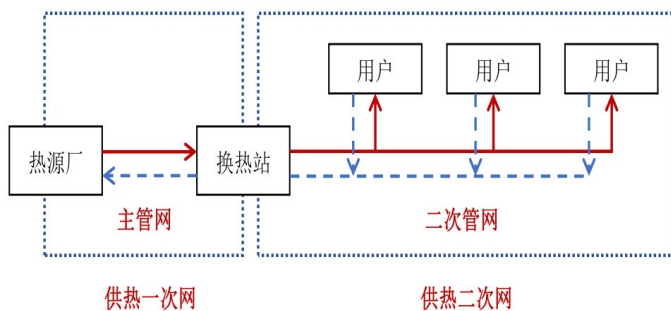


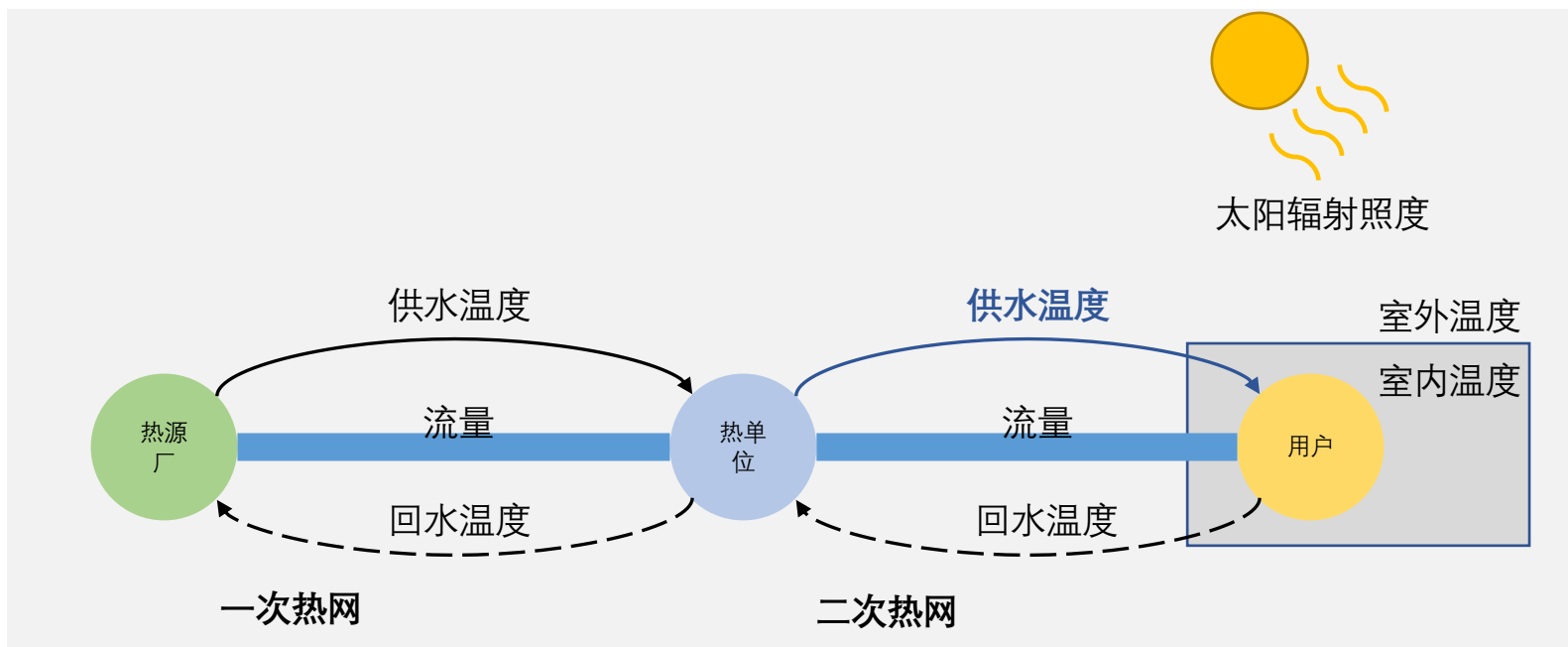
# 项目说明

## 简介

随着城市的发展,供热面积越来越大,集中供热的规模也随之增大,出现了热源紧张、供热不足的现象。集中供热规模增大的同时,也使热网系统平衡调节变得复杂。如何将有限的热源供给更多的用户,起到更好的供热作用,成为了热力公司需要解决的当务之急。



# 供热流程



$$\text{供热量} = (\text{供水温度} - \text{回水温度}) * \text{流量} * \text{比热容}$$

# 供热数据

## 1. 数据分为train和validation两个部分

Train: 2021/12/8—2022/02/25  
Test: 2022/02/26—2022/03/30

## 2. Excel含有两个Sheets

Sheet 1: 室外气象数据、室内温度数据、热网数据  
Sheet 2: 室外太阳辐射照度

## 3. 数据说明

date: 时间  
pri\_supp\_t: 一次网供水温度 (摄氏度)  
pri\_back\_t: 一次网回水温度 (摄氏度)  
pri\_flow: 一次网流量 (T/h)  
sec\_supp\_t: 二次网供水温度 (摄氏度)  
sec\_back\_t: 二次网回水温度 (摄氏度)  
sec\_flow: 二次网流量 (T/h)  
outdoor: 室外温度 (摄氏度)  
indoor: 平均室内温度 (摄氏度)  
irradiance: 太阳辐射照度

## Sheet 1

date	pri_supp_t	pri_back_t	pri_flow	sec_supp_t	sec_back_t	sec_flow	outdoor	indoor
2021/12/8 22:40	71.36	32.413	5.509	35.313	31.861	61.888	4.8	24.435
2021/12/8 22:50	71.212	32.404	5.536	35.266	31.844	62.464	4.8	24.145
2021/12/8 23:00	71.015	32.39	5.658	35.243	31.838	63.818	4.8	24.0923077
2021/12/8 23:10	70.846	32.398	5.572	35.247	31.821	62.271	4.75	24.1042857

## Sheet 2

date	irradiance
12/9/2021 6:00	0
12/9/2021 7:00	6.3
12/9/2021 8:00	80.896
12/9/2021 9:00	219.306

注: Sheet 1取样频率为10min, Sheet 2取样频率为1hour

# 目标与条件

## 目标

1. 开发算法对该供热系统进行仿真，可自行定义仿真模型的输入，输出为平均室内温度。
2. 开发算法对供热系统进行优化，实现通过控制二次网水温，智能调节室内温度，维持室内温度在20 ~ 24度之间的同时，提升能效。

