higher score)
  - Top 1/5 get 15 points, second 1/5 get 12 points, etc.

- Report (20 points)

# Penalty

Format penalty - **10 points (Maximum)**

**If you have any incorrect file format or name, then you will get -5 points.**

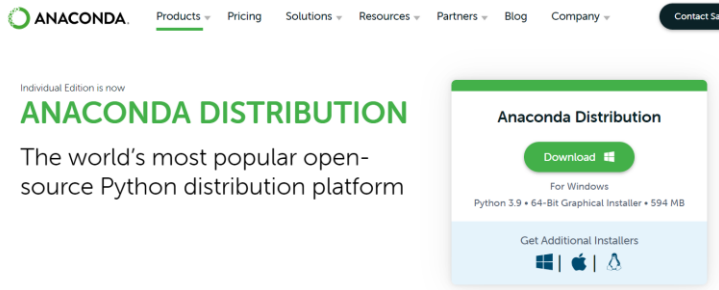
- Submit the result in the wrong name **-5 points**
- Submit the result in the wrong format **-5 points**

Late penalty - **20% per day**

- 1 day => 80%, 2 day => 60%...

# Reference

# Anaconda




1. Create Environment

   conda create –n [environment name] python=[python version]

2. Activate Environment

   conda activate [environment name]

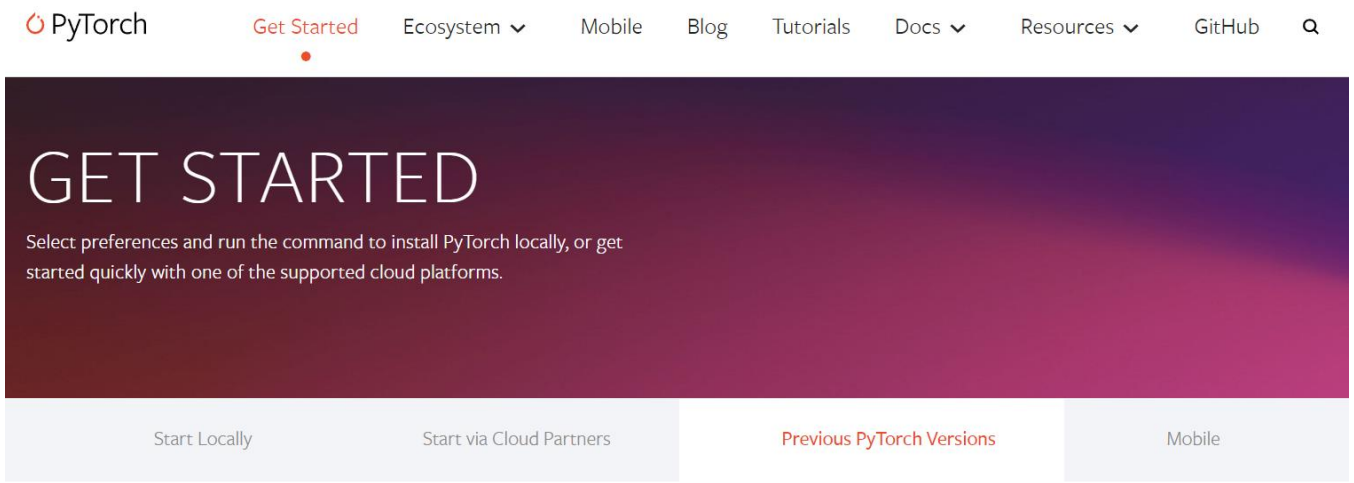
3. Export Environment (You should activate your current environment then you can export your environment)

   conda env export > environment.yml

# Pytorch

Tutorial1: https://www.youtube.com/watch?v=85uJ9hSaXig

Tutorial2: https://www.youtube.com/watch?v=VbqNn20FoHM



# Pytorch – Data Process

1. Import package

```
from torch.utils import data
```

2. Construct dataset

```
class SportLoader(data.Dataset):
    def __init__(self, mode, transform=None):
        self.mode = mode
        self.sport = # read the file from csv file
        self.img_name = # image name list from the csv file
        self.label = # label list from the csv file
        self.transform = transform # image transform

    def __len__(self):
        return len(self.img_name)

    def __getitem__(self, index):

        image_path = self.mode+"/"+self.img_name[index]
        self.img = io.imread(image_path)
        self.target = self.label[index]

        if self.transform:
            self.img = self.transform(self.img)

        return self.img, self.target
```

