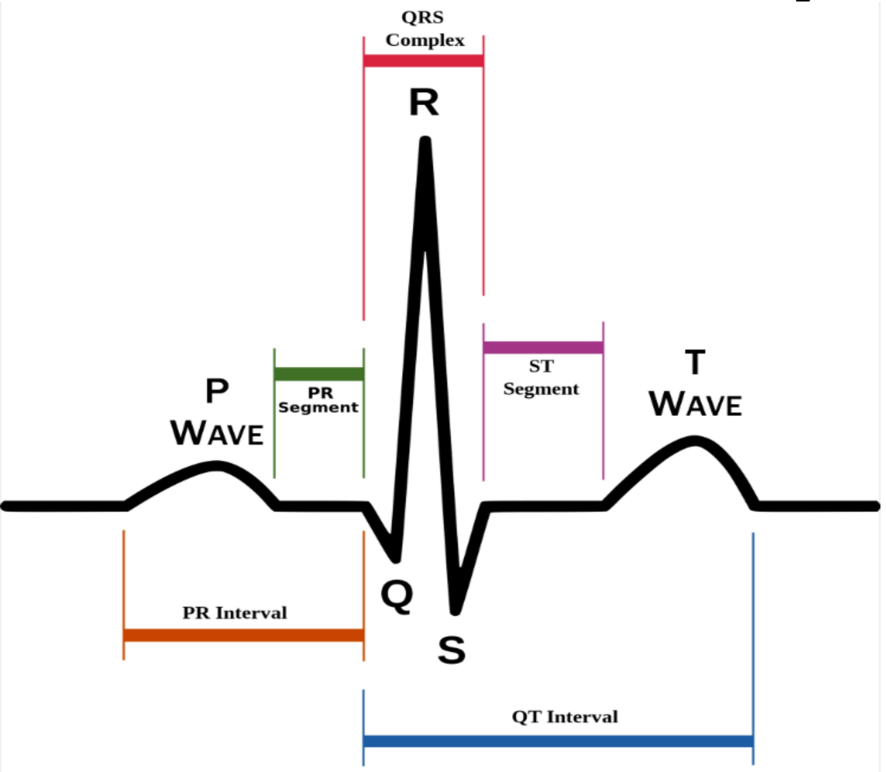
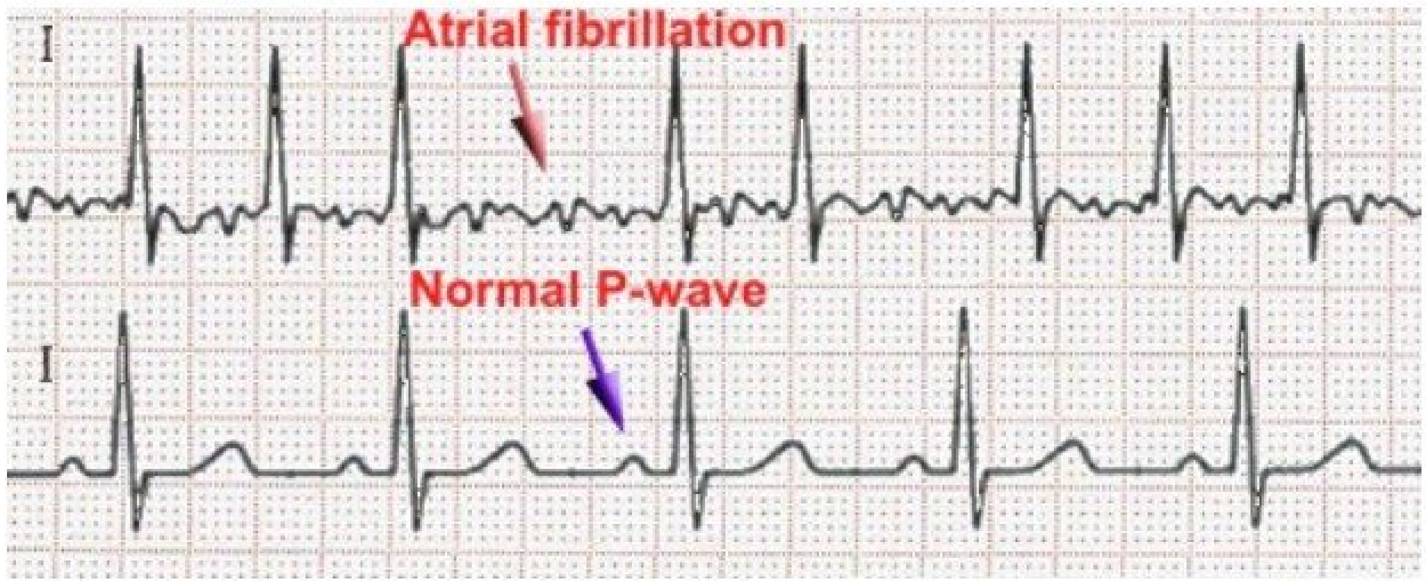
现代信号处理

Project2 第二次讲解
2022.12.8

Project2: 心电信号房颤检测





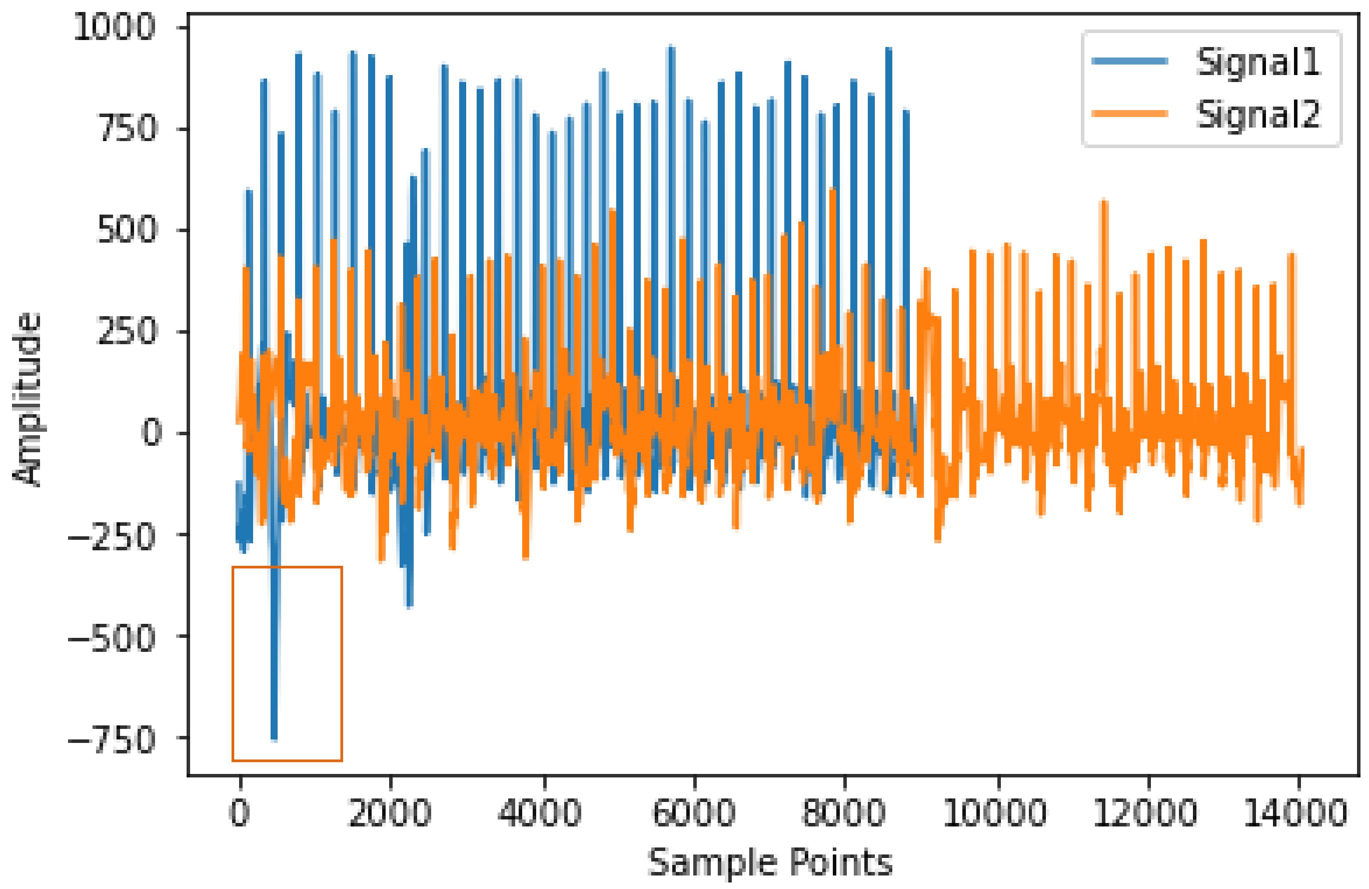
目标：本次Project针对心电图信号作为输入，通过机器学习的方法构建完备的房颤诊断系统

数据：经过固定时间采样的离散心电图信号，长度不固定

方法：1.基于特征提取的方法进行模型驱动的机器学习；2.基于神经网络的数据驱动的深度学习

数据预处理

数据预处理



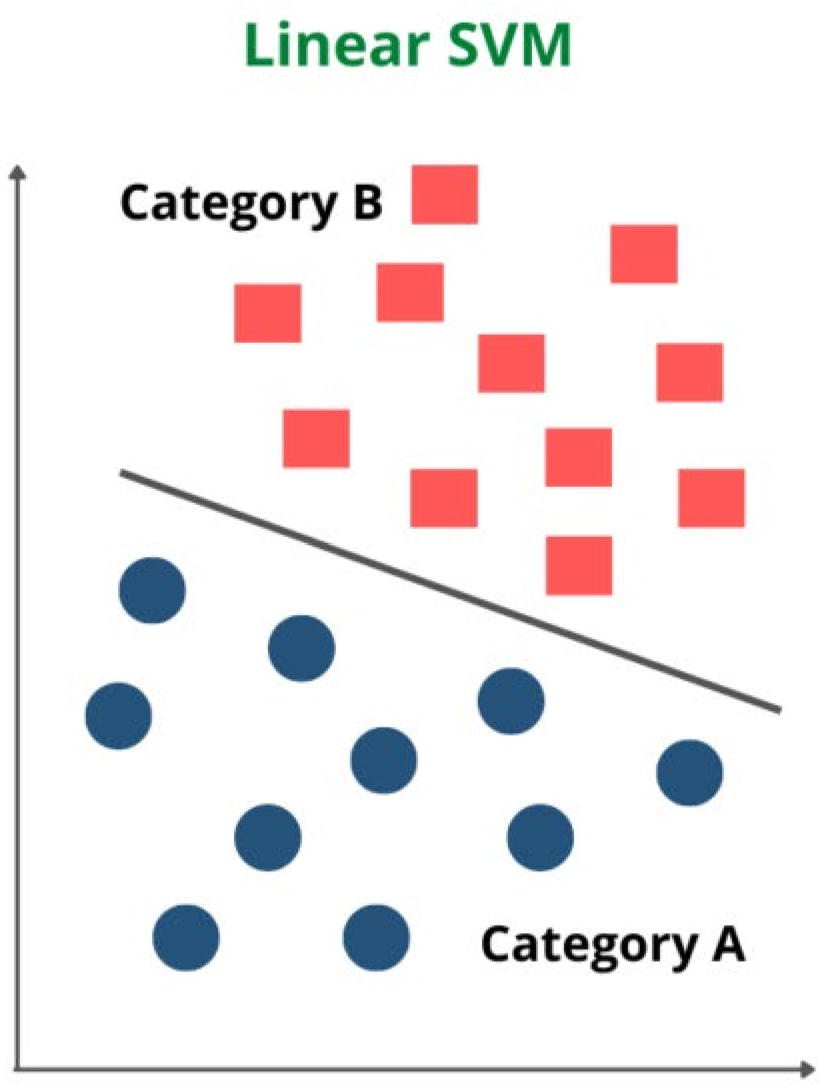
数据特点：

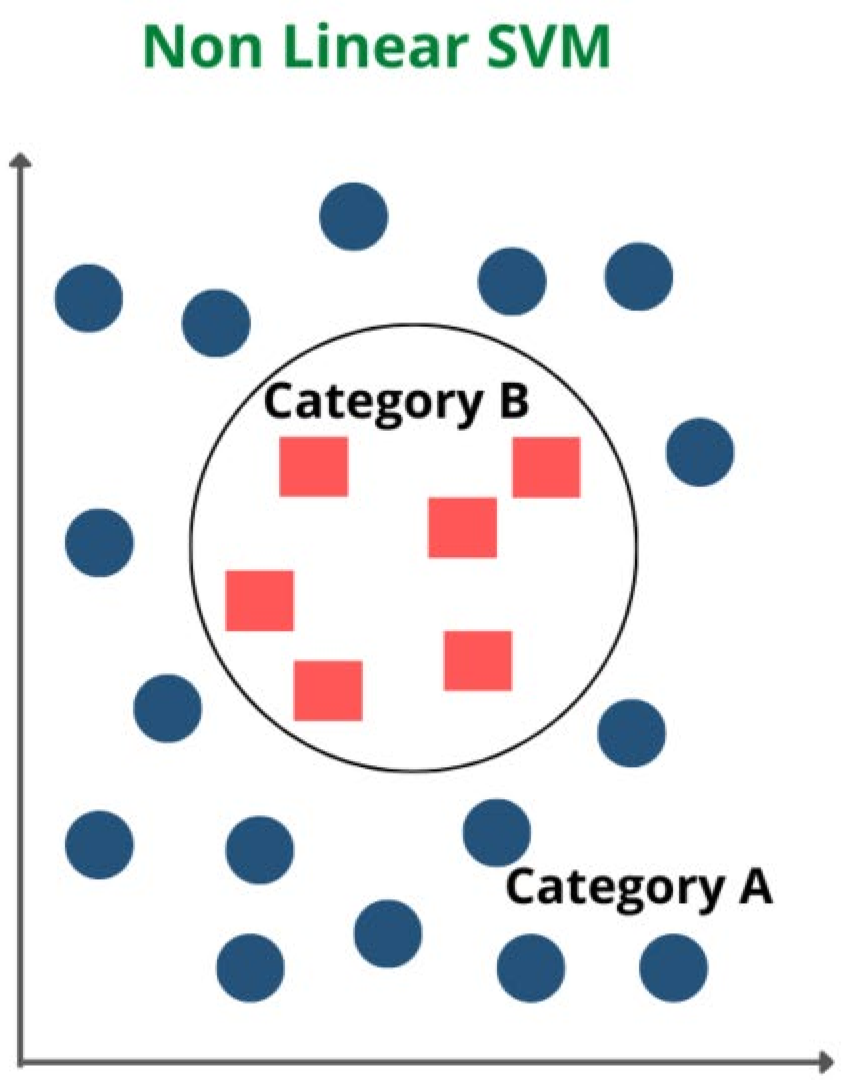
1. 不同信号，长度不一致
2. 不同信号，幅值差异较大
3. 存在离群值，干扰结果
4. 采样率为300Hz，数据长度较大

