Introduction to AI （CS103） – 08

AI Platform Introduction

Jimmy Liu 刘江

2023-11-10

Lecture 8

1. Reviews & Homework
2. Python
3. Machine Learning Platform
4. Deep Learning Platform

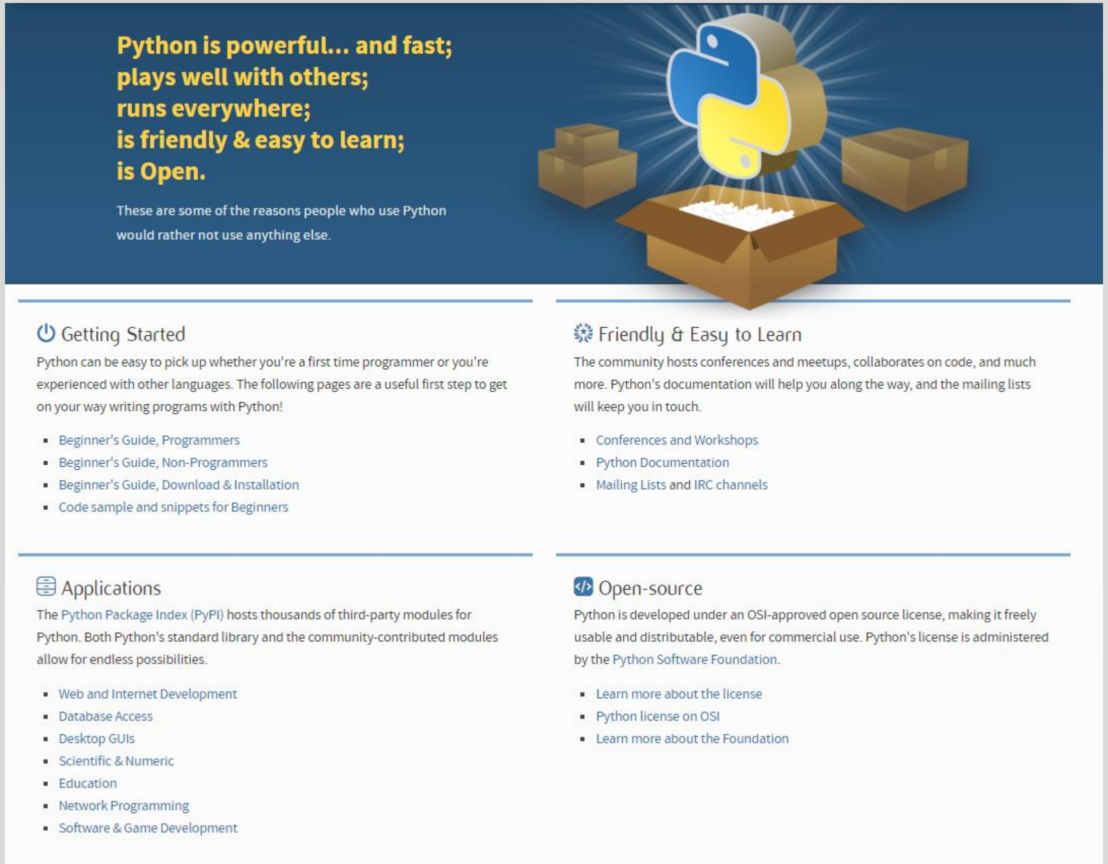
Reviews

Homework

Lecture 8

1. Reviews & Homework
2. Python
3. Machine Learning Platform
4. Deep Learning Platform

Python



In recent years, Python has rapidly emerged due to its simplicity, efficiency, ease of learning, rich third-party libraries, and wide application.

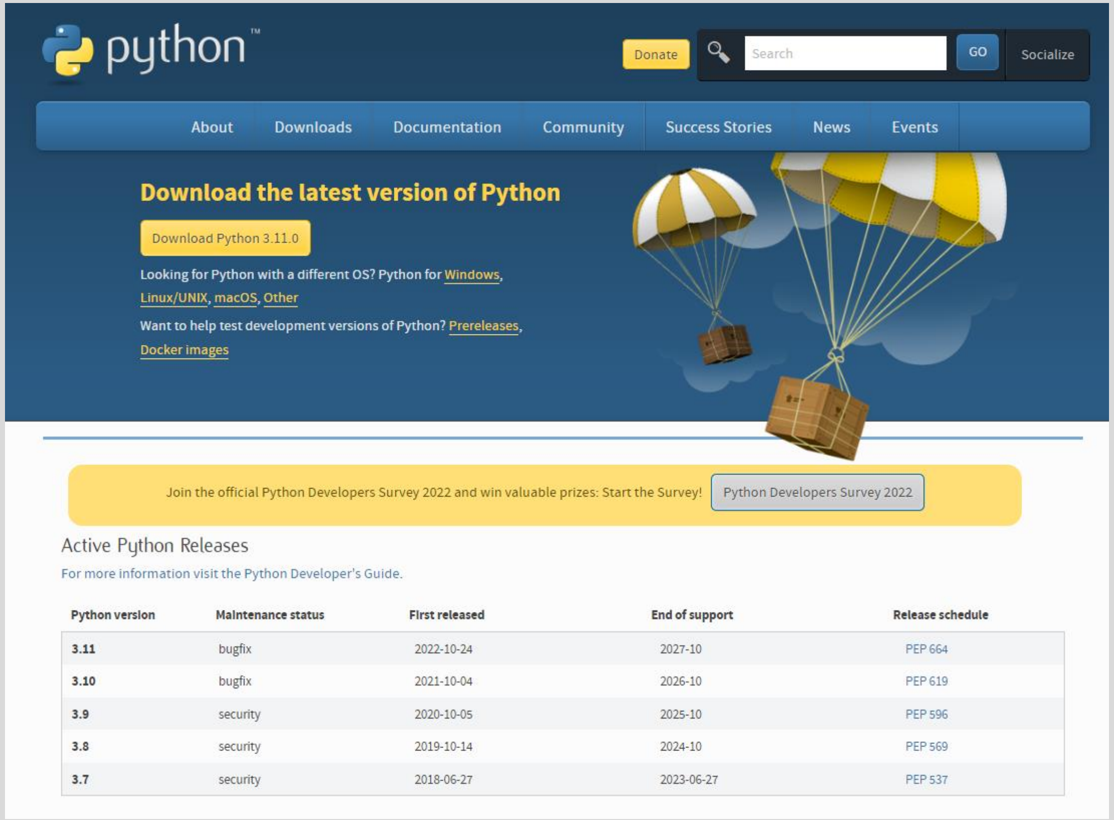
Python

According to the TIOBE Index website, as of September 2023, Python continues to top the list with a popularity of 14.16%.

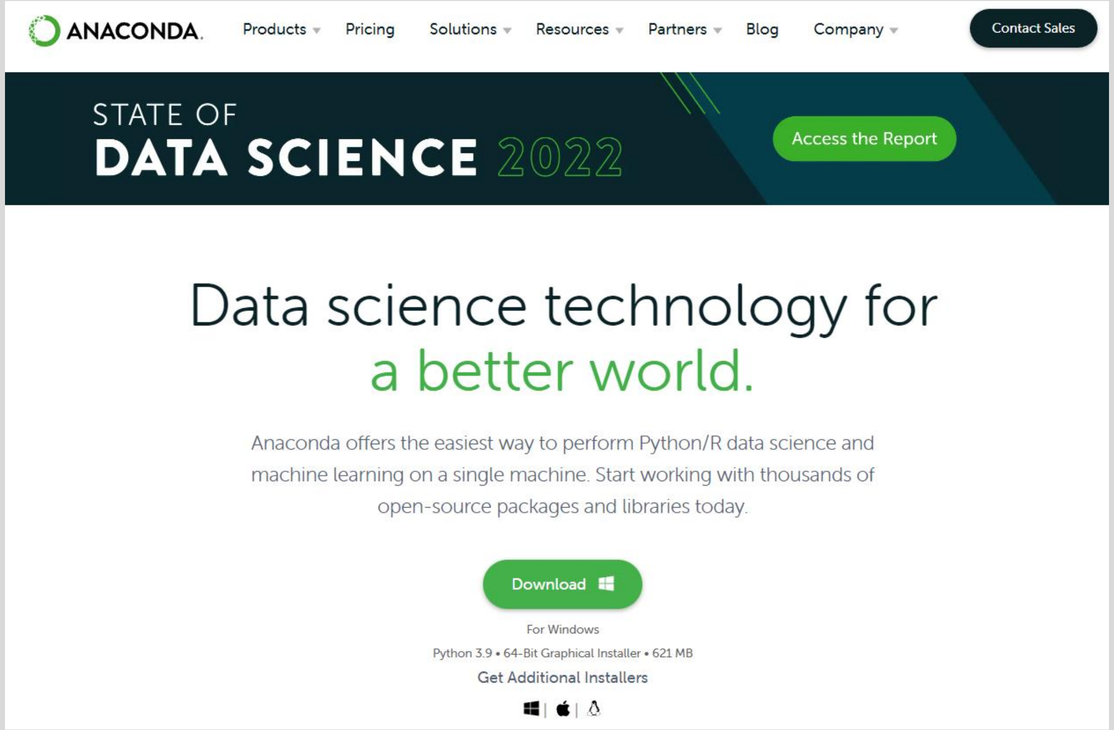
Sep 2023	Sep 2022	Change	Programming Language	Ratings	Change
1	1		Python	14.16%	-1.58%
2	2		C	11.27%	-2.70%
3	4	^	C++	10.65%	+0.90%
4	3	⌄	Java	9.49%	-2.23%
5	**5**		**C#**	**7.31%**	**+2.42%**
6	7	^	JavaScript	3.30%	+0.48%
7	6	⌄	Visual Basic	2.22%	-2.18%
8	10	^	PHP	1.55%	-0.13%
9	8	⌄	Assembly language	1.53%	-0.96%
10	9	⌄	SQL	1.44%	-0.57%

TIOBE Index 2023年编程语言排行榜

Python installation



[Download Python]



[Download Anaconda]

Python installation (optional)

Download PyCharm

Download Jupyter

Basic Knowledge

Data Types

Name	Notation	Declaration e.g.
Integers	int	a = 10
Floating	float	b = 3.14
Complex	complex	c = 1 + 2j
String	str	d = 'Python'

Arithmetic Operators

Name	Notation	Examples
Addition	+	a + b
Subtraction	-	c - b
Multiplication	*	x*y
Division	/	x/z
Modulus	%	x%a
Exponent	**	a**x

Basic Knowledge

Basic Data Structures

Name	Nation	Declaration e.g.
Tuple	tuple	b = (1,2.5, 'data')
List	list	c = [1,2.5,'data']
Dictionary	dict	d = {'Name': 'Kobe', 'Country':'US'}
Set	set	e = set(['u','d','ud','d','du'])

- 元组（tuple）只有几种方法可以更改。
- 列表（list）比元组更灵活。
- 字典（dict）是一个键值对存储对象。
- 集合（set）是对象中唯一的无序集合对象。

```python
l = [1, 2.1, 'asd']

l.append([2, 3])  # add an element
print(l)
l.extend([5, 6, 7])
print(l)  # extend the content from a new list
l.insert(1, 'start')
print(l)  # insert an element before index 1
l.remove('asd')
print(l)  # remove an element which is 'asd'
```

```
[1, 2.1, 'asd', [2, 3]]
[1, 2.1, 'asd', [2, 3], 5, 6, 7]
[1, 'start', 2.1, 'asd', [2, 3], 5, 6, 7]
[1, 'start', 2.1, [2, 3], 5, 6, 7]
```

Basic Knowledge

Conditions

```python
classes = 5
if classes == 3 or classes == 4:
    print('worker')
elif classes == 2:
    print('leader')
elif classes == 1:
    print('boss')
elif classes < 0:
    print('error')
else:
    print('classes N')
```

Loops

for...in ... : statement A
是循环中最常用的语句，通常与range（start,end,step）一起使用,start为起始值，end为结束值，step为步长。 例如，range(0,8,1) 给出[0, 1, 2, 3, 4, 5, 6, 7]

2/

while ...: statement A
将会执行A语句，直到满足while的条件。

```python
# for和range的例子 example  of  for  and  range
# 初始值默认值为default start of range is 0
# 步长默认值为default step of range is 1
for i in range(2, 10, 3):
    print(i)
    l= i**2
    print(l)

# white to sum   up 1 to 100
a = 0
sumup = 0
while  a < 100 :
    a + 1
    sumup += a
    print ( sumup)
```

Basic Knowledge

Function Declaration

- 方法定义如下(Functions are defined as)

```
def   TheNameOfFunction(paral, para2):
      ...
      return Outcome
```

- 函数（方法）返回的输出结果会在函数被调用的地方出现。

```
def  MaxOfTwo (x1, x2):
     if  x1 >= x2:
         return x1
     else:
         return x2

a = 1
b = 2
c = MaxOfTwo(a, b)
print(c)
```

Learning Materials

https://www.python.org/
https://www.runoob.com/python/python-tutorial.html
https://github.com/MurphyWan/Python-first-Practice

Other library recommended to learn：

- *NumPy*
- *Pandas*
- *Scikit-Learn* (Machine Learning)
- *Scipy*
- Deep Learning Library
  - *TensorFlow*
  - *PyTorch*

Q1: Any question?

