



计算机视觉与图形图像处理

课程设计

《使用深度神经网络进行汉字手写体识别》

姓 名:

班 级:

专 业: 人工智能

学 号:

2024年4月

**目 录**

**[1 实验思路 1](#_Toc19412)**

[1.1实验设备与实验环境 1](#_Toc28369)

[1.2 整体思路与结构 1](#_Toc6569)

**[2 数据集处理 2](#_Toc14220)**

[2.1数据集划分 2](#_Toc26047)

[2.2数据预处理 3](#_Toc511)

**[3 训练模型的选取及模型描述 5](#_Toc15715)**

[3.2 模型训练的设计 7](#_Toc3260)

[3.3 训练效果评价 8](#_Toc17492)

[3.4 过拟合的避免措施 9](#_Toc28182)

[3.5 对模型深度的尝试 15](#_Toc31455)

[3.6 对ResNet的尝试 17](#_Toc841)

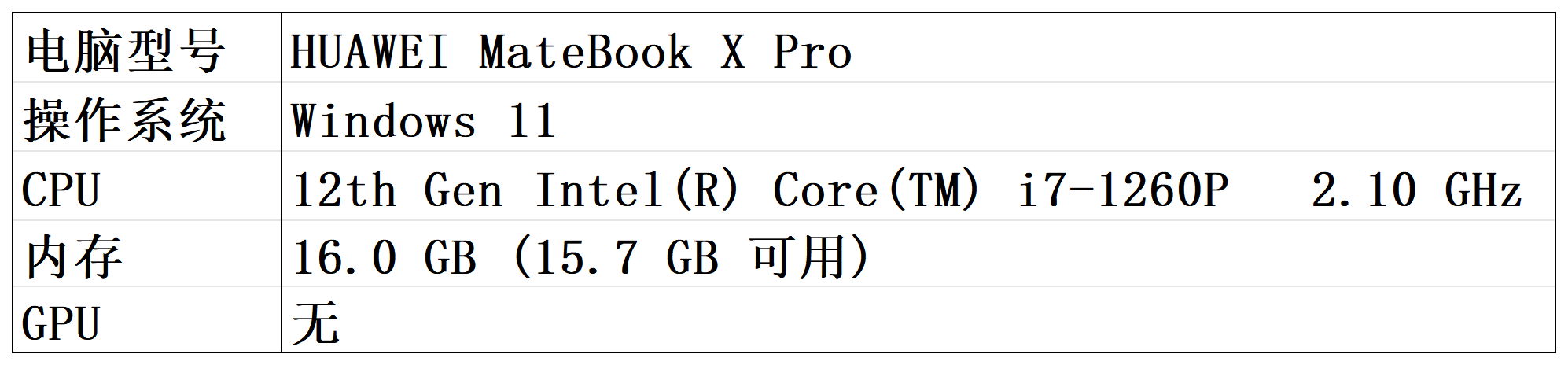
**[4 模型测试部分设计 20](#_Toc14062)**

**[5 实验心得 21](#_Toc27079)**

# **1 实验思路**

**1.1实验设备与实验环境**

实验设备：



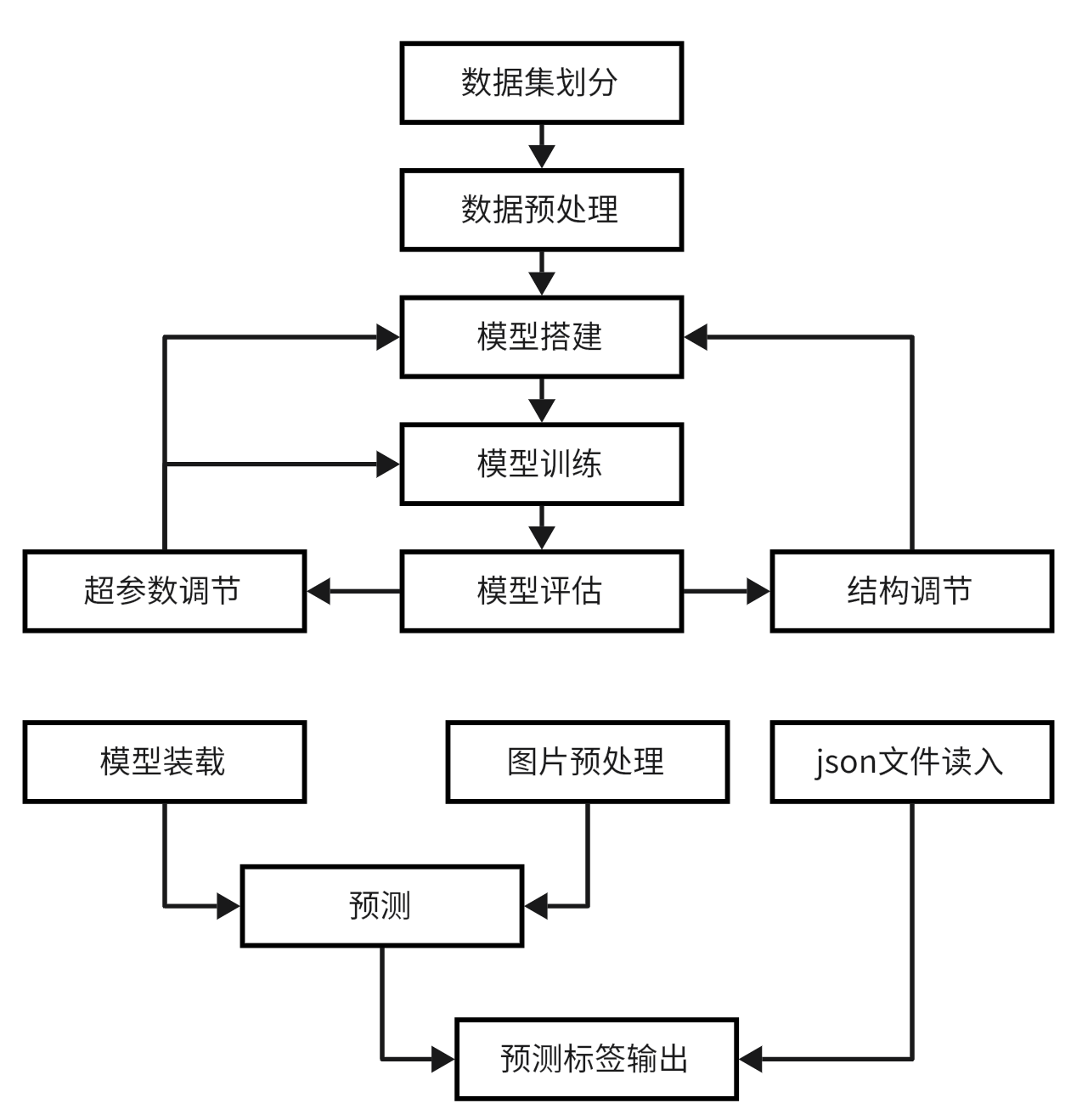
实验环境：

# 图_Sheet1

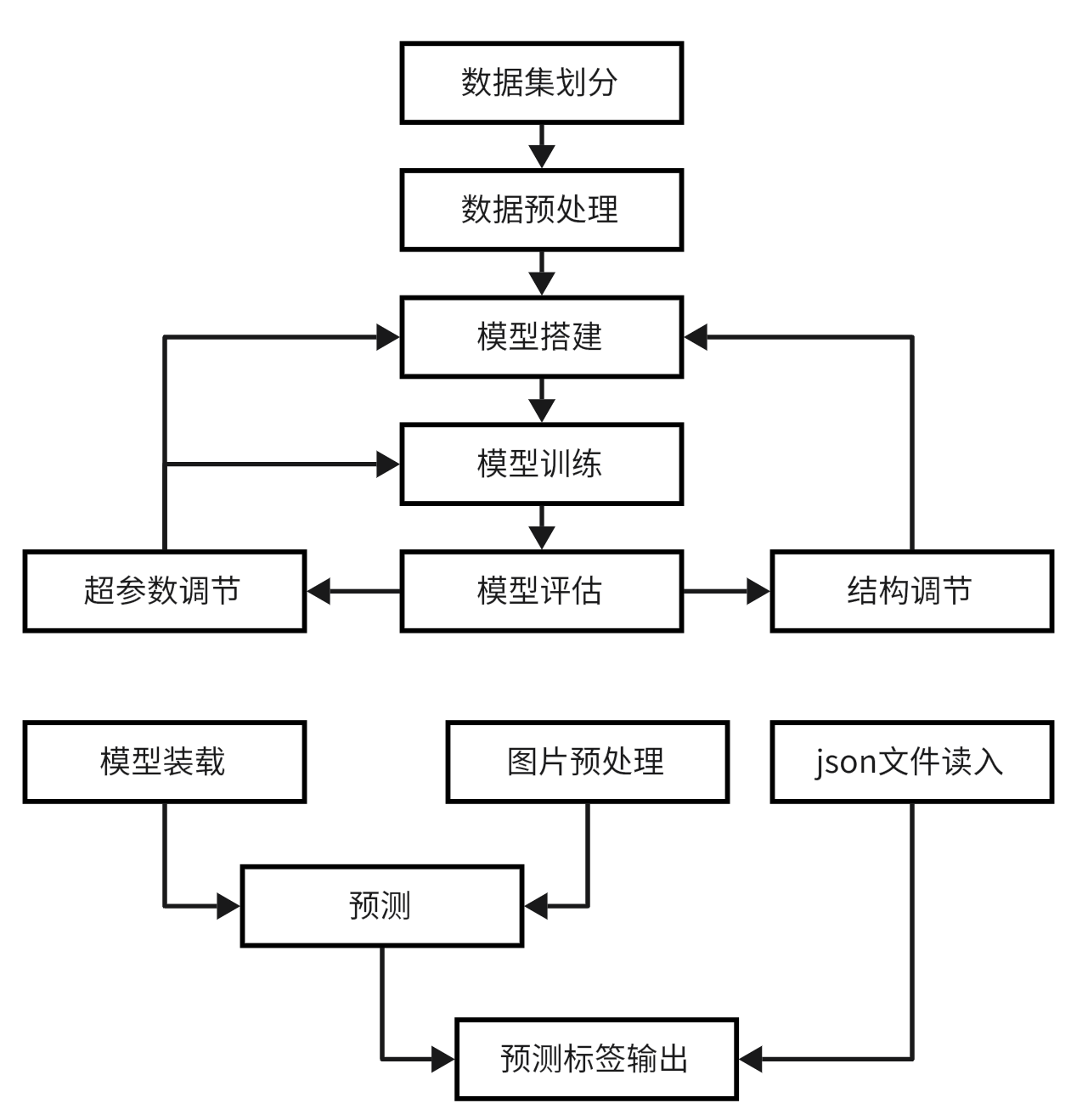
**1.2 整体思路与结构**

按照流程分为模型训练部分和模型测试部分。

训练部分：



