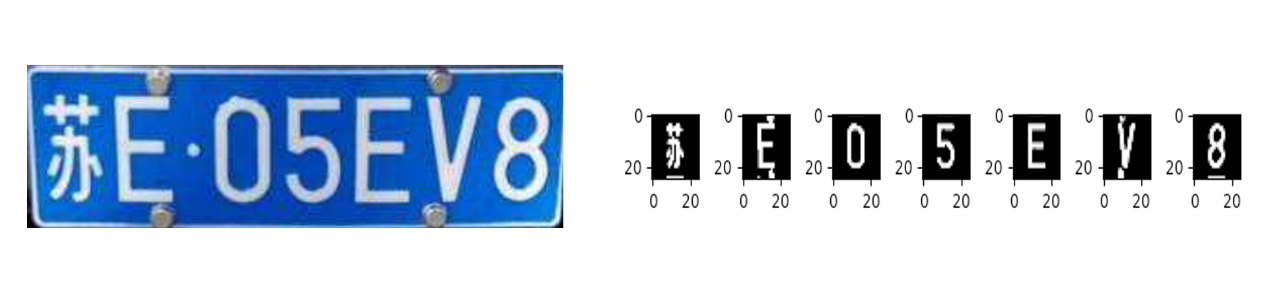
图像处理算法一：

如下图所示，能直接从包括有各类背景的车辆图中提取对应的车牌信息，并对车牌进行一定程度上的校正。



图像处理算法二：

 如下图所示，通过上述所获取得到的车牌图像，进行二值化及分割操作，将对应每个字符进行切分操作，并保存到对应文件中。

LeNet5模型参数解释：

# 设置LeNet5网络模型

class LeNet(nn.Module):

def \_\_init\_\_(self):

super(LeNet, self).\_\_init\_\_()

self.conv = nn.Sequential(

nn.Conv2d(1, 6, 5), # in\_channels, out\_channels, kernel\_size

nn.Sigmoid(),

nn.MaxPool2d(2, 2), # kernel\_size, stride

nn.Conv2d(6, 16, 5),

nn.Sigmoid(),

nn.MaxPool2d(2, 2)

)

上面这个Sequential，其中主要定义的内容包括有卷积层Conv2d，激活函数Sigmoid，池化层Pooling，其中相关的参数设置是nn.Conv2d(1, 6, 5)，第一个数1的含义是当前输入的图像为单通道，对数据像素的维度没有要求，第二个数6的函数是通过一张图片的输入，可以转化为6个通道的输入，意思就是一张图片的一个通道输入最终能转化为6个通道输出，第三个数5的函数就是所谓的核函数，这个就是理解为5\*5维度的矩阵，这个矩阵就是在进行卷积运算中使用到的函数（不理解的话，可以看一下这个链接https://zhuanlan.zhihu.com/p/163017446）。同样的，nn.Sigmoid()就是激活函数，然后.MaxPool2d(2, 2)就是最大池化层，其中每次移动的stride为2。下面参数的nn.Conv2d(6, 16, 5)是等同于上面的，但是这里有一个小细节就是第一个数6跟上面的nn.Conv2d(1, 6, 5)的第二个数是关联在一起的。

# 调用

self.fc = nn.Sequential(

nn.Linear(16\*4, 120),

nn.Sigmoid(),

nn.Linear(120, 84),

nn.Sigmoid(),

nn.Linear(84, 65) #

)

这个位置的话，就是考虑全连接层了，其中65为当前最后输出的个数nn.Conv2d(6, 16, 5),这里的第二个数作为遮脸linear的输入，其中考虑划分为4个维度，然后转化为120维度，下面依次都是这样，考虑将120转化为84的维度，最后一个linear就是最终的输入，需要考虑将84转化为65的维度，因为最后我们的数据有65个分类，就是对应车牌里面的字母、数字、字共65个不同的分类。

# 前向传播

def forward(self, img):

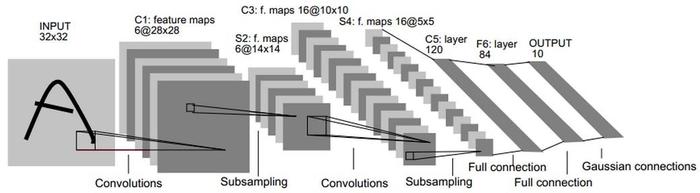
feature = self.conv(img)

output = self.fc(feature.view(img.shape[0], -1))

return output

这个地方就是考虑，对应的loss损失函数往前调用。（不理解的话，可以看一下这个链接https://blog.csdn.net/bitcarmanlee/article/details/78819025）

下面是LeNet5的网络模型架构:

 我们也是采用LeNet5的架构，但是因为我们的数据跟最原始的LeNet5数据不同，所以我们需要根据这个LeNet5网络架构进行我们的网络模型设计。