任务1：数据预处理

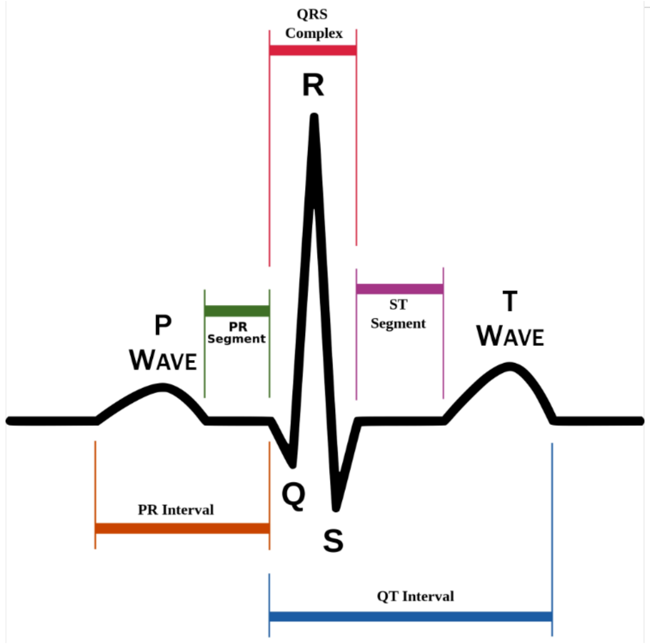
基于特征提取的模型驱动

特征提取

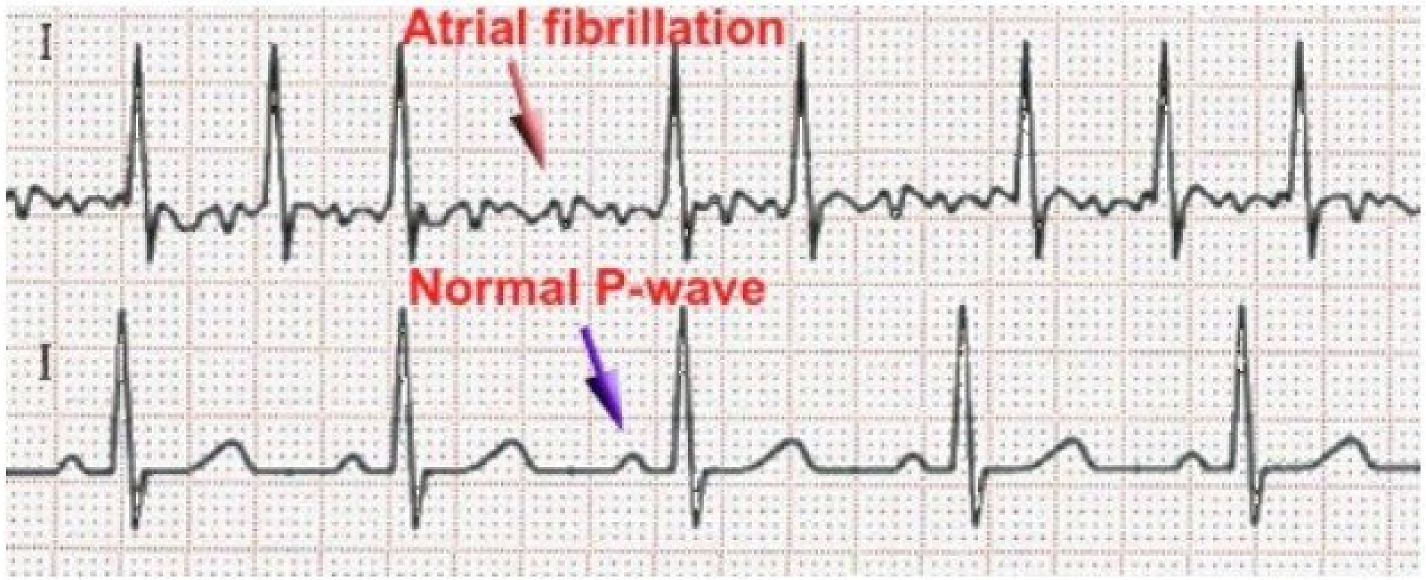

Linear SVM
Category B
Category A


Non Linear SVM
Category B
Category A

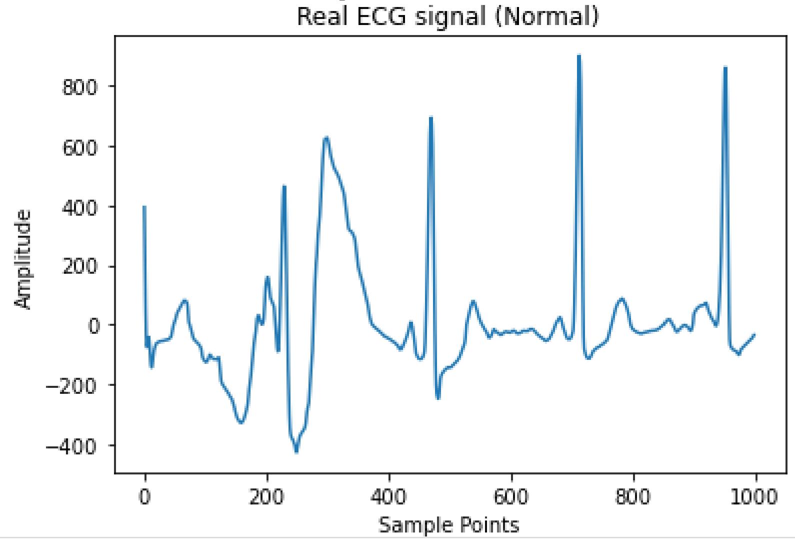
特征提取



难点一：特征选择



难点二：特征提取



1. 分析能够决定ECG信号的一些可提取特征：
Tips：①P 波振幅②QRS波振幅③T 波振幅④PR间期⑤QRS间期⑥QT间期⑦ST间期⑧PR段水平⑨ST段水平⑩RR间期…小波变换等

2. 理论上特征是越多越好，但是需要显式从离散信号中解耦出

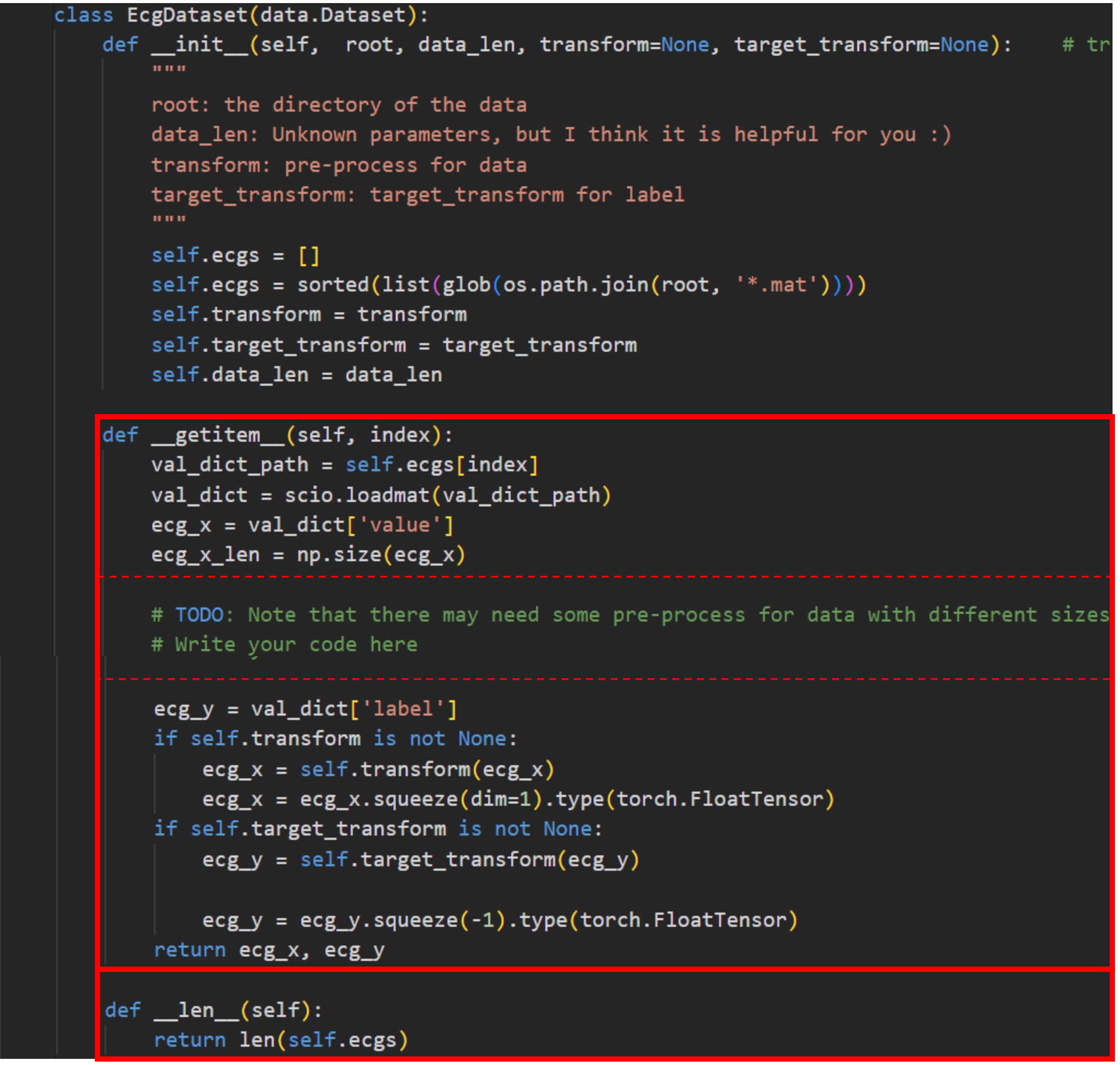
3.此部分数据基于上部分所分离的训练、测试数据

4. 此部分可使用任何模型，包括但不限于SVM、KNN、逻辑回归、XGBoost

基于数据驱动的深度学习

Outline

1. Dataloader



```python
class EcgDataset(data.Dataset):
    def __init__(self,  root, data_len, transform=None, target_transform=None):    # tr
        """
        root: the directory of the data
        data_len: Unknown parameters, but I think it is helpful for you :)
        transform: pre-process for data
        target_transform: target_transform for label
        """
        self.ecgs = []
        self.ecgs = sorted(list(glob(os.path.join(root, '*.mat'))))
        self.transform = transform
        self.target_transform = target_transform
        self.data_len = data_len

    def __getitem__(self, index):
        val_dict_path = self.ecgs[index]
        val_dict = scio.loadmat(val_dict_path)
        ecg_x = val_dict['value']
        ecg_x_len = np.size(ecg_x)

        # TODO: Note that there may need some pre-process for data with different sizes
        # Write your code here

        ecg_y = val_dict['label']
        if self.transform is not None:
            ecg_x = self.transform(ecg_x)
            ecg_x = ecg_x.squeeze(dim=1).type(torch.FloatTensor)
        if self.target_transform is not None:
            ecg_y = self.target_transform(ecg_y)

            ecg_y = ecg_y.squeeze(-1).type(torch.FloatTensor)
        return ecg_x, ecg_y

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

作用：
读取数据，处理数据，并为模型训练，提供数据及对应标签。

```python
def __getitem__(self, index):
```

1. 输入index，返回数据 ([1, 1, data_len]) 和标签 ([1, 1, 1])

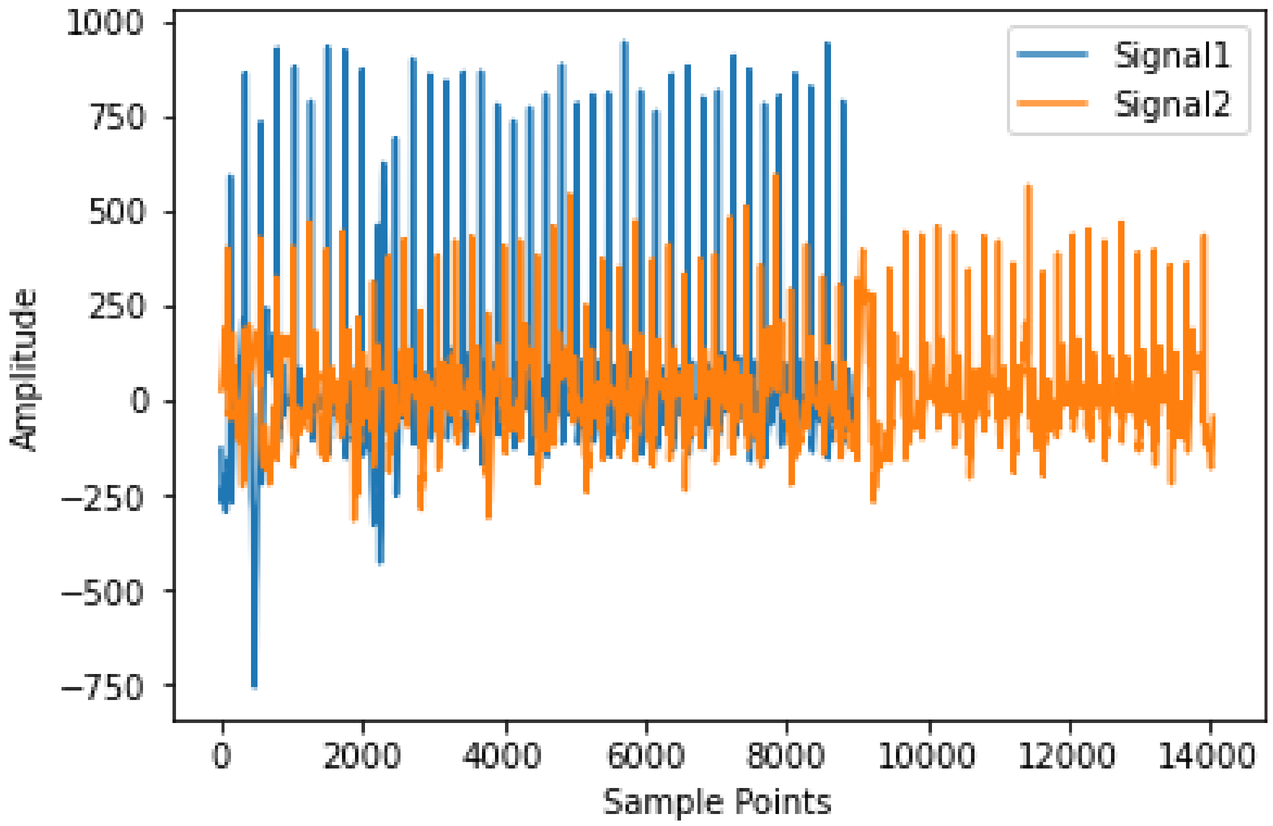
↑

Batch