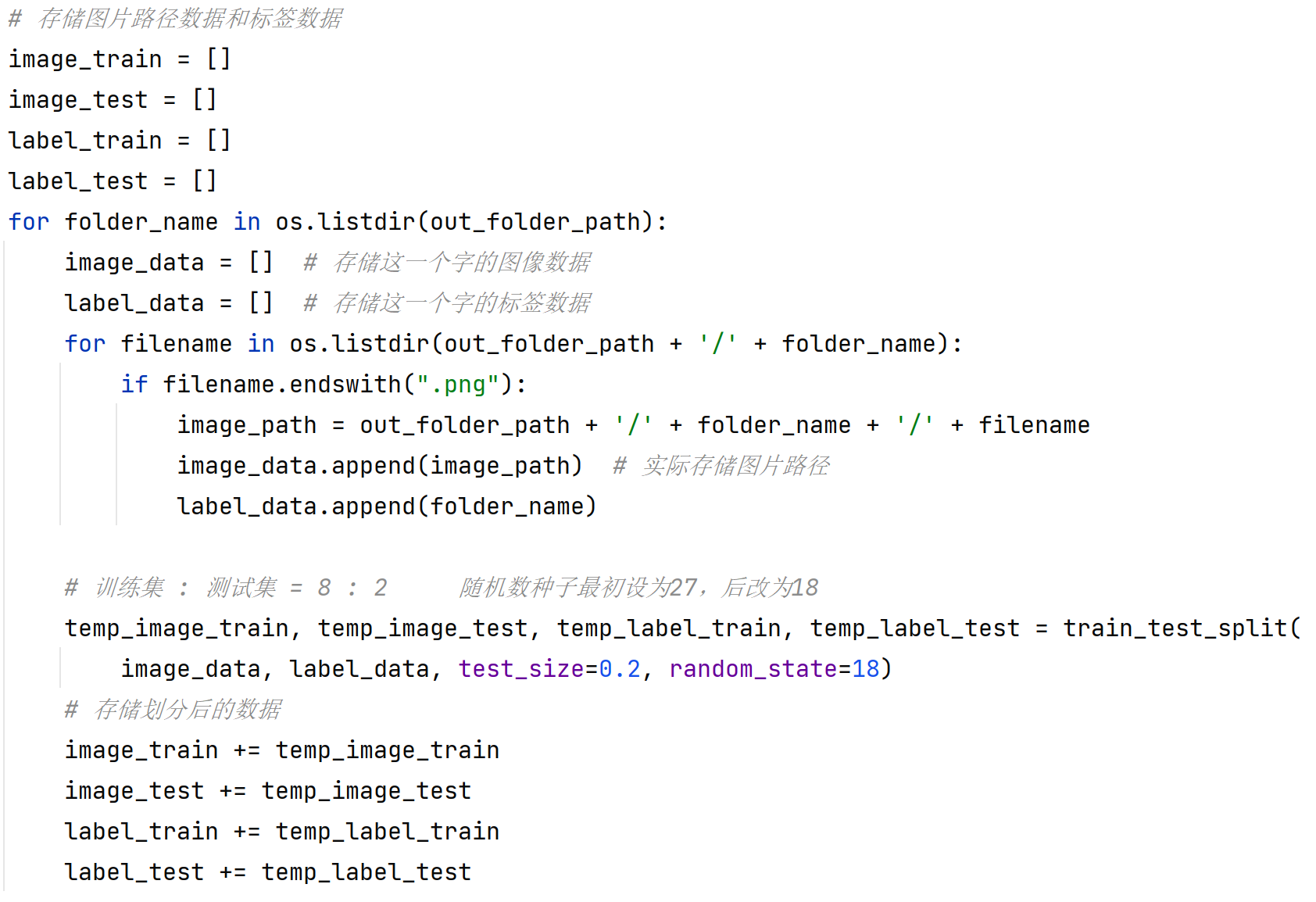
测试部分：



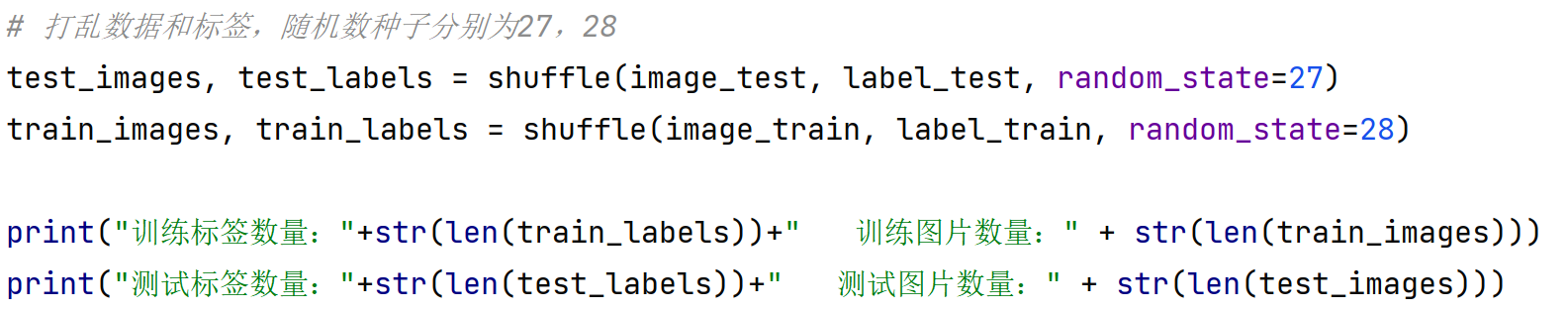
# **2 数据集处理**

## **2.1数据集划分**

采用同一汉字的80张图片按照8：2的比例划分为训练集与测试集：

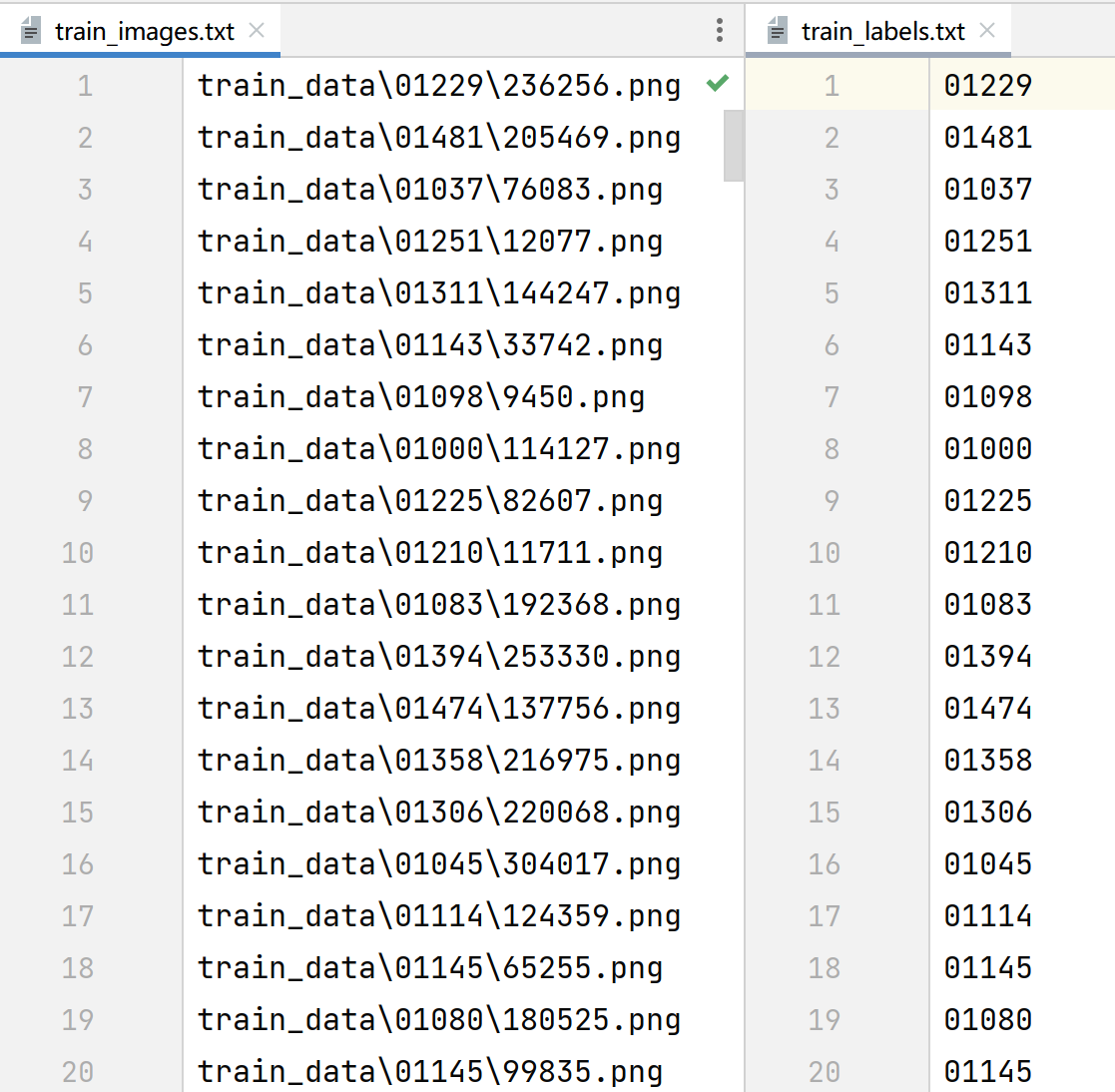


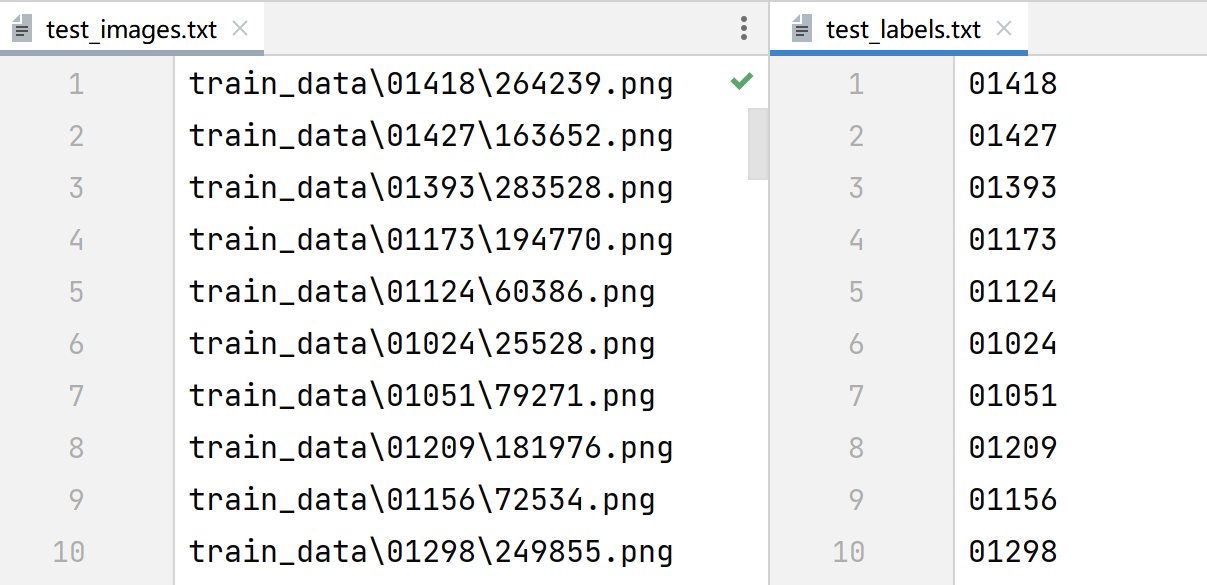
最后将所有训练集与测试集打乱：



得到训练图片31839张；测试图片8002张。

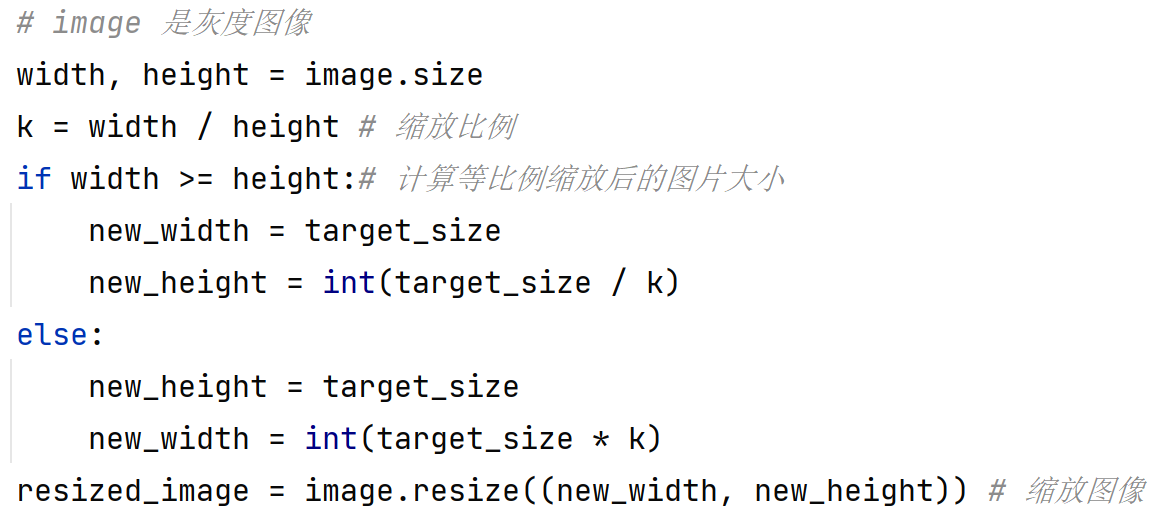
将划分后的数据以txt文件形式保存于doc文件夹中：



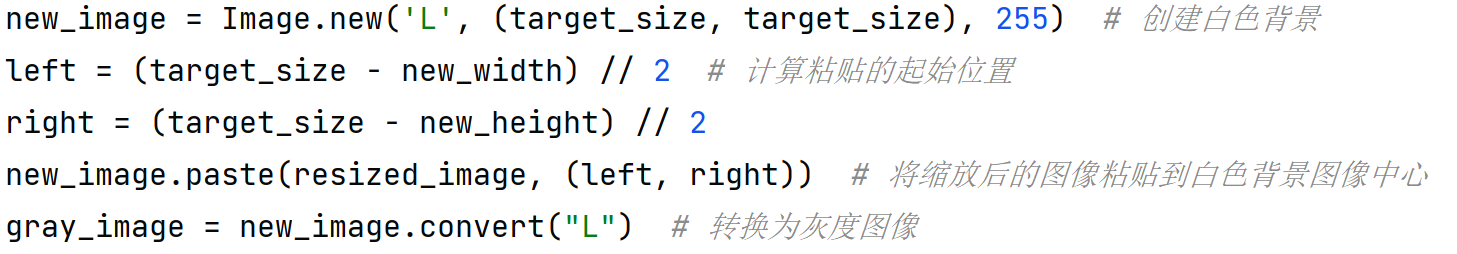


## **2.2数据预处理**

首先将图片统一按照长边等比例放缩至target\_size = 64（pixel）：



将放缩后的图片resized\_image按照短边居中，加白边补齐64（pixel）并转换为灰度图像（原定128（pixel），但是受限于电脑性能只能改成64（pixel）：

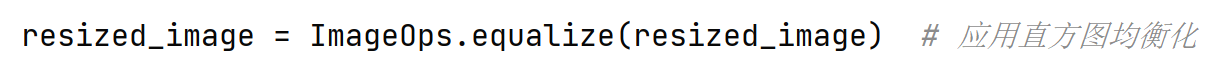


得到的效果：

z1   

z1  z1 

读入数据后进行直方图均衡化，以增强对比度：

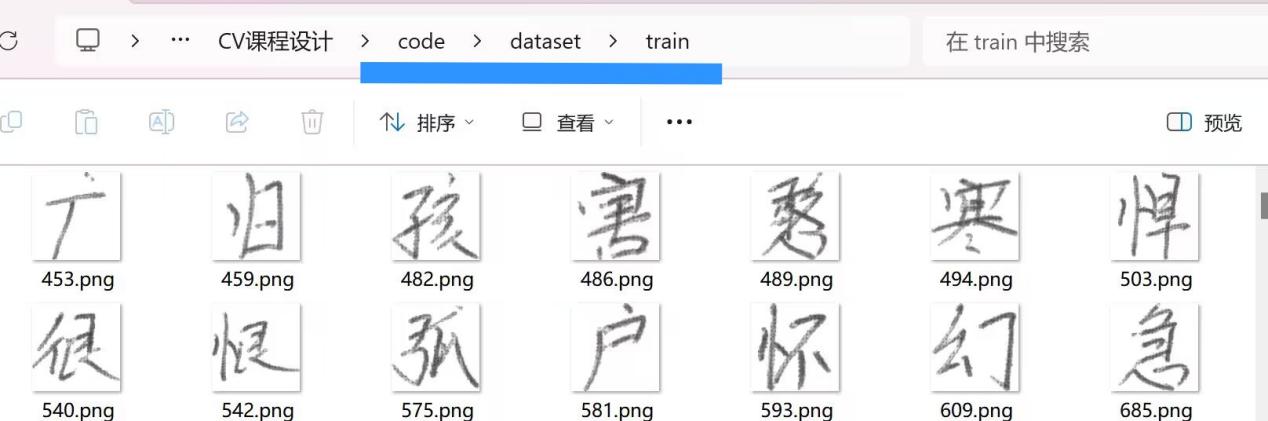


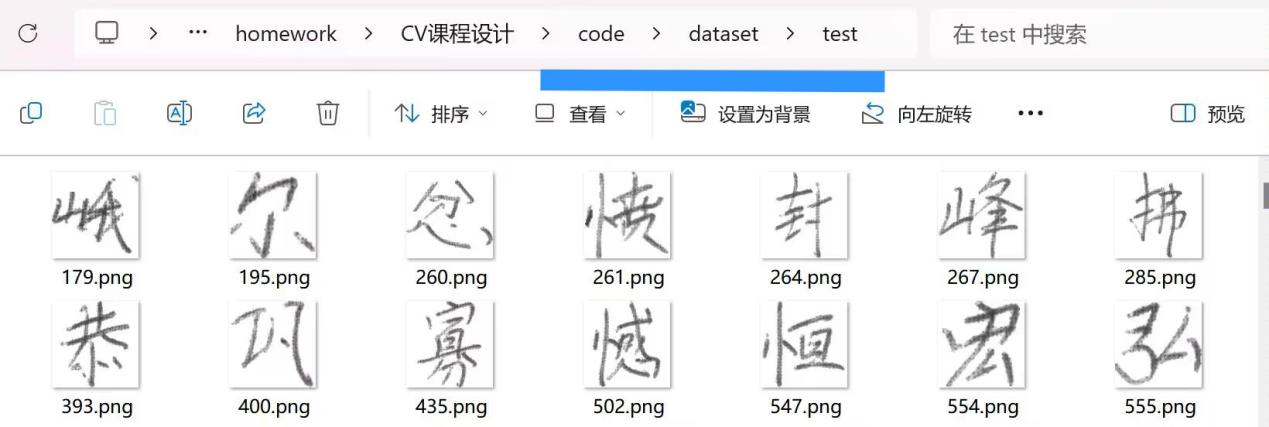
下图是直方图均衡化一次与两次的对比，效果差别不大，一次直方图均衡化的效果略微好一点：