Lecture 8

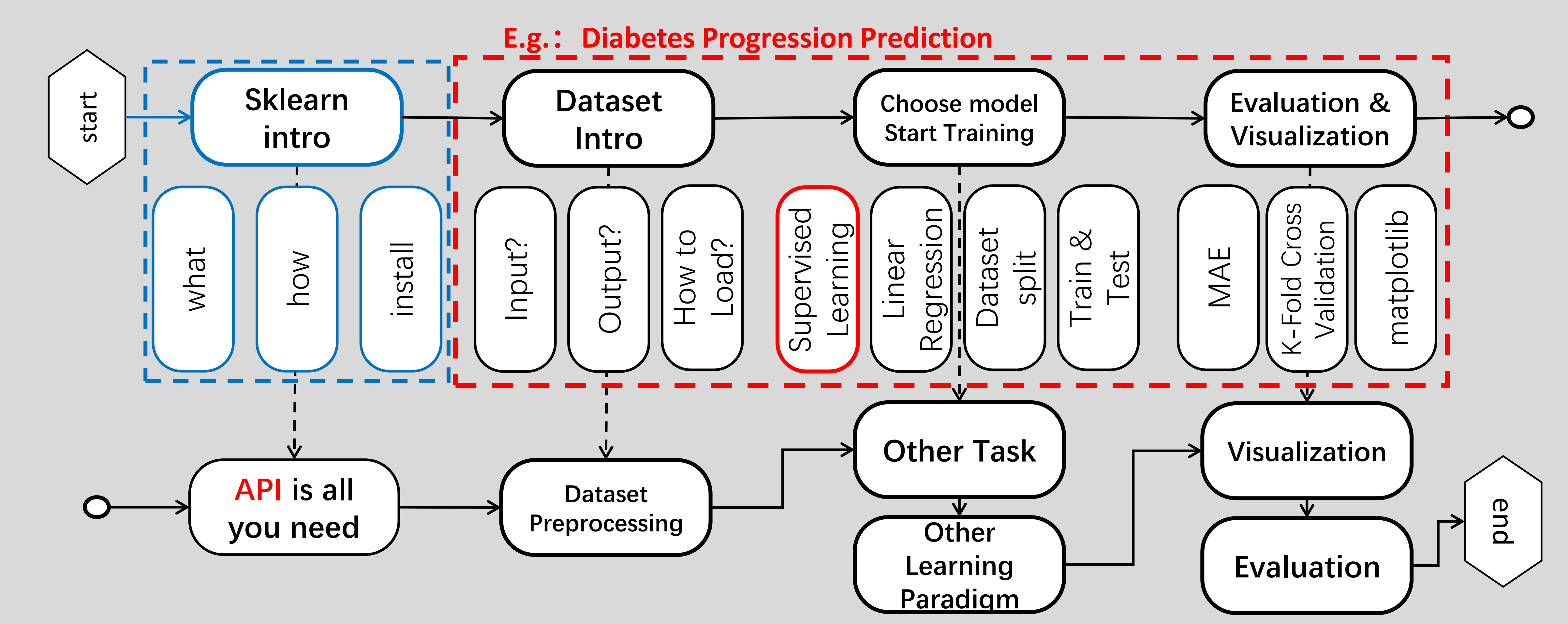
1 Reviews & Homework

2 Python

3 Machine Learning Platform

4 Deep Learning Platform

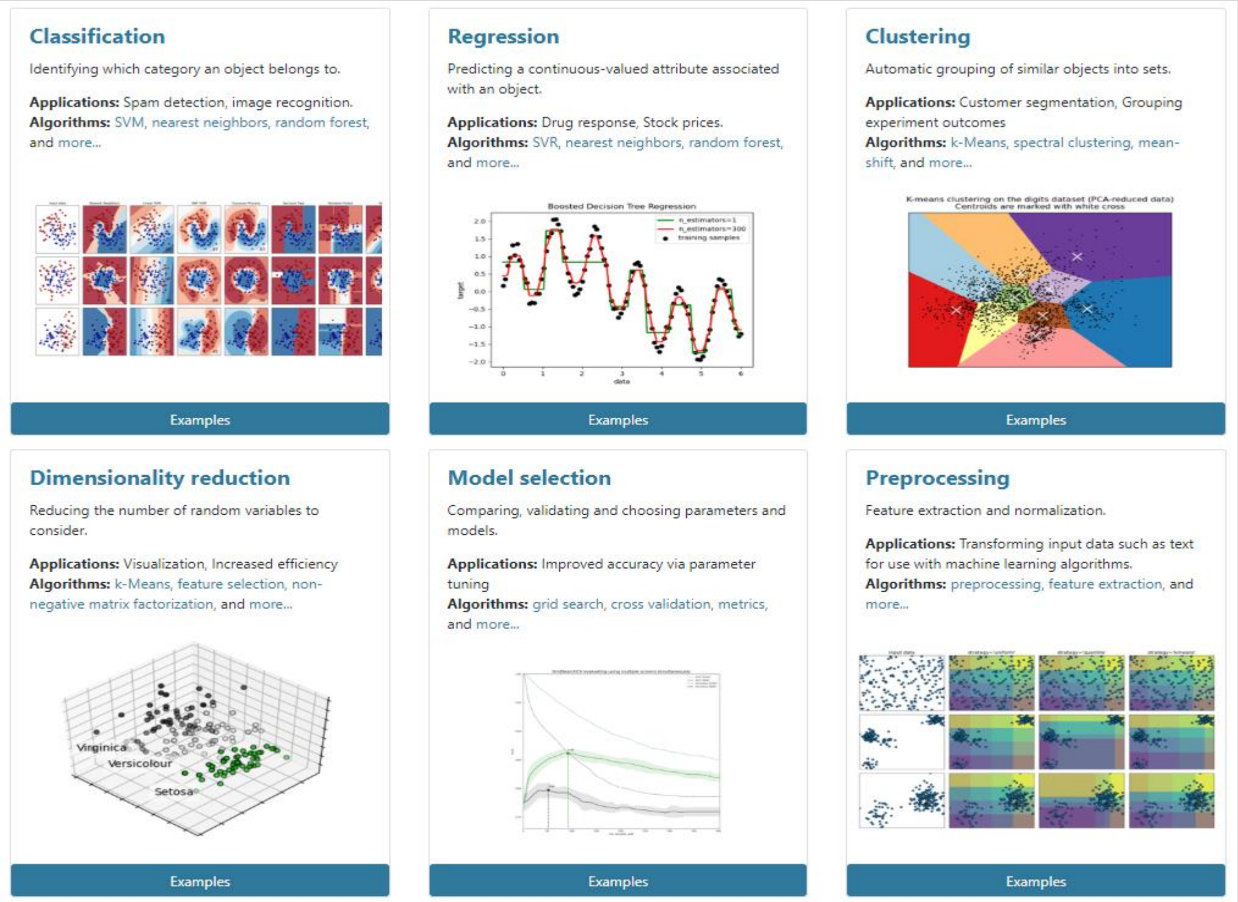
Machine Learning Platform



Scikit-Learn

Scikit-Learn is currently one of the most popular machine learning libraries. It efficiently implements various commonly used machine learning algorithms, with clean code, unified style, and rich and practical online documentation. This makes it one of the most popular libraries in the field of machine learning.

Scikit-Learn



The code functions of Scikit-Learn can be roughly divided into six parts: **classification, regression, clustering, dimensionality reduction, model selection, and preprocessing**, as shown in the left figure.

Scikit-Learn installation

Installing the latest release

Operating System Windows macOS Linux

Packager pip conda

Use pip virtualenv

Install the 64bit version of Python 3, for instance from https://www.python.org.
Then run:

```
$ pip install -U scikit-learn
```

In order to check your installation you can use

```
$ python -m pip show scikit-learn # to see which version and where scikit-learn is installed
$ python -m pip freeze # to see all packages installed in the active virtualenv
$ python -c "import sklearn; sklearn.show_versions()"
```

There are many ways to install scikit-learn：

- Official Release： Installing scikit-learn — scikit-learn 1.3.0 documentation
- Special versions： Installing scikit-learn — scikit-learn 1.3.0 documentation
- Build from source： Installing the development version of scikit-learn — scikit-learn 1.3.0 documentation

My favorite: Run the following command
> pip install scikit-learn

Scikit-Learn Dependencies

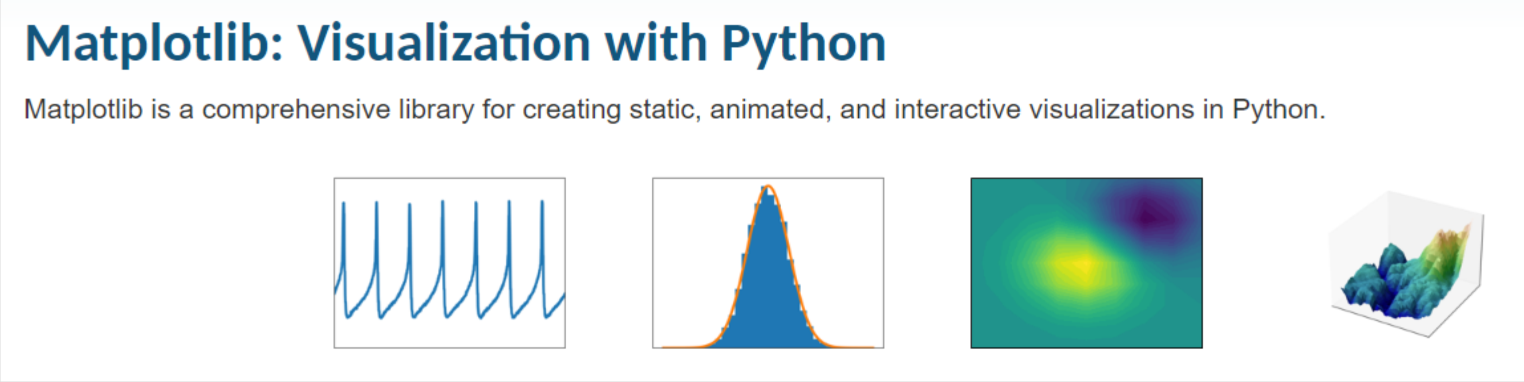
Dependencies

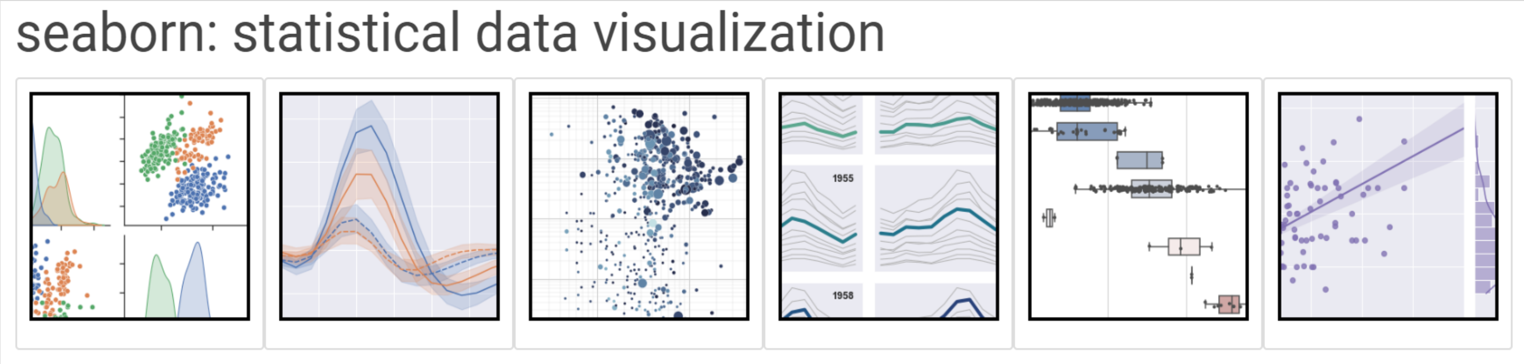
scikit-learn requires:

- Python
- NumPy
- SciPy
- joblib
- threadpoolctl
- **Matplotlib**
- **Seaborn**
- Jupyter
- **Statsmodels**

https://matplotlib.org/

http://seaborn.pydata.org/index.html





API is all you need

User Guide

1. Supervised learning
2. Unsupervised learning
3. Model selection and evaluation
4. Inspection
5. Visualizations
6. Dataset transformations
7. Dataset loading utilities
8. Computing with scikit-learn

https://scikit-learn.org/stable/user_guide.html

3. Model selection and evaluation

3.1. Cross-validation: evaluating estimator performance
- 3.1.1. Computing cross-validated metrics
- 3.1.2. Cross validation iterators
- 3.1.3. A note on shuffling
- 3.1.4. Cross validation and model selection

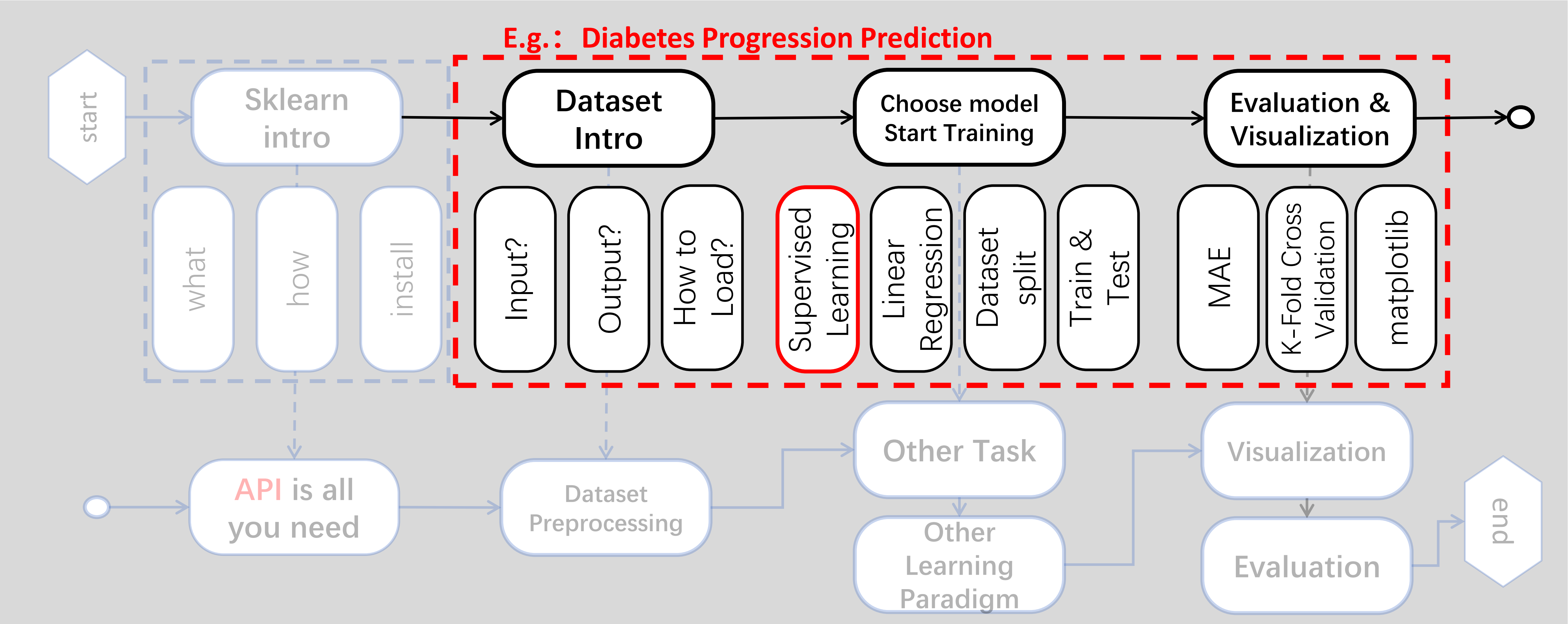
3.2. Tuning the hyper-parameters of an estimator
- 3.2.1. Exhaustive Grid Search
- 3.2.2. Randomized Parameter Optimization
- 3.2.3. Tips for parameter search
- 3.2.4. Alternatives to brute force parameter search

3.3. Metrics and scoring: quantifying the quality of predictions
- 3.3.1. The `scoring` parameter: defining model evaluation rules
- 3.3.2. Classification metrics
- 3.3.3. Multilabel ranking metrics
- 3.3.4. Regression metrics
- 3.3.5. Clustering metrics
- 3.3.6. Dummy estimators

3.4. Model persistence
- 3.4.1. Persistence example
- 3.4.2. Security & maintainability limitations

3.5. Validation curves: plotting scores to evaluate models
- 3.5.1. Validation curve
- 3.5.2. Learning curve

Machine Learning Platform



Diabetes Progression Prediction: An Example

- Let's start from a simple tutorial dataset provided by sklearn.

- The task is to predict the diabetes progression by some information of the patients:
  - Age
  - Sex
  - BMI
  - BP (average blood pressure)
  - S1-S6 (six blood serum measurement)

Load the Dataset

- What we want (predict):
  - disease progression 1 year after (a value)

- In real problems, data are always needed to be collected and processed **by yourself** but not just simply given to you using a single command. This may be a painful procedure. However, data is the **most important** stuff.

```python
import numpy as np
from sklearn.datasets import load_diabetes

X, y = load_diabetes(return_X_y=True)
```

Show the Dataset

- You may feel strange with values of age and sex and so on. This is because the value has been normalized. We'll talk about this later.
- The whole dataset is too big to be shown, so we just show the first 5 samples.

```python
print("Data:")
print(X[:5])
print("Target:")
print(y[:5])
print("Total samples:", len(X))
print("Number of features for each sample:", len(X[0]))
print("Shape of the input:", X.shape)
print("Shape of the target:", y.shape)
```

```
Data:
[[ 0.03807591  0.05068012  0.06169621  0.02187235 -0.0442235  -0.03482076
  -0.04340085 -0.00259226  0.01990842 -0.01764613]
 [-0.00188202 -0.04464164 -0.05147406 -0.02632783 -0.00844872 -0.01916334
   0.07441156 -0.03949338 -0.06832974 -0.09220405]
 [ 0.08529891  0.05068012  0.04445121 -0.00567061 -0.04559945 -0.03419447
  -0.03235593 -0.00259226  0.00286377 -0.02593034]
 [-0.08906294 -0.04464164 -0.01159501 -0.03665645  0.01219057  0.02499059
  -0.03603757  0.03430886  0.02269202 -0.00936191]
 [ 0.00538306 -0.04464164 -0.03638469  0.02187235  0.00393485  0.01559614
   0.00814208 -0.00259226 -0.03199144 -0.04664087]]
Target:
[151.  75. 141. 206. 135.]
Total samples: 442
Number of features for each sample: 10
Shape of the input: (442, 10)
Shape of the target: (442,)
```

Show the Dataset

- **Supervised learning**: Training data include both inputs and outputs
  - Data collection: Start with training data $\mathcal{D}$ from which experience is learned.
  - Data representation: Encode $\mathcal{D}$ to be the input to the learning system.
  - Modelling: Choose hypothesis space $\mathcal{H}$ --- a set of possible models for $\mathcal{D}$.
  - Learning: Find the best hypothesis $h \in \mathcal{H}$ according to some objective.
  - Model selection: Select the best model according to some criteria.
- Two categories:
  - Classification
  - Regression

Model Selection

- Our task is a regression task, so choose among regression models.
- Here we simply choose **linear regression** model as our model.

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error

model = LinearRegression()
print(model)
# This decides whether we use KFold cross validation or not later
use_k_fold = True
```

$$y = WX$$

$$W^{*} = (X^{T}X)^{-1}X^{T}y$$

Train-Test Split

- To evaluate the generalization of the model, we shouldn't use the whole dataset to train the model. We should train the model on a part of the dataset and test it on the remaining part.
- Use the function *train_test_split*, we can achieve this easily

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
print("Number of train samples:", len(X_train))
print("Number of test samples:", len(X_test))
```

```
Number of train samples: 353
Number of test samples: 89
```

Train-Predict-Test

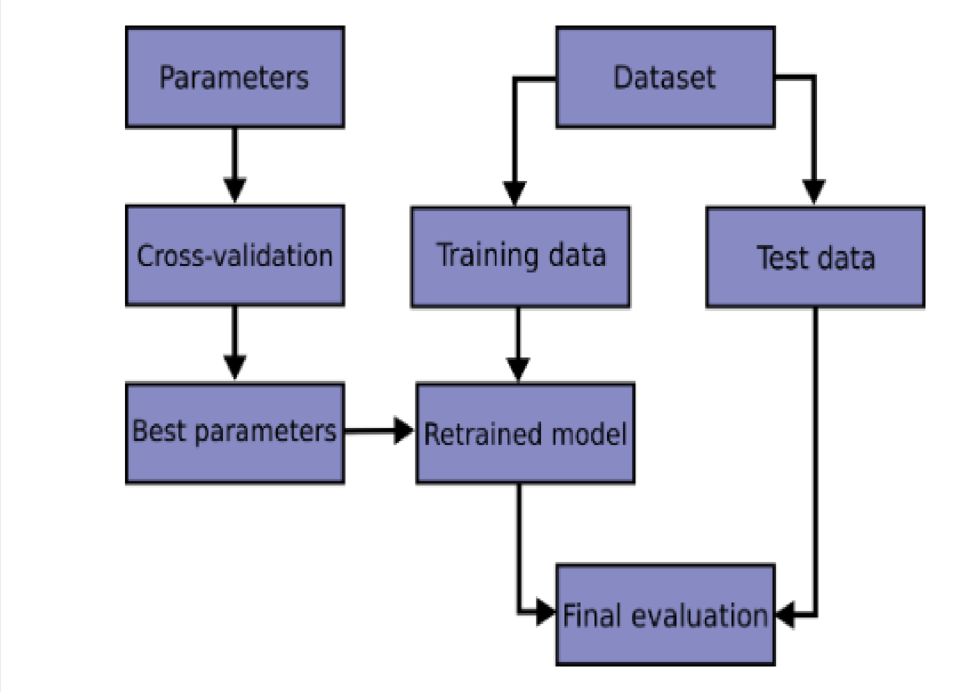
- Train: fit
- Predict: predict
- Test: mean_absolute_error
  - where $\tilde{y}\_i$ is the predicted value, y_i is the ground truth value
  - (we also has many other metric which will be decribed later)

$$MAE = \sum_i |\tilde{y}_i - y_i|$$

```python
if not use_k_fold:
    # train the model
    model.fit(X_train, y_train)

    # predict and test
    y_pred = model.predict(X_test)

    # evaluate metric
    mae = mean_absolute_error(y_pred, y_test)
    print("Error: %.4f" % mae)
```

