**课题执行过程中的问题及解决方法**

1. **最开始尝试机器学习 以手写数字识别练手：**

验证运行手写数字识别样例时发现正确率很低，约为40%的正确率，在确保格式按照张量的格式调整之后，我们仔细阅读代码以及将归一化的图片打印后对比我们发现我们似乎差一步对归一化后的图片取反（所以我们之前识别的图片可以说原本识别的黑色的笔记但我们的图片其实是白色的笔迹，空白变成了数字，这也解释了我们手写的数字“1”都被识别成了数字”0”）。

1. **传统方法识别音频**

在验证我们使用数理方法音准的识别准确率时，我们发现效果不够理想，于是将识别错误的音频进行分析，发现在第一步音频裁剪就出现了问题，所以我们针对这些音频优化了切割模型，之后又调整了当频域计算和时域计算不统一时的处理机制，将两计算结果不统一时，计算系数的阈值调大，使更大比例的音以频域计算的结果返回，提高了近5%的准确率。

1. **深度学习输入层数据格式**

我们最先想输入3维张量，但仔细思考后发现每一个坐标点对应的意义不明确，导致无法训练出对应的网络，和老师讨论、思考之后我们用类似彩色图片的输入（包含rgb3个参数）形式输入。

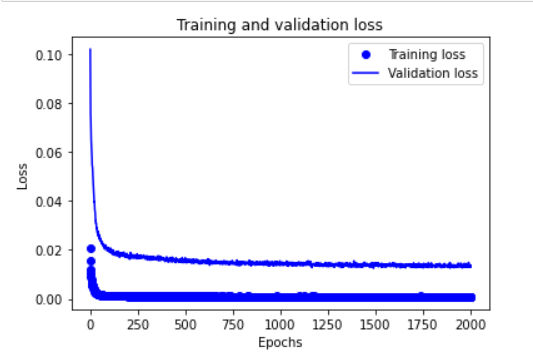
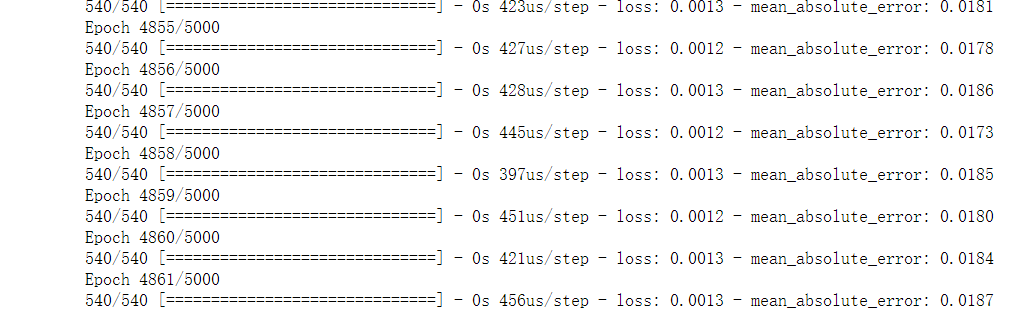
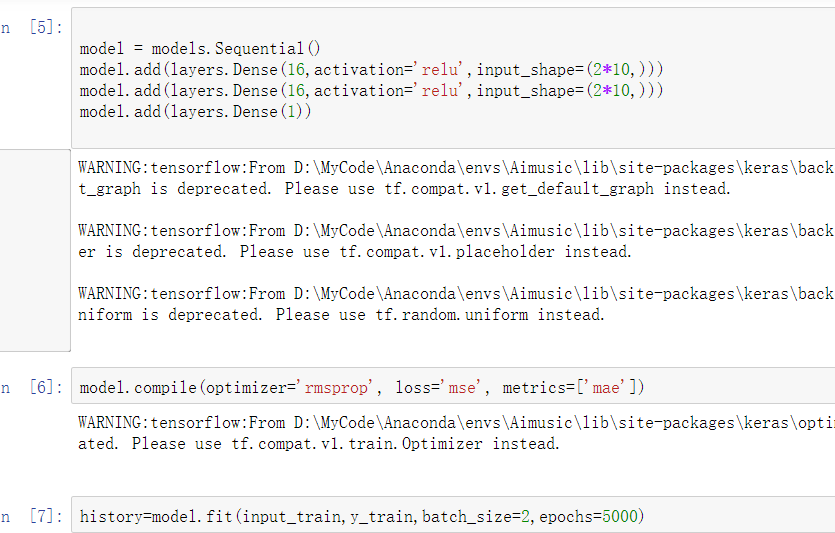
**4.特征工程处理**