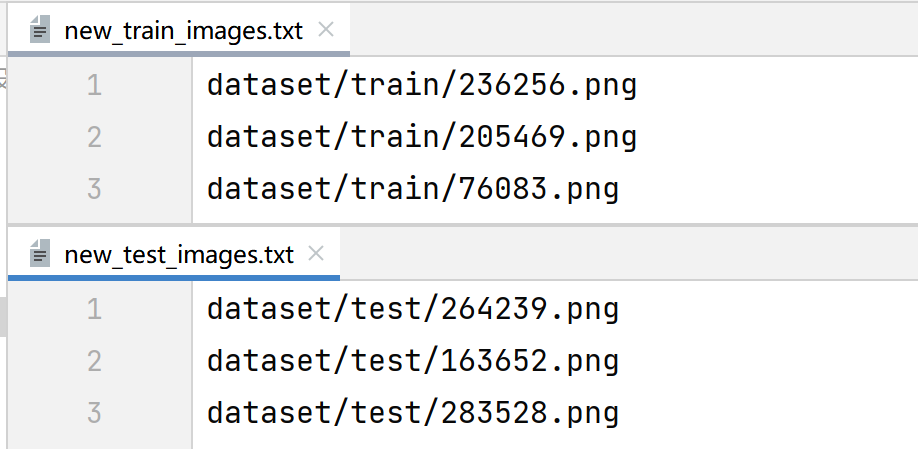
   

将预处理后的图像，保存在新建的dataset文件夹中的test文件夹和train文件夹中（只用运行一次即可，节省时间）：

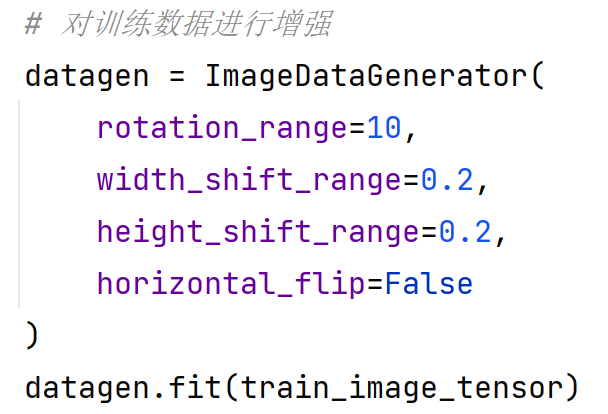




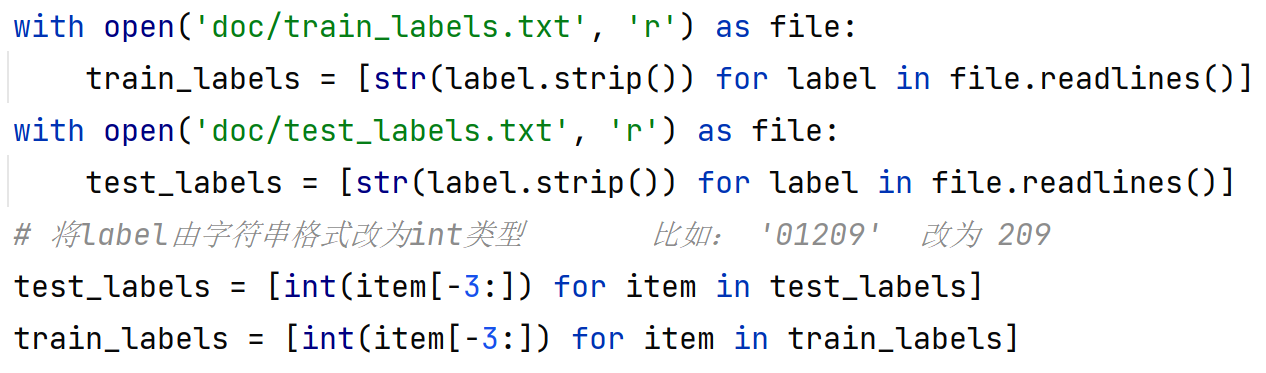
将图片的新相对路径按原有顺序以txt的形式保存至doc文件夹中的新文件：



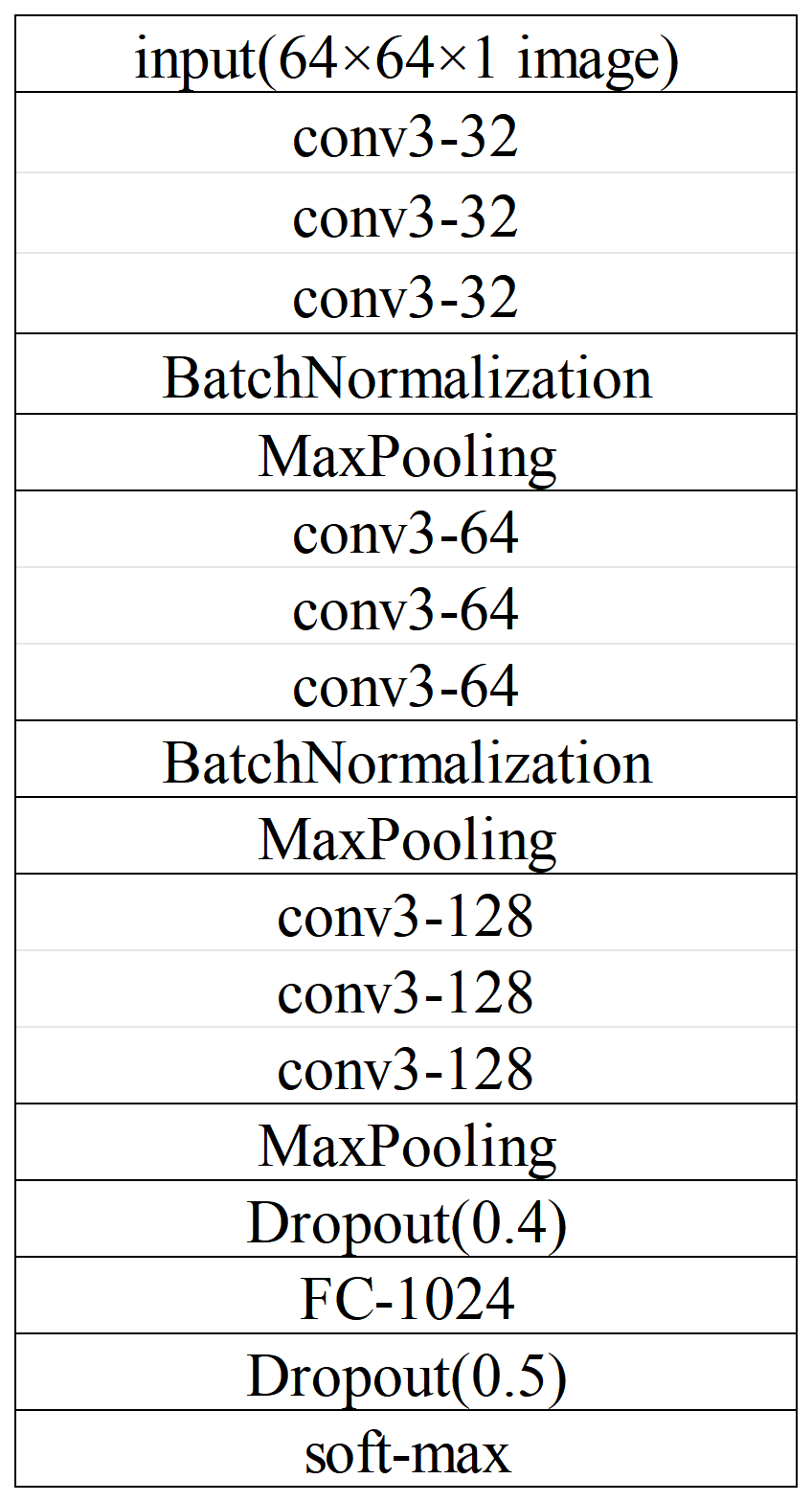
再对训练数据进行数据增强：



对于标签，采用取后三位的方式（比如“01073”在模型中记为73，整数）：

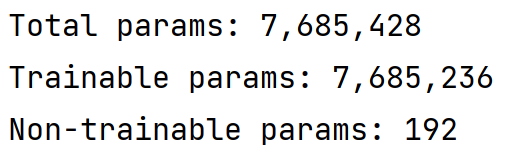


**3 训练模型的选取及模型描述**

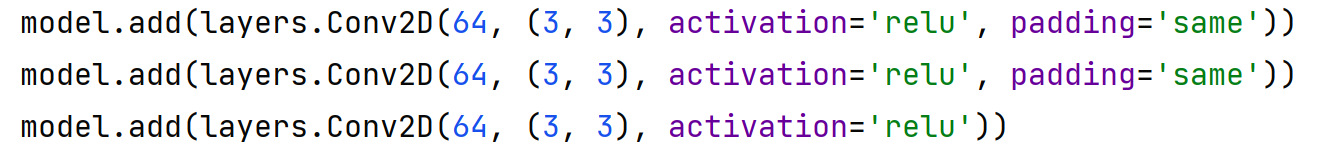
**3.1 最终模型结构**

采用了经典的**CNN**结构：

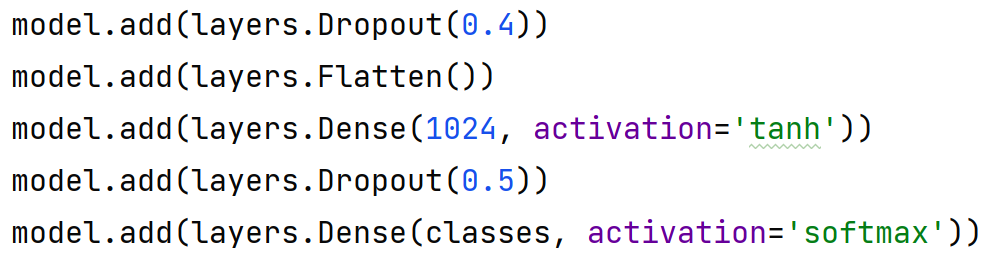
由9个卷积层，2个BN层，2个全连接层，2个dropout层，2个最大池化层，按右图顺序堆叠。



由于输入图像尺寸较小，因此三个相连的卷积层中，其中两个都使 padding=‘same’，在卷积中**自动填充**，使输出特征图的尺寸与输入特征图的尺寸相同，尽量保持特征图的空间维度，比如：



对于第一个全连接层,考虑到ReLU激活函数对于大于零的输入直接输出输入值，不做任何限制，可能会导致神经网络在输出层使用softmax激活函数时出现数值不稳定的问题，因此将输出层前的全连接层激活函数设置为**tanh**。