## 说明

1. 供热的调控变量是二次网供水温度，即升温或者降温。
2. 热流为水，即比热容取水的数值。

## 条件

1. 调控频率为小时，即1小时更改一次调控操作。
2. 一小时内升温降温不超过2度。

# Conda——简介

Conda是一个免费、易于安装的包管理器、环境管理器和 Python 发行版，包含 1,500 多个开源包，并提供 免费社区支持。

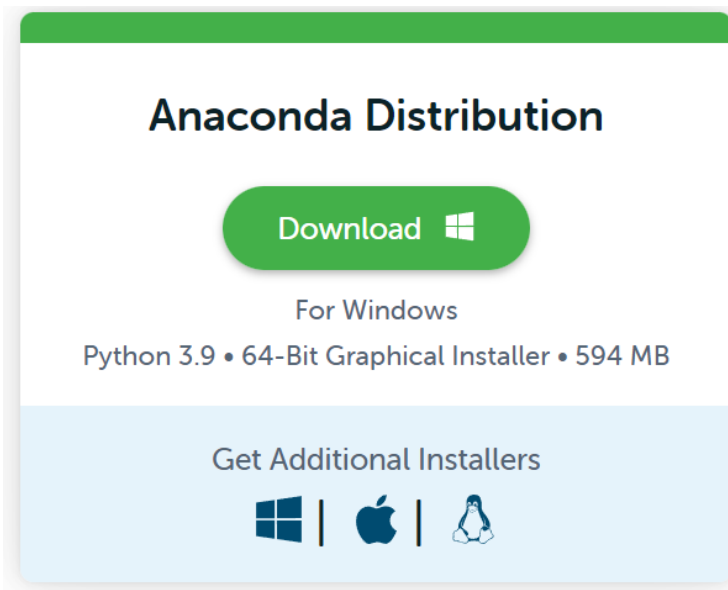
Conda 与平台无关，因此无论在 Windows、macOS 还是 Linux 上都可以使用它。

## 简单概括

Conda 是一个开源的软件包管理系统和环境管理系统，用于安装多个版本的软件包及其依赖关系。。



## Miniconda



<https://anaconda.org.cn/>

# Conda——使用



## 常用命令

- 查询环境

*conda env list*

```
(base) C:\Users\城北徐公>conda env list
# conda environments:
#
base                    * D:\Anaconda\conda
FootScanner_Case       D:\Anaconda\conda\envs\FootScanner_Case
arts_1425               D:\Anaconda\conda\envs\arts_1425
gaiic2022               D:\Anaconda\conda\envs\gaiic2022
```

- 创建环境

*conda create -n arts\_1425*

```
(base) C:\Users\城北徐公>conda activate arts_1425
```

- 激活环境

*conda activate arts\_1425*

```
(arts_1425) C:\Users\城北徐公>
```

- 安装包

*conda install python==3.6*

```
Downloading and Extracting Packages
python-3.6.0          | 31.7 MB | #####
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

- 删除包

*conda uninstall python==3.6*

```
(arts_1425) C:\Users\城北徐公>conda list
# packages in environment at D:\Anaconda\conda\envs\arts_1425:
#
# Name                        Version      Build    Channel
certifi                       2016.2.28    py36_0   http://mirr
pip                           9.0.1        py36_1   http://mirr
python                        3.6.0        0        http://mirr
setuptools                    36.4.0       py36_1   http://mirr
vc                             14           0        http://mirr
vs2015_runtime                14.0.25420   0        http://mirr
wheel                         0.29.0       py36_0   http://mirr
wincertstore                   0.2          py36_0   http://mirr
```

- 查看当前环境已安装包

*conda list*

<https://anaconda.org.cn/>

# Conda——换源



Windows: *C:\用户\你的用户名\.condarc*

Linux: */home/你的用户名/.condarc*

清华源:

channels:

- <https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/main/>
  - <https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/free/>
  - <https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge/>
- ssl\_verify: true

中科大源:

channels:

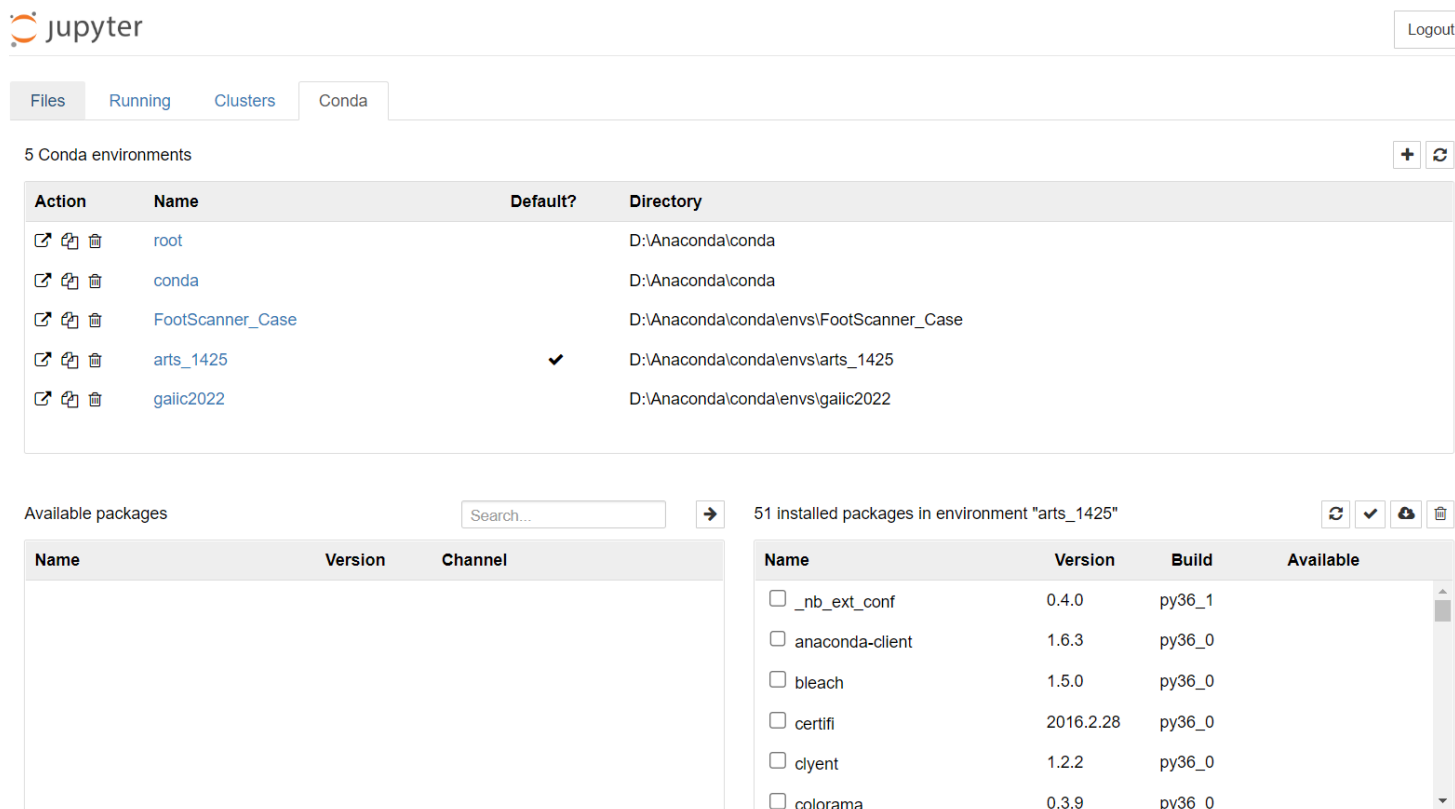
- <https://mirrors.ustc.edu.cn/anaconda/pkg/main/>
  - <https://mirrors.ustc.edu.cn/anaconda/pkg/free/>
  - <https://mirrors.ustc.edu.cn/anaconda/cloud/conda-forge/>
- ssl\_verify: true

<https://anaconda.org.cn/>

# Conda——Jupyter Notebook
















安装插件:


```
conda install nb_conda
```







The image shows the Jupyter Notebook interface with the 'Conda' tab selected. It displays 5 Conda environments and a list of installed packages for the 'arts\_1425' environment.

**5 Conda environments**

Action	Name	Default?	Directory
  	root		D:\Anaconda\conda
  	conda		D:\Anaconda\conda
  	FootScanner_Case		D:\Anaconda\conda\envs\FootScanner_Case
  	arts_1425	✓	D:\Anaconda\conda\envs\arts_1425
  	galic2022		D:\Anaconda\conda\envs\galic2022

**Available packages**  

Name	Version	Channel
------	---------	---------

**51 installed packages in environment "arts\_1425"**    

Name	Version	Build	Available
<input type="checkbox"/> _nb_ext_conf	0.4.0	py36_1	
<input type="checkbox"/> anaconda-client	1.6.3	py36_0	
<input type="checkbox"/> bleach	1.5.0	py36_0	
<input type="checkbox"/> certifi	2016.2.28	py36_0	
<input type="checkbox"/> clyent	1.2.2	py36_0	
<input type="checkbox"/> colorama	0.3.9	py36_0	

<https://anaconda.org.cn/>



# Debug——PDB

## Pdb调试包：

```
import pdb
pdb.set_trace #进入调试
```

pdb 是 python 自带的一个包，为 python 程序提供了一种交互的源代码调试功能，主要特性包括设置断点、单步调试、进入函数调试、查看当前代码、查看栈片段、动态改变变量的值等。

```
import pdb
pdb.set_trace()

def debug_function(i):
    print("The number is : ",i)

x = 10
for i in range(x):
    debug_function(i)
```

(Pdb)

--Return--

> <ipython-input-5-8d571bf8480b>(2)<module>()->None

-> pdb.set\_trace()

# Debug——PDB

(Pdb) help

Documented commands (type help <topic>):

=====

EOF	c	d	h	list	q	rv	undisplay
a	cl	debug	help	ll	quit	s	unt
alias	clear	disable	ignore	longlist	r	source	until
args	commands	display	interact	n	restart	step	up
b	condition	down	j	next	return	tbreak	w
break	cont	enable	jump	p	retval	u	whatis
bt	continue	exit	l	pp	run	unalias	where

命令	说明	命令	说明
List / l	查看当前位置前后代码	Step / s	执行下一步，进入函数
Longlist / ll	查看当前函数代码	next / n	执行下一步，跳过函数
Break / b	插入断点	p	打印变量
tbreak	插入临时断点	a	打印函数参数值
Clear / cl	清除断点	whatis	打印变量类型
Continue / c	执行到下一个断点	Exit / q	退出调试

<https://docs.python.org/zh-cn/3/library/pdb.html>

# Debug——PDB

## Pdb实例：

### 查看代码

```
(Pdb) 1
1     import pdb
2 -> pdb.set_trace()
3
4     def debug_function(i):
5         print("The number is : ",i)
6
7     x = 10
8     for i in range(x):
9         debug_function(i)
[EOF]
```

### 查看变量及类型

```
(Pdb) c
The number is : 0
> <ipython-input-10-7b1e18993015>(9)<module>()->None
-> for i in range(x):
(Pdb) p x
10
(Pdb) p s
0
(Pdb) whatis s
<class 'int'>
```

### 插入断点

```
(Pdb) b 5
Breakpoint 10 at <ipython-input-9-7b1e18993015>:5
(Pdb) b 7
Breakpoint 11 at <ipython-input-9-7b1e18993015>:7
(Pdb) b

Num Type          Disp Enb   Where
10 breakpoint keep yes    at <ipython-input-9-7b1e18993015>:5
11 breakpoint keep yes    at <ipython-input-9-7b1e18993015>:7
```

*help* 命令 查看使用说明。  
或者翻阅官方文档。

<https://docs.python.org/zh-cn/3/library/pdb.html>

# NumPy——简介



## NumPy:

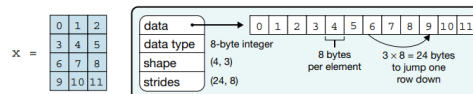
NumPy(Numerical Python) 是 Python 语言的一个扩展程序库, 支持大量的维度数组与矩阵运算, 此外也针对数组运算提供大量的数学函数库。

## 特点

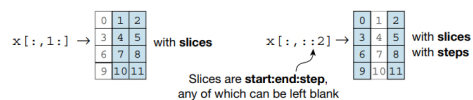
- 一个强大的N维数组对象 ndarray
- 广播功能函数
- 整合 C/C++/Fortran 代码的工具
- 线性代数、傅里叶变换、随机数生成等功能

## Review

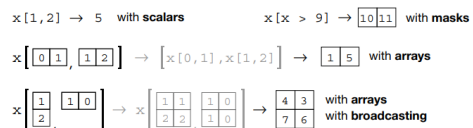
### a Data structure



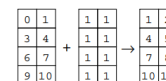
### b Indexing (view)



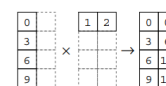
### c Indexing (copy)



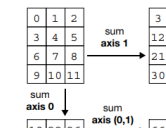
### d Vectorization



### e Broadcasting



### f Reduction



### g Example