我们利用librosa的峰值检测函数，设定好一定的参数，检测出峰，但发现有些频率对应的峰值过高或者过低（不在我们要求的频域范围内），而且还出现了负值反向峰值的情况（因为我们之前对数据进行过处理，使用了分贝的形式，log对数导致出现了负数），我们重新预处理了数据，并进行了归一化的处理。

**5.模型构建的报错**

刚开始构建网络的时候一直报错，在反复参考书中的示例代码以及查看所调用函数的函数说明后，我们修改出了正确输入层的维度参数，并且将数据转换为numpy的array格式后成功运行。

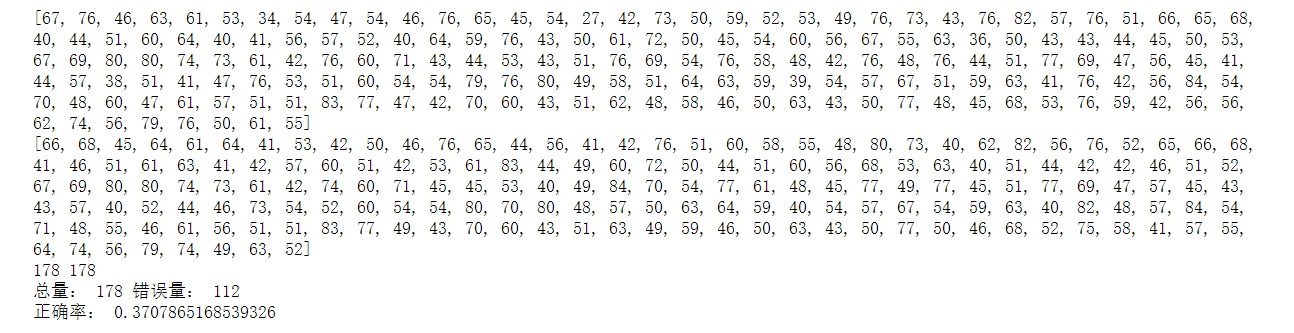
**6.训练结果很不理想**



在最初训练的时候，效果十分不理想，损失函数下降到一定时便不再下降（约0.1左右），我们仔细检查归一化的数据以及作为输入的峰值检测的数据，发现频率过高的许多音并未检测到峰值，导致代表该音的整个序列都是0作为输入，而这样样本的输入会导致输出结果很奇怪，而且我们的模型前一个输入会影响后边的序列，导致训练出的结果很差。

于是，我们考虑去掉上述的数据，最终得到的结果如上图。

然后我们将预测到的频率值（目前是回归模型）转换成音高，与数据集的标签比较，计算出正确率，但正确率也不高。



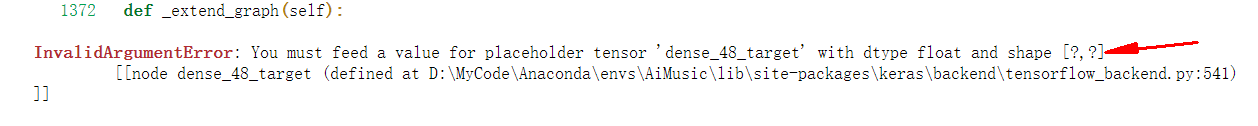
我们思考可能对于机器学习的回归模型不断优化的是预测出频率，而不是音高，所以可能分类模型更加适合。