**损失函数**是交叉熵损失函数中的一种：sparse\_categorical\_crossentropy（稀疏分类交叉熵损失函数），标签是整数形式时常用的损失函数。

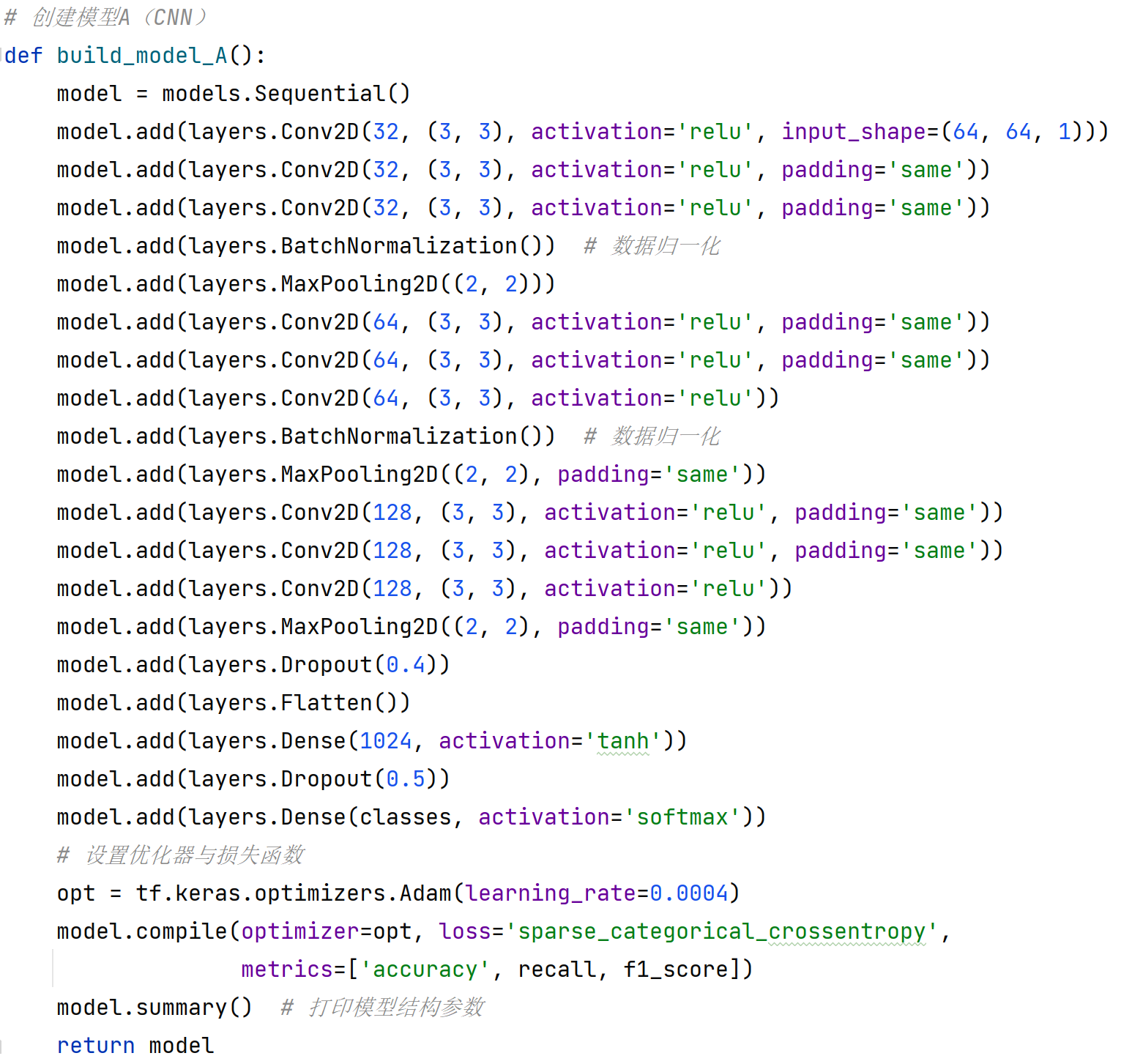
评估指标为准确率与召回率：

**准确率**用于衡量模型预测正确的样本数占总样本数的比例，准确率越高，表示模型在整体（多个类别）数据上的分类能力越好。

**召回率**衡量模型能够正确预测出每个类别的真正例样本的能力，帮助评估模型对每个类别的覆盖能力。

采用**Adam优化器**，初始学习率为0.0004（考虑到模型略复杂，设置了较小的学习率），它会根据每个参数的梯度情况进行自适应地调整学习率。

完整定义代码：



**3.2 模型训练的设计**

定义了四个回调函数：



2个**提前停止**的回调，为了防止过拟合，5轮内指标val\_accuracy和accuracy不再改善，则停止训练。

1个**CSVLogger**回调，将训练过程中测试集与验证集的参数变化以CSV的形式保存在“model”文件夹中。

1个**ModelCheckpoint**回调，监视ModelA的验证集上的准确率，并保存准确率最高的模型。



设定**epoch**为50，因为设置了提前停止，所以不用太担心epoch太大过拟合。

综合考虑到训练速度和训练效果，将每次训练时使用的样本数量**batch\_size**设置为64。

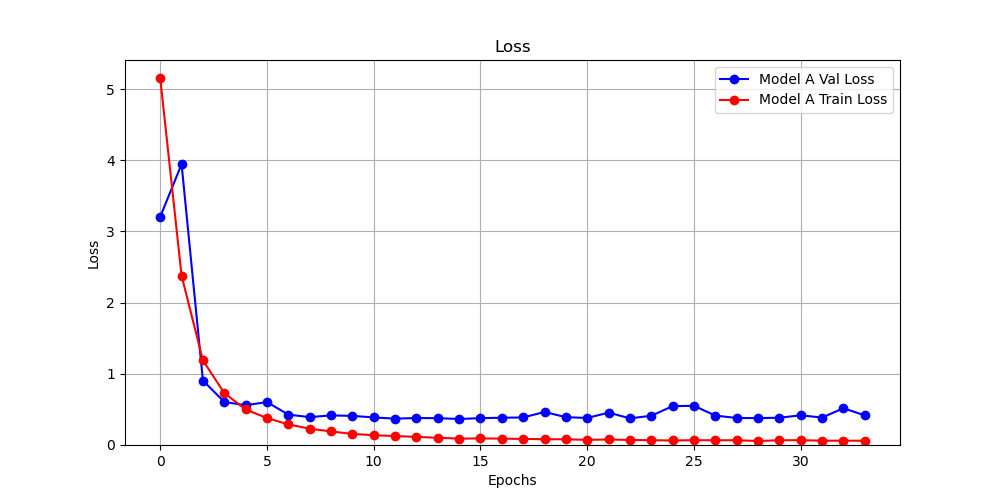
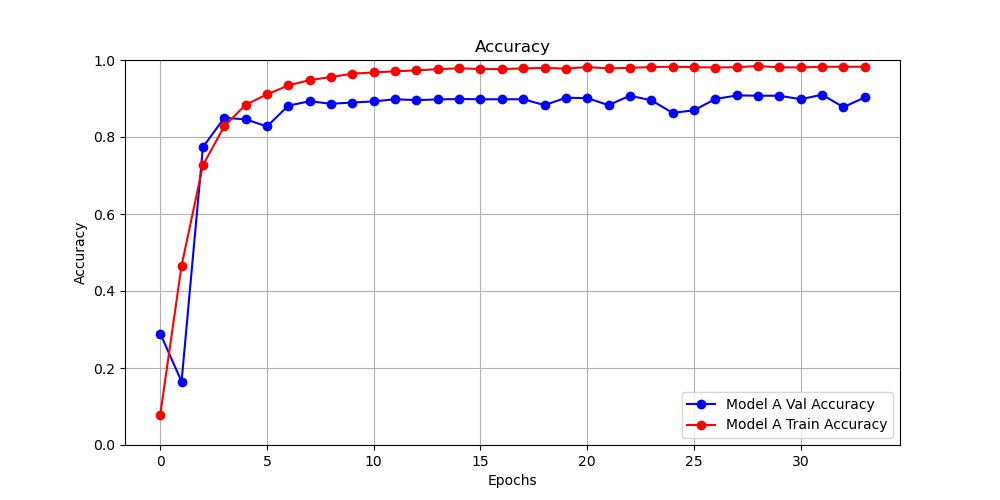
最后将训练好的模型A以HDF5格式保存在名为“end\_model\_A.h5”保存到“model”文件夹中。

