**Data Warehouse** Technologies

ETL extract transform load Extract, Load, and Transform

From China University of Electronic Science and Technology of China 2014 major in Electronic engineering  
**Federal Reserve**

**Pythonsqlite3  
Current Expected Credit Losses** (CECL) is a credit loss accounting standard (model) that was issued by the Financial Accounting Standards

Insert question:

<https://leetcode.com/discuss/interview-question/system-design/295671/Amazon-or-Data-Engineer-Role-or-Database-Design-Question>

answer 1Default value

2I will spin up a new node of the same database and make it actively available (active-passive). Once the databases have synced up, I will disconnect the old node (so that the new node become active now) and insert the new column into the old node. Now, add this old node again to the Active Availability group to let it sync with the new node. Now both databases will be in sync having new column.AWS aws copy,replace



### 1.3训练模型

在训练过程中使用ImageDataGenerator实现数据增强，并通过flow\_from\_directory根据文件名划分label：

优化算法选择了SGD

而激活函数选择了硬饱和的ReLU。因为tanh和sigmoid在训练后期，产生了因没有进行归一化而梯度消失训练困难的问题。如下图

批尺寸在多次尝试之后最终选择了128，迭代50次（在这之后val\_loss呈上升趋势，val\_acc呈下降趋势)

### 2.1图像预处理

使用公开库opencv实现人脸识别，然后对识别到的人脸进行了裁切以及翻转，处理之后将图像进行几何归一化，通过双线内插值算法将图像统一重塑为48\*48像素。

然后使用之前训练好的模型进行同时预测，对多个处理过的脸部预测结果进行线性加权融合，最后得出预测结果

Polybimial kernel, SVM, Radial Kernel

**Reinforcement learning cart pole**

**Oberservation action(buy sale tqqq sqqq) reward(gain, loss)**

**States: rising going down**

**Policy function Qlearning continuing to discrate, Q table**

class DQN():

# DQN Agent

def \_\_init\_\_(self, env): #初始化

def create\_Q\_network(self): #创建Q网络

def create\_training\_method(self): #创建训练方法

def perceive(self,state,action,reward,next\_state,done): #感知存储信息

def train\_Q\_network(self): #训练网络

def egreedy\_action(self,state): #输出带随机的动作

def action(self,state): #输出动作

点积

两数组（向量）间的点积就是简单地将第一组中的每个元素和对应的元素相乘，然后把它们结合在一起假设我们想找到数组A和B之间的点积，只需要简单计算A[0]B[0] + A[1]B[1]…即可。我们会用这个公式将状态（数组）和另一个数组（我们的策略）相乘。移动一点就是一个状态，有无数个状态。因此在DQN中，提出了一种价值函数近似。

价值函数近似什么是价值函数近似？也就是使用函数来表示Q：因为在实际中，并不知道Q的分布，因此只能够使用此函数近似的表示Q：回想我们的Qlearning，输入一种状态，输出一组值，分别是不同动作对应的Q值。**Q**为**动作效用函数**（action-utility function）

target = (reward + self.gamma \*

np.amax(self.model.predict(next\_state)[0]))

1. next\_state为下一状态
2. target为下一状态对应的Q值
3. 而当前状态的Q值则是根据下一状态的Q值以及当前的奖赏来计算得到，就是上面的更新公式
4. 同时，当前的Q值还可以通过当前的状态输入给神经网络来得到

这样，通过3步和4步，神经网络的目标就是将这两个不同方式得到的Q值误差最小化，因此神经网络就此搭建完成。有同学问了，如何编码上面的一些动作？这就是考验大家的编码能力，如果刚入门，没有关系，我们慢慢道来。