```
In [1]: import numpy as np
In [2]: x = np.arange(12)
In [3]: x = x.reshape(4, 3)

In [4]: x
Out[4]:
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])

In [5]: np.mean(x, axis=0)
Out[5]: array([4.5, 5.5, 6.5])

In [6]: x = x - np.mean(x, axis=0)

In [7]: x
Out[7]:
array([[-4.5, -4.5, -4.5],
       [-1.5, -1.5, -1.5],
       [ 1.5,  1.5,  1.5],
       [ 4.5,  4.5,  4.5]])
```

## Array programming with NumPy(2020)

# NumPy——ndarray



N 维数组对象 ndarray，它是一系列同类型数据的集合，以 0 下标为开始进行集合中元素的索引。

```
import numpy as np
```

np 约定俗成

```
data = np.array([1, 4, 2, 5], dtype=np.float64)  
print(data)  
print(data.dtype)
```

指定数据类型

```
[ 1.  4.  2.  5.]  
float64
```

指定数值和维度

```
data = np.array([[1, 2, 5], [7, 9, 8], [6, 5, 4]])  
print(data)  
print(data.shape)
```

[ ]为维度的界限

```
[[1 2 5]  
 [7 9 8]  
 [6 5 4]]  
(3, 3)
```

## 数据类型

bool_	uint32
int_	uint64
intc	float_
int8	float16
int16	float32
int32	float64
int64	complex_
uint8	complex64
uint16	complex128

<https://numpy.org/doc/>

# NumPy——索引



冒号：如果只放置一个参数，如 [4]，将返回与该索引相对应的单个元素。如果为 [4:]，表示从该索引开始以后的所有项都将被提取。如果使用了两个参数，如 [4:8]，那么则提取两个索引(不包括停止索引)之间的项。

```
range = np.arange(12).reshape(3,4)
print(range)
print(range[0,1])
print(range[:,1])
print(range[0,:])
print(range[0:2,1:3])
```

返回第1行第2列元素

返回第2列元素

返回第1行元素

返回第1, 2行第2, 3列元素

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

```
1
[1 5 9]
[0 1 2 3]
[[1 2]
 [5 6]]
```

[0:2, 1:3] 左闭右开

# NumPy——创建数组



创建空数组

```
List = np.empty( (3, 3), dtype=np.float64 )
```

创建零数组

```
List = np.zeros( (3, 3), dtype=np.float64 )
```

创建全1数组

```
List = np.ones( (3, 3), dtype=np.float64 )
```

创建全x数组

```
List = np.full( (3, 3), 9, dtype=np.float64 )
```

```
list = np.full((3, 3), 9, dtype=np.float64)
print(list)
print(list.shape)
print(list.dtype)
```

```
[[ 9.  9.  9.]
 [ 9.  9.  9.]
 [ 9.  9.  9.]]
(3, 3)
float64
```

数据类型

填充数值

形状大小

<https://numpy.org/doc/>

```
list = np.ones((3, 3), dtype=np.float64) * 9
print(list)
print(list.shape)
print(list.dtype)
```

```
[[ 9.  9.  9.]
 [ 9.  9.  9.]
 [ 9.  9.  9.]]
(3, 3)
float64
```

Full函数等价操作

# NumPy——范围数组



`numpy.arange(start, stop, step, dtype)` #起始值 终止值 步长 数据类型

```
range = np.arange(8, 14, 2, dtype=np.float32)
print(range)
```

```
[ 8.  10.  12.]
```

左闭右开

`numpy.linspace(start, stop, num, endpoint, retstep, dtype)`  
#起始值 终止值 样本数量 是否包含终止值 是否显示间隔 数据类型

```
line = np.linspace(5, 10, 5, endpoint=True, retstep=True, dtype=np.float32)
print(line)
```

```
(array([ 5. ,  6.25,  7.5 ,  8.75, 10. ], dtype=float32), 1.25)
```

间隔

`numpy.logspace(start, stop, num, endpoint, base, dtype)`  
#起始值 终止值 样本数量 是否包含终止值 log底数 数据类型

```
log = np.logspace(1, 8, 8, endpoint=True, base=2, dtype=np.float32)
print(log)
```

```
[ 2.   4.   8.  16.  32.  64. 128. 256.]
```

<https://numpy.org/doc/>



# NumPy——数组操作



修改数组大小

```
numpy.reshape(arr, newshape, order= 'C' ) #数组 形状 顺序
```

翻转数组

```
numpy.transpose(arr, axes)#数组 维度顺序
```

arr.T #数组转置

修改维度

```
numpy.expand_dims(arr, axis)#数组 维度位置
```

```
numpy.squeeze(arr, axis)#数组 一维维度位置
```

连接数组

```
numpy.concatenate((a1, a2, ...), axis)#数组 方向
```

分割数组

```
numpy.split(ary, indices_or_sections, axis)#数组 分割数量/位置 方向
```

```
range = np.arange(12).reshape(3, 4)
print(range)
print(np.transpose(range))
print(range.T)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
[[ 0  4  8]
 [ 1  5  9]
 [ 2  6 10]
 [ 3  7 11]]
[[ 0  4  8]
 [ 1  5  9]
 [ 2  6 10]
 [ 3  7 11]]
```

# NumPy——数组统计



最值

`numpy.amin(arr, axis)` #数组 方向

`numpy.amax(arr, axis)` #数组 方向

中位数

`numpy.median(arr, axis)` #数组 方向

均值

`numpy.mean(arr, axis)` #数组 方向

加权均值

`numpy.average(arr, weights,axis)` #数组 权重 方向

标准差

`numpy.std(arr,axis)` #数组 方向

方差

`numpy.var(arr,axis)` #数组 方向

<https://numpy.org/doc/>

```
range = np.arange(12).reshape(3,4)
print(np.amin(range,axis=0))
print(np.amax(range,axis=0))
print(np.median(range,axis=0))
print(np.average(range,weights=range,axis=0))
print(np.std(range,axis=0))
print(np.var(range,axis=0))
```