```python
def __len__(self):
```

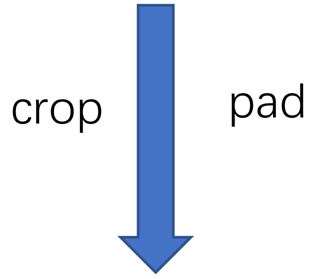
2. 数据集长度

1. Dataloader



直接读取的数据维度:

Signal1: [1, 9000], Signal2: [1, 14054],…,



模型要求输入的维度:

Model_input: [b, 1, n]

Crop:
信号大于n时，选择合适区间裁剪，尽量不包含离群值

Pad:
信号小于n时，对信号进行填充（0，复制，反射等等）

任务2：处理不定长输入问题，跑通代码

2. Model

Multiscaled-CNN(MS-CNN):

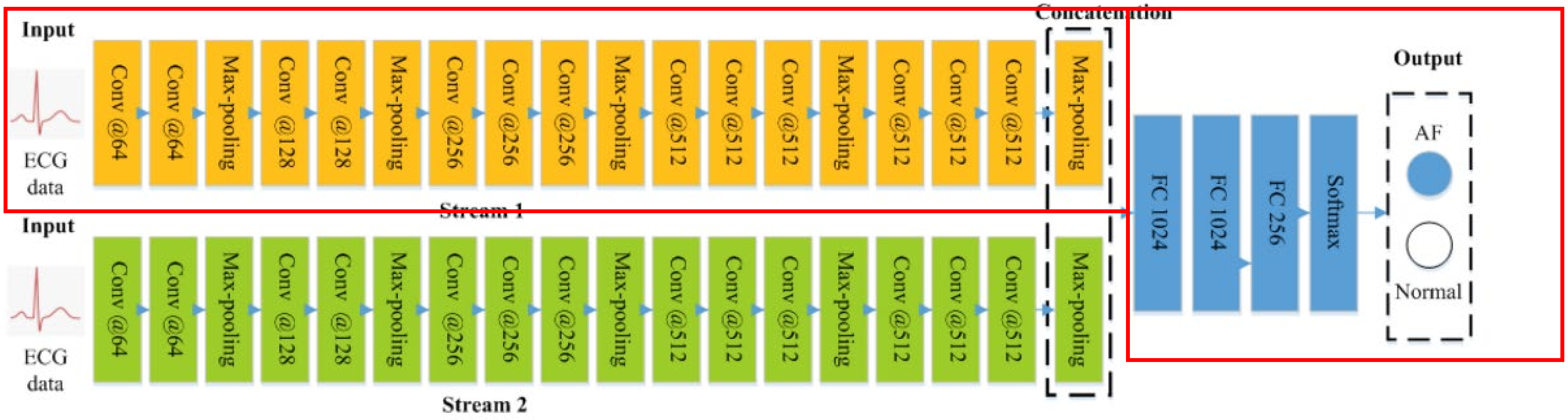


Fig. 2. The network architecture of the MS-CNN. The MS-CNN has two stream networks, which consist of 13 layer convolutional neural network (Conv), 5 maxpooling layers, and 3 fully connected layers (FC).

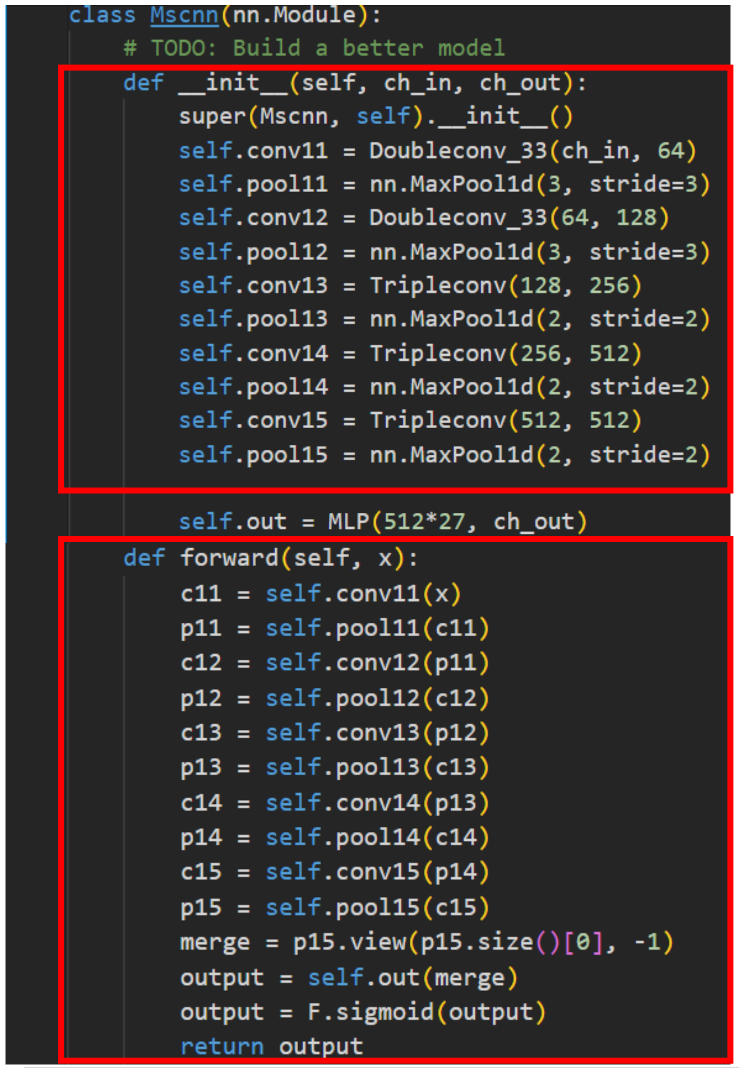
1744 IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, VOL. 22, NO. 6, NOVEMBER 2018

Multiscaled Fusion of Deep Convolutional Neural Networks for Screening Atrial Fibrillation From Single Lead Short ECG Recordings

Xiaomao Fan, Qihang Yao, Yunpeng Cai, Fen Miao, Fangmin Sun, and Ye Li, *Member, IEEE*

2. Model

Multiscaled-CNN(MS-CNN):



```
class Mscnn(nn.Module):
    # TODO: Build a better model
    def __init__(self, ch_in, ch_out):
        super(Mscnn, self).__init__()
        self.conv11 = Doubleconv_33(ch_in, 64)
        self.pool11 = nn.MaxPool1d(3, stride=3)
        self.conv12 = Doubleconv_33(64, 128)
        self.pool12 = nn.MaxPool1d(3, stride=3)
        self.conv13 = Tripleconv(128, 256)
        self.pool13 = nn.MaxPool1d(2, stride=2)
        self.conv14 = Tripleconv(256, 512)
        self.pool14 = nn.MaxPool1d(2, stride=2)
        self.conv15 = Tripleconv(512, 512)
        self.pool15 = nn.MaxPool1d(2, stride=2)

        self.out = MLP(512*27, ch_out)
    def forward(self, x):
        c11 = self.conv11(x)
        p11 = self.pool11(c11)
        c12 = self.conv12(p11)
        p12 = self.pool12(c12)
        c13 = self.conv13(p12)
        p13 = self.pool13(c13)
        c14 = self.conv14(p13)
        p14 = self.pool14(c14)
        c15 = self.conv15(p14)
        p15 = self.pool15(c15)
        merge = p15.view(p15.size()[0], -1)
        output = self.out(merge)
        output = F.sigmoid(output)
        return output
```

初始化函数

forward函数

1. Input x: one dimension ECG signal

[batch_size, channel_num, data_len]
([64, 1, 2400])

2. self.conv11: conv1d
[64, 1, 2400] -> [64, 64, 2396]

3. self.pool11: maxpool1d
[64, 64, 2396] -> [64, 64, 798]

4. self.out: mlp
p15->Merge: [64, 512, 27] -> [64, 13824]
merge->output: [64, 13824] -> [64, 1]

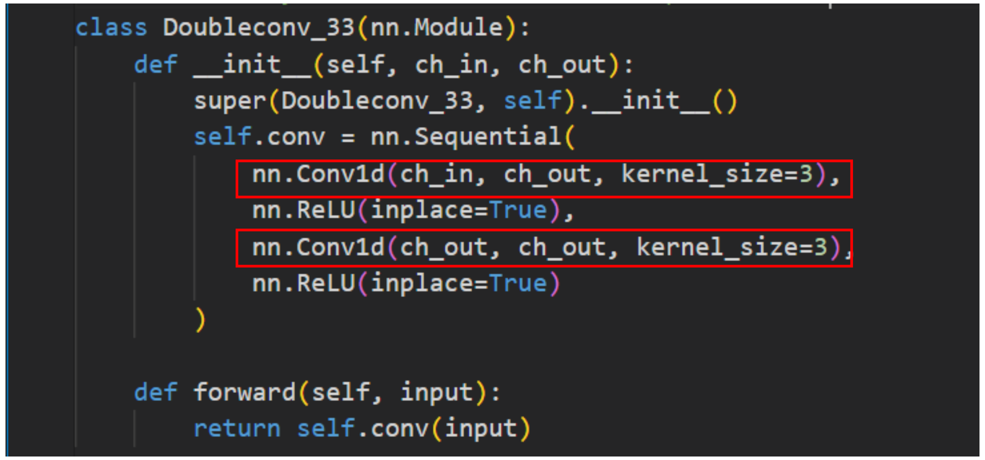
5. F.sigmoid: map to [0,1]

2. Model

SUSTech Southern University of Science and Technology 电子与电气工程系 MedICAL Medical Image Computing and Analysis Lab

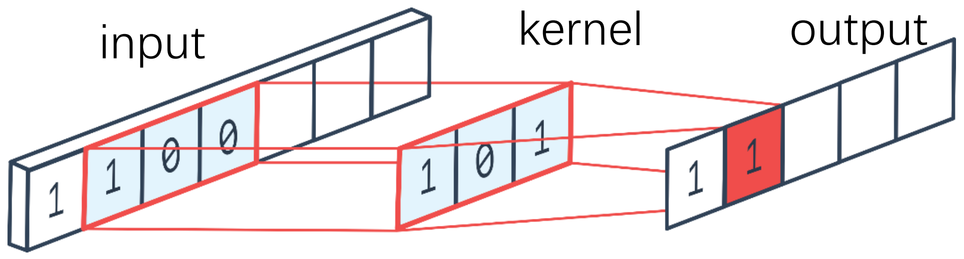
2. self.conv11: conv1d

[64, 1, 2400] -> [64, 64, 2396]



```
class Doubleconv_33(nn.Module):
    def __init__(self, ch_in, ch_out):
        super(Doubleconv_33, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv1d(ch_in, ch_out, kernel_size=3),
            nn.ReLU(inplace=True),
            nn.Conv1d(ch_out, ch_out, kernel_size=3),
            nn.ReLU(inplace=True)
        )

    def forward(self, input):
        return self.conv(input)
```



conv1d

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 both sides of the input. Default: 0
- **padding_mode** (*str, 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`

Shape:

- Input: (N, C_{in}, L_{in}) or (C_{in}, L_{in})
- Output: (N, C_{out}, L_{out}) or (C_{out}, L_{out}), where

$$L_{out} = \left\lfloor \frac{L_{in} + 2 \times \text{padding} - \text{dilation} \times (\text{kernel\_size} - 1) - 1}{\text{stride}} + 1 \right\rfloor$$

dilation和stride值都为1时:

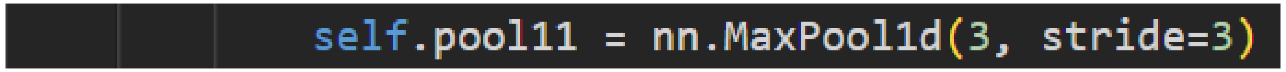
$$L_{\text{out}} = L_{in} + 2 \times padding - kernelsize + 1$$

$$padding = \frac{(kernelsize + 1)}{2}$$

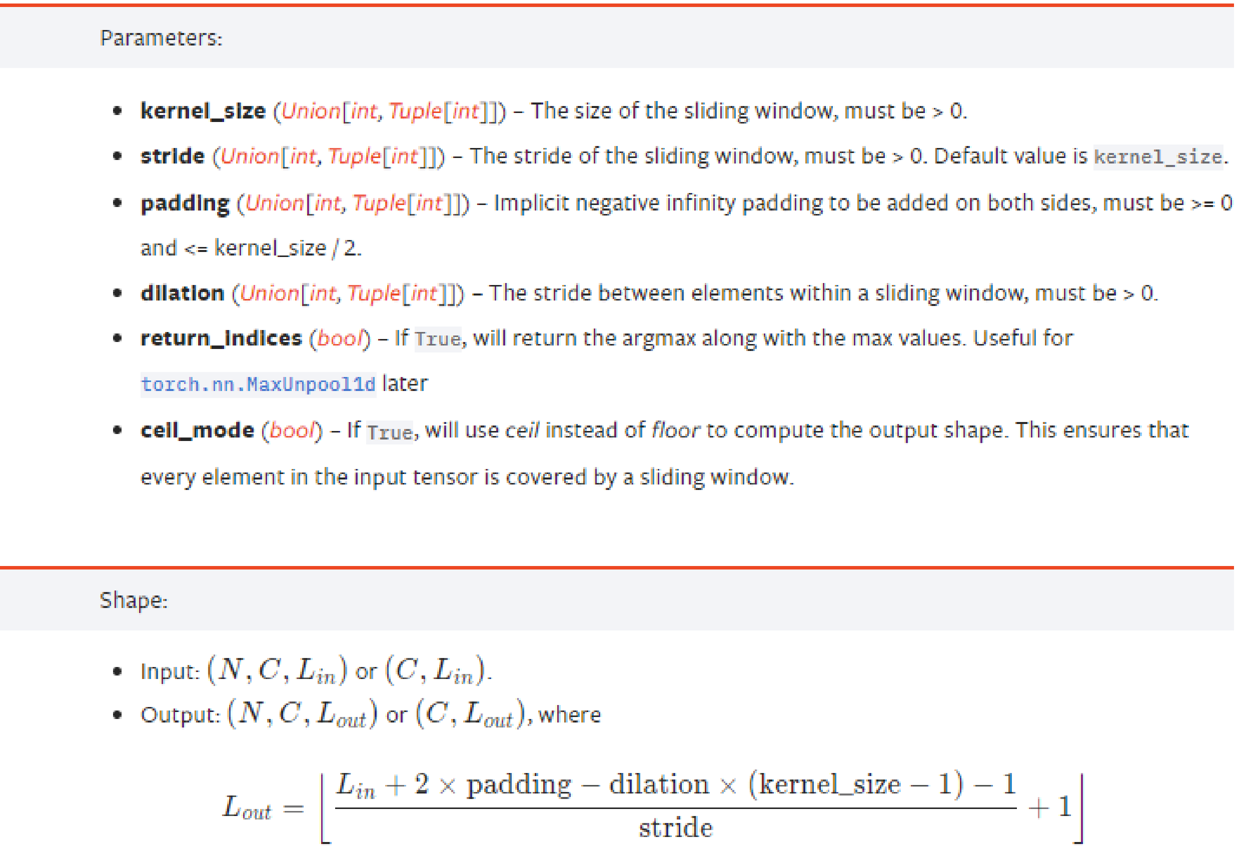
2. Model

3. self.pool11: maxpool1d

[64, 64, 2396] -> [64, 64, 798]



```
self.pool11 = nn.MaxPool1d(3, stride=3)
```



Parameters:

- **kernel_size** (*Union*[*int*, *Tuple*[*int*]]) – The size of the sliding window, must be > 0.
- **stride** (*Union*[*int*, *Tuple*[*int*]]) – The stride of the sliding window, must be > 0. Default value is `kernel_size`.
- **padding** (*Union*[*int*, *Tuple*[*int*]]) – Implicit negative infinity padding to be added on both sides, must be >= 0 and <= kernel_size / 2.
- **dilation** (*Union*[*int*, *Tuple*[*int*]]) – The stride between elements within a sliding window, must be > 0.
- **return_indices** (*bool*) – If `True`, will return the argmax along with the max values. Useful for `torch.nn.MaxUnpool1d` later
- **ceil_mode** (*bool*) – If `True`, will use *ceil* instead of *floor* to compute the output shape. This ensures that every element in the input tensor is covered by a sliding window.

Shape:

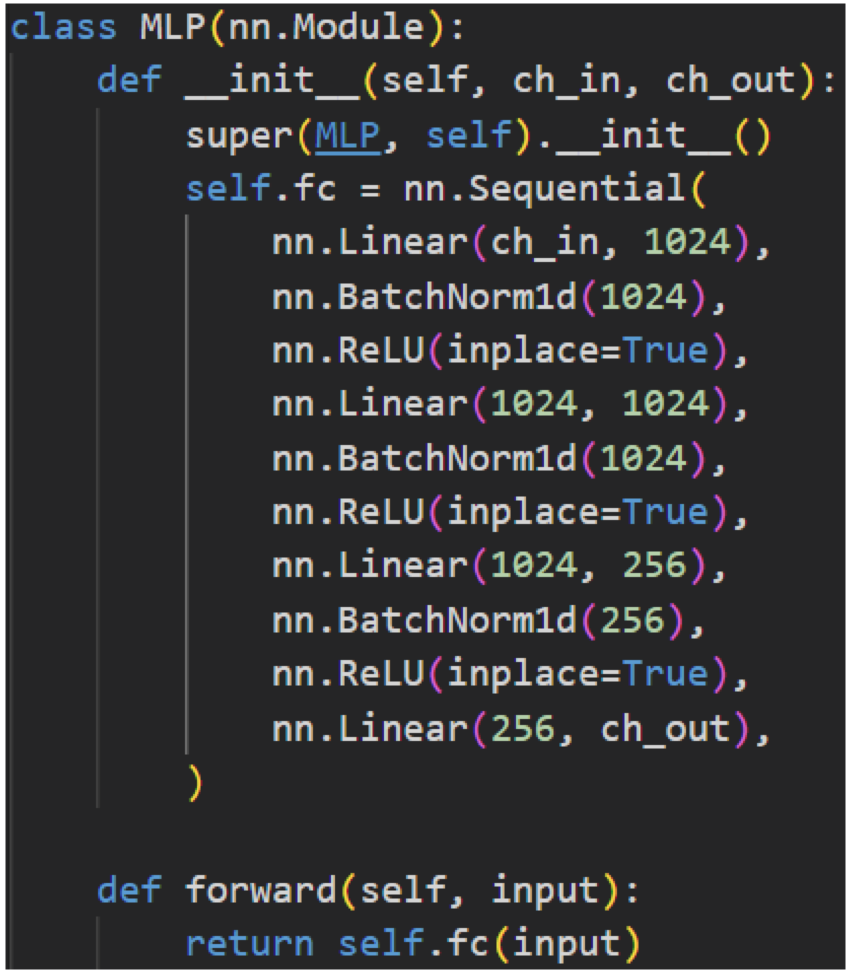
- Input: (N, C, L_{in}) or (C, L_{in}).
- Output: (N, C, L_{out}) or (C, L_{out}), where

$$L_{out} = \left\lfloor \frac{L_{in} + 2 \times \text{padding} - \text{dilation} \times (\text{kernel\_size} - 1) - 1}{\text{stride}} + 1 \right\rfloor$$

2. Model

4. self.out: mlp

p15->Merge: [64, 512, 27] -> [64, 13824]
merge->output: [64, 13824] -> [64, 1]



