TASK 1：

1. 加载导入数据
   1. DataLoader
   2. Transform使用
      1. transforms.Resize方法改变图像大小
      2. transforms.ToTensor方法是将图像转换成张量，同时会进行归一化的一个操作，将张量的值从0-255转到0-1
      3. transforms.Normalize方法将数据进行标准化（一般参数估计为0.5）
      4. transforms.Compose 包装一系列transforms方法
2. 构建神经网络
   1. **Model**
      1. **卷积层：**
         1. 导入的图片被ToTensor转换为二维的张量，通过卷积核（二维矩阵）的移动与运算提取关键信息
         2. Padding：通过加边保存边缘信息
         3. 彩色图片，RNG格式，它包含红，黄，蓝三个通道，而灰色图片只有一个通道。
         4. 输入通道数等于卷积核通道个数、卷积核个数等于输出通道个数
         5. 增加通道数就代表着增加特征，而造成通道数增加的操作其实就是卷积核的增加，不同的卷积核可以提取到不同特征
      2. **池化层**：取区域平均or最大（maxpool）,减少信息量
      3. **全连接层**
      4. **激活函数**
         1. 非线性函数，线性隐含层没有意义，层数再多也只是线性组合，结果计算的是logistics线性回归问题，添加非线性函数才可以计算复杂的情况
         2. Relu：优势与sigmod和tanh相比在输入x很大时斜率为1，而后两者梯度接近0，会拖慢训练的速度
            1. Leaky Relu:x<0时，g(x)=-0.01x
         3. Sigmoid将数据控制在0~1之间，通常用在二元分类任务中
            1. G(x)=sigmoid(x),有g’(x)=g(x)(1-g(x))
         4. Tanh将数据控制在-1~1之间（中心化效果更好），大部分情况下比sigmoid更优越
            1. G(x)=tanh(x),有g’(x)=1-g(x)^(2)
         5. Softmax函数可以将一个K维的实数向量（通常是神经网络最后一层的输出）压缩（归一化）到(0, 1)之间，并且使得所有元素的和为1
   2. **损失、成本函数**（用梯度下降法使loss不断下降直到找到最低点）
      1. W与b参数每次减去梯度（偏导）乘步长（学习率）实现反向传播
      2. 交叉熵损失函数CrossEntropyLoss()
   3. **优化器**
      1. 更新参数用
      2. 学习速率lr不能太大或太小
      3. optimizer.zero\_grad()：将优化器中所有参数的梯度清零。在每个批次数据进行反向传播之前，需要先将之前计算得到的梯度清零，以避免梯度累积的影响。
      4. backward()，自动计算损失函数关于模型参数的梯度，即反向传播过程。
      5. optimizer.step()：用于更新模型参数的步骤，在梯度计算完成后调用。
3. 训练、测试、评测
   1. Tensorboard
   2. 指标评测
      1. correct = (outputs.argmax(1) == targets).sum()计算预测正确的个数

TASK2：

1. 问题：RNN网络中出现梯度消失与爆炸的情况
   1. 因为网络层数过深时，反向传播到很前面时各层梯度通过链式法则乘在一起可能成指数型变化，而RNN处理的序列化数据相当于层数很多很多的神经网络