**3.3 训练效果评价**

采用两幅图表直观展示训练过程中训练集与测试集准确率与损失的变化：



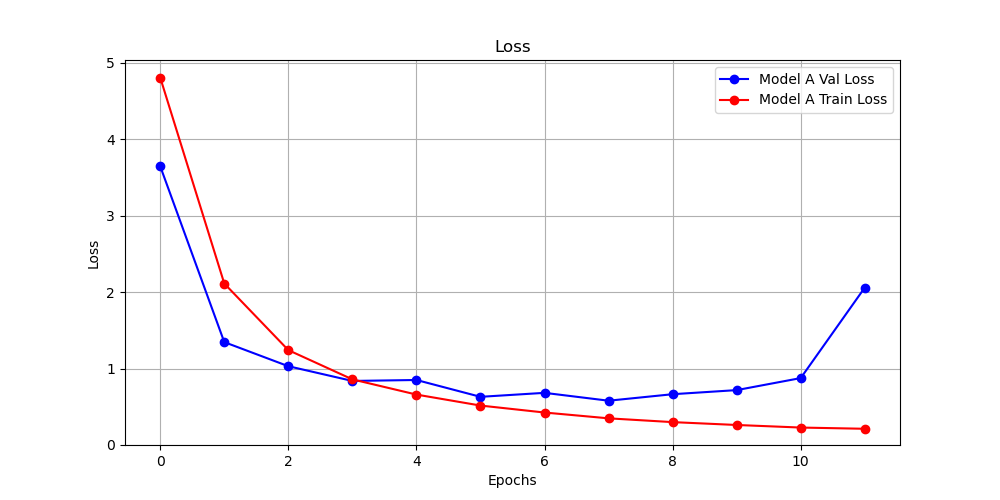
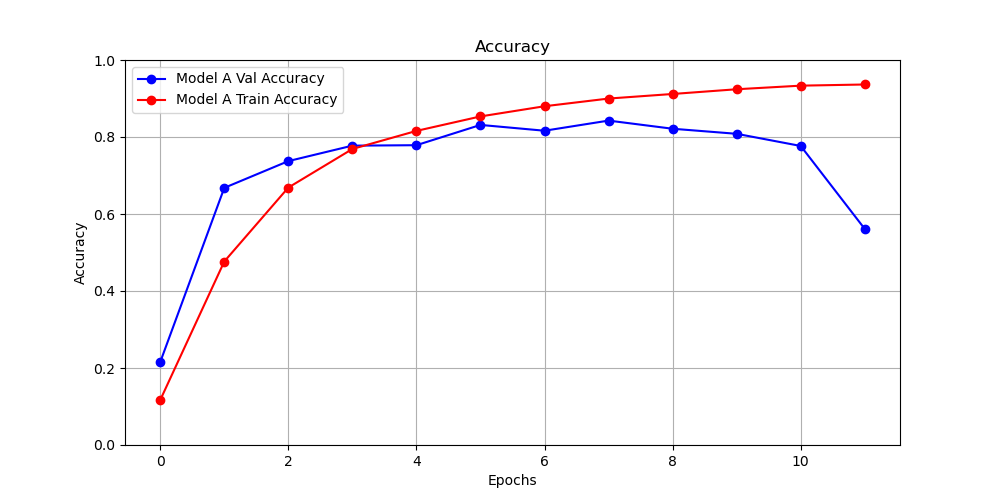
比如上述模型的某一次训练：



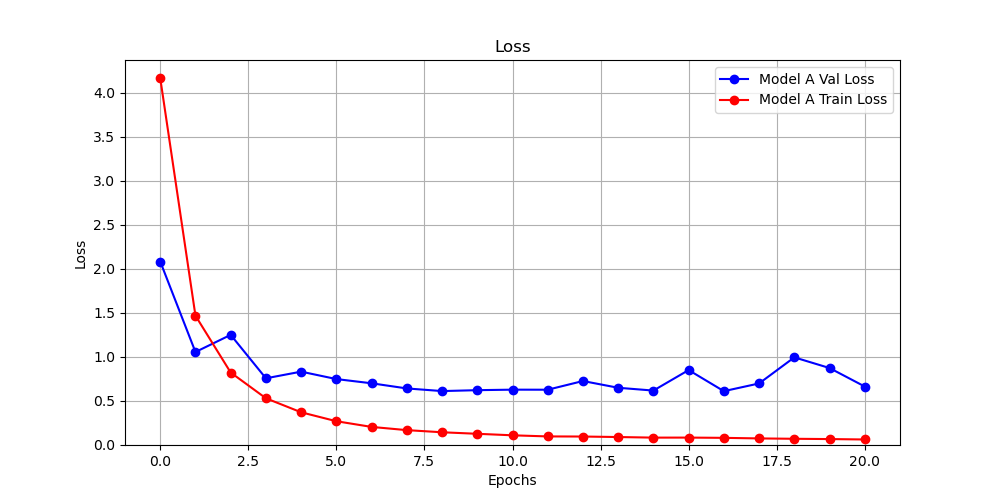
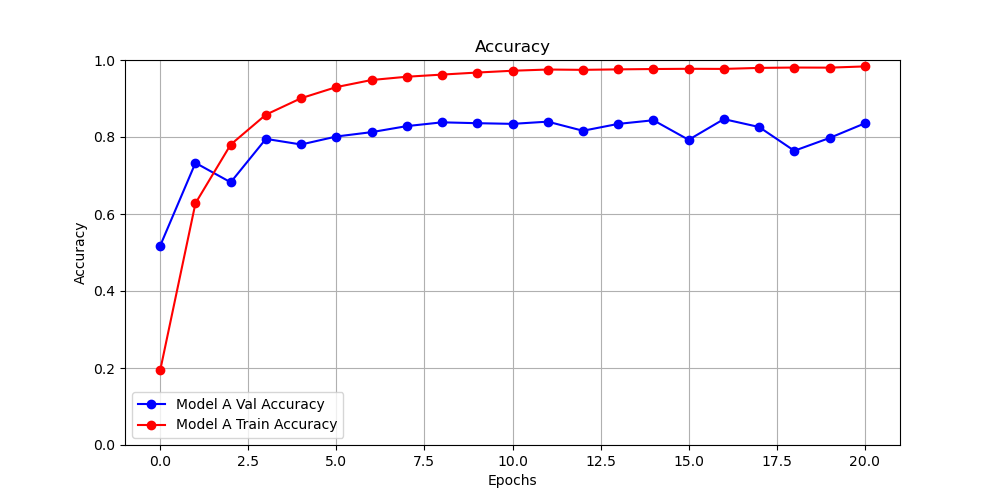
初期模型未完全调整好，因此会产生比较大的波动，但最终趋于稳定。

**3.4 过拟合的避免措施**

跑出过一个典型的过拟合曲线：



如上图，在训练集准确率逐渐趋于1，损失逐渐趋于0时时，训练集准确率逐渐下降，损失逐渐上升，或训练集评估数据由平稳转为较大的波动，比如：



除了设置“提前停止”的回调、设置数据增强以外，我还尝试了2种措施：

**3.4.1 添加Dropout层、L1\_L2正则化项，限制模型的复杂度**

在训练阶段，**Dropout层**会随机地将一部分神经元的输出置零（或按一定概率置零）减少模型的过拟合，提高泛化能力；而在验证阶段，Dropout层不会再随机地丢弃神经元的输出，而是保留所有神经元的输出，以获得更稳定的预测结果。

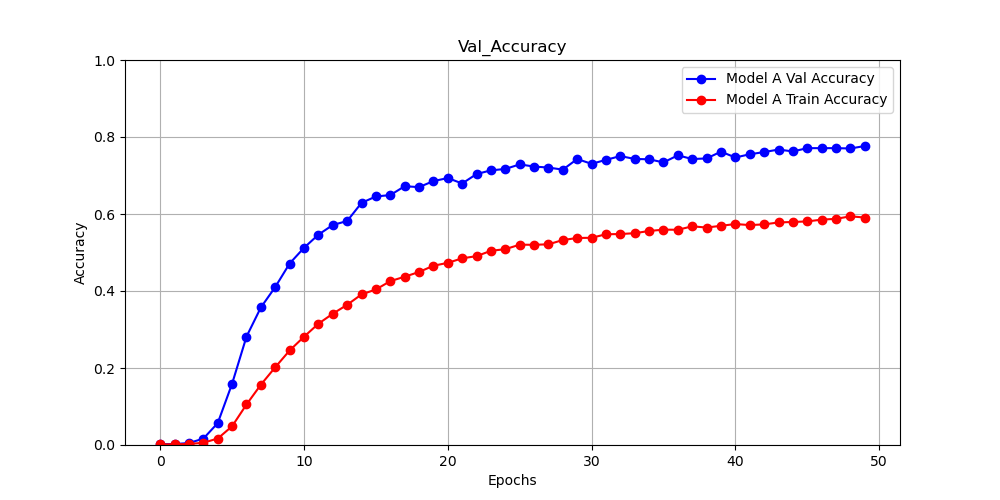
**弹性网络正则化（L1\_L2）**结合了L1范数正则化与L2范数正则化，既能够产生稀疏的权重（L1），也能够让权重参数更加平滑（L2），控制模型的复杂度，防止过拟合。

因此，用最简单的模型（3个卷积层）进行了尝试：

设置了2个dropout(0.5)，分别位于展平层之前，1024全连接层之前以及1024全连接层之后全连接输出层之前。设置了一个L1\_L2正则化项给1024全连接层：