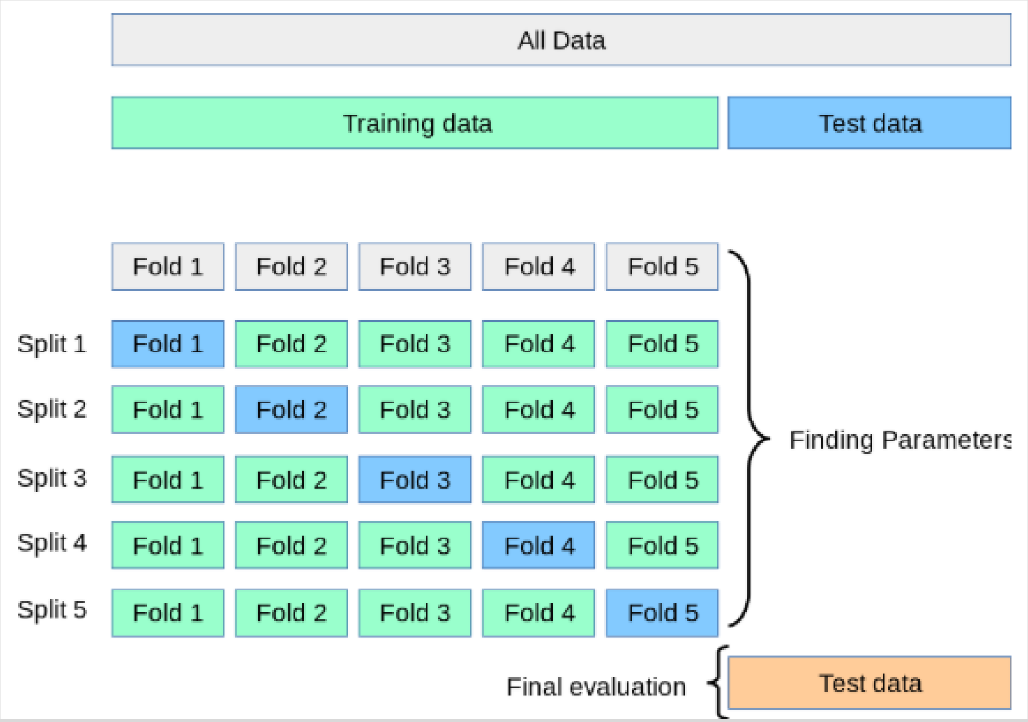
```
Error: 46.1742
```

Cross Validation



StatQuest,

K-Fold Cross Validation



https://scikit-learn.org/stable/modules/cross_validation.html#computing-cross-validated-metrics

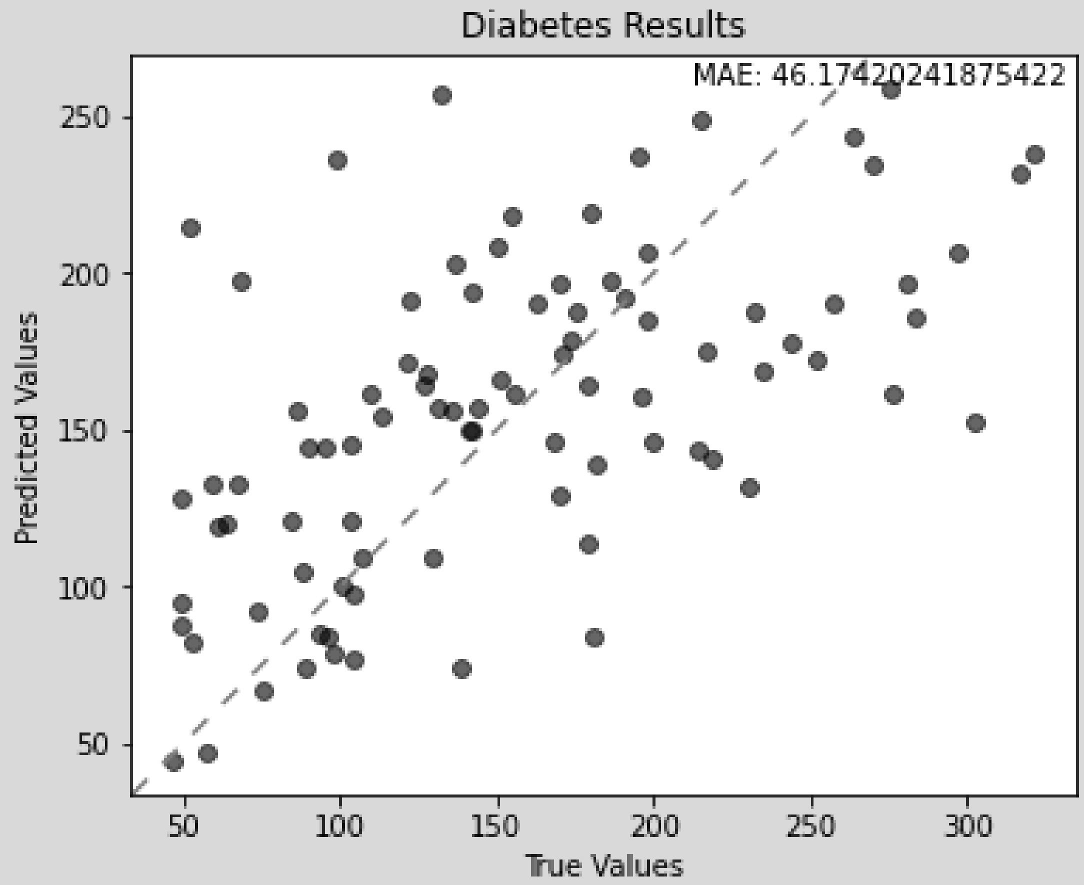
```python
if use_k_fold:
    # enable K-Fold
    from sklearn.model_selection import KFold

    kf = KFold(n_splits=5, shuffle=True)
    y_pred = np.zeros(len(X_test))
    score = 0
    for k, (train, test) in enumerate(kf.split(X_train, y_train)):
        model.fit(X_train[train], y_train[train])
        print("[fold {0}] score: {1:.5f}".format(
            k,
            model.score(X_train[test], y_train[test]),
        ))
        score += model.score(X_train[test], y_train[test])
        y_pred += model.predict(X_test)
    y_pred /= kf.get_n_splits()
    score /= kf.get_n_splits()
    mae = mean_absolute_error(y_pred, y_test)
    print("Error: %.4f, Score: %.4f" % (mae, score))
```

```
[fold 0] score: 0.44948
[fold 1] score: 0.55197
[fold 2] score: 0.54490
[fold 3] score: 0.57494
[fold 4] score: 0.47988
Error: 46.1702, Score: 0.5202
```

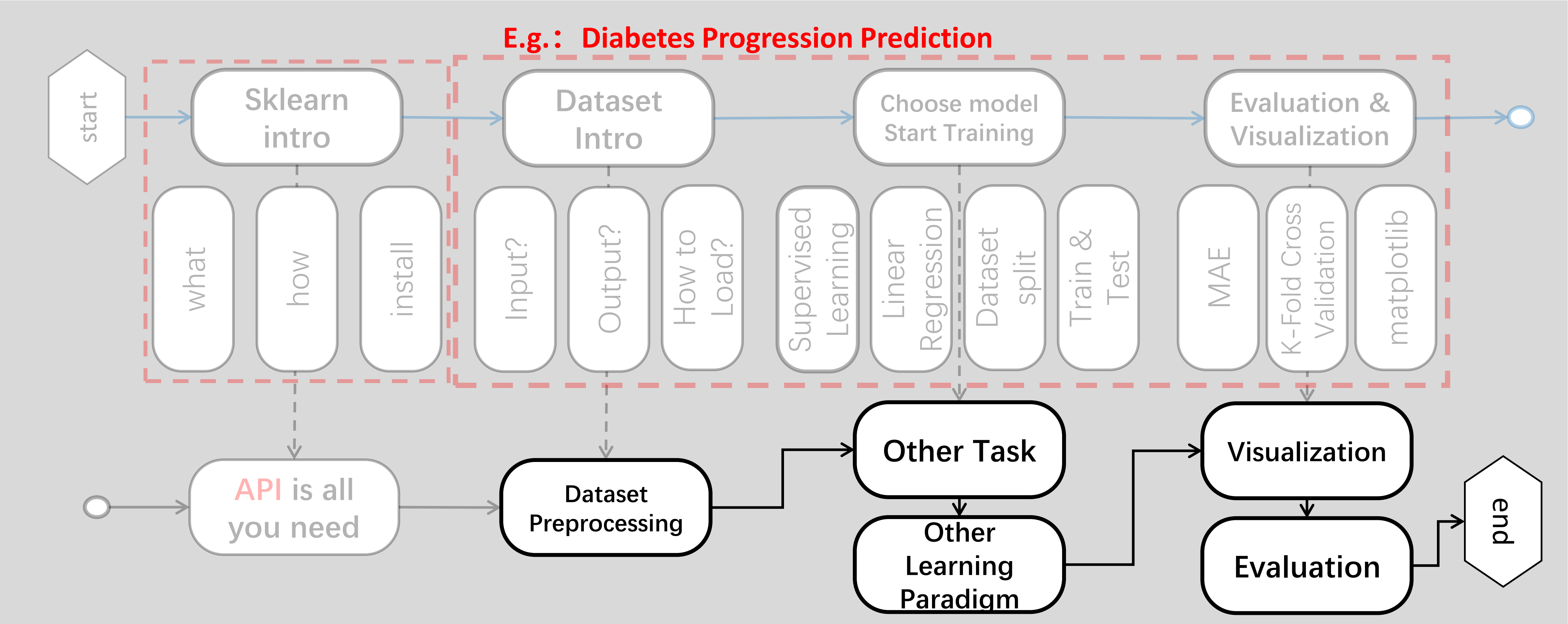
Visualization

- A simple value is not intuitive!
- We have to visualize the result!
- *matplotlib*

```
import matplotlib.pyplot as plt
%matplotlib inline

fig, ax = plt.subplots(1, 1, figsize=(6, 6))
plt.title("Diabetes Results")
plt.xlabel("True Values")
plt.ylabel("Predicted Values")
plt.axline((200, 200), slope=1, color='grey', linestyle=(0, (5, 5)))
plt.scatter(y_test, y_pred, color='black', alpha=.6)

# generate the label on the top right corner
xmin, xmax = ax.get_xlim()
ymin, ymax = ax.get_ylim()
text = 'MAE: ' + str(mae)
plt.text(xmax - 0.01 * xmax, ymax - 0.01 * ymax, text, verticalalignment='top',
         horizontalalignment='right', fontsize=10)
plt.axis('scaled')
plt.show()
```



Machine Learning Platform



Preprocessing Data

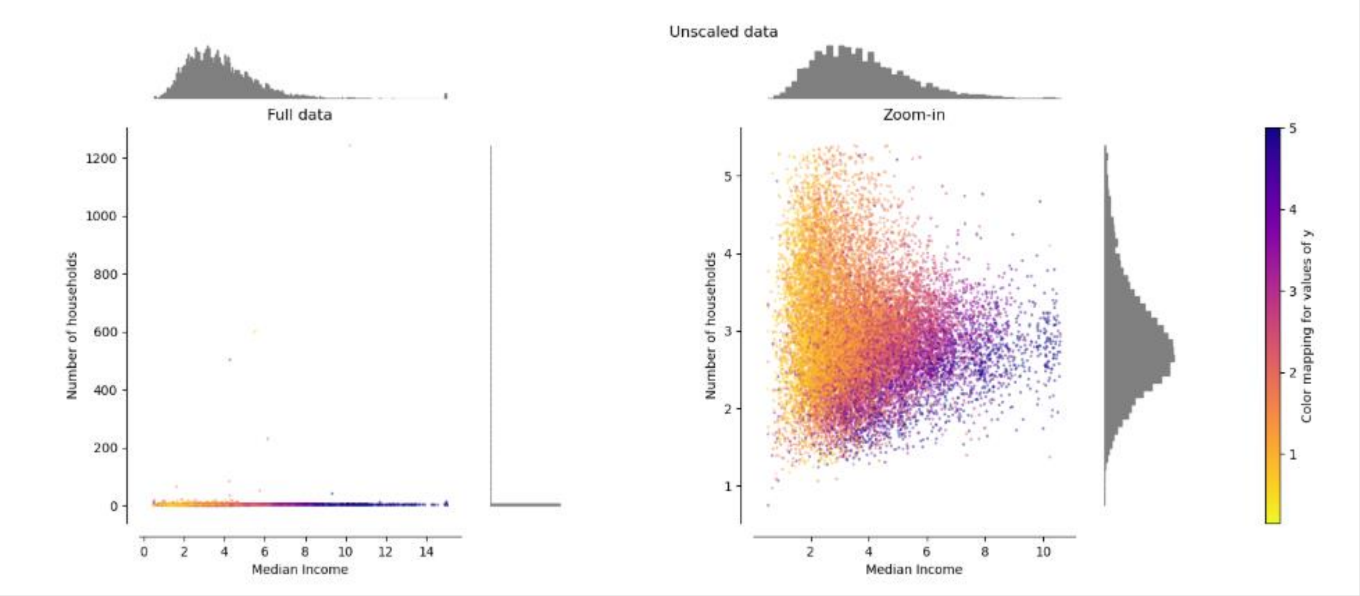
- The sklearn.preprocessing package provides several common utility functions and transformer classes to change raw features into representations more suitable for downstream tasks.