```
[0 1 2 3]
[ 8  9 10 11]
[ 4.  5.  6.  7.]
[ 6.66666667  7.13333333  7.77777778  8.52380952]
[ 3.26598632  3.26598632  3.26598632  3.26598632]
[10.66666667 10.66666667 10.66666667 10.66666667]
```

# SciPy——简介



## SciPy:

Scipy 是一个用于数学、科学、工程领域的常用软件包，可以处理最优化、线性代数、积分、插值、拟合、特殊函数、快速傅里叶变换、信号处理、图像处理、常微分方程求解器等。

通常来说，NumPy 和 SciPy 的协同工作可以高效解决很多问题。

[Numpy 1.17.0 Reference Guide, \[HTML+zip\], \[PDF\]](#)

[Numpy 1.17.0 User Guide, \[PDF\]](#)

[Numpy 1.16.1 Reference Guide, \[HTML+zip\], \[PDF\]](#)

[Numpy 1.16.1 User Guide, \[PDF\]](#)

[Numpy 1.16.0 Reference Guide, \[HTML+zip\], \[PDF\]](#)

[Numpy 1.16.0 User Guide, \[PDF\]](#)

[Numpy 1.15.4 Reference Guide, \[HTML+zip\], \[PDF\]](#)

[SciPy 1.9.0 Documentation, \[HTML+zip\]](#)

[SciPy 1.8.1 Documentation, \[HTML+zip\], \[PDF\]](#)

[SciPy 1.8.0 Documentation, \[HTML+zip\], \[PDF\]](#)

[SciPy 1.7.1 Reference Guide, \[HTML+zip\], \[PDF\]](#)

[SciPy 1.7.0 Reference Guide, \[HTML+zip\], \[PDF\]](#)

[SciPy 1.6.3 Reference Guide, \[HTML+zip\], \[PDF\]](#)

[SciPy 1.6.2 Reference Guide, \[HTML+zip\], \[PDF\]](#)

NumPy和SciPy文档官网

<https://docs.scipy.org/doc/>

scipy.cluster	向量量化
scipy.constants	数学常量
scipy.fft	快速傅里叶变换
scipy.integrate	积分
scipy.interpolate	插值
scipy.io	数据输入输出
scipy.linalg	线性代数
scipy.misc	图像处理
scipy.ndimage	N 维图像
scipy.odr	正交距离回归
scipy.optimize	优化算法
scipy.signal	信号处理
scipy.sparse	稀疏矩阵
scipy.spatial	空间数据结构和算法
scipy.special	特殊数学函数
scipy.stats	统计函数

# SciPy——数学常量



## scipy.constants:

scipy的constants模块包含了大量用于科学计算的常数

```
print(constants.speed_of_light)
print(constants.speed_of_sound)
print(constants.pi)
print(constants.golden)
print(constants.g)
print(constants.hour)
print(constants.year)
```

299792458.0

340.5

3.141592653589793

1.618033988749895

9.80665

3600.0

31536000.0

['Avogadro', 'Boltzmann', 'Btu', 'Btu\_IT', 'Btu\_th', 'C2F', 'C2K', 'ConstantWarning', 'F2C', 'F2K', 'G', 'Julian\_year', 'K2C', 'K2F', 'N\_A', 'Planck', 'R', 'Rydberg', 'Stefan\_Boltzmann', 'Tester', 'Wien', '\_\_all\_\_', '\_\_builtins\_\_', '\_\_cached\_\_', '\_\_doc\_\_', '\_\_file\_\_', '\_\_loader\_\_', '\_\_name\_\_', '\_\_package\_\_', '\_\_path\_\_', '\_\_spec\_\_', '\_obsolete\_constants', 'absolute\_import', 'acre', 'alpha', 'angstrom', 'arcmin', 'arcminute', 'arcsec', 'arcsecond', 'astronomical\_unit', 'atm', 'atmosphere', 'atomic\_mass', 'atto', 'au', 'bar', 'barrel', 'bbl', 'c', 'calorie', 'calorie\_IT', 'calorie\_th', 'carat', 'centi', 'codata', 'constants', 'convert\_temperature', 'day', 'deci', 'degree', 'degree\_Fahrenheit', 'deka', 'division', 'dyn', 'dyne', 'e', 'eV', 'electron\_mass', 'electron\_volt', 'elementary\_charge', 'epsilon\_0', 'erg', 'exa', 'exbi', 'femto', 'fermi', 'find', 'fine\_structure', 'fluid\_ounce', 'fluid\_ounce\_US', 'fluid\_ounce\_imp', 'foot', 'g', 'gallon', 'gallon\_US', 'gallon\_imp', 'gas\_constant', 'gibi', 'giga', 'golden', 'golden\_ratio', 'grain', 'gram', 'gravitational\_constant', 'h', 'hbar', 'hectare', 'hecto', 'horsepower', 'hour', 'hp', 'inch', 'k', 'kgf', 'kibi', 'kilo', 'kilogram\_force', 'kmh', 'knot', 'lambda2nu', 'lb', 'lbf', 'light\_year', 'liter', 'litre', 'long\_ton', 'm\_e', 'm\_n', 'm\_p', 'm\_u', 'mach', 'mebi', 'mega', 'metric\_ton', 'micro', 'micron', 'mil', 'mile', 'milli', 'minute', 'mmHg', 'mph', 'mu\_0', 'nano', 'nautical\_mile', 'neutron\_mass', 'nu2lambda', 'ounce', 'oz', 'parsec', 'pebi', 'peta', 'physical\_constants', 'pi', 'pico', 'point', 'pound', 'pound\_force', 'precision', 'print\_function', 'proton\_mass', 'psi', 'pt', 'short\_ton', 'sigma', 'speed\_of\_light', 'speed\_of\_sound', 'stone', 'survey\_foot', 'survey\_mile', 'tebi', 'tera', 'test', 'ton\_TNT', 'torr', 'troy\_ounce', 'troy\_pound', 'u', 'unit', 'value', 'week', 'yard', 'year', 'yobi', 'yotta', 'zebi', 'zepto', 'zero\_Celsius', 'zetta']

所有常量

<https://docs.scipy.org/doc/>

# SciPy——读写



## scipy.io:

scipy提供io模块，用于从各种文件格式读取数据和将数据写入各种文件格式。

读写MATLAB文件