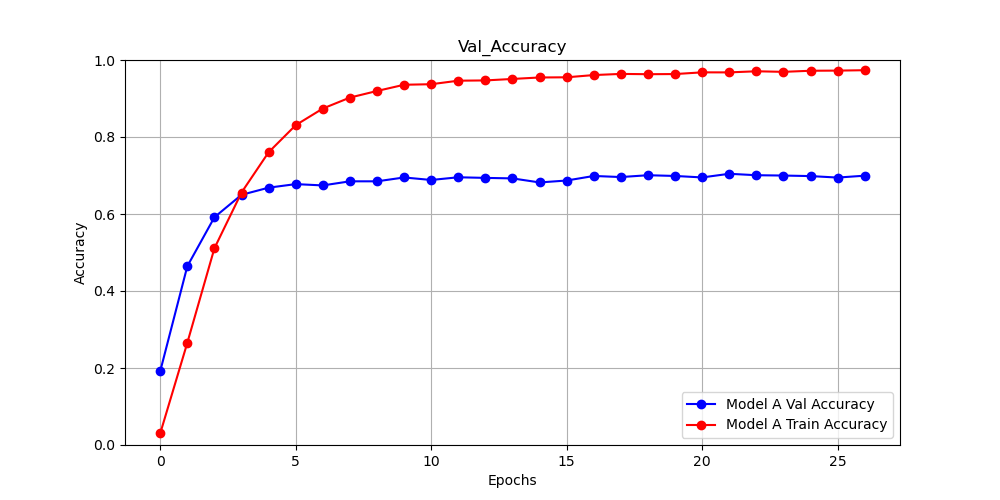
但是所得到的训练曲线见下：

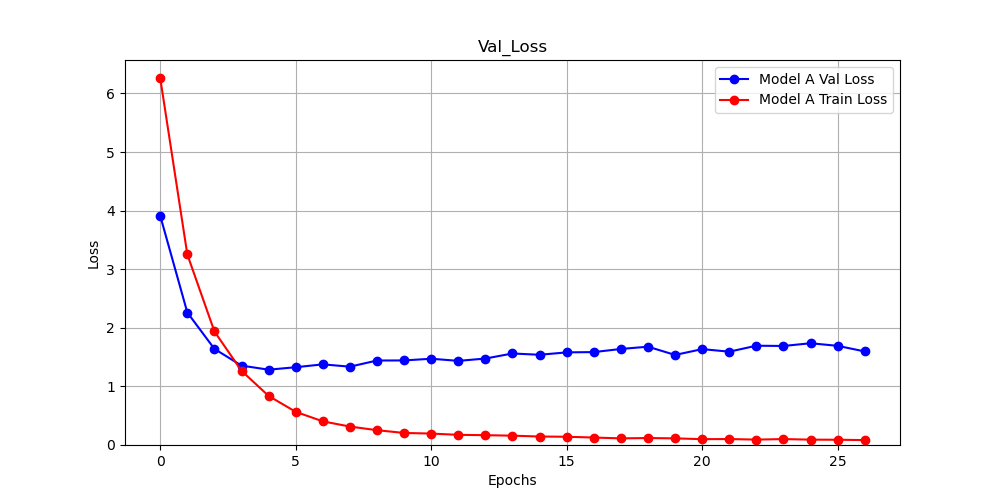


训练集上模型的准确率远低于验证集。

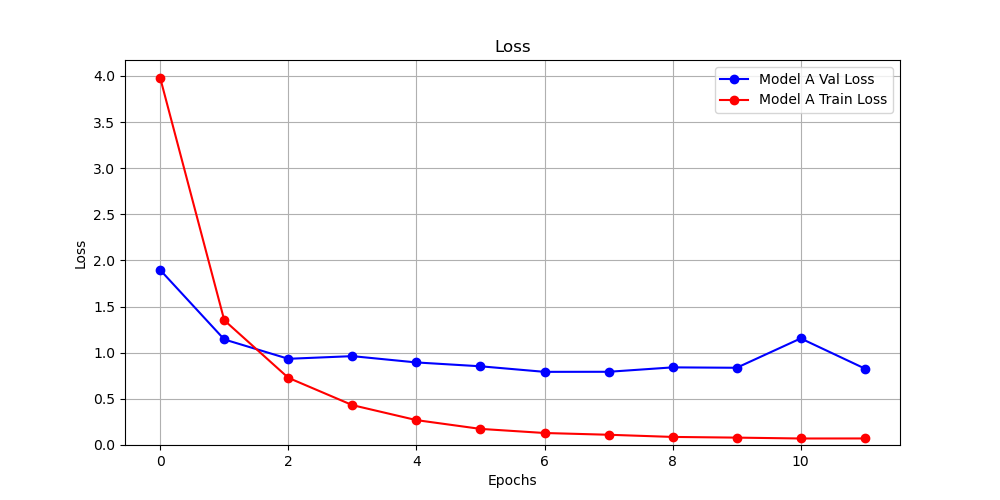
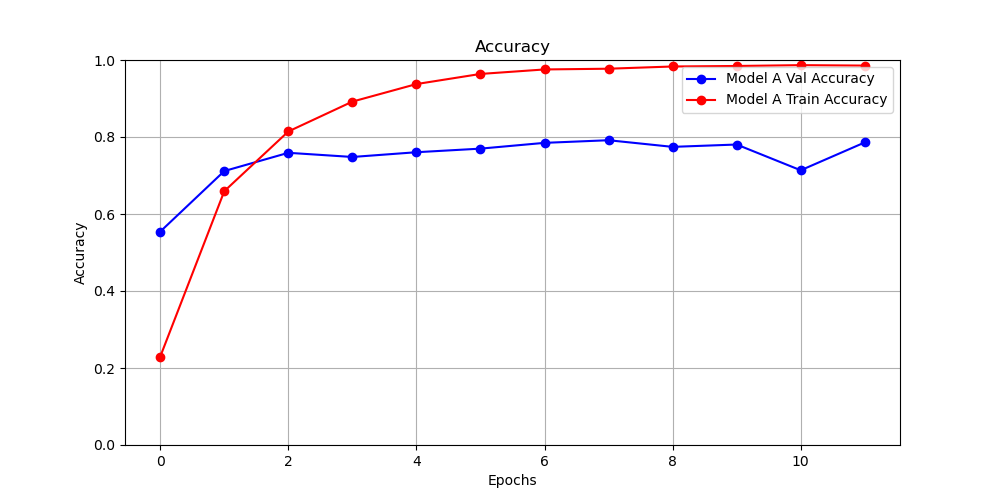
我认为可能的原因是L1\_L2正则化项过于强力，以及Dropout层过多、参数过大，当模型中的Dropout层过多时，训练时模型会受到更强的随机性影响，导致模型在训练集上的表现不佳。而在测试阶段，由于Dropout的行为不再存在，模型的性能可能会更稳定，导致测试集准确率高于训练集。

因此调整了Dropout的概率参数，降低训练阶段的随机性，使模型在训练阶段仍能减少过拟合，又不会影响模型在训练集上的表现。因此首先删除了L1\_L2正则项，将2个Dropout层的概率参数都改为0.25，并修改了初始学习率，得到以下训练过程：



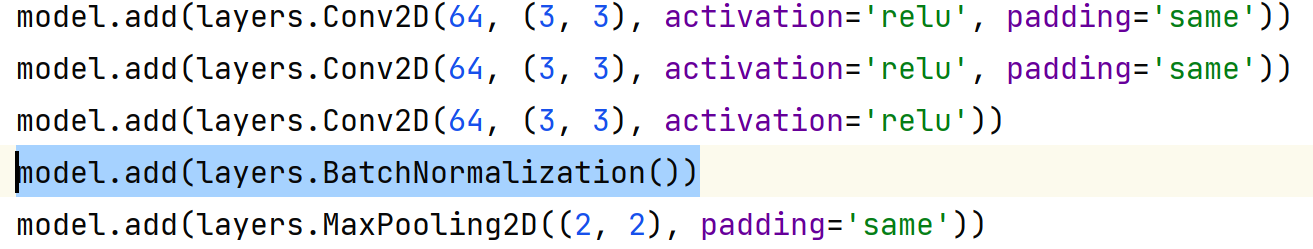


模型准确率确实有提升，但是当epoch>10之后训练存在少许过拟合的迹象，因此将第二个Dropout层参数改回0.5，并修改学习率、对结构增加了一些改变（见3.4.2）后得到：



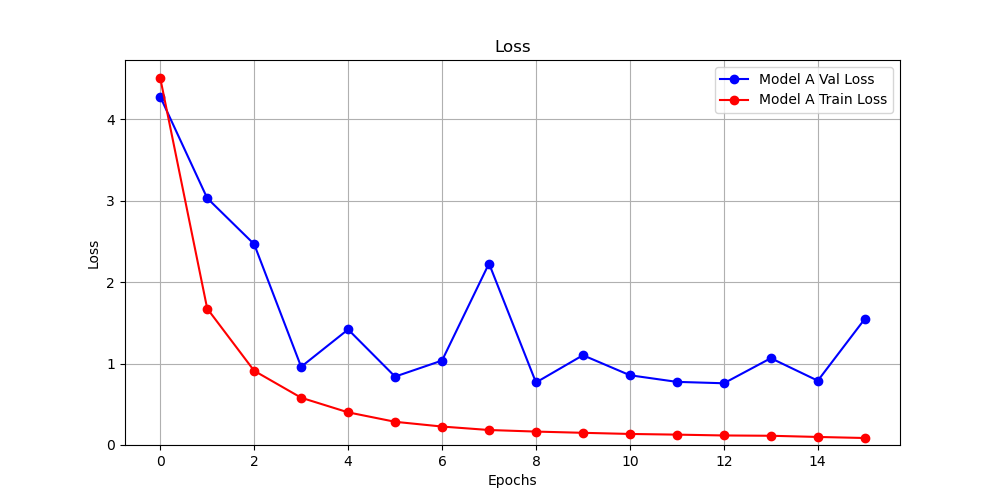
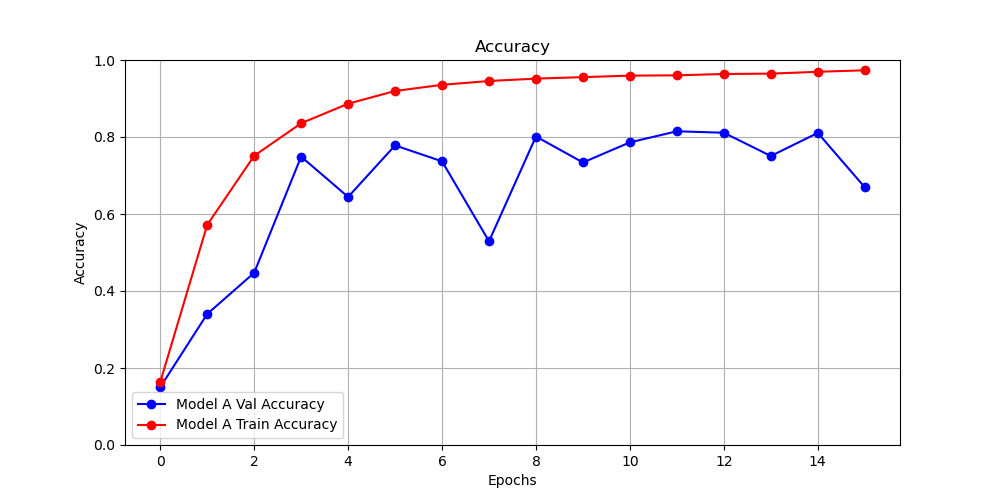
**3.4.2 添加数据归一化（批标准化）层**

Batch\_Normalization对每个batch的输入都进行归一化处理（均值接近0，方差接近1），并且引入了一些自动调整的参数，使得模型可以学习到每个特征的最佳均值和方差的调整，使得下一层的输入分布稳定。一般用于加速收敛，提高模型的训练速度和稳定性；也有一定的正则化效果，可以减少过拟合：