- In general, learning algorithms benefit from the standardization of datasets. If there are some outliers in the collection, scaling or transformation is generally required.

1. [Standardization, or mean removal and variance scaling]
2. [Non-linear transformation]
3. [Normalization]
4. [Encoding categorical features]
5. [Discretization]
6. [Imputation of missing values]
7. [Generating polynomial features]
8. [Custom transformers]

Decomposing Signals in Components

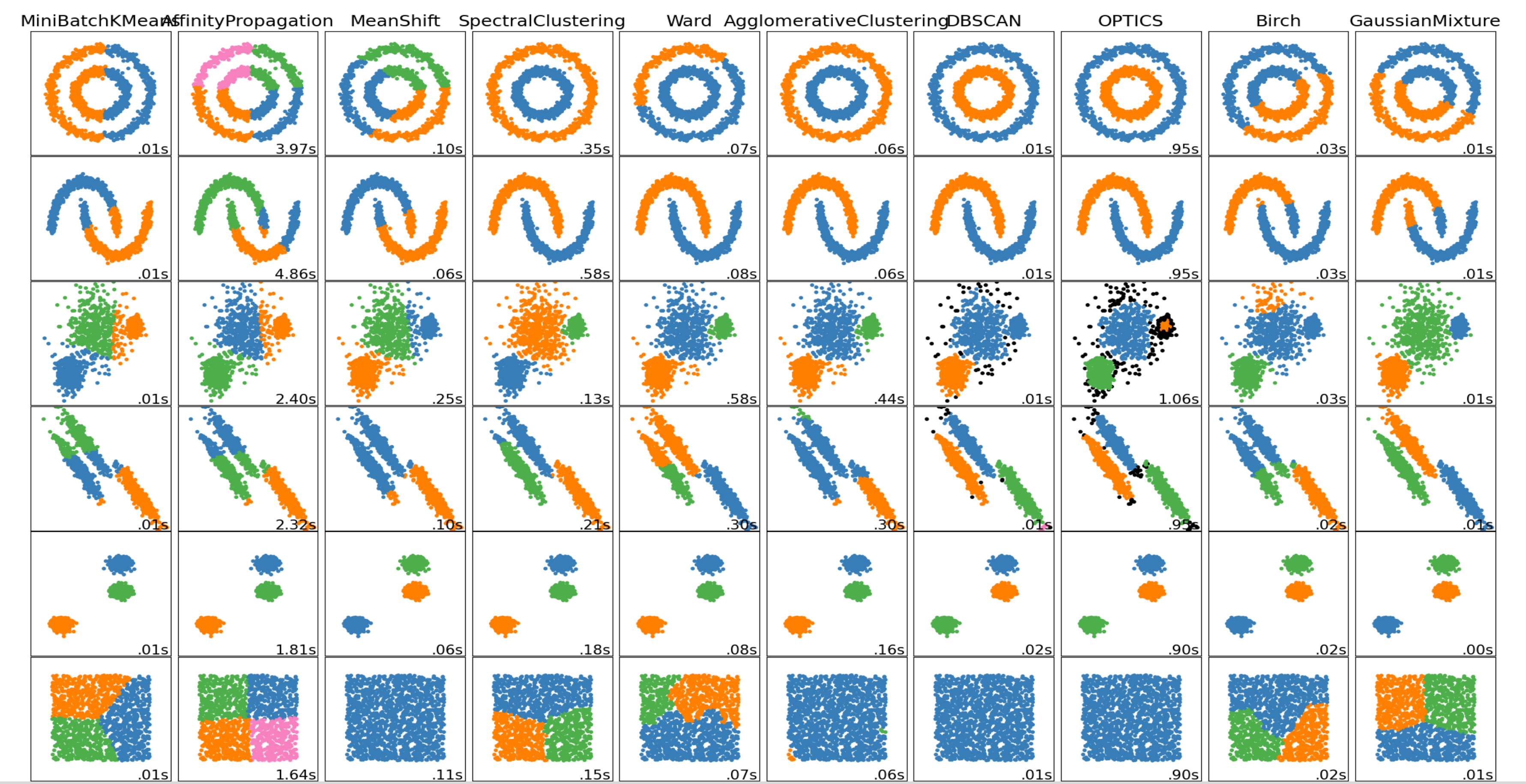
1. Principal component analysis (PCA)
2. Truncated singular value decomposition and latent semantic analysis
3. Dictionary Learning
4. Factor Analysis
5. Independent component analysis (ICA)
6. Non-negative matrix factorization (NMF or NNMF)
7. Latent Dirichlet Allocation (LDA)



Supervised Learning Algorithms

1. **Linear Models**
2. Linear and Quadratic Discriminant Analysis
3. Kernel ridge regression
4. **Support Vector Machines**
5. Stochastic Gradient Descent
6. **Nearest Neighbors**
7. Gaussian Processes
8. Cross decomposition
9. **Naive Bayes**
10. **Decision Trees**
11. **Ensemble methods**
12. Multiclass and multilabel algorithms
13. **Feature selection**
14. Semi-Supervised
15. Isotonic regression
16. Probability calibration
17. **Neural network models (supervised)**

Clustering


MiniBatchKMeans
AffinityPropagation
MeanShift
SpectralClustering
Ward
AgglomerativeClustering
DBSCAN
OPTICS
Birch
GaussianMixture
.01s
3.97s
.10s
.35s
.07s
.06s
.01s
.95s
.03s
.01s
.01s
4.86s
.06s
.58s
.08s
.06s
.01s
.95s
.03s
.01s
.01s
2.40s
.25s
.13s
.58s
.44s
.01s
1.06s
.03s
.01s
.01s
2.32s
.10s
.21s
.30s
.30s
.01s
.95s
.02s
.01s
.01s
1.81s
.06s
.18s
.08s
.16s
.02s
.90s
.02s
.00s
.01s
1.64s
.11s
.15s
.07s
.06s
.01s
.90s
.02s
.01s

Clustering

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points
OPTICS	minimum cluster membership	Very large `n_samples`, large `n_clusters`	Non-flat geometry, uneven cluster sizes, variable cluster density	Distances between points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large `n_clusters` and `n_samples`	Large dataset, outlier removal, data reduction.	Euclidean distance between points

Classification Metrics

Scoring	Function	Comment
Classification		
'accuracy'	`metrics.accuracy_score`	
'balanced_accuracy'	`metrics.balanced_accuracy_score`	
'average_precision'	`metrics.average_precision_score`	
'neg_brier_score'	`metrics.brier_score_loss`	
'f1'	`metrics.f1_score`	for binary targets
'f1_micro'	`metrics.f1_score`	micro-averaged
'f1_macro'	`metrics.f1_score`	macro-averaged
'f1_weighted'	`metrics.f1_score`	weighted average
'f1_samples'	`metrics.f1_score`	by multilabel sample
'neg_log_loss'	`metrics.log_loss`	requires `predict_proba` support
'precision' etc.	`metrics.precision_score`	suffixes apply as with 'f1'
'recall' etc.	`metrics.recall_score`	suffixes apply as with 'f1'
'jaccard' etc.	`metrics.jaccard_score`	suffixes apply as with 'f1'
'roc_auc'	`metrics.roc_auc_score`	
'roc_auc_ovr'	`metrics.roc_auc_score`	
'roc_auc_ovo'	`metrics.roc_auc_score`	
'roc_auc_ovr_weighted'	`metrics.roc_auc_score`	

Classification Metrics

Some of these are restricted to the binary classification case:

`precision_recall_curve`(y_true, probas_pred, *)	Compute precision-recall pairs for different probability thresholds
`roc_curve`(y_true, y_score, *[, pos_label, ...])	Compute Receiver operating characteristic (ROC)

Others also work in the multiclass case:

`balanced_accuracy_score`(y_true, y_pred, *[, ...])	Compute the balanced accuracy
`cohen_kappa_score`(y1, y2, *[, labels, ...])	Cohen's kappa: a statistic that measures inter-annotator agreement.
`confusion_matrix`(y_true, y_pred, *[, ...])	Compute confusion matrix to evaluate the accuracy of a classification.
`hinge_loss`(y_true, pred_decision, *[, ...])	Average hinge loss (non-regularized)
`matthews_corrcoef`(y_true, y_pred, *[, ...])	Compute the Matthews correlation coefficient (MCC)
`roc_auc_score`(y_true, y_score, *[, average, ...])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

Classification Metrics

Some also work in the multilabel case:

`accuracy_score`(y_true, y_pred, *[, ...])	Accuracy classification score.
`classification_report`(y_true, y_pred, *[, ...])	Build a text report showing the main classification metrics.
`f1_score`(y_true, y_pred, *[, labels, ...])	Compute the F1 score, also known as balanced F-score or F-measure
`fbeta_score`(y_true, y_pred, *, beta[, ...])	Compute the F-beta score
`hamming_loss`(y_true, y_pred, *[, sample_weight])	Compute the average Hamming loss.
`jaccard_score`(y_true, y_pred, *[, labels, ...])	Jaccard similarity coefficient score
`log_loss`(y_true, y_pred, *[, eps, ...])	Log loss, aka logistic loss or cross-entropy loss.
`multilabel_confusion_matrix`(y_true, y_pred, *)	Compute a confusion matrix for each class or sample
`precision_recall_fscore_support`(y_true, ...)	Compute precision, recall, F-measure and support for each class
`precision_score`(y_true, y_pred, *[, labels, ...])	Compute the precision
`recall_score`(y_true, y_pred, *[, labels, ...])	Compute the recall
`roc_auc_score`(y_true, y_score, *[, average, ...])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
`zero_one_loss`(y_true, y_pred, *[, ...])	Zero-one classification loss.

And some work with binary and multilabel (but not multiclass) problems:

`average_precision_score`(y_true, y_score, *)	Compute average precision (AP) from prediction scores

Confusion Matrix

		True condition				
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$	F_1 score $= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$		

Q2: Any question?