还有一点我们也认为是原因所在，模型是根据loss函数优化的，而我们最终评估的其实是音高，但我们目前也没有弄清如何将自己的验证函数放入机器学习的损失函数，这两个问题有待解决。

**7.自编损失函数**

之前的2D、3D张量模型效果不尽人意（正确率只有不到百分之20），于是我们重新从输入开始检查，发现输入数据是并未进行音频裁剪的预处理的数据，于是我们重新对所有音频进行了裁切的预处理，再归一化，送入模型，得到了正确的预测率分别为51%、57%，但还说未能达到我们的预期，于是我们思考，可能是损失函数未能对应上，因为对于回归模型，我们采用的是relu作为损失函数，其实可以看见损失最终是收敛的，也没有出现过拟合的情况，但其实我们真正的输出值并非是一个频率点，而是一个包含一定频率范围的“音高”，于是，这可能导致损失的求法有所出入，于是我们查询了损失函数的接口，其中输入分别是一个真值一个输入，返回值即当前情况下的距离（或称作损失），本身此函数是不难的，但复杂的是，传入的参数数据类型是tensor（张量），在计算的时候不同于简单的数、向量，并且出现了在运算中张量的shape是未知的情况，这导致我们在尝试自定义损失函数的时候遇到了阻碍，目前还未解决。

而同时，我们思考，可能如今的输入数据也需要调整，因为我们目前的打算是让机器学习处理时域和频域两种计算出的有出入的数据，（但如果机器学习的效果很好，我们可以考虑直接替代），所有我们可能需要调整输入数据，把训练样本就干脆换成有出入的数据。



**8.放弃特征工程**

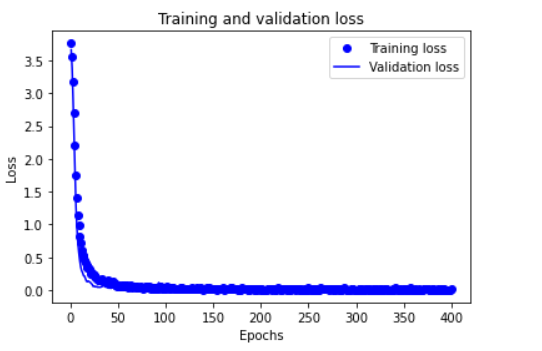
在前几周中，我们尝试了频域做特征工程，将10个峰作为输入、将时域采样1600个样本点作为输入、尝试了回归模型、分类模型、3D向量、2D向量的变换、CNN预测的模型…但之前的尝试都没有取得比较好的效果，我们也一度有些低落和怀疑。

不过在老师的建议下“可能有些时候最简单的方法也许就能完成”，于是我也进行排除，发现我们还有一条直接把频域数据归一化后输入这一条路没走，并不进行更进一步的特征提取，在训练时我也预感这次应该会有很好的结果，果然，这次的正确率比较令人满意。

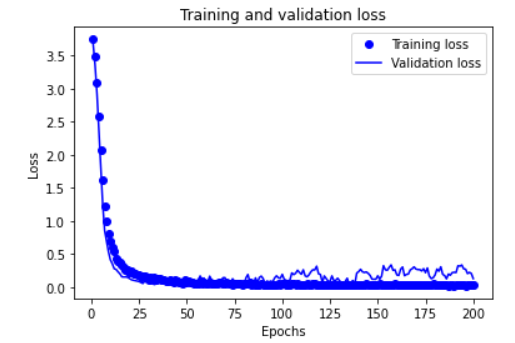
但我们也觉得之前的工作也没有白做，不仅熟悉了修改网络、搭建模型的过程，也让我们可以在确定了输入数据之后，可以在各个网络中进行选择和优化，而且也一定程度上磨砺了我们的耐心。