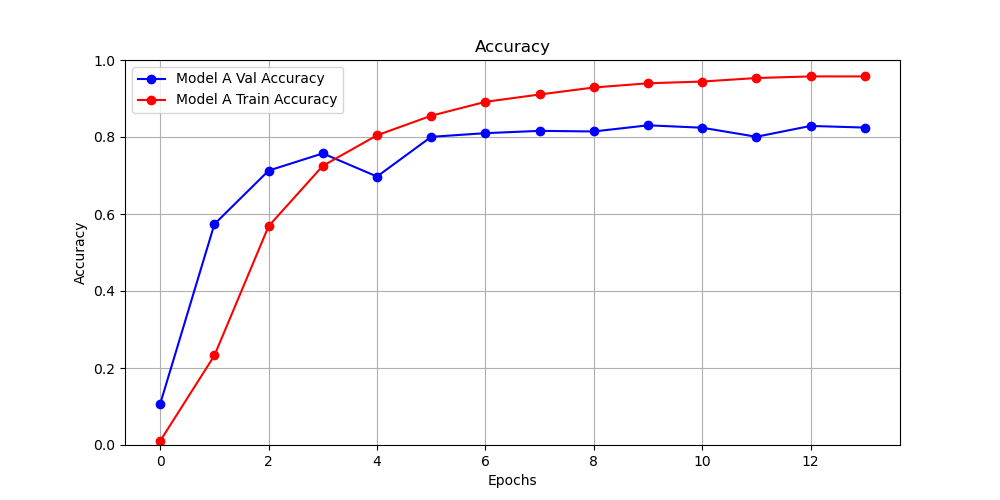


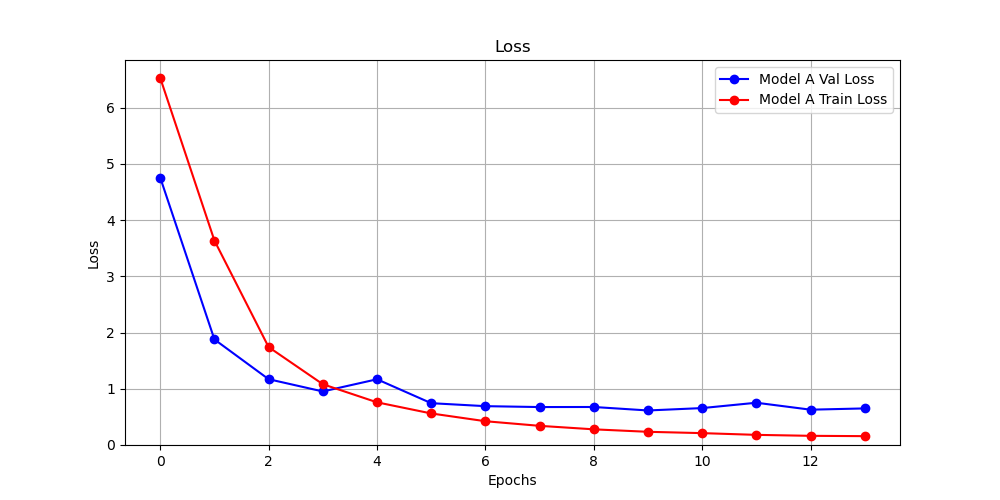
与Dropout不同的是Dropout对全连接层效果更好，全连接层通常包含大量的参数，训练过程中随机将一部分神经元置0，减少神经元之间的依赖关系，从而减少过拟合，而对于卷积层，通常用于提取图像的特征，参数数量比较少，本身就不需要过多的正则化，用Batch\_Normalization可以减少梯度消失和梯度爆炸问题，还有助于提高特征的稳定性和表示能力，已经足够了。

但是添加Batch\_Normalization层之后却得到了如下结果：



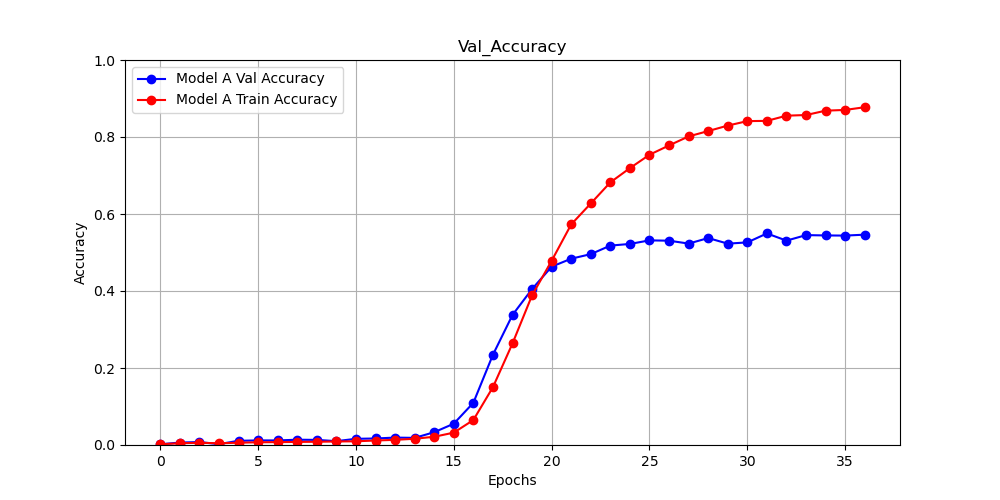
存在比较明显的波动情况，我认为是因为Batch\_Normalization层可以加速模型的收敛速度，使得模型更快地学习到有效的特征表示，从而可能需要更小的学习率来避免训练过程中的不稳定性。降低学习率后达了到了下图的效果，更加稳定的模型学习：





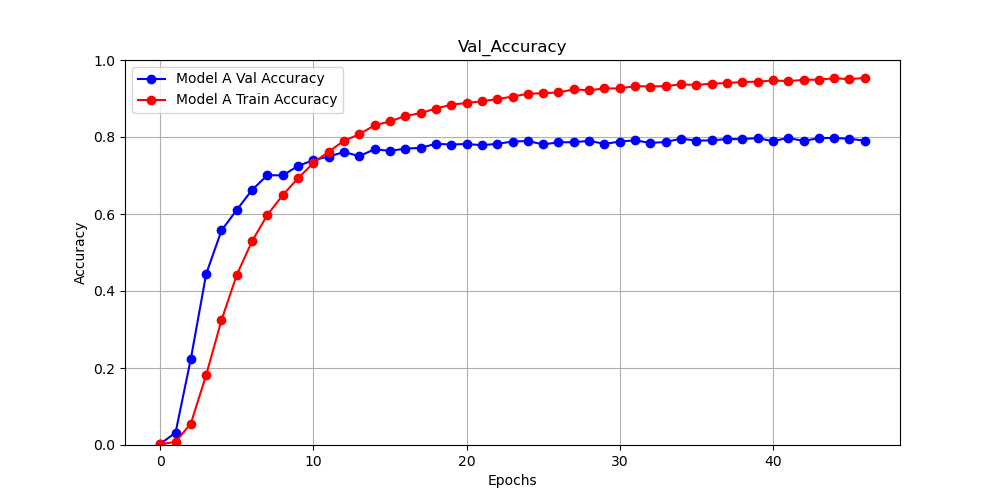
**3.5 对模型深度的尝试**

最初，考虑到电脑的性能（轻薄本，无GPU），我先进行了3个卷积层、3个池化层，2个全连接层的搭建，且数据预处理仅用resize修改为需要的尺寸，无正则化，归一化，得到的结果：



验证集准确率大约在0.55左右，损失高达2。

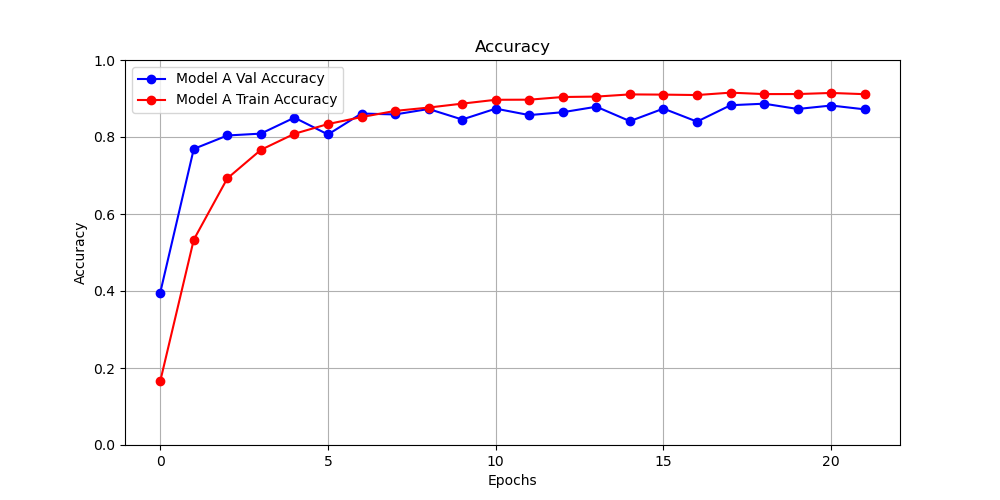
但是在加入了图像预处理、正则化，修改了batch\_size和学习率后，模型训练曲线：

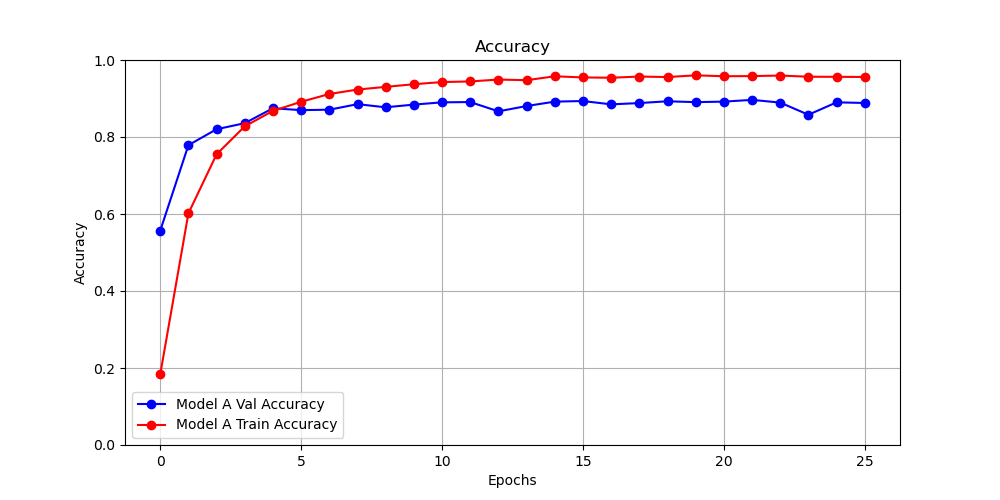


模型在验证集上准确率趋于0.8，损失也降到1左右。

现在仅靠“小技巧”很难让模型能力更强了，因此尝试对模型的深度（模型的复杂度）进行调整，在训练耗时从1个epoch需100s，到210s，再到300s的同时，模型拟合能力也有了一定程度的提升。

以下一次是增加到6个卷积层和9个卷积层（最终版本）的效果，可以看出，在准确率和损失上，9层比6层略优，而由于模型更复杂时需要更低的学习率达到收敛，9层在降低学习率后所得到的训练曲线也更加稳定：





**3.6 对ResNet的尝试**

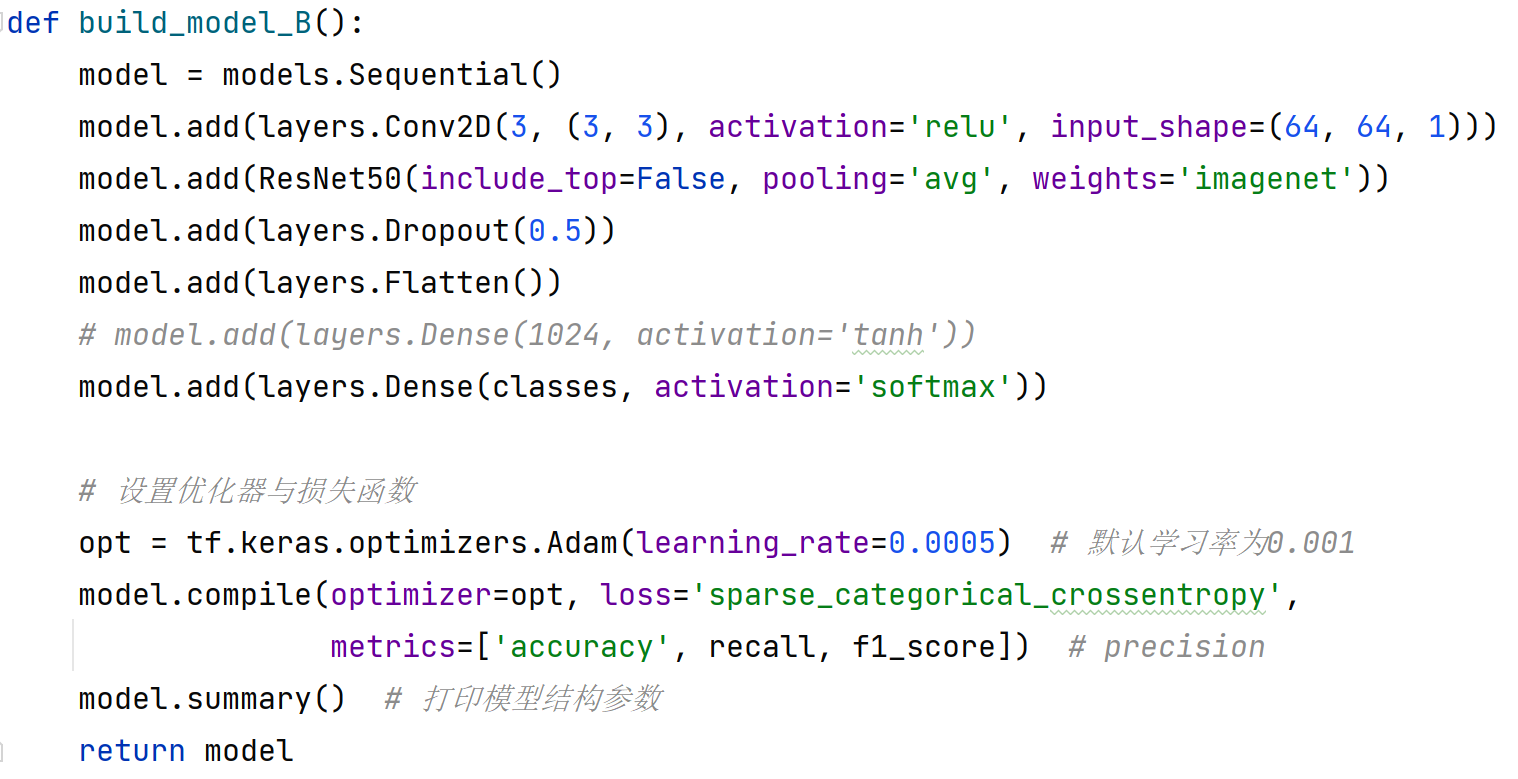
ResNet通过引入残差连接来解决深层神经网络训练中的梯度消失和梯度爆炸问题，使得可以训练更深的网络结构。