Lecture 8

1. Reviews & Homework
2. Python
3. Machine Learning Platform
4. Deep Learning Platform

History

- 2002--Torch
- 2011--Torch7

Merits: flexible、dynamic、user-friendly
Demerits: based on Lua

Caffe

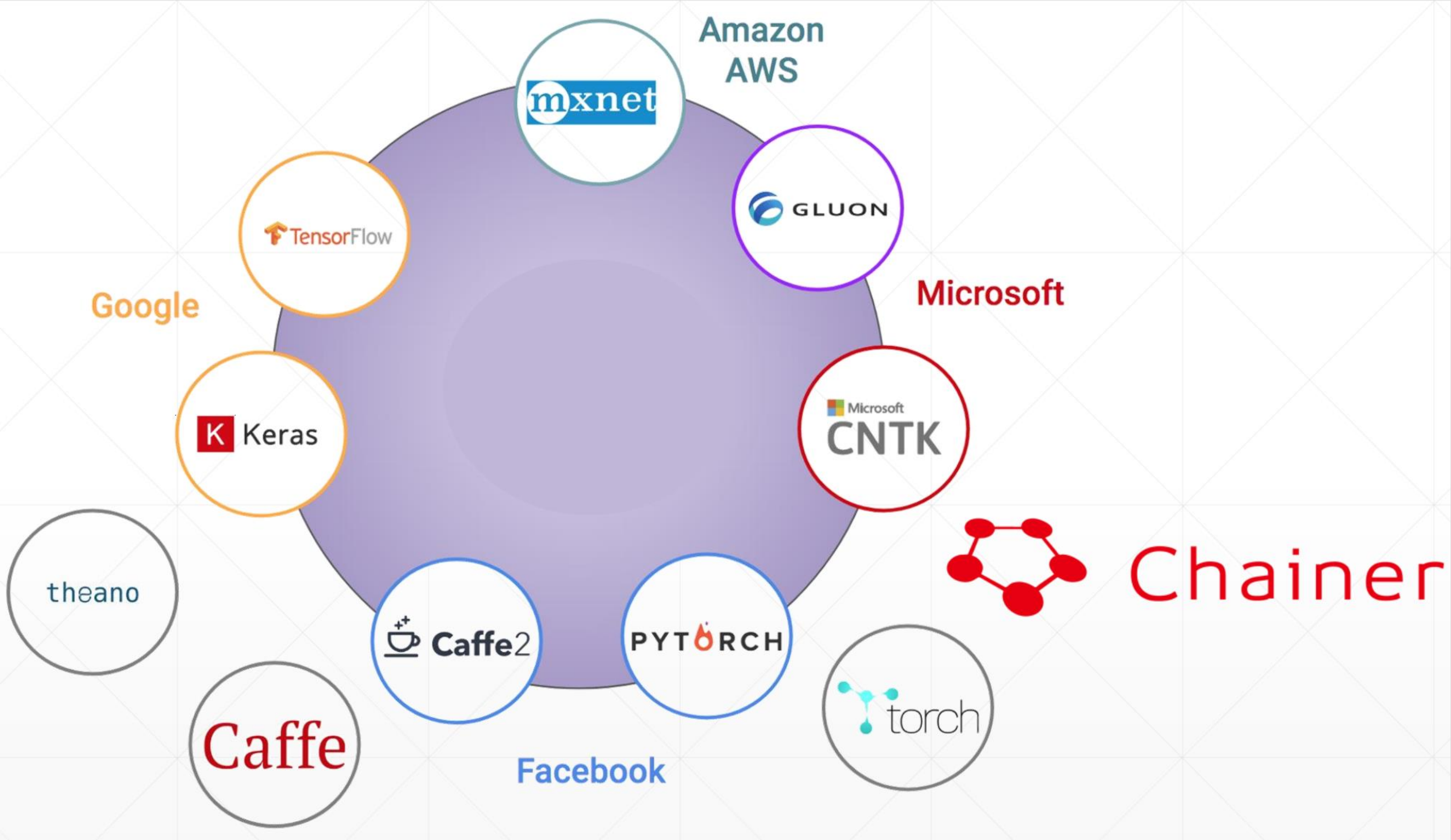
- 2013--Caffe
- 2017--Caffe2

Merits: fast(based on C++)
Demerits: not flexible

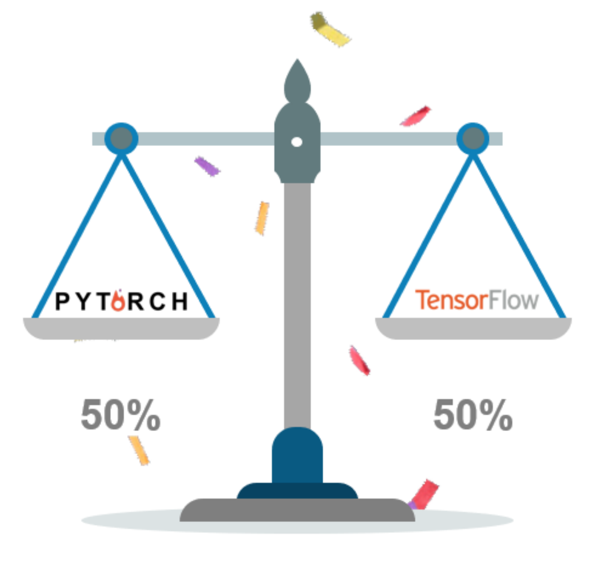
- 2017--PyTorch
- 2018--PyTorch v0.4

PyTorch

History



阿里： X-Deep Learning
小米： Mobile AI Compute Engine



History

	PyTorch	TensorFlow 1	TensorFlow 2
性能	★★★★	★★★★	★★★★
生态	★★★★	★★★★★	★★
工业界	★★★	★★★★★	★★
学术界	★★★★★	★★★	★★
上手难度	★★★★★	★★	★★★★
易用性	★★★★★	★	★★★★
兼容性	★★★★	★	★
发展前景	★★★★	0	★★★★

How to build a model?

- Install pytorch

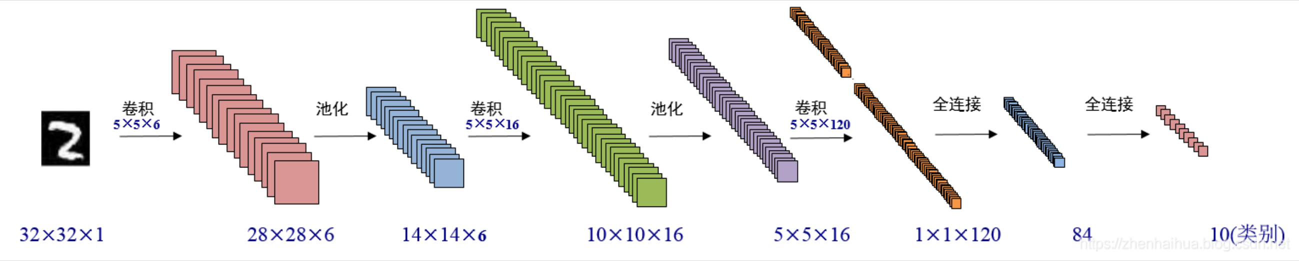
START LOCALLY

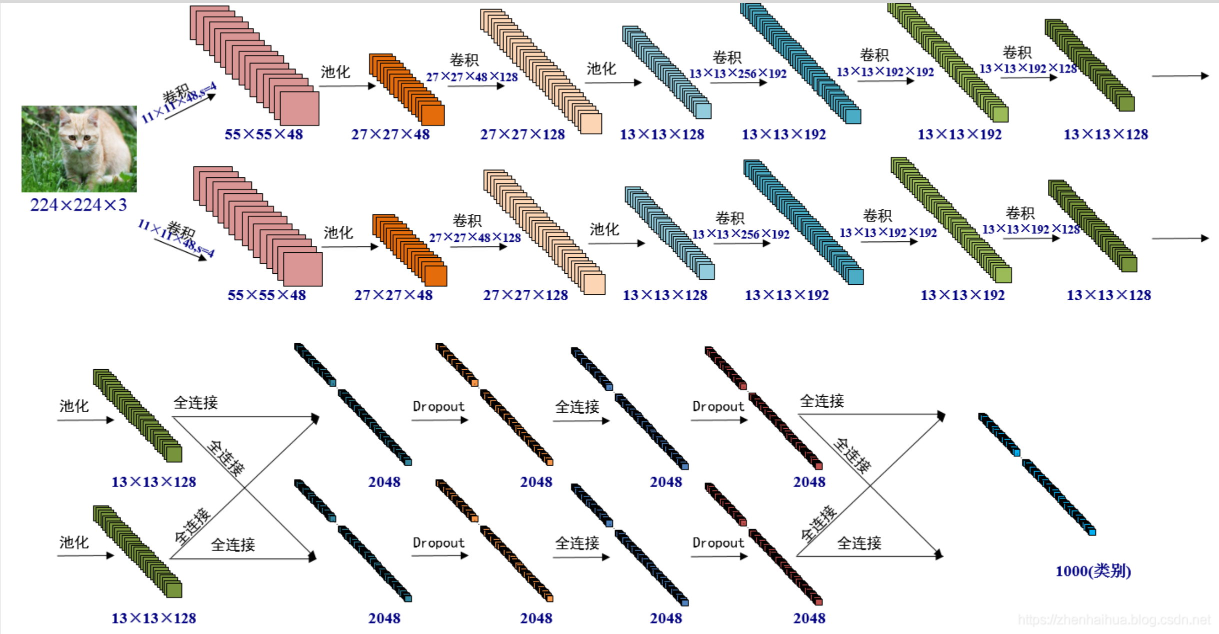
Select your preferences and run the install command. Stable represents the most currently tested and supported version of PyTorch. This should be suitable for many users. Preview is available if you want the latest, not fully tested and supported, builds that are generated nightly. Please ensure that you have **met the prerequisites below (e.g., numpy)**, depending on your package manager. Anaconda is our recommended package manager since it installs all dependencies. You can also install previous versions of PyTorch. Note that LibTorch is only available for C++.

PyTorch Build	Stable (1.13.0)	Preview (Nightly)	LTS (1.8.2)	
Your OS	Linux	Mac	Windows	
Package	Conda	Pip	LibTorch	Source
Language	Python	C++ / Java		
Compute Platform	CUDA 11.6	CUDA 11.7	~~ROCm 5.2~~	CPU
Run this Command:	`conda install pytorch torchvision torchaudio cpuonly -c pytorch`			

How to build a model?

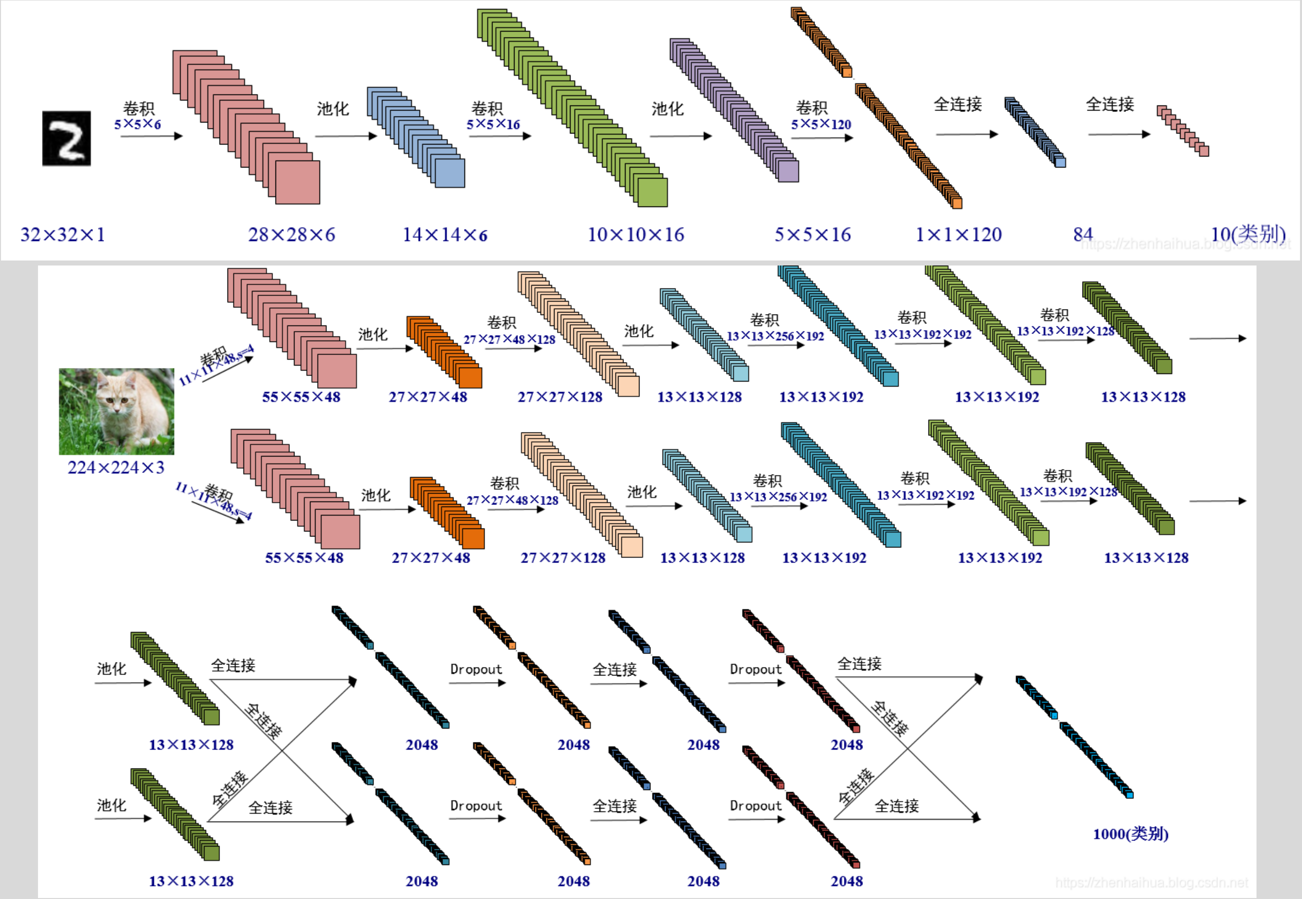
- Install pytorch
- Decide the model you want to realize