**9.过拟合问题**

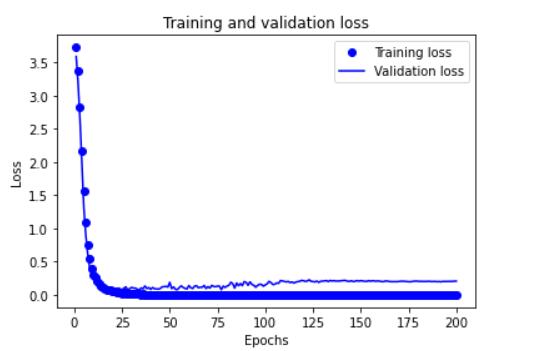
在上一周我们已经优化了模型，大幅提升了预测率，但我们发现我们的模型输出的图像始终没有出现过拟合的情况，始终只是下降有点奇怪



于是我们仔细核对DeepLearning的参考书，发现我们并未将输入样本分出测试集，于是我们重新加入测试集发现了过拟合的情况：

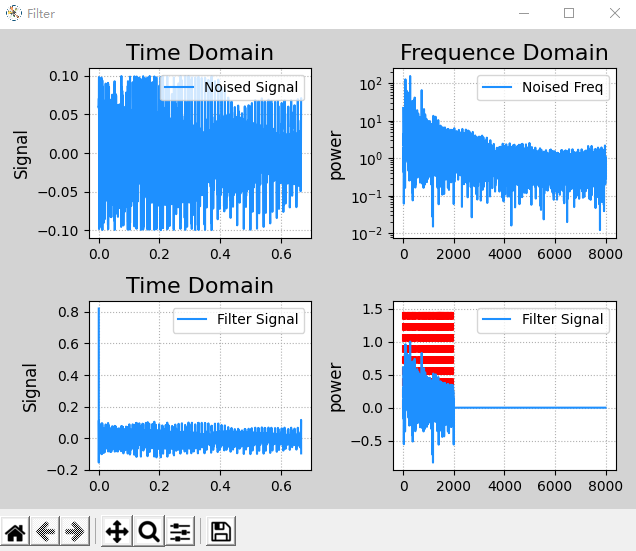
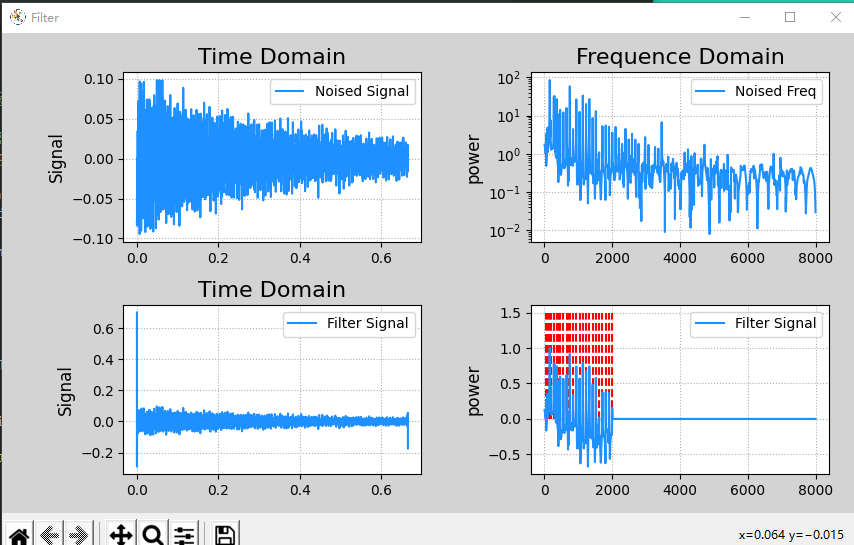


所以我们打算减小模型的过拟合情况，书中介绍了很多种思路，比如K-折法（针对数据量不大的情况）、增加Drop层，丢弃一些层，最后我们采用了增添drop层的方法，已经取得了比较好的抑制过拟合的效果，最后使预测率也上升了百分之1。