```
from scipy import io as spio
m = np.arange(16).reshape(4,4)
spio.savemat('file.mat', {'np_data': m})
data = spio.loadmat('file.mat')
data['np_data']
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

其它格式文件

- .sav
- .mtx
- .dat
- .nc
- .hb
- .wav
- .arff

# SciPy——线性代数



## scipy.linalg:

scipy提供linalg模块，该库包含了线性代数所需的所有功能，例如求点积、内积、行列式、矩阵的逆和求解线性矩阵方程等等。

### 求行列式

```
from scipy import linalg
data = np.arange(1, 5).reshape(2, 2)
print(data)
linalg.det(data)
```

```
[[1 2]
 [3 4]]
-2.0
```

### 求逆相乘

```
data = np.arange(4, 8).reshape(2, 2)
print(data)
data_inv = linalg.inv(data)
print(data_inv)
print(data@data_inv)
```

```
[[4 5]
 [6 7]]
[[ -3.5  2.5]
 [ 3.  -2. ]]
[[ 1.00000000e+00 -1.77635684e-15]
 [ 3.55271368e-15  1.00000000e+00]]
```

@ 矩阵乘法

# SciPy——积分



## scipy.integrate:

该模块提供了几种积分方法，包括普通的微分方程积分器和多重积分器。

`integrate.quad(f, a, b)` #单变量积分

$$\int_0^1 \cos^2(x) + 1 \, dx$$

```
import numpy as np
from scipy import integrate

def f(x):
    return np.cos(x) ** 2 + 1

val, err = integrate.quad(f, 0, 1)
print(val, err)

1.7273243567064205 1.917715271811305e-14
```

`integrate.dblquad(f, y_a, y_b, x_a, x_b)`

#二重积分

$$\int_0^{\frac{1}{2}} \int_0^{1-2y} xy \, dx \, dy$$

```
def f(x, y):
    return x*y

val, err = integrate.dblquad(f, 0, 1/2, lambda x: 0, lambda x: 1-2*x)
print(val, err)

0.010416666666666668 4.101620128472366e-16
```

内层在前

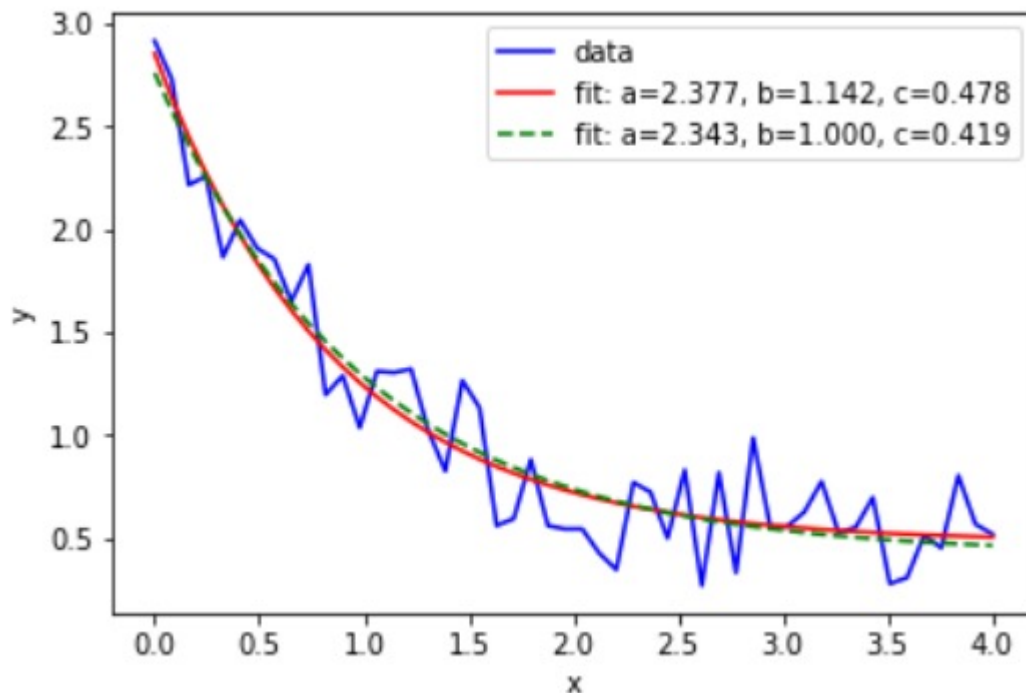
内层上下限表达式形式

# SciPy——最优化



## scipy.optimize :

scipy提供optimize模块，该模块提供函数最小化、曲线拟合和求根的算法。。



curve\_fit函数做拟合

<https://docs.scipy.org/doc/>



# SciPy——快速傅里叶



## scipy.fft:

scipy提供fft模块，可以让用户计算快速傅里叶变换。

```
from scipy import fft, ifft
fft(np.exp(2j * np.pi * np.arange(8) / 8))

array([ -3.44509285e-16 +1.22464680e-16j,
         8.00000000e+00 -5.68502218e-15j,
         3.44509285e-16 +1.22464680e-16j,
         1.44328993e-15 +1.77635684e-15j,
         9.95799250e-17 +1.22464680e-16j,
         0.00000000e+00 +1.64244978e-15j,
        -9.95799250e-17 +1.22464680e-16j,
        -1.44328993e-15 +1.77635684e-15j])
```

```
ifft([0, 4, 0, 0])

array([ 1.+0.j,  0.+1.j, -1.+0.j,  0.-1.j])
```

# PyTorch——介绍



## PyTorch:

PyTorch是一个基于Torch的Python开源机器学习库，用于自然语言处理等应用程序。

PyTorch提供了两个高级功能：

- 1.具有强大的GPU加速的张量计算
  - 2.包含自动求导系统的深度神经网络
- 无缝替换NumPy
  - 让神经网络的实现变得更加容易。

PyTorch Build	Stable (1.12.1)		Preview (Nightly)		LTS (1.8.2)	
Your OS	Linux		Mac		Windows	
Package	Conda	Pip		LibTorch		Source
Language	Python			C++ / Java		
Compute Platform	CUDA 10.2	CUDA 11.3	CUDA 11.6	ROCm 5.1.1		CPU
Run this Command:	CUDA-10.2 PyTorch builds are no longer available for Windows, please use CUDA-11.6					