```
class MLP(nn.Module):
    def __init__(self, ch_in, ch_out):
        super(MLP, self).__init__()
        self.fc = nn.Sequential(
            nn.Linear(ch_in, 1024),
            nn.BatchNorm1d(1024),
            nn.ReLU(inplace=True),
            nn.Linear(1024, 1024),
            nn.BatchNorm1d(1024),
            nn.ReLU(inplace=True),
            nn.Linear(1024, 256),
            nn.BatchNorm1d(256),
            nn.ReLU(inplace=True),
            nn.Linear(256, ch_out),
        )

    def forward(self, input):
        return self.fc(input)
```

[64, 13824] -> [64, 1024]

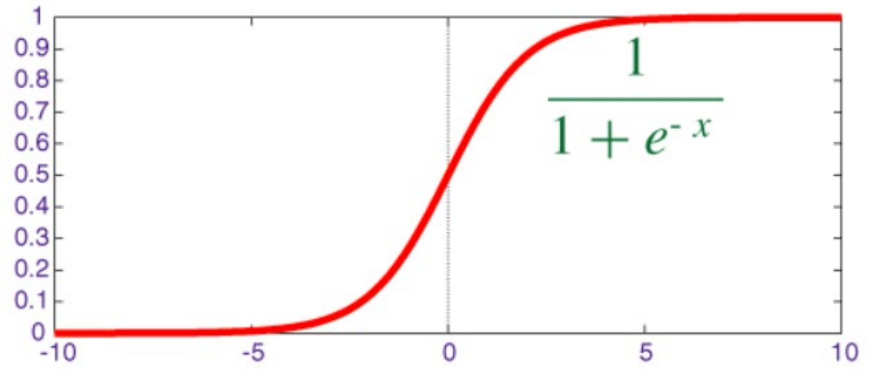
[64, 1024] -> [64, 1024]

[64, 1024] -> [64, 256]

[64, 256] -> [64, 1]

5. F.sigmoid: map to (0,1)

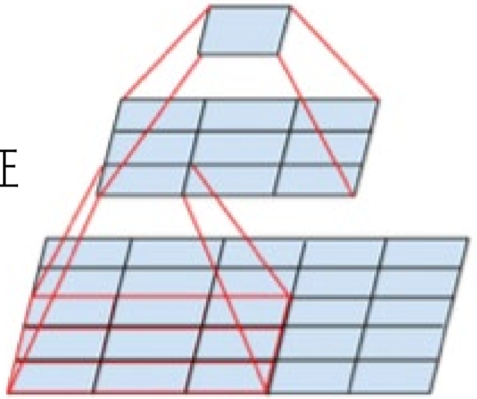
在设定阈值为0.5情况，当输出结果大于0.5时，判定为正常；小于0.5时，判定为异常



2. Model

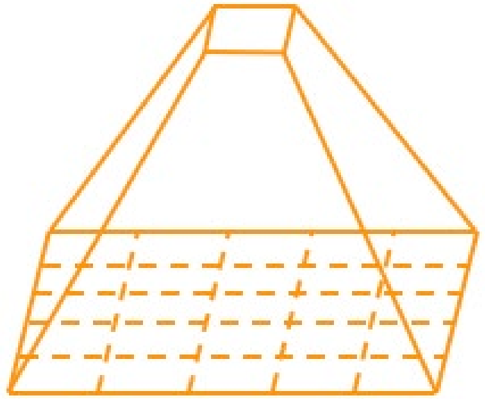
不同大小的卷积核具有不同的感受野

小感受野，
侧重局部特征



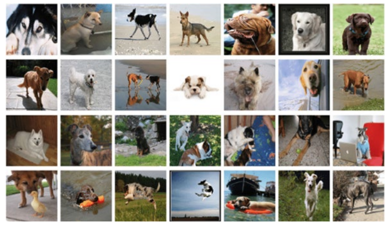
two successive
3x3 convolutions

大感受野，
侧重较全局特征



5x5 convolution

感受野对分类网络影响，
以斑马分类为例：



2. Model

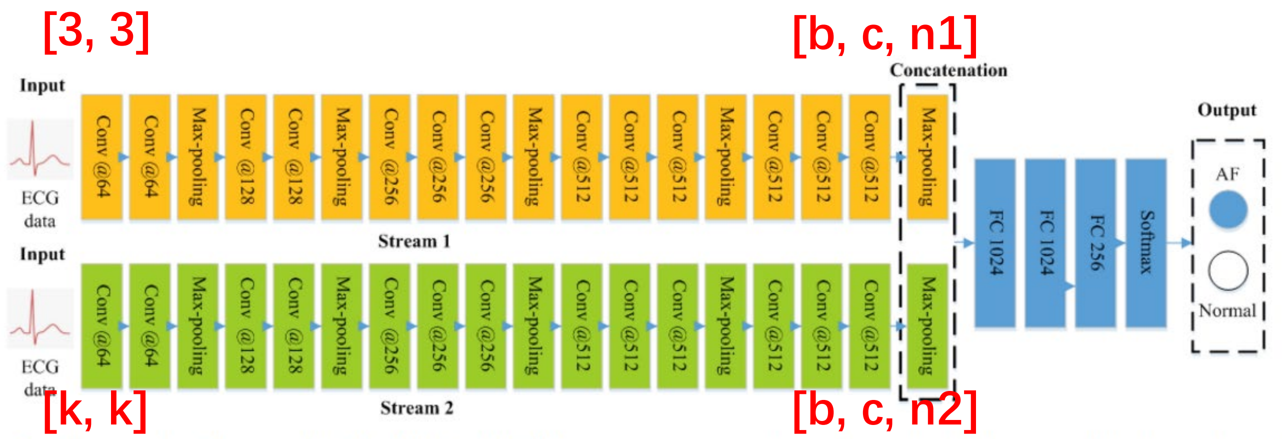


Fig. 2. The network architecture of the MS-CNN. The MS-CNN has two stream networks, which consist of 13 layer convolutional neural network (Conv), 5 maxpooling layers, and 3 fully connected layers (FC).

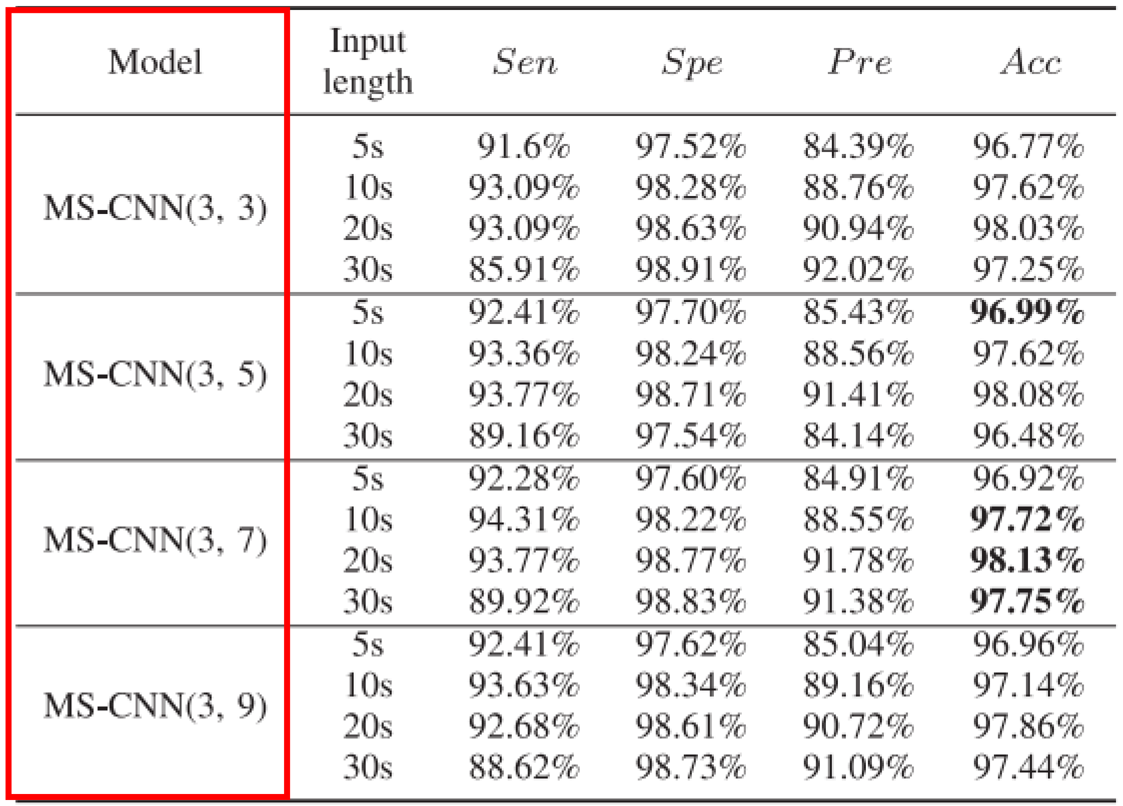
Concatenation：特征融合

Cat([b, c, n1], [b, c, n2], dim=1) --> [b, 2c, n]

n1 = n2

Con1d(kernel_size, **padding**)

TABLE III
CLASSIFICATION PERFORMANCE OF THE MS-CNN



Model	Input length	*Sen*	*Spe*	*Pre*	*Acc*
MS-CNN(3, 3)	5s	91.6%	97.52%	84.39%	96.77%
	10s	93.09%	98.28%	88.76%	97.62%
	20s	93.09%	98.63%	90.94%	98.03%
	30s	85.91%	98.91%	92.02%	97.25%
MS-CNN(3, 5)	5s	92.41%	97.70%	85.43%	**96.99%**
	10s	93.36%	98.24%	88.56%	97.62%
	20s	93.77%	98.71%	91.41%	98.08%
	30s	89.16%	97.54%	84.14%	96.48%
MS-CNN(3, 7)	5s	92.28%	97.60%	84.91%	96.92%
	10s	94.31%	98.22%	88.55%	**97.72%**
	20s	93.77%	98.77%	91.78%	**98.13%**
	30s	89.92%	98.83%	91.38%	**97.75%**
MS-CNN(3, 9)	5s	92.41%	97.62%	85.04%	96.96%
	10s	93.63%	98.34%	89.16%	97.14%
	20s	92.68%	98.61%	90.72%	97.86%
	30s	88.62%	98.73%	91.09%	97.44%

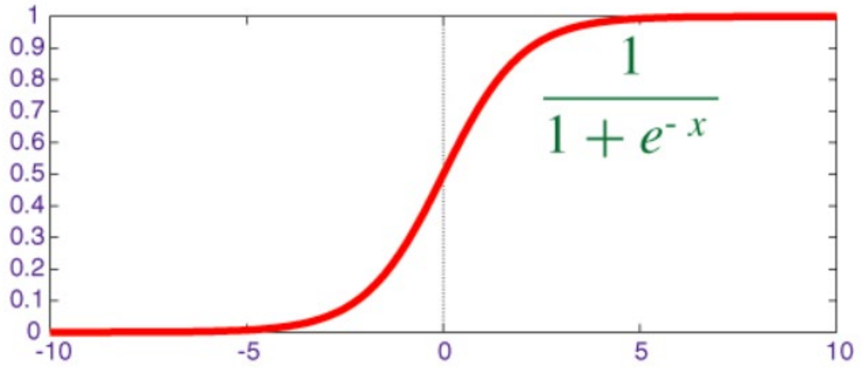
任务3：完成stream2，观察多尺度卷积融合对结果的影响

3. Loss

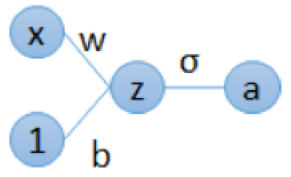
1. MSE Loss