→ Initialize image and label data

→ Numbers of image and label

→ Return  single image and label

# Pytorch – Data Loader

1. Import package

```
from torch.utils.data import DataLoader
```

2. Select your mode in datasets(Page14)

```
train_dataset=SportLoader("train")
valid_dataset=SportLoader("val")
test_dataset=SportLoader("test")
```

3. Setting batch_size in the dataloader

```
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
valid_loader = DataLoader(valid_dataset, batch_size=64, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=True)
```

# Pytorch – Build Model

1. Import package

```
import torch.nn as nn
import torch.nn.functional as F
```

2. Build model

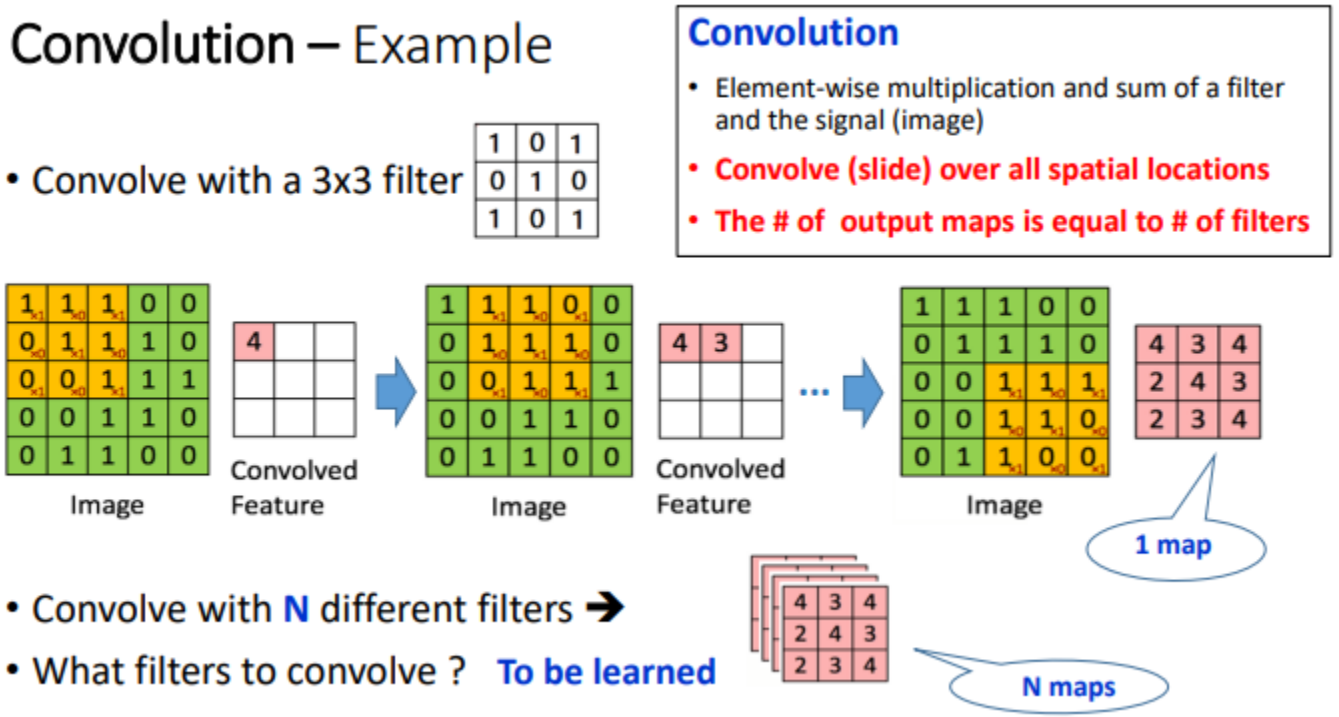
```
class Network(nn.Module):
    def __init__(self):
        super(Network, self).__init__()

        self.conv1 = nn.Conv2d(in_channels=3, out_channels=12, kernel_size=5, stride=1, padding=1)
        self.conv2 = nn.Conv2d(in_channels=12, out_channels=12, kernel_size=5, stride=1, padding=1)
        self.pool = nn.MaxPool2d(2,2)
        self.conv4 = nn.Conv2d(in_channels=12, out_channels=24, kernel_size=5, stride=1, padding=1)
        self.conv5 = nn.Conv2d(in_channels=24, out_channels=24, kernel_size=5, stride=1, padding=1)
        self.fc1 = nn.Linear(24*10*10, 10)

    def forward(self, input):
        output = F.relu(self.conv1(input))
        output = F.relu(self.conv2(output))
        output = self.pool(output)
        output = F.relu(self.conv4(output))
        output = F.relu(self.conv5(output))
        output = output.view(-1, 24*10*10)
        output = self.fc1(output)

        return output
```