<https://pytorch.org/tutorials/>

# PyTorch——张量



## Tensors :

Tensors 类似于 NumPy 的 ndarrays ,  
同时 Tensors 可以使用 GPU 进行计算。

创建空张量

```
torch.empty(size, dtype) #大小 类型
```

创建随机张量

```
torch.rand(size, dtype) #大小 类型
```

创建全0张量

```
torch.zeros(size, dtype) #大小 类型
```

创建全1张量

```
torch.ones(size, dtype) #大小 类型
```

Tensors 与 ndarrays 拥有相同的运算  
操作方法, 如转置、索引、切片、数学  
运算、线性代数、随机采样等。

```
torch.empty_like(tensor, dtype) #目标 类型
```

```
torch.rand_like(tensor, dtype) #目标 类型
```

```
torch.zeros_like(tensor, dtype) #目标 类型
```

```
torch.ones_like(tensor, dtype) #目标 类型
```

# PyTorch——张量



使用GPU

```
if torch.cuda.is_available(): tensor = tensor.to('cuda')
```

张量变换为numpy数组

```
Tensor.numpy()
```

numpy数组变换为张量

```
torch.from_numpy(arr)
```

改变张量形状

```
Tensor.view(shape)
```

可以利用 GPU 的性能进行计算NumPy 的替代品

<https://pytorch.org/tutorials/>

# PyTorch——自动求导



## torch.autograd:

autograd是 PyTorch 的自动差分引擎，可为神经网络训练提供支持。

## 更新参数代码:

<code>optimizer.zero_grad()</code>	----->	梯度重置
<code>loss.backward()</code>	----->	反向传播计算梯度
<code>optimizer.step()</code>	----->	根据梯度更新一步参数

Pytorch为我们封装好自动求导 API，我们在写一个自己的网络的时候，几乎不用过于去关注求导问题，因为这些 API 已经在私底下处理好了这些事情。

## 一般流程:

搭建目标的模型，处理数据的载入，调用optimizer 和 loss function，开始训练和测试。

<https://pytorch.org/tutorials/>

# PyTorch——MNIST



## MNIST:

MNIST包含70,000张手写数字0~9的黑白照片: 60,000张用于训练, 10,000张用于测试。图像是灰度居中, 28x28像素的。

## 导入包:

```
import torch
import torchvision
from torch.utils.data import DataLoader
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
```

torchvision包含一些常用的数据集、模型、转换函数等。

用来处理模型输入数据的一个工具类。

构建运行神经网络的类

包含激活函数, 损失函数, softmax函数等。

包含 SGD 等优化器, 可在反向传播步骤中更新 Parameter 的权重。

# PyTorch——MNIST



导入数据:

```
train_loader = torch.utils.data.DataLoader(  
    torchvision.datasets.MNIST('./data/', train=True, download=True,  
        transform=torchvision.transforms.Compose([  
            torchvision.transforms.ToTensor(),  
            torchvision.transforms.Normalize(  
                (0.1307,), (0.3081,))  
        ])),  
    batch_size=batch_size_train, shuffle=True)
```



# PyTorch——MNIST



## CNN网络模型:

```
class Net(nn.Module): -----> (b, 1, 28, 28)
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 10, kernel_size=5) -----> (b, 10, 24, 24)
        self.conv2 = nn.Conv2d(10, 20, kernel_size=5) -----> (b, 20, 8, 8)
        self.conv2_drop = nn.Dropout2d()
        self.fc1 = nn.Linear(320, 50) -----> (b, 50)
        self.fc2 = nn.Linear(50, 10) -----> (b, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), 2)) -----> (b, 10, 12, 12)
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2)) -----> (b, 20, 4, 4)
        x = x.view(-1, 320) -----> (b, 320)
        x = F.relu(self.fc1(x))
        x = F.dropout(x, training=self.training)
        x = self.fc2(x)
        return F.log_softmax(x, dim=1)
```



# PyTorch——MNIST



训练搭建：

使用设备定义：

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

模型和优化器定义：

```
network = Net().to(device)
optimizer = optim.SGD(network.parameters(), lr=learning_rate,
momentum=momentum)

if os.path.exists('model.pth'):
    network_state_dict = torch.load('model.pth')
    network.load_state_dict(network_state_dict)
if os.path.exists('optimizer.pth'):
    optimizer_state_dict = torch.load('optimizer.pth')
    optimizer.load_state_dict(optimizer_state_dict)
```

<https://pytorch.org/tutorials/>

# PyTorch——MNIST



训练过程：

```
def train(epoch):
    network.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        optimizer.zero_grad()
        output = network(data.to(device))
        loss = F.nll_loss(output, target)
        loss.backward()
        optimizer.step()

    .....

    torch.save(network.state_dict(), './model.pth')
    torch.save(optimizer.state_dict(), './optimizer.pth')
```

# PyTorch——MNIST



## RNN网络模型：

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.rnn = nn.RNN(28, 128, 1, batch_first=True,
nonlinearity='relu')
        self.fc1 = nn.Linear(128, 10)
```

输入维度 隐藏层维度 层数  
batch首位 激活函数

```
    def forward(self, x):
        x, h_n = self.rnn(x, None)
        x = self.fc1(x[:, -1, :])
        return F.log_softmax(x, dim=1)
```

初始参数为0  
最后一个特征连接全连接层

