```
criterion = torch.nn.MSELoss()
```

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$



假设给定输入为x，label为y，其中y的取值为0或者1，是一个分类问题。我们要训练一个最简单的Logistic Regression来学习一个函数f(x)使得它能较好的拟合label，如下图所示。



其中 $z(x) = w * x + b$ ， $a(z) = \sigma(z) = \frac{1}{1+e^{-z}}$ 。

最小均方误差，MSE（Mean Squared Error）Loss

$$L_{mse} = \frac{1}{2}(a-y)^2$$

$$\frac{\partial L_{mse}}{\partial w} = \frac{\partial L}{\partial a} * \frac{\partial a}{\partial z} * \frac{\partial z}{\partial w} = (a-y) * \boxed{\sigma^{'}(z)} * x$$

当真实值为1，预测值为0时，sigmoid导数接近0，参数权重更新慢

3. Loss

2. Cross Entrop Loss

衡量估计模型与真实概率分布之间差异情况

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i)$$

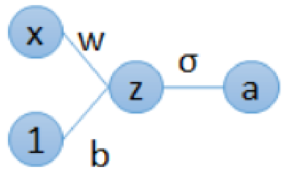
m: 类别个数
y_i: 真实标签
$\hat{y}_i$: 预测概率

Example:

	Normal(1)	Abnormal(0)	CE Loss
Model 1	0.8	0.2	0.223
Model 2	0.6	0.4	0.511

$-1 * \ln 0.8$
$-1 * \ln 0.6$

假设给定输入为x，label为y，其中y的取值为0或者1，是一个分类问题。我们要训练一个最简单的Logistic Regression来学习一个函数f(x)使得它能较好的拟合label，如下图所示。



其中 $z(x) = w * x + b$， $a(z) = \sigma(z) = \frac{1}{1+e^{-z}}$ 。

交叉熵误差CEE（Cross Entropy Error）Loss

$$L_{cee} = -(y * ln(a) + (1-y) * ln(1-a))$$

$$\frac{\partial L_{cee}}{\partial w} = (-\frac{y}{a} + \frac{1-y}{1-a}) * \sigma'(z) * x$$

由于 $\sigma'(z) = \sigma(z) * (1-\sigma(z)) = a * (1-a)$ ，则:

$$\frac{\partial L_{cee}}{\partial w} = (ay - y + a - ay) * x = (a - y) * x$$

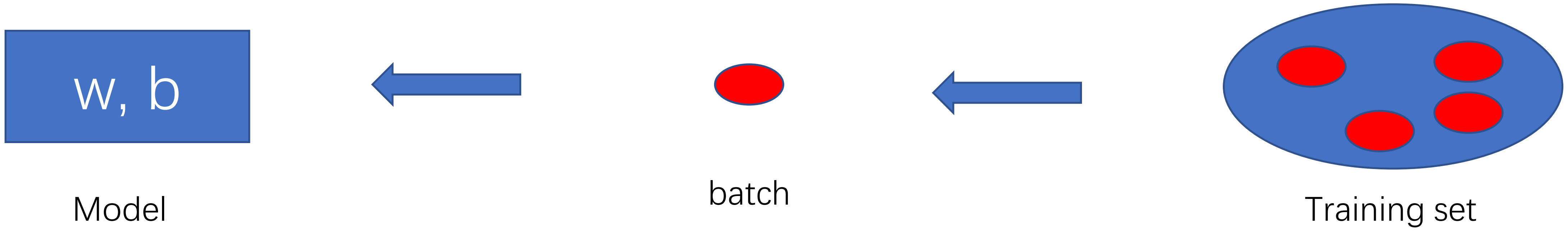
当真实y和预测a距离大时， 梯度大， 更新快。

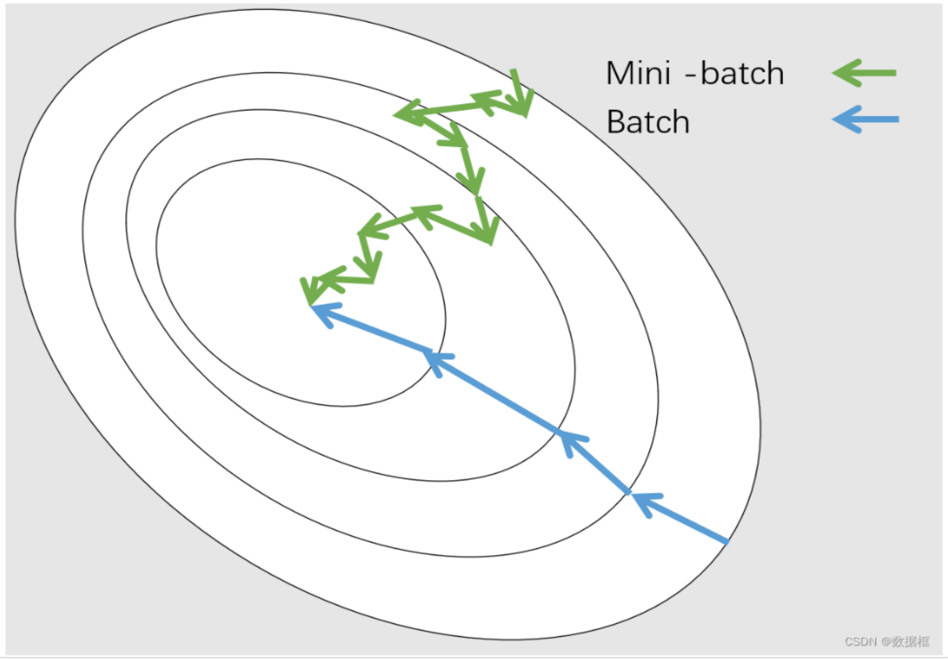
证明参考博客：https://zhuanlan.zhihu.com/p/104130889

任务4：选择更合适的损失函数

4. Hyperparameters

1. Batch_size：1,2,4,8,16,32,64,⋯



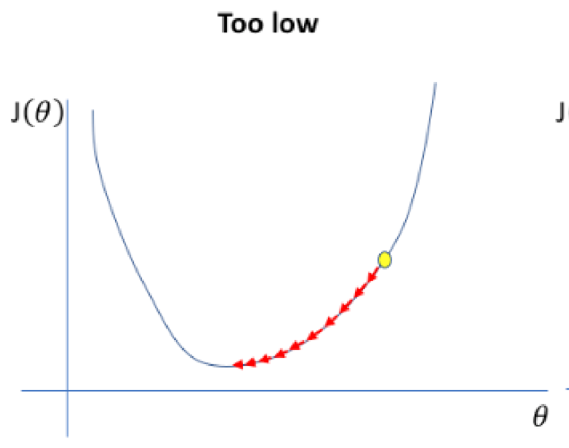


过大：显存不够
过小：难以收敛

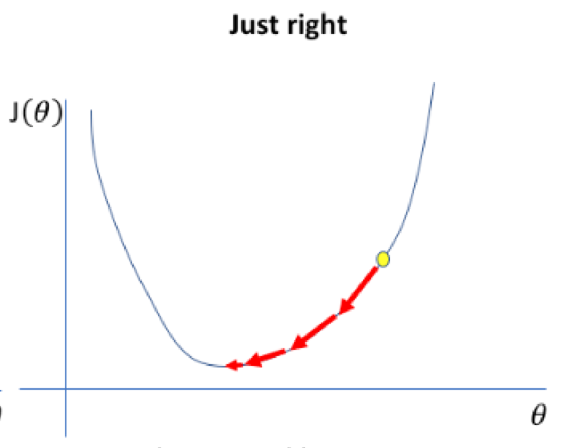
4. Hyperparameters

2.1 Learning rate (lr)

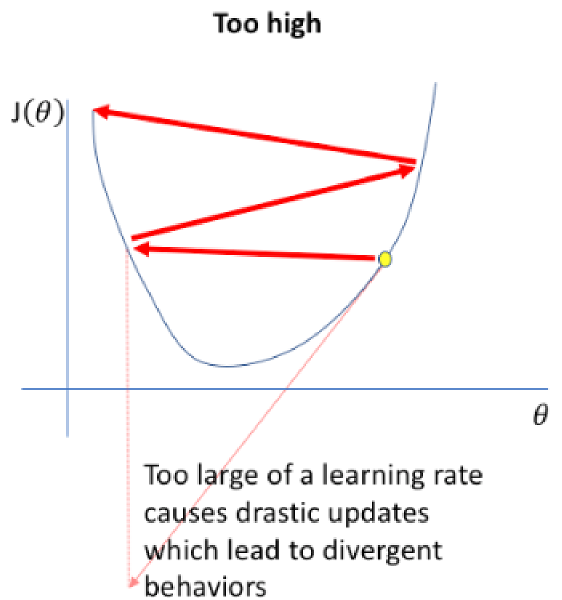
new_*weight* = *old*_weight - learning_rate * gradient



A small learning rate requires many updates before reaching the minimum point



The optimal learning rate swiftly reaches the minimum point

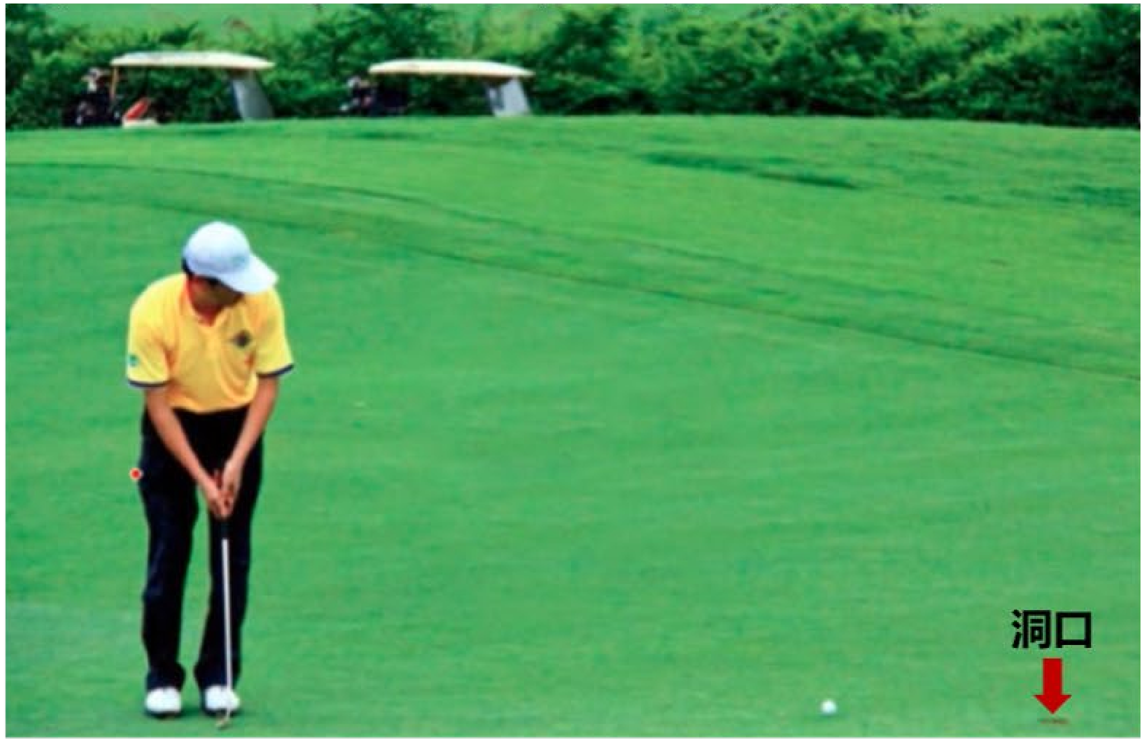


Too large of a learning rate causes drastic updates which lead to divergent behaviors

4. Hyperparameters

2.2 Learning rate schedule

前期 大力击打——大学习率



后期 轻轻击打——小学习率

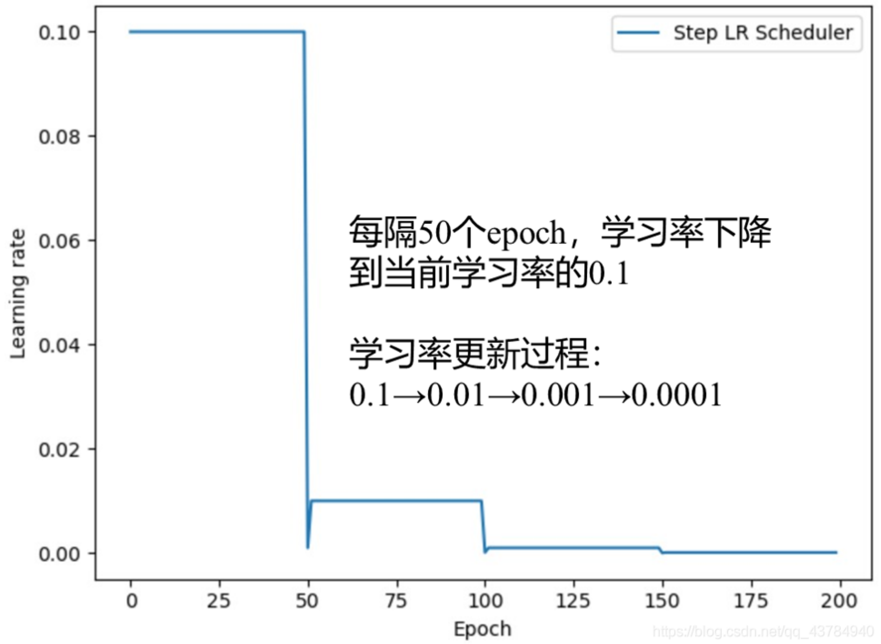
图片来自:

https://www.cnblogs.com/peachtea/p/13532209.html

4. Hyperparameters

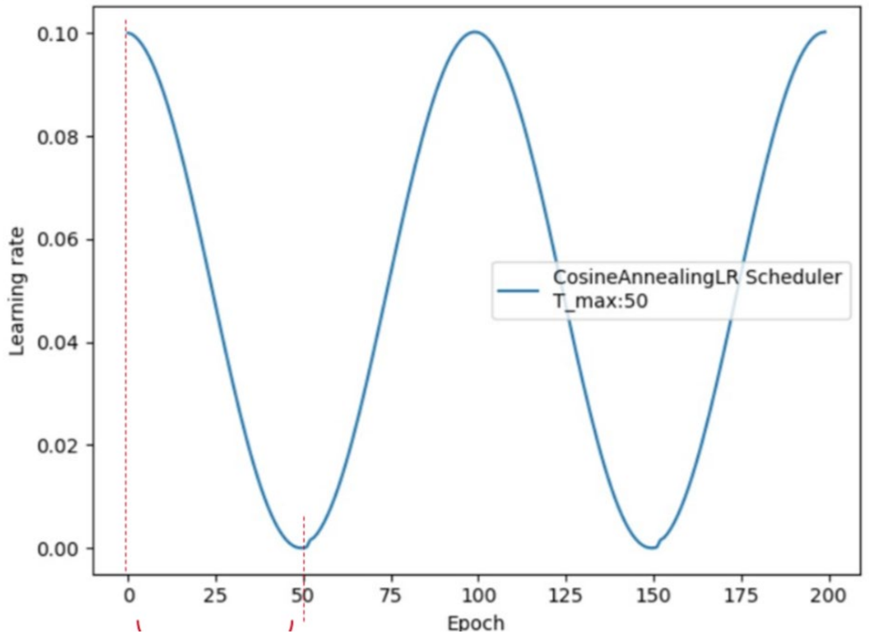
2.2 Learning rate schedule

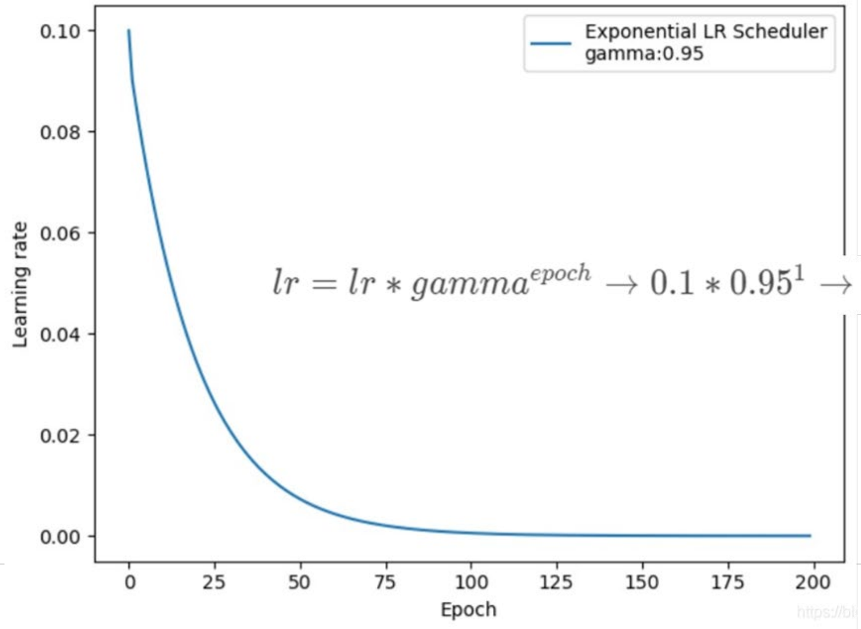
- StepLR:
  等间隔下降
- MultiStepLR:
  自定义间隔下降
- ExponentialLR:
  指数下降
- CosineAnnealingLR:
  余弦周期变换
- ReduceLROnPlateau:
  当loss或指标不变时，自动调整



每隔50个epoch，学习率下降到当前学习率的0.1

学习率更新过程：
0.1→0.01→0.001→0.0001





$$lr = lr * gamma^{epoch} \rightarrow 0.1 * 0.95^1 \rightarrow 0.1 * 0.95^2$$

图片来自：

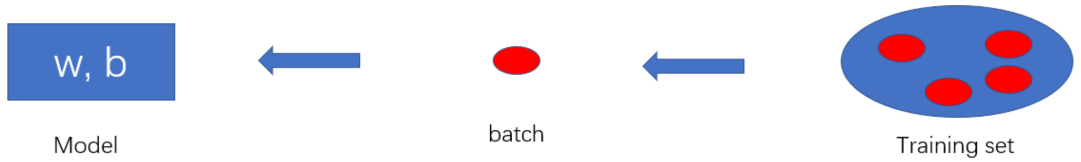
https://www.cnblogs.com/peachtea/p/13532209.html

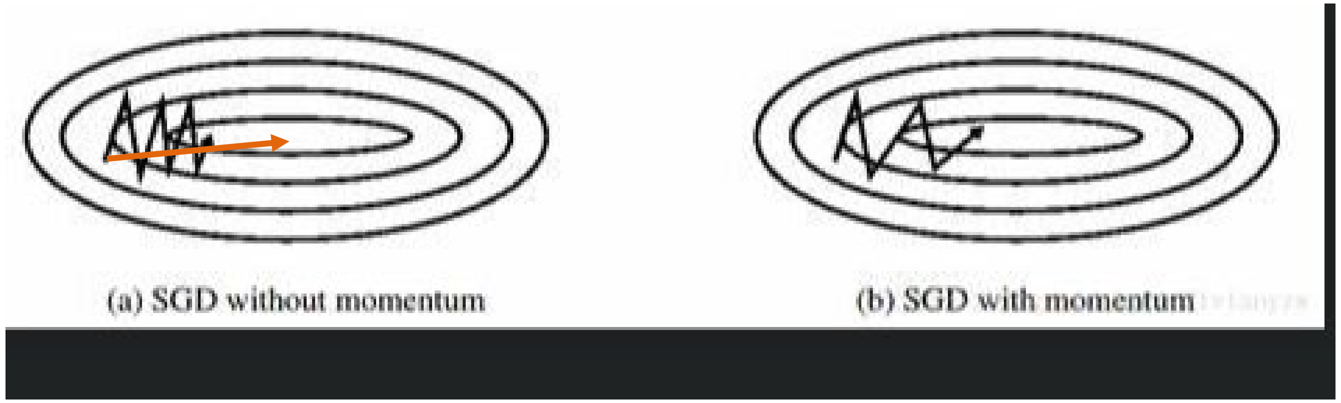
4. Hyperparameters

3. Optimizer

3.1 Stochastic Gradient Descent （SGD）

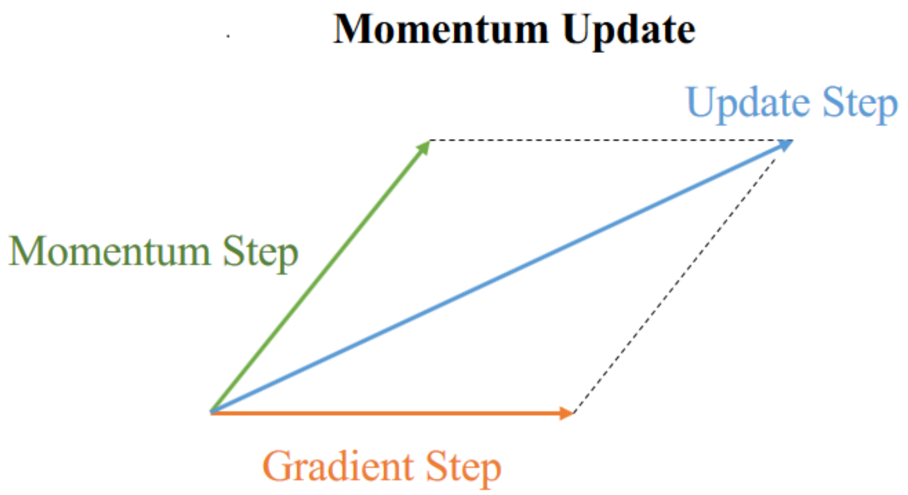
```
optimizer = torch.optim.SGD(model.parameters(), lr=0.001)
```





AdaGrad, Adam等等

https://blog.csdn.net/qq_44614524/article/details/114241259



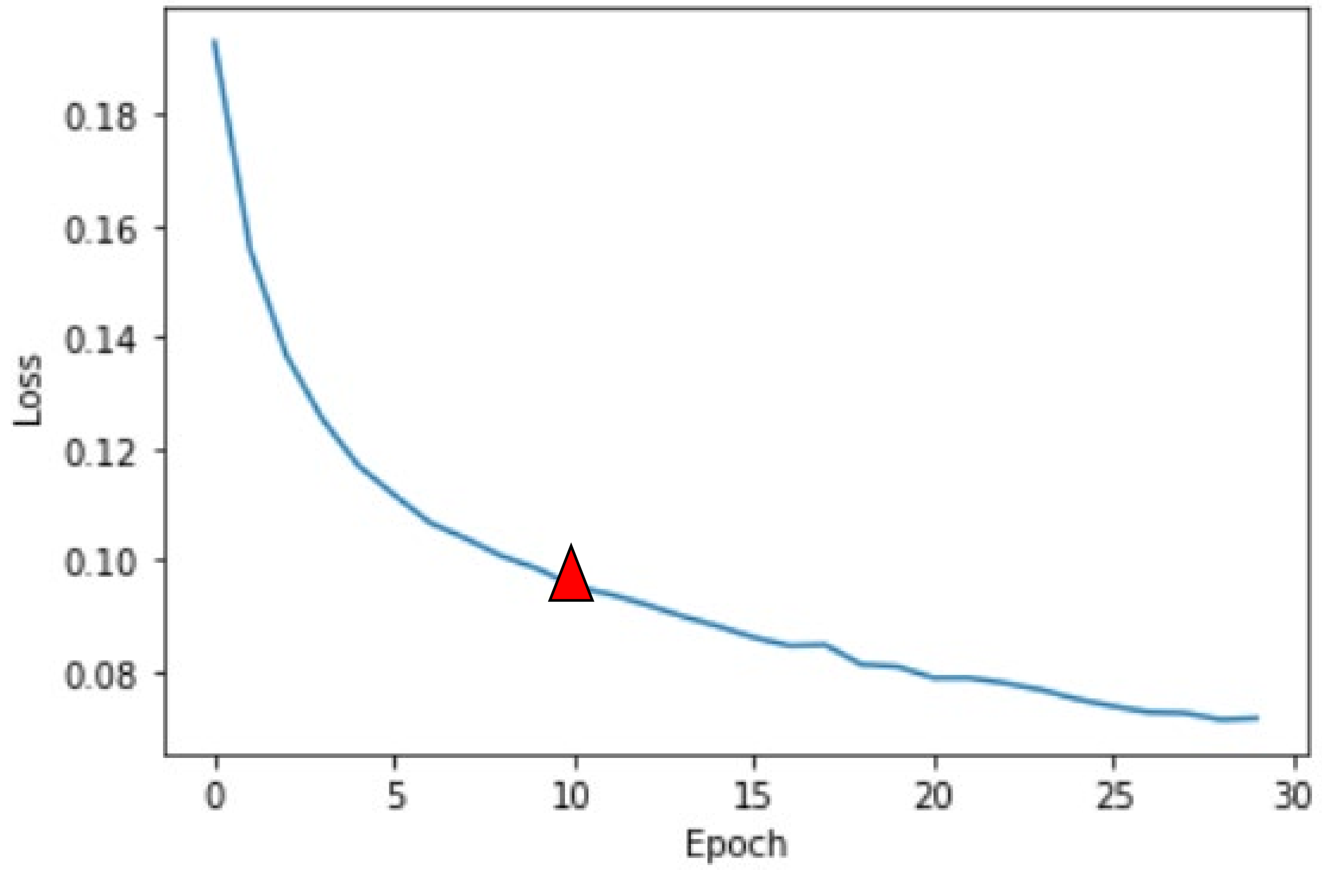
3.2 SGD with momentum

$$v_t = \gamma v_{t-1} + \eta g_t。$$
$$\theta_{t+1} = \theta_t - v_t。$$

4. Hyperparameters

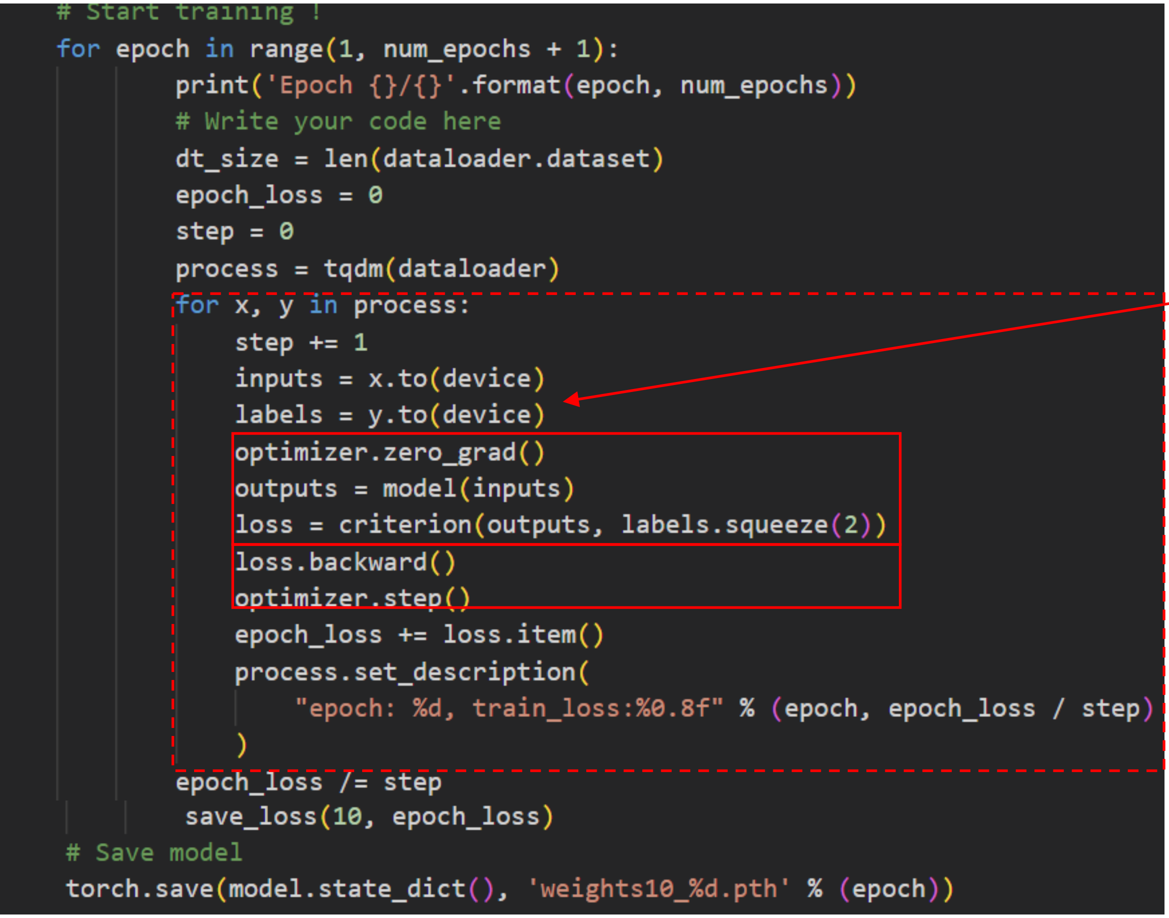
4. Epoch: 训练完所有数据，称为一轮epoch