# Pytorch – Conv2D



# Pytorch – Conv2D

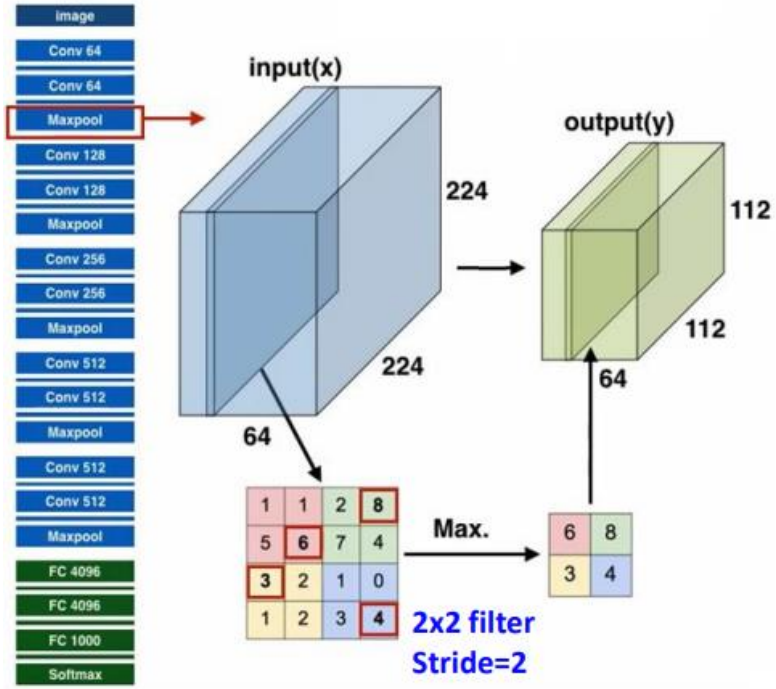
## CONV2D

CLASS `torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)` [SOURCE]

Parameters

- **in_channels** (*int*) – Number of channels in the input image
- **out_channels** (*int*) – Number of channels produced by the convolution
- **kernel_size** (*int or tuple*) – Size of the convolving kernel
- **stride** (*int or tuple, optional*) – Stride of the convolution. Default: 1
- **padding** (*int, tuple or str, optional*) – Padding added to all four sides of the input. Default: 0
- **padding_mode** (*string, optional*) – `'zeros'`, `'reflect'`, `'replicate'` or `'circular'`. Default: `'zeros'`
- **dilation** (*int or tuple, optional*) – Spacing between kernel elements. Default: 1
- **groups** (*int, optional*) – Number of blocked connections from input channels to output channels. Default: 1
- **bias** (*bool, optional*) – If `True`, adds a learnable bias to the output. Default: `True`