我尝试了ResNet50与ResNet18两种残差网络。

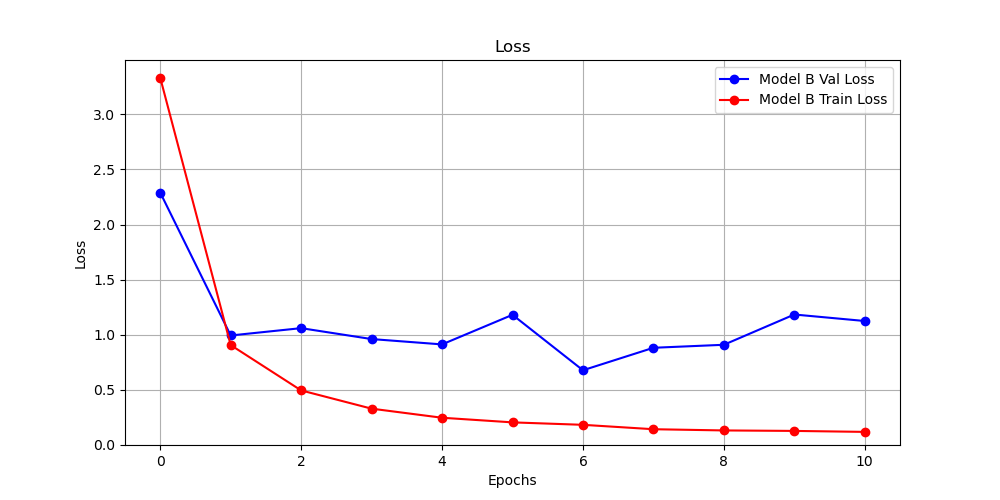
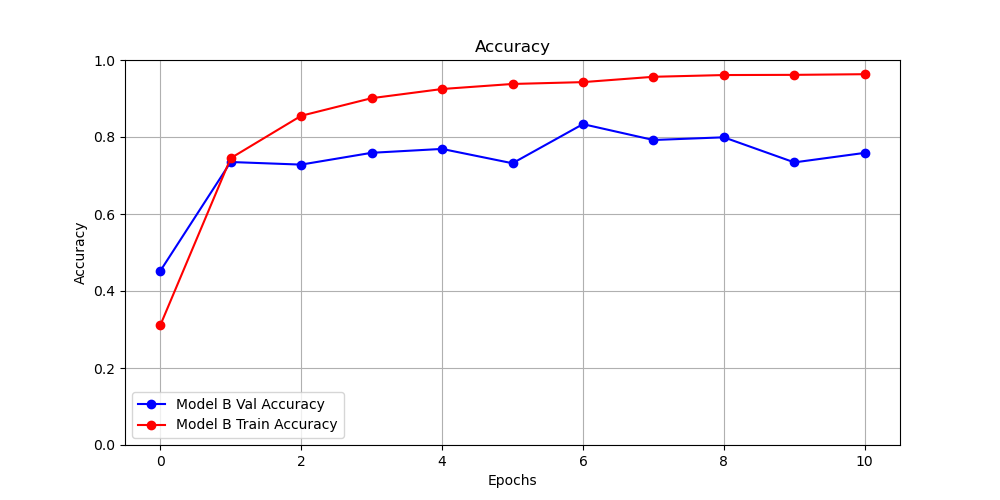
**3.6.1 ResNet50**

调用了keras中的**ResNet50**的函数，由于其要求输入为3通道图像，而我都输入时灰度图像（只有1通道），因此在ResNet前设置了一个卷积，将其转换为3个特征图。取消了ResNet50自带的全连接层。保留了展平层前的Dropout层，希望能用于减少过拟合。

训练时1个epoch的耗时大约在900-1000s。



得到训练曲线：



模型性能相比我自己搭建的CNN要略微差一些，epoch>6后测试集摇摆准确率不稳定，我认为可能有两个原因：

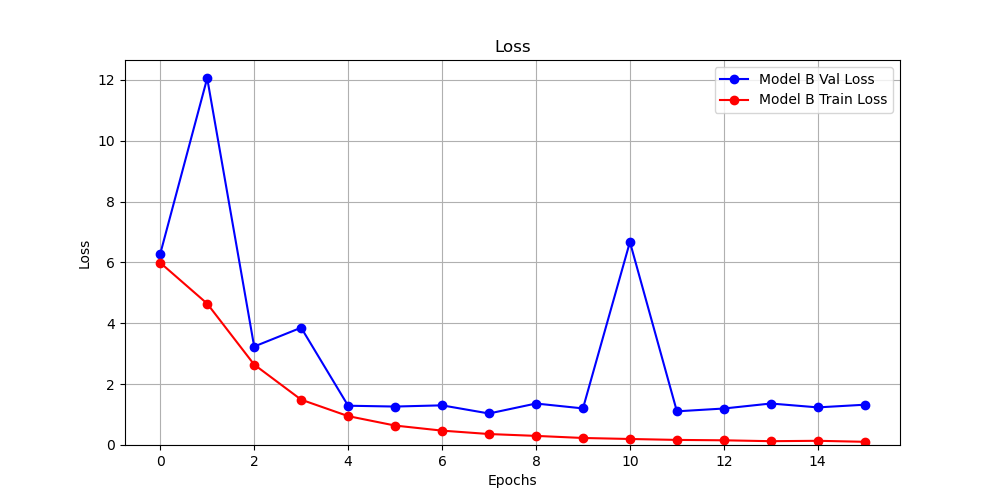
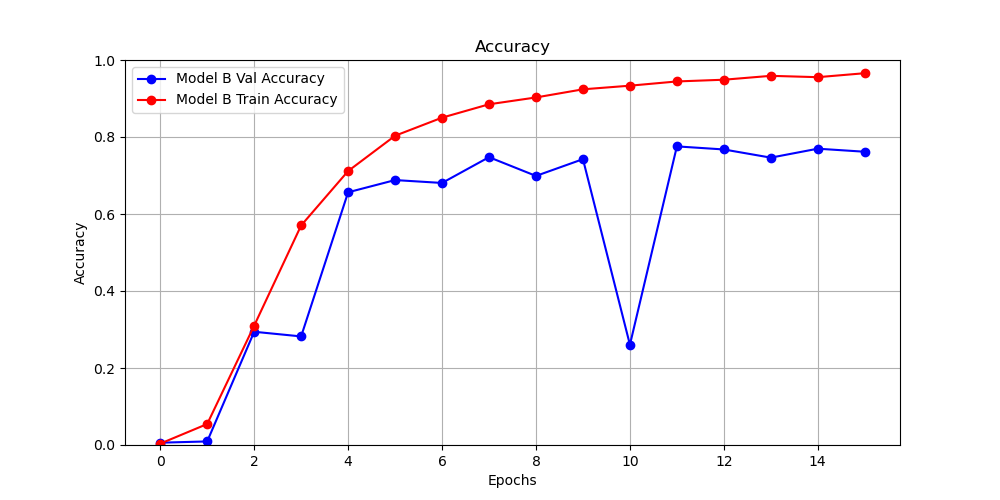
①ResNet50本身已经是一个非常深的网络结构，又采用了较大的学习率导致模型训练情况不稳定。

②ResNet50具有强大的特征学习能力和泛化能力，而且已经通过跳跃连接和批量归一化（BN层）等技术进行了有效的正则化。我添加的Dropout层可能会导致训练过程变得更加困难，同时降低网络的性能。

综合考虑，我将模型换成了结构更加简单的残差网络——**ResNet18**。

**3.6.1 ResNet18**

效果不如我自己搭的CNN，且考虑到训练耗时1epoch需要500s左右，因此没有采用：



**4 模型测试部分设计**

模型测试部分按顺序由数据读入与预处理、训练好的模型载入、预测、解读预测结果组成：



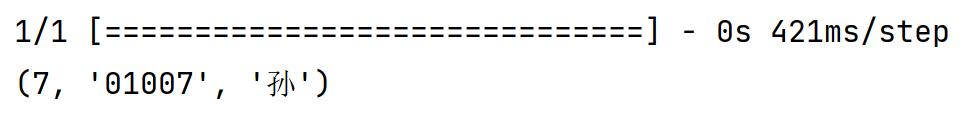
**数据预处理**片依旧是统一按照长边等比例放缩至64（pixel），进行一次直方图均衡化，将放缩后的图片按照短边居中，加白边补齐64（pixel）,并转换为灰度图像。

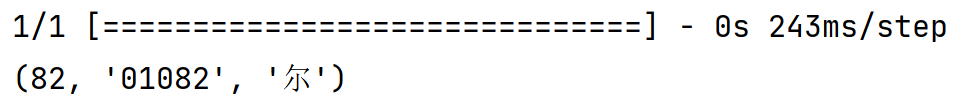
加载的模型“best\_model\_A.h5”是训练过程中通过回调ModelCheckpoint监视ModelA的验证集上的准确率，保存在“model”文件夹的验证集准确率最高的模型。

读取的json表格是“Char\_dict.json”,用于查找标签对应的汉字。

最终输出模型中存储的标签、实际的标签、汉字。

比如：

**** 

**5 实验心得**

本次课程设计中我共完整训练了19个模型，手动中途停止了4个，平均一个模型20个epoch，平均1个epoch耗时400s，其实还有很多模型的搭建想法，但是受限于电脑性能和时间，只能在这里停下了。

因为是第一次做类别这么多的神经网络分类项目，而且数据量也比较大，所以仔细考虑过数据划分（原本是73分，后面改成82分以后训练效果变好了，可能是因为训练量变大了）与数据预处理，尽量在不影响训练出的模型性能的情况下减小数据量，降低训练时间。现在看来效果还不错。

相比直接随意地调用一个现有的模型，自建CNN需要花更很多时间用于防止过拟合的措施进行调整、对模型深度进行调整、对CNN的层顺序进行调等整，也增强了我调整模型结构、优化超参数的能力。

但是现在看来还有很多地方需要改进：

包括模型效果的可视化，尝试了混淆矩阵和热力图，但是混淆矩阵由于横竖坐标都是500，而我没有想到一个合适的减小坐标长度的方法，最终也没有放进报告。以及，没能妥善运用**云平台**，也是这次课程设计的一个遗憾之处。我也尝试引入**Transformer**，但是因为代码能力还是不足，最终没能成功运行。

总之这次课程设计使我度过了充实的一周，心情随着模型性能上下起伏的过程虽然痛苦，但也满足。

感谢你看到这里。