```
num_epochs = 10
```



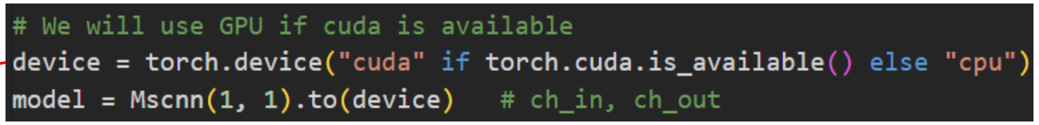
任务5：选择合适的超参数。

5. Training



```
# Start training !
for epoch in range(1, num_epochs + 1):
        print('Epoch {}/{}'.format(epoch, num_epochs))
        # Write your code here
        dt_size = len(dataloader.dataset)
        epoch_loss = 0
        step = 0
        process = tqdm(dataloader)
        for x, y in process:
            step += 1
            inputs = x.to(device)
            labels = y.to(device)
            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, labels.squeeze(2))
            loss.backward()
            optimizer.step()
            epoch_loss += loss.item()
            process.set_description(
                "epoch: %d, train_loss:%0.8f" % (epoch, epoch_loss / step)
            )
        epoch_loss /= step
        save_loss(10, epoch_loss)
# Save model
torch.save(model.state_dict(), 'weights10_%d.pth' % (epoch))
```

1. 加载数据，数据和模型要放在同一个设备上