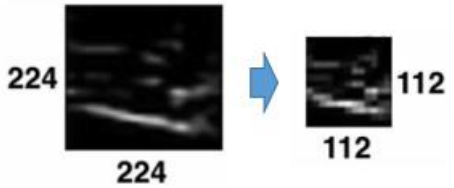
# Pytorch – MaxPool2D




## Max Pooling

1. Reduce dimension
2. produce an abstracted form of the representation

➔ avoid over-fitting



# Pytorch – MaxPool2D

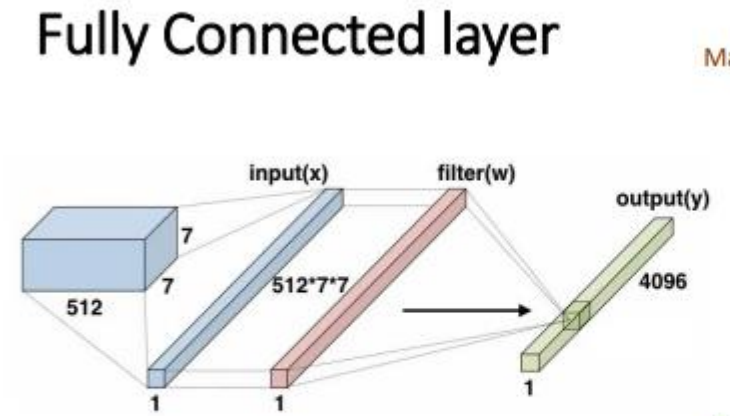
## MAXPOOL2D

CLASS `torch.nn.MaxPool2d(kernel_size, stride=None, padding=0, dilation=1, return_indices=False, ceil_mode=False)` [SOURCE]

Parameters

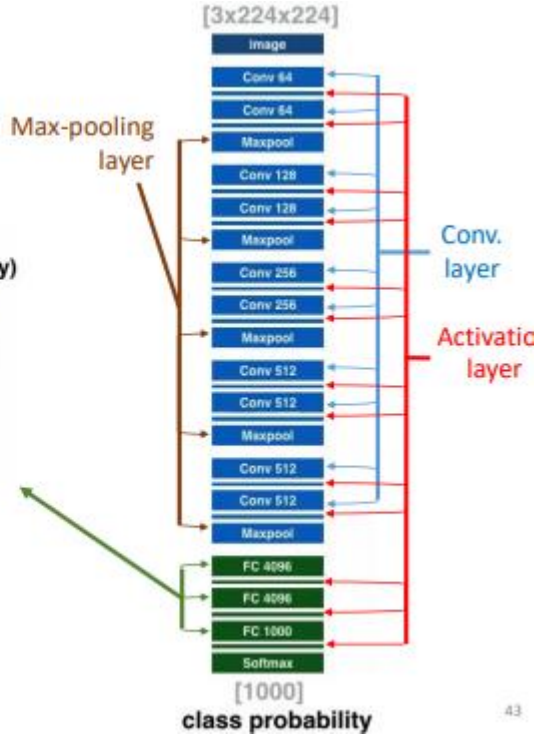
- **kernel_size** – the size of the window to take a max over
- **stride** – the stride of the window. Default value is `kernel_size`
- **padding** – implicit zero padding to be added on both sides
- **dilation** – a parameter that controls the stride of elements in the window
- **return_indices** – if `True`, will return the max indices along with the outputs. Useful for `torch.nn.MaxUnpool2d` later
- **ceil_mode** – when True, will use *ceil* instead of *floor* to compute the output shape