```
optimizer.zero_grad()
data = data.view(-1, 28, 28)
output = network(data.to(device))
loss = F.nll_loss(output, target)
loss.backward()
optimizer.step()
```

注意维度的转换

























# PyTorch——MNIST



预测结果:

Prediction: 3 	Prediction: 0 	Prediction: 3 	Prediction: 3 
Prediction: 1 	Prediction: 3 	Prediction: 1 	Prediction: 3 
Prediction: 5 	Prediction: 7 	Prediction: 9 	Prediction: 1 
Prediction: 3 	Prediction: 2 	Prediction: 4 	Prediction: 9 
Prediction: 3 	Prediction: 9 	Prediction: 4 	Prediction: 8 
Prediction: 5 	Prediction: 0 	Prediction: 7 	Prediction: 1 

CNN

Prediction: 9 	Prediction: 2 	Prediction: 5 	Prediction: 6 
Prediction: 3 	Prediction: 5 	Prediction: 2 	Prediction: 6 
Prediction: 5 	Prediction: 3 	Prediction: 7 	Prediction: 5 
Prediction: 6 	Prediction: 9 	Prediction: 9 	Prediction: 1 
Prediction: 7 	Prediction: 4 	Prediction: 6 	Prediction: 3 
Prediction: 2 	Prediction: 1 	Prediction: 2 	Prediction: 3 

RNN

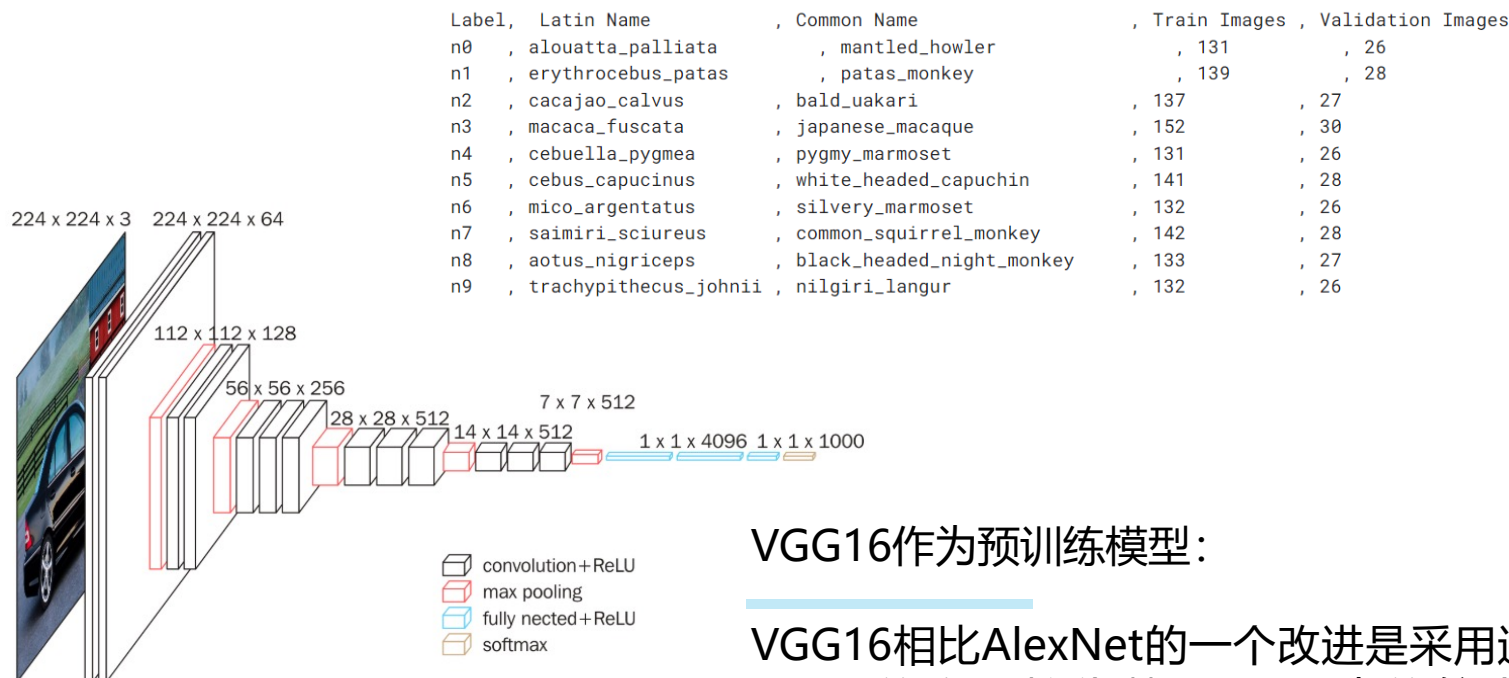
# PyTorch——预训练



[10 Monkey Species | Kaggle](https://www.kaggle.com/datasets/slothkong/10-monkey-species)

数据集包含1400张10类猴子照片，400 \* 300像素或者更大。

<https://www.kaggle.com/datasets/slothkong/10-monkey-species>



VGG16作为预训练模型：

VGG16相比AlexNet的一个改进是采用连续的几个3x3的卷积核代替AlexNet中的较大卷积核(11x11, 7x7, 5x5)。

<https://pytorch.org/tutorials/>

# PyTorch——预训练



## 数据处理：

```
tran_data_transforms = transforms.Compose([
    transforms.RandomSizedCrop(224), -----> 随机裁剪
    transforms.RandomHorizontalFlip(p=0.5), -----> 随机翻转
    transforms.ToTensor(), -----> 转为tensor
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]) -----> 标准化
])

val_data_transforms = transforms.Compose([
    transforms.Resize(256), -----> 重置分辨率
    transforms.CenterCrop(224), -----> 中心裁剪
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

train_data = datasets.ImageFolder("archive/training/training", transform=train_data_transforms)
val_data = datasets.ImageFolder("archive/validation/validation", transform=val_data_transforms)
```

读取图片

```
train_loader = torch.utils.data.DataLoader(train_data, batch_size=batch_size_train, shuffle=True)
test_loader = torch.utils.data.DataLoader(val_data, batch_size=batch_size_test, shuffle=True)
```

## 加载数据

<https://pytorch.org/tutorials/>

# PyTorch——预训练



## 网络模型：

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        vgg16 = models.vgg16(pretrained=True)
        vgg = vgg16.features
        for param in vgg.parameters():
            param.requires_grad_(False)
        self.vgg = vgg
        self.classifier = nn.Sequential(
            nn.Linear(25088, 512),
            nn.ReLU(),
            nn.Linear(512, 256),
            nn.ReLU(),
            nn.Linear(256, 10),
            nn.Softmax(dim=1)
        )

    def forward(self, x):
        x = self.vgg(x)
        x = x.view(x.size(0), -1)
        x = self.classifier(x)
        return x
```

加载预训练模型

获取特征提取层

冻结梯度参数

定义分类容器头

# PyTorch——预训练



预测结果：

Prediction: 1



Prediction: 7



Prediction: 1



Prediction: 4



Prediction: 7



Prediction: 5



Prediction: 4



Prediction: 2



Prediction: 0



Prediction: 4



Prediction: 5



Prediction: 4



Prediction: 9



Prediction: 0



Prediction: 8



Prediction: 9



Prediction: 1



Prediction: 1



Prediction: 8



Prediction: 1



Prediction: 9



Prediction: 0



Prediction: 7



Prediction: 8





## 练习

- 根据个人情况实操上述Conda, PDB, NumPy, SciPy和PyTorch内容
- 根据给出部分MNIST代码复现CNN和RNN训练测试过程。
- 使用预训练+微调模型, 复现10类猴子分类问题。