```
# We will use GPU if cuda is available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = Mscnn(1, 1).to(device)   # ch_in, ch_out
```

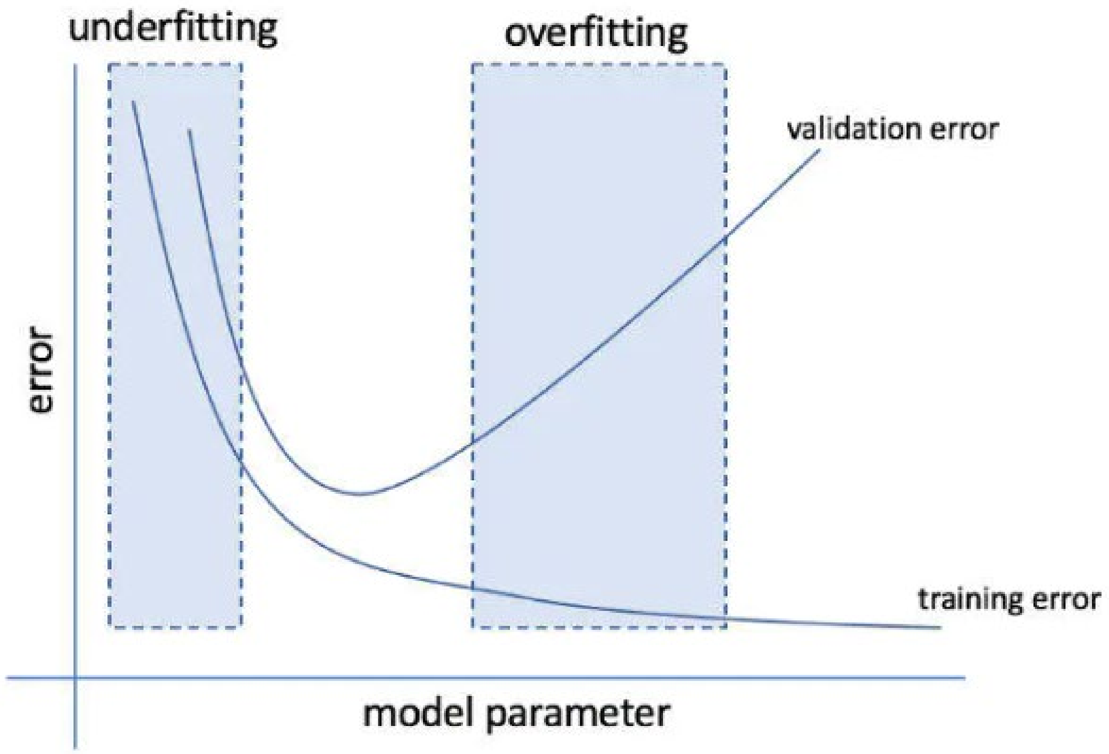
2. 优化器梯度置0，前向传播，计算损失函数

3. 反向传播，更新网络参数

5. Training

问题：

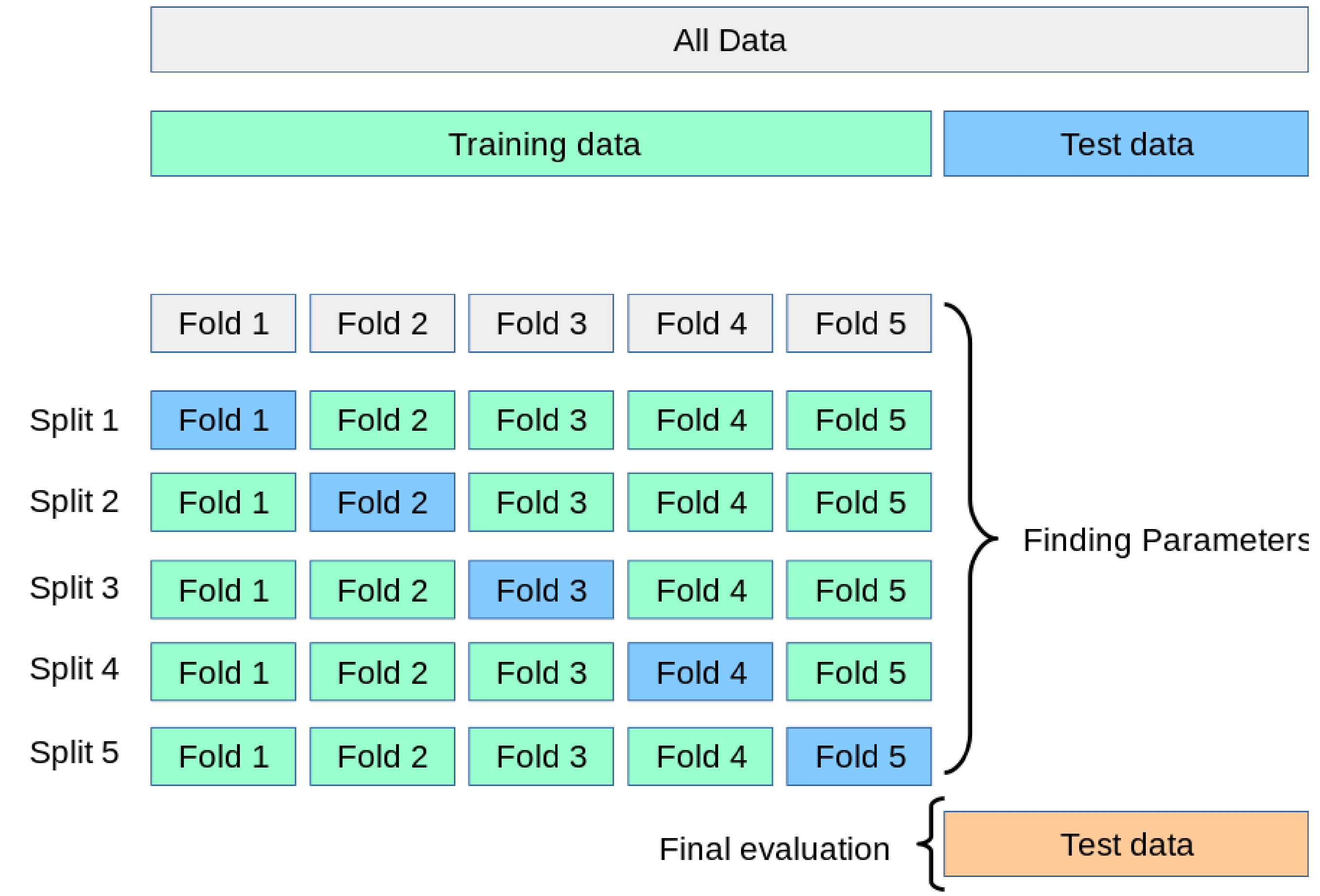
1. 只保留最后一个epoch结果，无法确定最优模型
2. 无法保证模型的泛化能力
3. 可能出现过拟合和欠拟合



5. Training

K-fold cross-validation:

Example: 5-fold cross-validation:



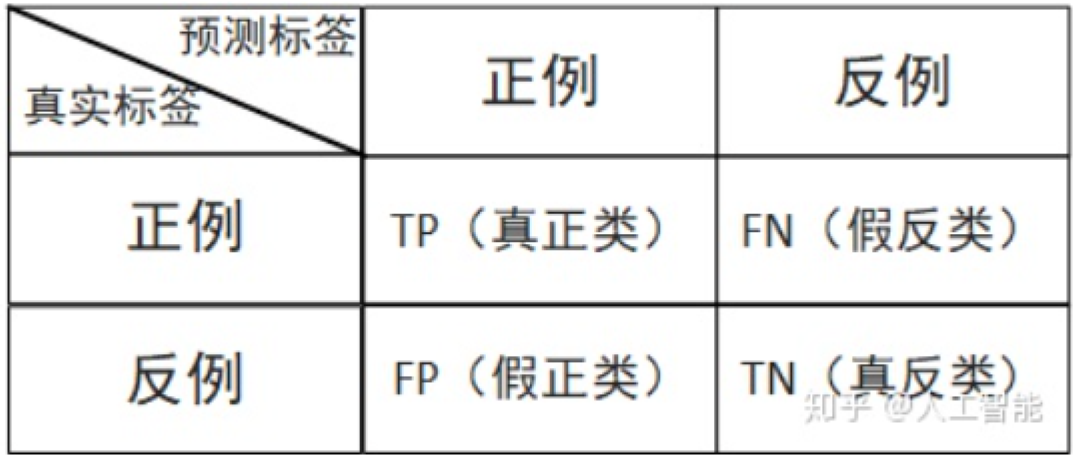
任务6：交叉验证，选择最佳模型（sklearn.model_selection）

6. Evaluation metric

Unbalanced test set:

	Normal	Abnormal
Test set	1570	207

Confuse Matrix:



真实标签 \ 预测标签	正例	反例
正例	TP（真正类）	FN（假反类）
反例	FP（假正类）	TN（真反类）

1. Accuracy (Acc)

$$Accuracy = \frac{n_{correct}}{n_{total}} = \frac{TP+TN}{TP+FN+FP+TN}$$

缺点：不适用于不平衡数据

全预测为正常下：
Acc = 0.883

6. Evaluation metric

2. Precision

$$Precision = \frac{TP}{TP+FP}$$

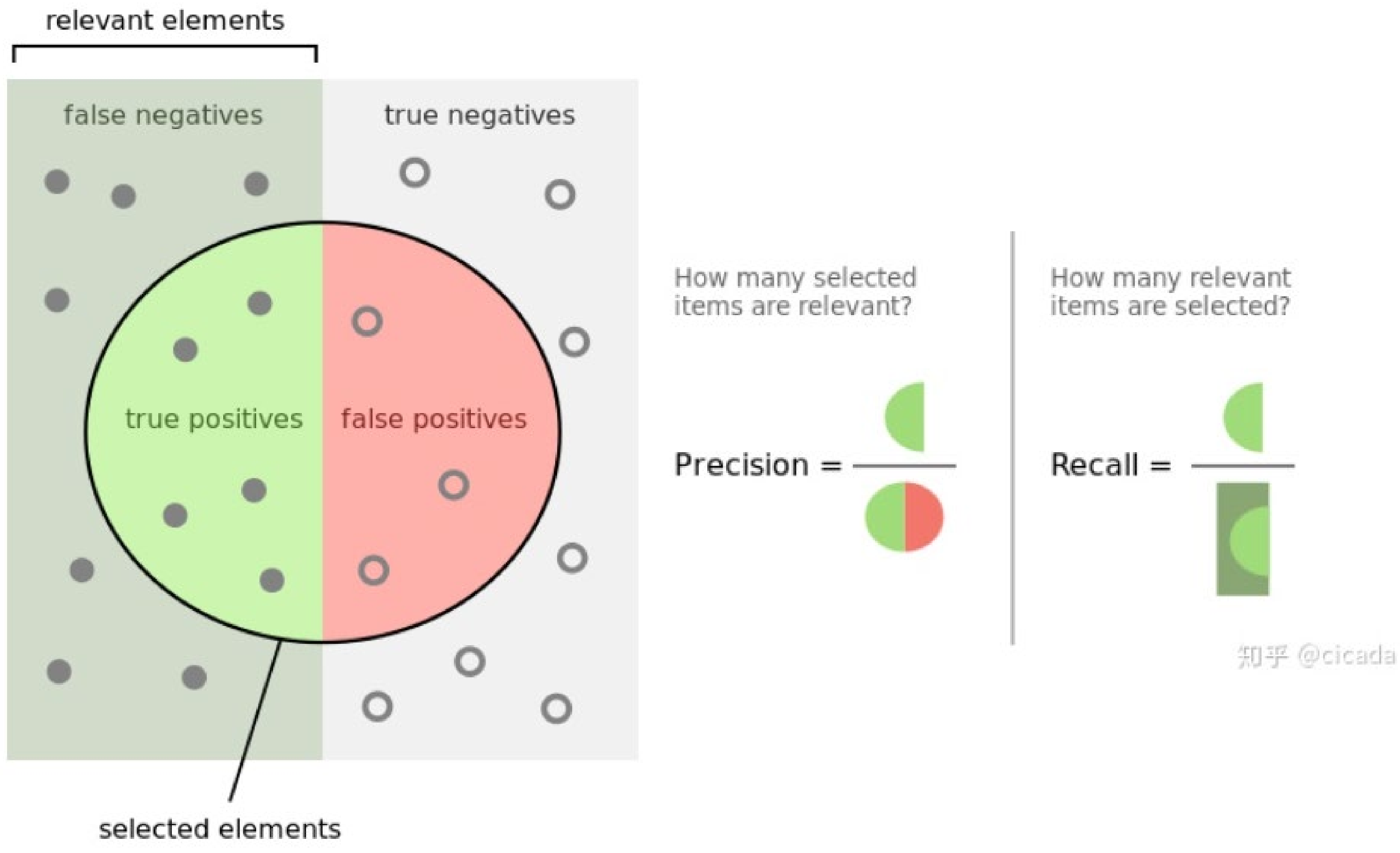
矛盾 ⇕ 统一

3. Recall

$$Recall = \frac{TP}{TP+FN}$$

4. F1 Score

$$F1 = \frac{2*Precision*Recall}{Precision+Recall}$$

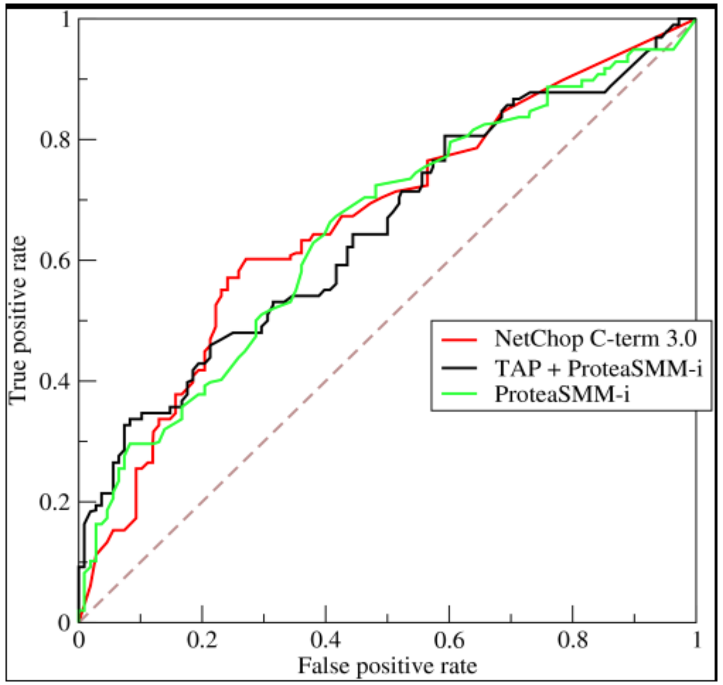


6. Evaluation metric

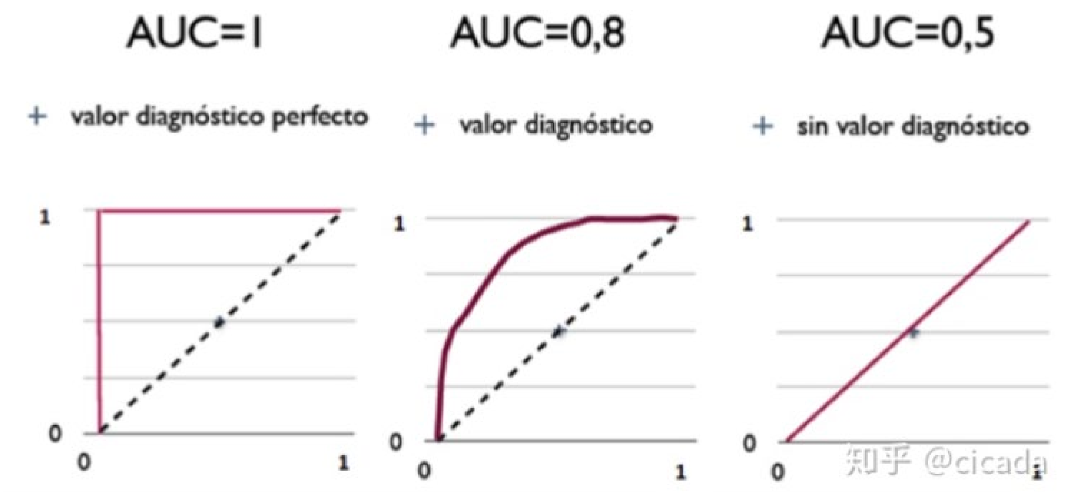
5. Receiver Operating Characteristic Curve (ROC曲线)

$$FPR = \frac{FP}{FP+TN}$$

$$TPR = \frac{TP}{TP+FN}$$



Threshold: 1 --> 0



完美预测

特定阈值下，有预测价值

完全随机

6. Area Under Curve (AUC)

Range: （0.5, 1）

6. Evaluation metric

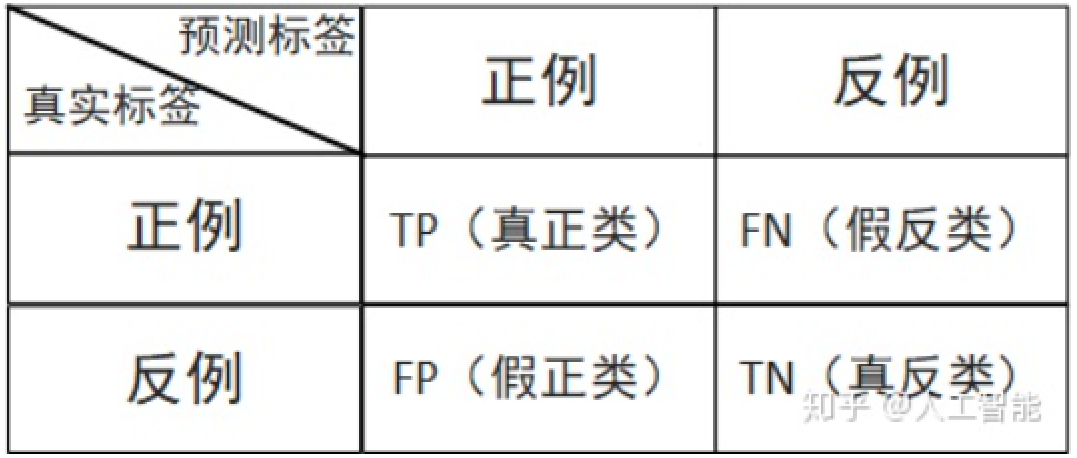
7. Kappa

$$kappa = \frac{p_o - p_e}{1 - p_e}$$

Range:（0, 1）

$$p_o = \frac{\text{对角线元素之和}}{\text{整个矩阵元素之和}}$$ ，其实就是acc。

$$p_e = \frac{\sum_i \text{第}i\text{行元素之和} * \text{第}i\text{列元素之和}}{(\sum \text{矩阵所有元素})^2}$$



预测标签 / 真实标签	正例	反例
正例	TP（真正类）	FN（假反类）
反例	FP（假正类）	TN（真反类）

完全随机： Kappa = 0

0.5	0.5
0.5	0.5

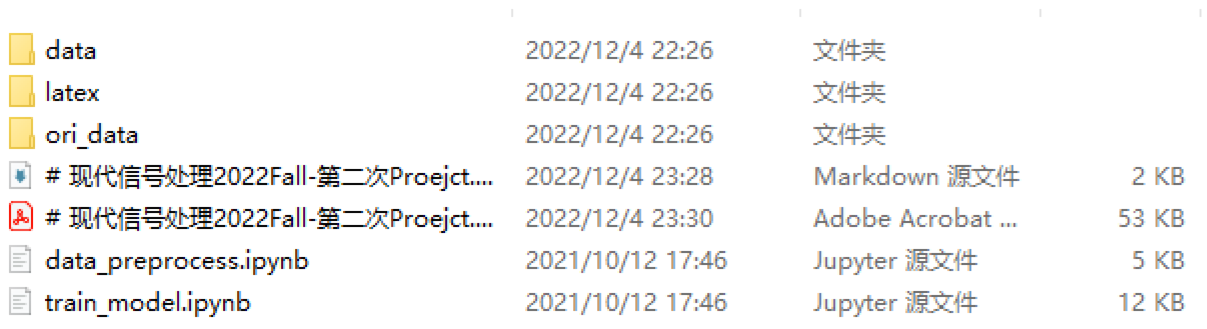
完全一致： Kappa = 1

1	0
0	1

任务7：针对不平衡的测试集，选择合适评价指标（sklearn.metric）

如何开始Project?
——深度学习部分

Demo:

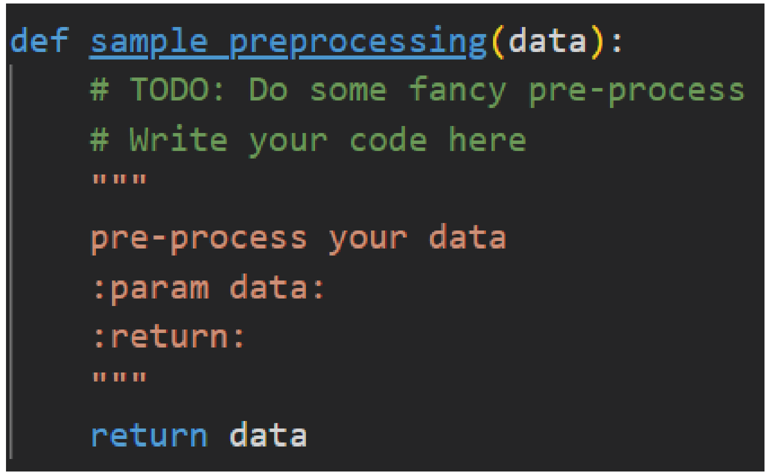


data	2022/12/4 22:26	文件夹	
latex	2022/12/4 22:26	文件夹	
ori_data	2022/12/4 22:26	文件夹	
# 现代信号处理2022Fall-第二次Proejct....	2022/12/4 23:28	Markdown 源文件	2 KB
# 现代信号处理2022Fall-第二次Proejct....	2022/12/4 23:30	Adobe Acrobat ...	53 KB
data_preprocess.ipynb	2021/10/12 17:46	Jupyter 源文件	5 KB
train_model.ipynb	2021/10/12 17:46	Jupyter 源文件	12 KB

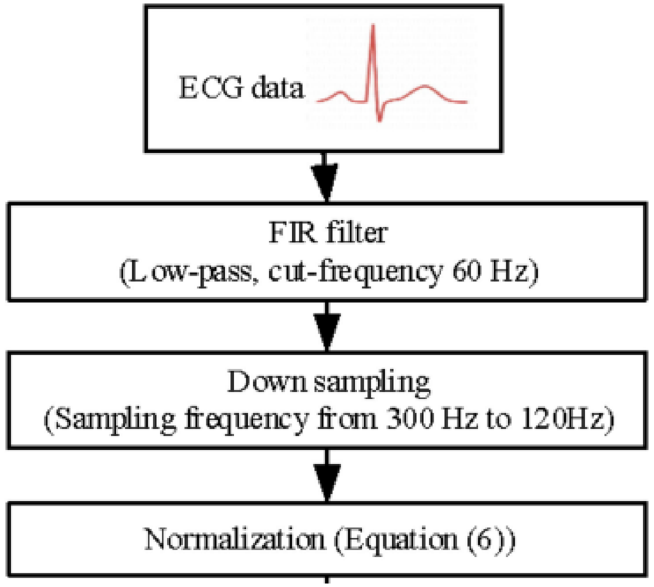
data：预处理后的数据（初始仅包括两个csv文件，用于划分训练测试集）
ori_data：原始数据，包括ECG信号的各种记录
data_preprocess.ipynb：对原始数据进行训练测试集划分，同时需要定义你的预处理方式
train_model.ipynb：定义模型和损失函数，优化器，进行训练并最终验证

1: 数据预处理

运行data_preprocess.ipynb进行数据预处理