# Pytorch – Linear




**Q1: How many parameters in this layer?**

**Q2: Can we use different sizes of input images for this network? Why?**



# Pytorch – Linear

## LINEAR

CLASS `torch.nn.Linear(`*in_features*, *out_features*, *bias=True*, *device=None*, *dtype=None*`)` [SOURCE]

Parameters

- **in_features** – size of each input sample
- **out_features** – size of each output sample
- **bias** – If set to `False`, the layer will not learn an additive bias. Default: `True`

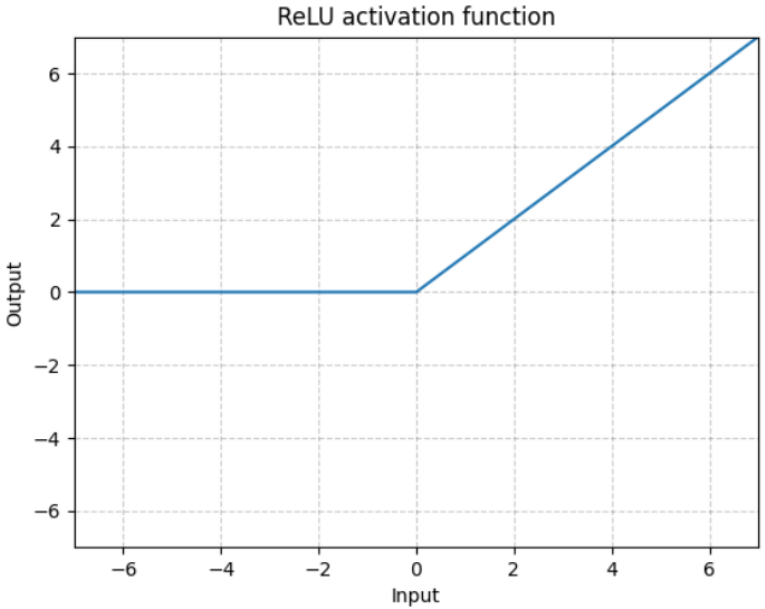
# Pytorch – relu

## RELU

CLASS `torch.nn.ReLU(inplace=False)` [SOURCE]

Parameters

**inplace** – can optionally do the operation in-place. Default: `False`



# Pytorch – CrossEntropyLoss

## Entropy, Cross entropy, KL divergence

- **Entropy**

$H_P = E_{x\sim P}[\log_2 1/p(x)]$

$$= \sum_{i=1}^{n} p_i \log \frac{1}{p_i}$$

| | $X_1$ | $X_2$ | $X_3$ | $X_4$ |
|---|---|---|---|---|
| Probability **p** | 0.5 | 0.25 | 0.125 | 0.125 |
| **$I = \log_2(1/p)$** | 1 | 2 | 3 | 3 |
| Entropy $H_P$ | 0.5x1 + 0.25x2 + 0.125x3 + 0.125x3 = **1.75** | | | |

- Expected Entropies across all choices

- Skewed Probability Distribution: Low entropy
- Balanced Probability Distribution: High entropy

# Pytorch – CrossEntropyLoss

## CROSSENTROPYLOSS

CLASS `torch.nn.CrossEntropyLoss(weight=None, size_average=None, ignore_index=-100, reduce=None, reduction='mean', label_smoothing=0.0)` [SOURCE]

```
# Example of target with class indices
loss = nn.CrossEntropyLoss()
input = torch.randn(3, 5, requires_grad=True)
target = torch.empty(3, dtype=torch.long).random_(5)
output = loss(input, target)
output.backward()
```

# Pandas

Tutorial: https://pandas.pydata.org/docs/user_guide/10min.html

pandas

About us | Getting started | Documentation | Community | Contribute

## pandas

**pandas** is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

Install pandas now!