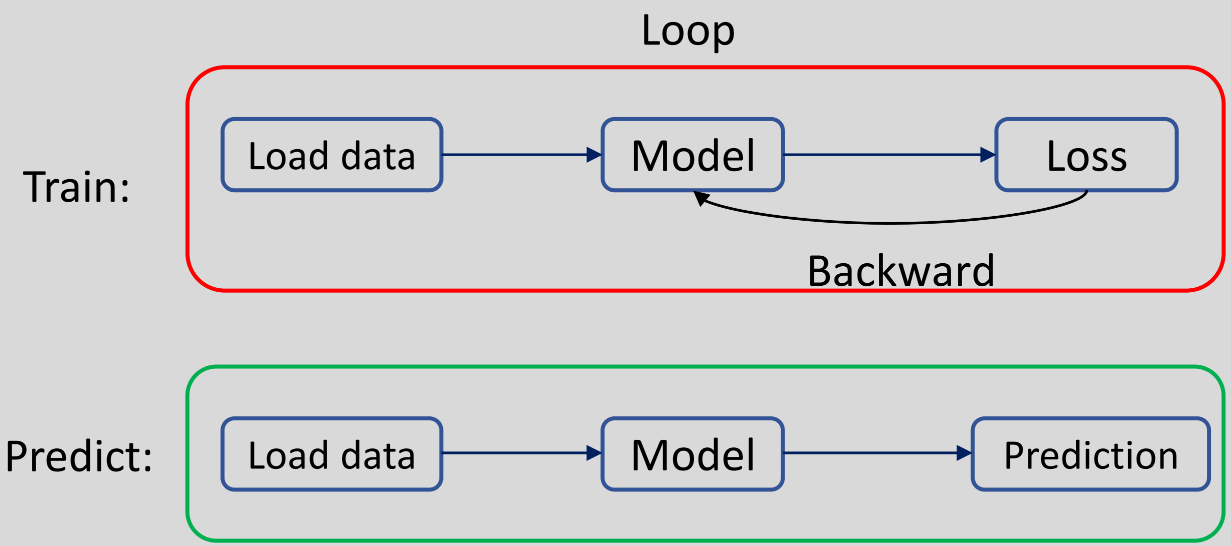
How to build a model?

- Install pytorch
- Decide the model you want to realize
- A model may have …
  - Convolutional Layers
  - Fully-connected Layers
  - Batch Normalization Layers
  - Pooling Layers
  - ……

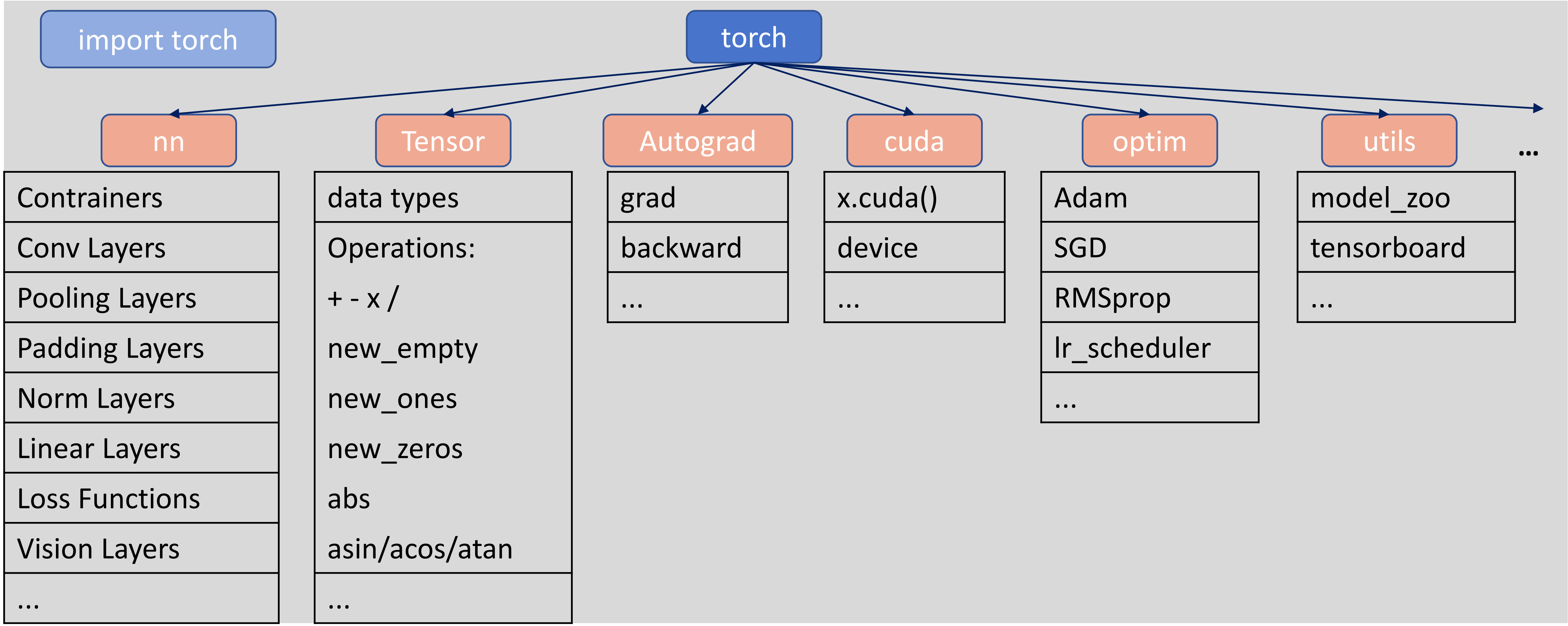


How to build a model?

- Install pytorch
- Decide the model you want to realize
- A model may have …
  - Convolutional Layers
  - Fully-connected Layers
  - Batch Normalization Layers
  - Pooling Layers
  - ……
- How to train a model?



Common API



import torch
torch
nn
Tensor
Autograd
cuda
optim
utils
...
Contrainers
Conv Layers
Pooling Layers
Padding Layers
Norm Layers
Linear Layers
Loss Functions
Vision Layers
...
data types
Operations:
+ - x /
new_empty
new_ones
new_zeros
abs
asin/acos/atan
...
grad
backward
...
x.cuda()
device
...
Adam
SGD
RMSprop
lr_scheduler
...
model_zoo
tensorboard
...

Common API

torch.nn

Containers

`Module`	Base class for all neural network modules.
`Sequential`	A sequential container.
`ModuleList`	Holds submodules in a list.
`ModuleDict`	Holds submodules in a dictionary.
`ParameterList`	Holds parameters in a list.
`ParameterDict`	Holds parameters in a dictionary.

Common API

torch.nn

Convolution Layers

`nn.Conv1d`	Applies a 1D convolution over an input signal composed of several input planes.
`nn.Conv2d`	Applies a 2D convolution over an input signal composed of several input planes.
`nn.Conv3d`	Applies a 3D convolution over an input signal composed of several input planes.
`nn.ConvTranspose1d`	Applies a 1D transposed convolution operator over an input image composed of several input planes.
`nn.ConvTranspose2d`	Applies a 2D transposed convolution operator over an input image composed of several input planes.
`nn.ConvTranspose3d`	Applies a 3D transposed convolution operator over an input image composed of several input planes.
`nn.Unfold`	Extracts sliding local blocks from a batched input tensor.
`nn.Fold`	Combines an array of sliding local blocks into a large containing tensor.

Common API

torch.nn

Pooling layers

`nn.MaxPool1d`	Applies a 1D max pooling over an input signal composed of several input planes.
`nn.MaxPool2d`	Applies a 2D max pooling over an input signal composed of several input planes.
`nn.MaxPool3d`	Applies a 3D max pooling over an input signal composed of several input planes.
`nn.MaxUnpool1d`	Computes a partial inverse of `MaxPool1d`.
`nn.MaxUnpool2d`	Computes a partial inverse of `MaxPool2d`.
`nn.MaxUnpool3d`	Computes a partial inverse of `MaxPool3d`.
`nn.AvgPool1d`	Applies a 1D average pooling over an input signal composed of several input planes.
`nn.AvgPool2d`	Applies a 2D average pooling over an input signal composed of several input planes.
`nn.AvgPool3d`	Applies a 3D average pooling over an input signal composed of several input planes.
`nn.FractionalMaxPool2d`	Applies a 2D fractional max pooling over an input signal composed of several input planes.
`nn.LPPool1d`	Applies a 1D power-average pooling over an input signal composed of several input planes.
`nn.LPPool2d`	Applies a 2D power-average pooling over an input signal composed of several input planes.
`nn.AdaptiveMaxPool1d`	Applies a 1D adaptive max pooling over an input signal composed of several input planes.
`nn.AdaptiveMaxPool2d`	Applies a 2D adaptive max pooling over an input signal composed of several input planes.
`nn.AdaptiveMaxPool3d`	Applies a 3D adaptive max pooling over an input signal composed of several input planes.
`nn.AdaptiveAvgPool1d`	Applies a 1D adaptive average pooling over an input signal composed of several input planes.
`nn.AdaptiveAvgPool2d`	Applies a 2D adaptive average pooling over an input signal composed of several input planes.
`nn.AdaptiveAvgPool3d`	Applies a 3D adaptive average pooling over an input signal composed of several input planes.