```
def sample_preprocessing(data):
    # TODO: Do some fancy pre-process
    # Write your code here
    """
    pre-process your data
    :param data:
    :return:
    """
    return data
```



ECG data
FIR filter
(Low-pass, cut-frequency 60 Hz)
Down sampling
(Sampling frequency from 300 Hz to 120Hz)
Normalization (Equation (6))

?

$$Normalized(X) = \frac{X - \overline{X}}{S} \qquad (6)$$

2: 跑通基础模型

在train_model.ipynb文件，解决数据不定长问题，跑通基础模型。

加分项

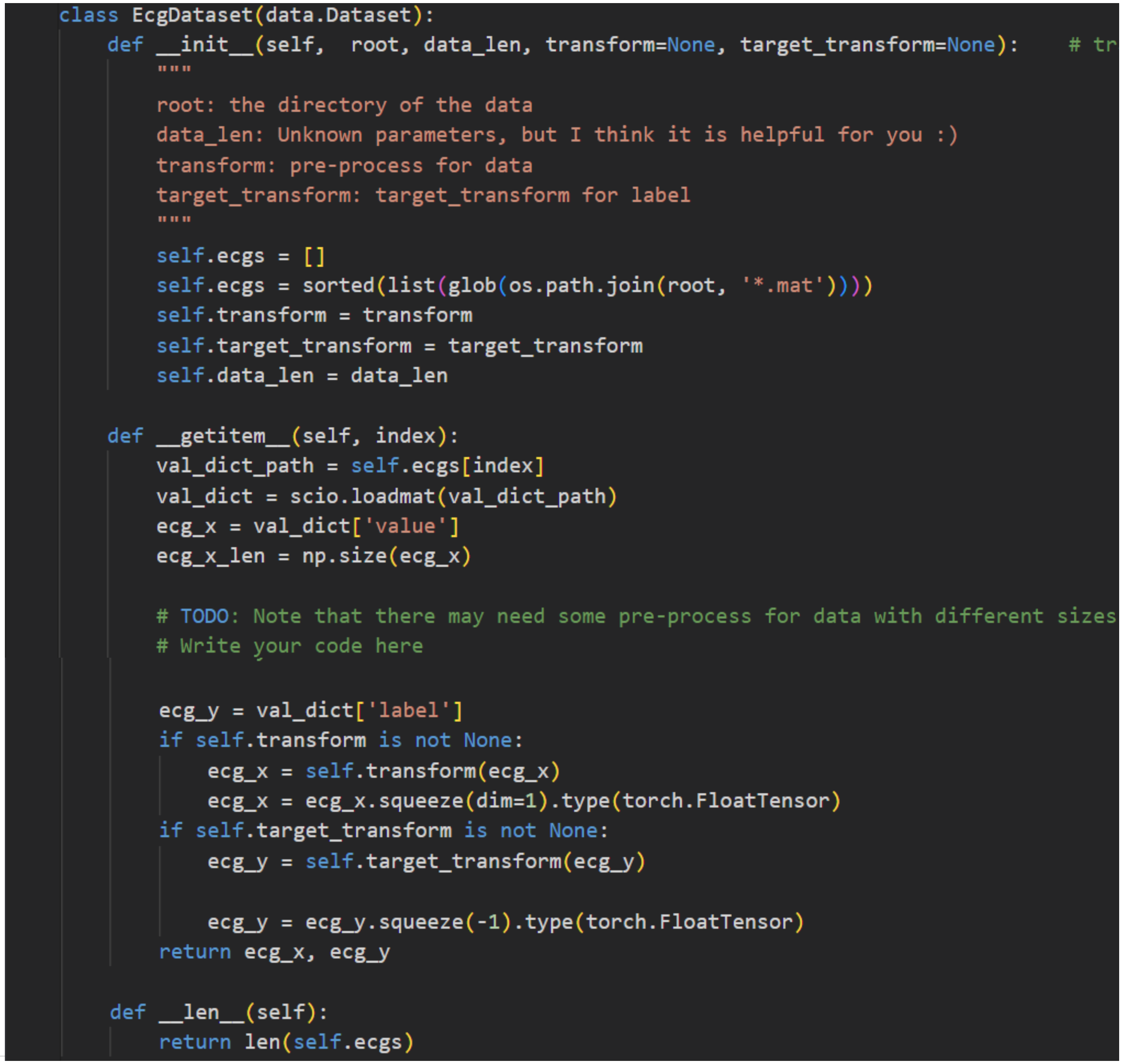
3: 修改网络结构(multi-scaled network)

4: 损失函数的选择

5: 超参数的调整，包括优化器，学习率，epoch，batchsize等等

6: 最优模型的选择

7: 合理的评估指标的选择



```
class EcgDataset(data.Dataset):
    def __init__(self,  root, data_len, transform=None, target_transform=None):    # tr
        """
        root: the directory of the data
        data_len: Unknown parameters, but I think it is helpful for you :)
        transform: pre-process for data
        target_transform: target_transform for label
        """
        self.ecgs = []
        self.ecgs = sorted(list(glob(os.path.join(root, '*.mat'))))
        self.transform = transform
        self.target_transform = target_transform
        self.data_len = data_len

    def __getitem__(self, index):
        val_dict_path = self.ecgs[index]
        val_dict = scio.loadmat(val_dict_path)
        ecg_x = val_dict['value']
        ecg_x_len = np.size(ecg_x)

        # TODO: Note that there may need some pre-process for data with different sizes
        # Write your code here


        ecg_y = val_dict['label']
        if self.transform is not None:
            ecg_x = self.transform(ecg_x)
            ecg_x = ecg_x.squeeze(dim=1).type(torch.FloatTensor)
        if self.target_transform is not None:
            ecg_y = self.target_transform(ecg_y)

            ecg_y = ecg_y.squeeze(-1).type(torch.FloatTensor)
        return ecg_x, ecg_y

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

报告要求

1. 报告要求为IEEE的会议模板，长度不限制，但应当包括Introduction，Methods，Results和Discussion部分。

2. 我们鼓励有着完备的实验对比，以及不足分析。

3. 请遵循学术规范，引用和表格符合基本要求。

4. 请于12.30前将报告和可运行代码发送至 12032946@mail.sustech.edu.cn

思考

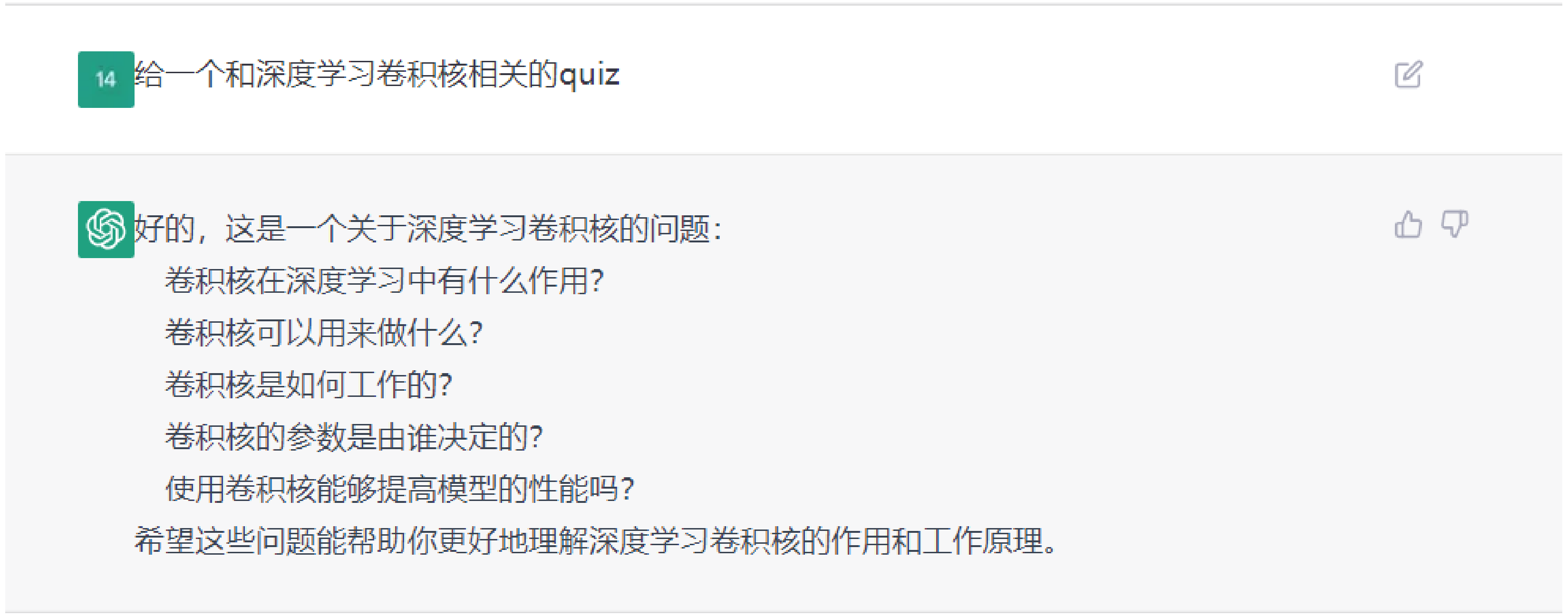
- 数据预处理的作用以及对不同的算法的性能有怎样的影响?
- 从ECG信号中提取出的特征是不是越多越好?
- 模型驱动的特征提取算法和数据驱动的特征提取算法之间相互对比都有哪些优点和缺点?

Q&A

Quiz:

1:

2: 你对这门课project的感受和建议



14 给一个和深度学习卷积核相关的quiz

好的，这是一个关于深度学习卷积核的问题：

卷积核在深度学习中有什么作用?

卷积核可以用来做什么?

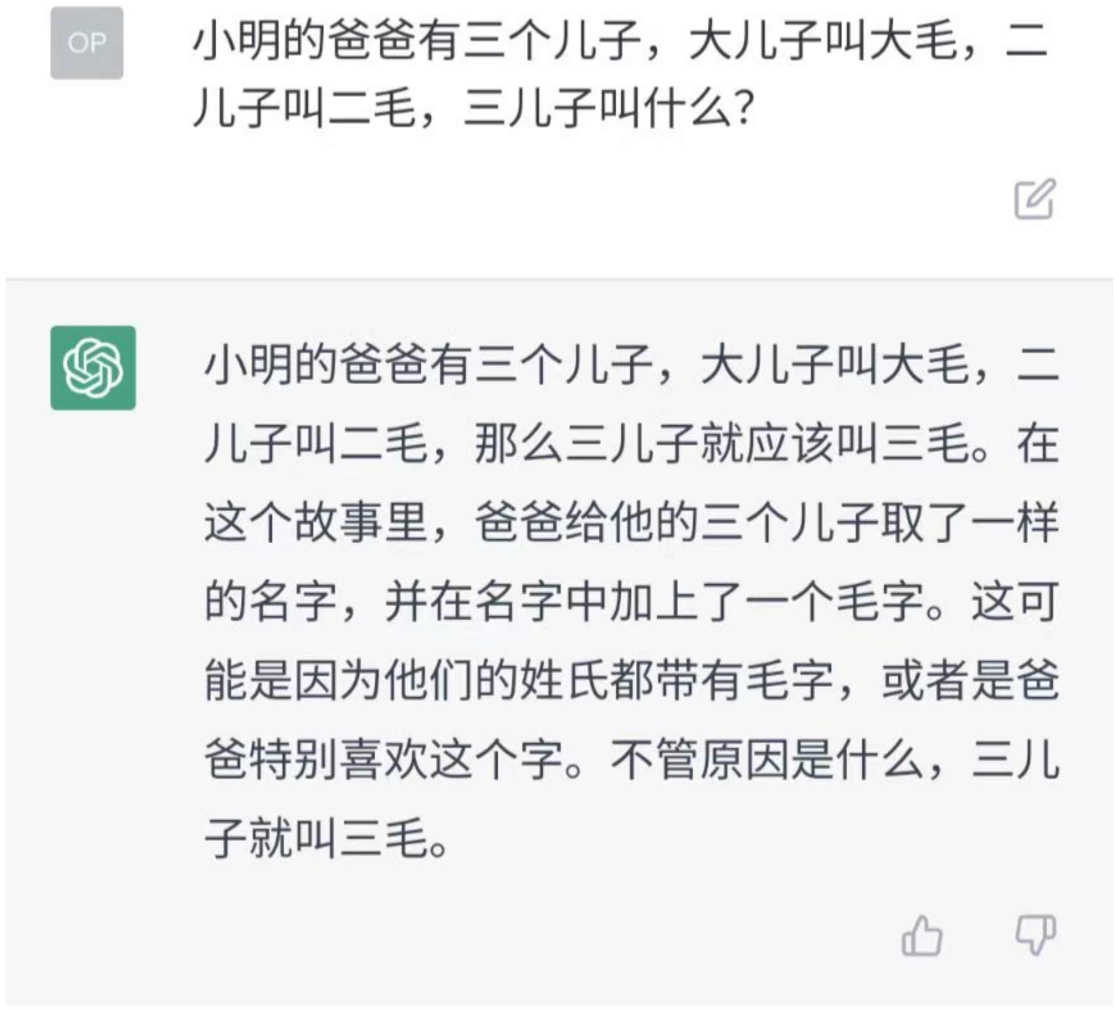
卷积核是如何工作的?

卷积核的参数是由谁决定的?

使用卷积核能够提高模型的性能吗?

希望这些问题能帮助你更好地理解深度学习卷积核的作用和工作原理。

不怎么擅长脑筋急转弯！！！



小明的爸爸有三个儿子，大儿子叫大毛，二儿子叫二毛，三儿子叫什么?

小明的爸爸有三个儿子，大儿子叫大毛，二儿子叫二毛，那么三儿子就应该叫三毛。在这个故事里，爸爸给他的三个儿子取了一样的名字，并在名字中加上了一个毛字。这可能是因为他们的姓氏都带有毛字，或者是爸爸特别喜欢这个字。不管原因是什么，三儿子就叫三毛。