### Getting started

- Install pandas
- Getting started

### Documentation

- User guide
- API reference
- Contributing to pandas
- Release notes

### Community

- About pandas
- Ask a question
- Ecosystem

With the support of:

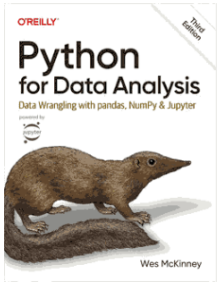
NUMFOCUS OPEN CODE = BETTER SCIENCE · TWO SIGMA · VOLTRON DATA · d-fine · Quansight Labs · NVIDIA

### Latest version: 1.4.4

- What's new in 1.4.4
- Release date: Aug 31, 2022
- Documentation (web)
- Download source code

### Follow us

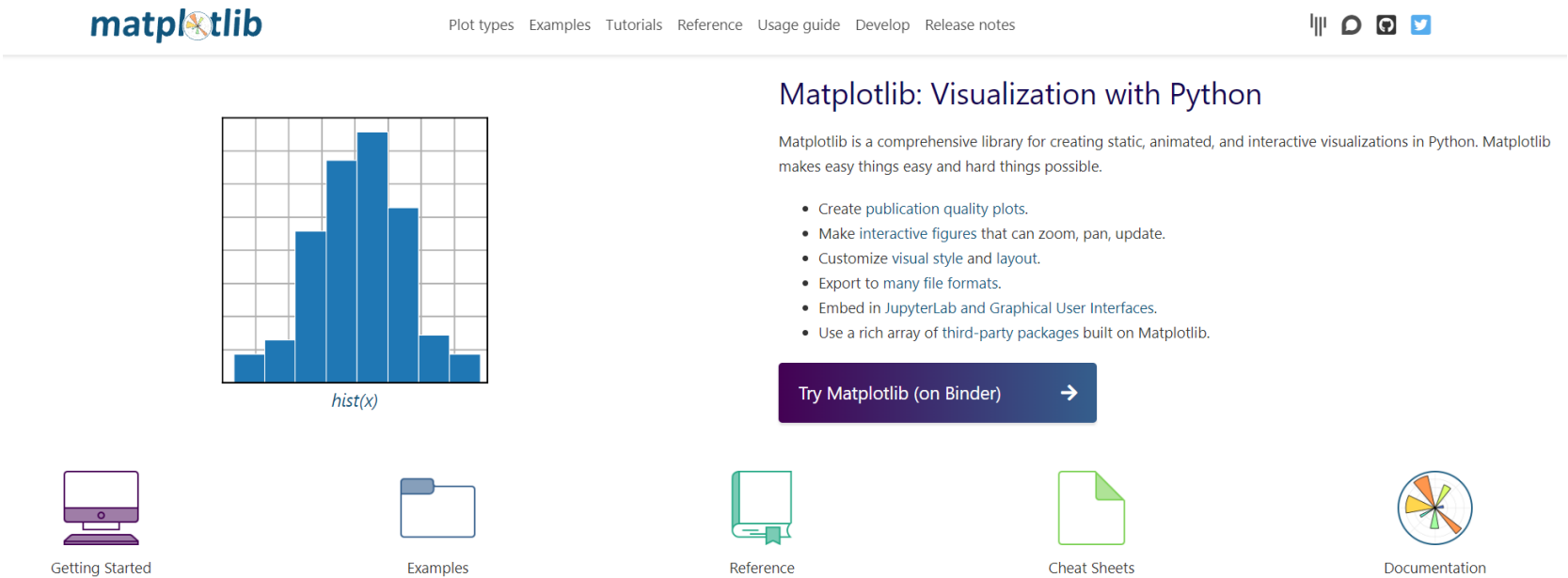
### Get the book




### Previous versions

# Matplotlib

Tutorial: https://matplotlib.org/stable/tutorials/index.html



# Reference

- Pytorch tutorial1: https://www.youtube.com/watch?v=85uJ9hSaXig
- Pytorch tutorial2: https://www.youtube.com/watch?v=VbqNn20FoHM
- Official tutorial: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html