Common API

torch.nn

Padding Layers

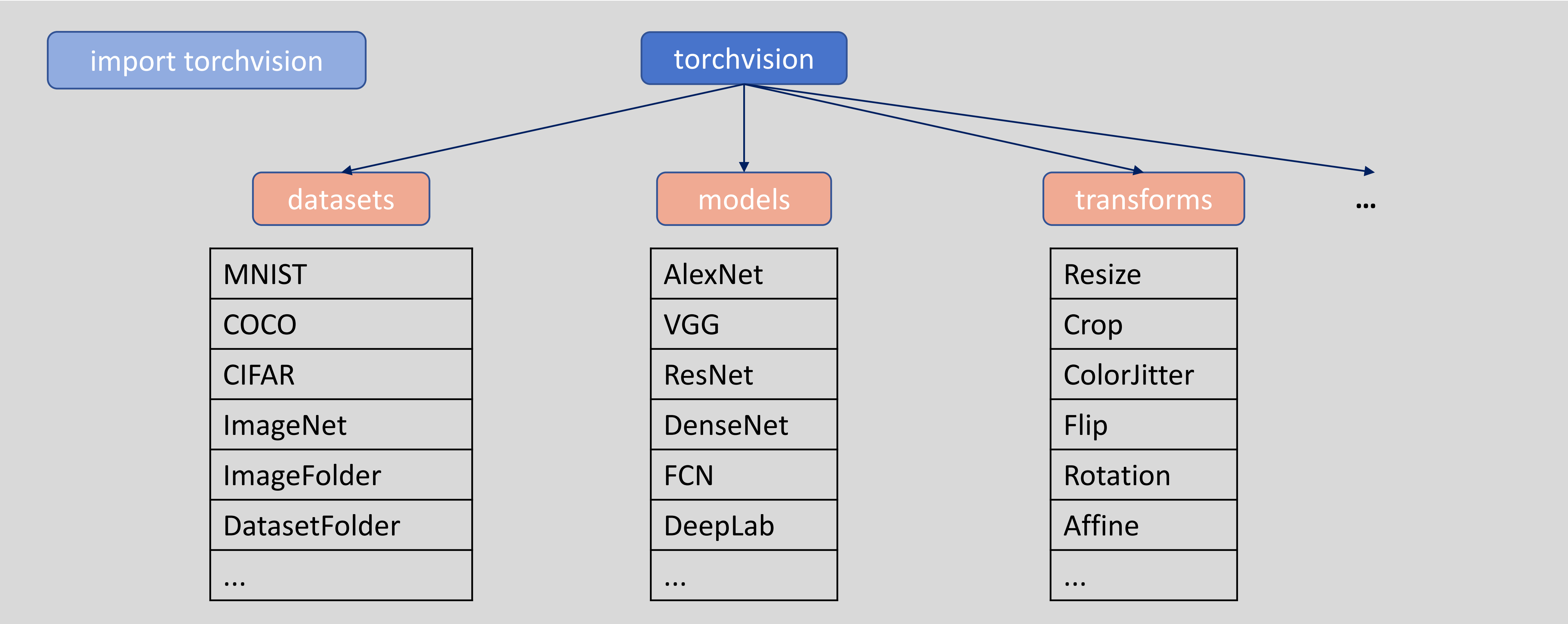
`nn.ReflectionPad1d`	Pads the input tensor using the reflection of the input boundary.
`nn.ReflectionPad2d`	Pads the input tensor using the reflection of the input boundary.
`nn.ReplicationPad1d`	Pads the input tensor using replication of the input boundary.
`nn.ReplicationPad2d`	Pads the input tensor using replication of the input boundary.
`nn.ReplicationPad3d`	Pads the input tensor using replication of the input boundary.
`nn.ZeroPad2d`	Pads the input tensor boundaries with zero.
`nn.ConstantPad1d`	Pads the input tensor boundaries with a constant value.
`nn.ConstantPad2d`	Pads the input tensor boundaries with a constant value.
`nn.ConstantPad3d`	Pads the input tensor boundaries with a constant value.

Common API

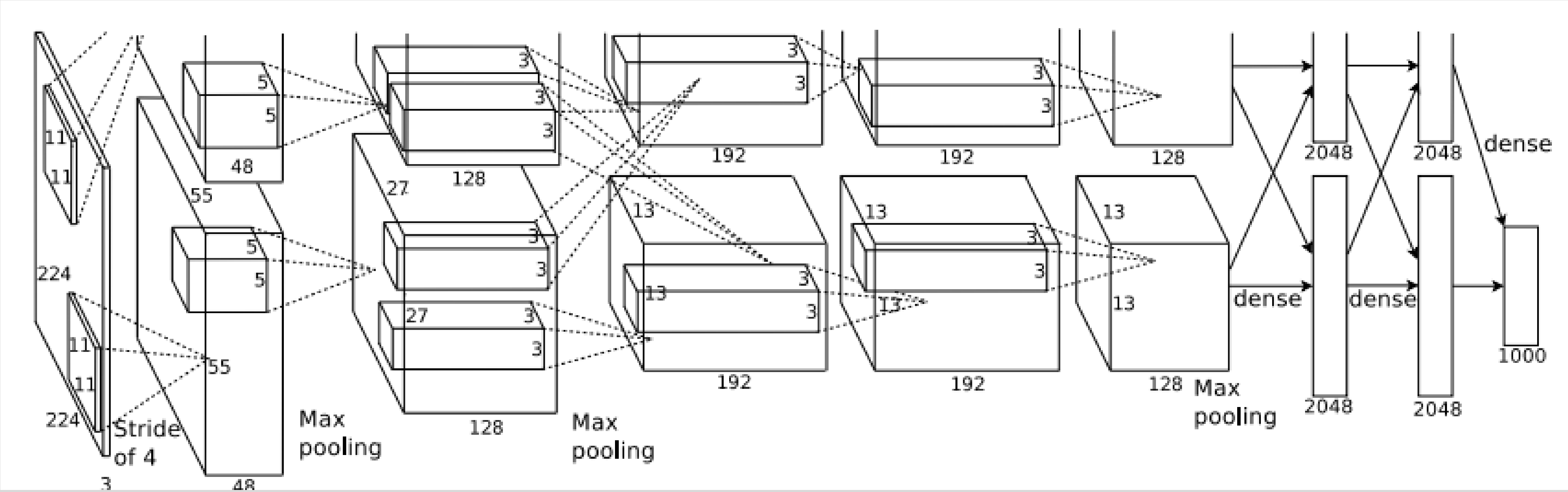
torch.Tensor

Data type	dtype	CPU tensor	GPU tensor
32-bit floating point	`torch.float32` or `torch.float`	`torch.FloatTensor`	`torch.cuda.FloatTensor`
64-bit floating point	`torch.float64` or `torch.double`	`torch.DoubleTensor`	`torch.cuda.DoubleTensor`
16-bit floating point 1	`torch.float16` or `torch.half`	`torch.HalfTensor`	`torch.cuda.HalfTensor`
16-bit floating point 2	`torch.bfloat16`	`torch.BFloat16Tensor`	`torch.cuda.BFloat16Tensor`

Common API


import torchvision
torchvision
datasets
models
transforms
...
MNIST
COCO
CIFAR
ImageNet
ImageFolder
DatasetFolder
...
AlexNet
VGG
ResNet
DenseNet
FCN
DeepLab
...
Resize
Crop
ColorJitter
Flip
Rotation
Affine
...

Example： AlexNet



Example： AlexNet – DataLoader & Dataset

```
loader = DataLoader(
    dataset=dataset,
    batch_size=self.batch_size,
    shuffle=False,
    num_workers=self.num_workers,
    pin_memory=True,
    drop_last=True,
)
```

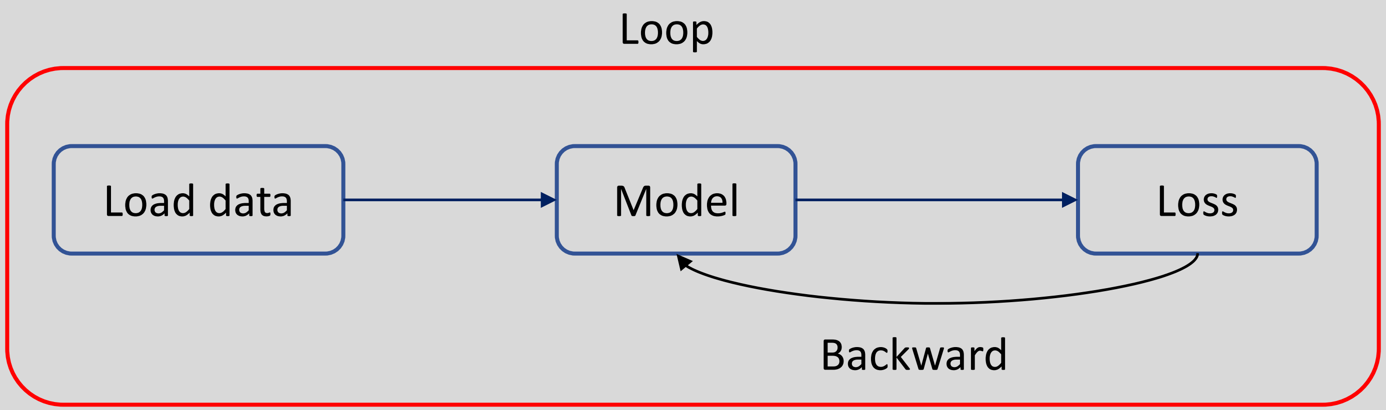
```
from PIL import Image
from skimage import io
from torch.utils.data import Dataset

class AlexDataset(Dataset):
    def __init__(self, root_path, transform, mode):
        super(AlexDataset, self).__init__()
        self.root = root_path
        self.mode = mode
        self.datas = []
        '''
            ......
        '''
        self.transform = transform

    def __getitem__(self, item):
        img_path, label = self.datas[item]
        img = Image.fromarray(io.imread(img_path)).convert("RGB")
        img = self.transform(img)
        return img, label

    def __len__(self):
        return len(self.datas)
```

Example： AlexNet – Model

```python
import torch
from torch import nn, Tensor


class AlexNet(nn.Module):
    def __init__(self, num_classes: int = 1000) -> None:
        super(AlexNet, self).__init__()

        self.features = nn.Sequential(
            nn.Conv2d(3, 64, (11, 11), (4, 4), (2, 2)),
            nn.ReLU(True),
            nn.MaxPool2d((3, 3), (2, 2)),

            nn.Conv2d(64, 192, (5, 5), (1, 1), (2, 2)),
            nn.ReLU(True),
            nn.MaxPool2d((3, 3), (2, 2)),

            nn.Conv2d(192, 384, (3, 3), (1, 1), (1, 1)),
            nn.ReLU(True),
            nn.Conv2d(384, 256, (3, 3), (1, 1), (1, 1)),
            nn.ReLU(True),
            nn.Conv2d(256, 256, (3, 3), (1, 1), (1, 1)),
            nn.ReLU(True),
            nn.MaxPool2d((3, 3), (2, 2)),
        )

        self.avgpool = nn.AdaptiveAvgPool2d((6, 6))
```

```python

        self.avgpool = nn.AdaptiveAvgPool2d((6, 6))

        self.classifier = nn.Sequential(
            nn.Dropout(0.5),
            nn.Linear(256 * 6 * 6, 4096),
            nn.ReLU(True),
            nn.Dropout(0.5),
            nn.Linear(4096, 4096),
            nn.ReLU(True),
            nn.Linear(4096, num_classes),
        )

    def forward(self, x: Tensor) -> Tensor:
        out = self.features(x)
        out = self.avgpool(out)
        out = torch.flatten(out, 1)
        out = self.classifier(out)
        return out

```

Example： AlexNet – Main



Learning Materials

Learning resources

电子资源 >

数据库

电子期刊

校外访问

电子资源使用规定

中文数据库

1 51CTO学院 访问

2 Choice金融终端数据库 访问

学术成果 >

读者荐购 >

教学参考书 >

新书推荐 >

工具与软件 >

馆际互借/文献传递 >

网络资源导航 >

网络公开课

发布时间：2021-07-20 276

序号	名称	简介
1	耶鲁开放课程	耶鲁开放课程是由耶鲁大学于2006 年6 月William and Flora Hewlett 基金会提出申请并于10 月获得资助，在课堂实录的基础上制作开放课程。2007 年12 月12 日，耶鲁大学的开放课程通过网络，免费向公众提供精选的基础课程及相关教学材料。这些课程涵盖了艺术、人文、社会科学和自然科学。
2	英国公开大学	英国公开大学是由英国十几所大学联合组建的。其网络公开课的一大特色，是把课程依难度分为"导论、中级、进阶、研究"四个等级，科目跨文学、法学、商学、教育、理工等领域。
3	麻省理工大学公开课程	麻省理工大学公开课程提供麻省理工学院从本科到研究生教育各层次的课程资源，以多媒体的形式呈现。这些课件资源包括每一门课程的主讲教师信息、讲义、教学大纲、试题、作业、阅读书目、教学法、课堂录像等。
4	中国教育在线	中国最大的综合教育门户，主要发布高考、考研、自考、成人高考、教师招聘、就业、留学等权威的招考、招生、就业、招聘、复习、辅导信息。
5	新浪公开课	汇集哈佛、耶鲁、斯坦福、麻省等全世界各大名校的著名教授视频课程，涉及人文、历史、经济、哲学、理工等各学科。
6	爱课程	是教育部、财政部 "十二五"期间启动实施的"高等学校本科教学质量与教学改革工程"支持建设的高等教育课程资源共享平台。
7	网易公开课	推出"全球名校视频公开课项目"，内容涵盖人文、社会、艺术、科学、金融等领域。可下载客户端。

Learning Materials

Learning resources

Course

中国大学MOOC

网易云课堂 我的职业课堂

腾讯课堂

bilibili

coursera

Docs

CSDN

知

PyTorch

pytorch.org

Code

GitHub

gitee

Q3: Any question?

Homework

Introduction to AI （CS103） – 08

AI Platform Introduction

Jimmy Liu 刘江

2023-11-10