**10.杂音问题**

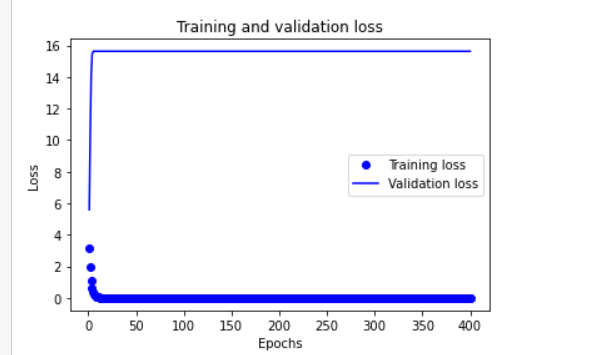
通过测试，我们发现将两个音在时域求和后就等效于同时弹奏两个音的和弦（我们目前以两个音合成的和弦做为第一步进行识别）。但我们发现，如果只进行直接求和而不进行其他操作的话，音频会出现“呲呲”的杂音，考虑到我们进行音频合成之前首先对音频进行了切割，为了保证同步，我们以音的幅值最大的点作为起始点，但这也导致了合成出来的音频幅值过大。如下图所示



于是我们在求和之后对整体音频进行了减弱再输出，部分缓解了出现“杂音”的问题。

**11.数据未打乱**

在第一次输入样本进行训练的时候，我们发现效果出奇的好，但损失却很大

而且损失是不断增加的。

然后我们检查了音频，发现音质非常糟糕，仔细检查后才知道，原来我们在合成和弦音频后，并没有对音频进行打乱，所以其实是顺序合成的，这就导致机器学习其实学到了一种最简单的规律那就是线性增加，在加入random函数打乱样本后才恢复。

**12.和弦合成音质问题**

针对音质问题，我们最先截取音频时域幅值最大值作为起始点进行合成，但这样的效果并不好，部分声音出现了破音，然后我们降低了音频的分贝，但效果仍不理想，最后我们尝试错开了两个音（相差40ms）发现这样的效果比以前好了很多

**13 AI谱曲输入问题**

乐谱中的休止符难以表示以及不同乐曲的节奏（拍）不同

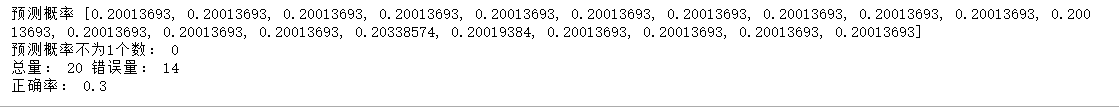
解决方法：将休止符同样地当作音高之一作为输入，乐曲的节奏体现在时长序列之中。

**14.数据错误问题**

数据集中有较大一部分的音符是存在问题的，这些问题是录制数据时产生的，我们需要将这些样本分离出来。

解决方法：利用机器学期分类模型，将不和谐的音符根据分类预测概率分离出来

**15．错误类型分类输入问题**

****

仿照之前的训练模型，我们将错误样本先进性了FFT的变换，将其频域值采样作为输入传入网络，但可见预测概率非常低，错误类型为5，但其预测概率几乎就是0.2，正确率也几乎是随机，在困扰了我们一段时间后，我们认为应该是输入的样本出现了问题，我们人工进行分类的标签是通过人工听觉分辨的错误类型：

0：音高弹错

1：弦没按紧

2：打到琴板

3：破音

4：按到品丝

但我们却把频域输入了网络，我们决定先尝试输入时域采样，或者输入一些其他特征，比如：过零率等重新进行训练。

**16.音频演奏问题**

在引入mingus库的时候遇到了很多困难，比如环境变量、原代码缺少类型转换，输出的音频格式不对，乐器类型不知，而且这个库在网络上的资料几乎没有，但我们都一步一步通过读代码、看原码、查找代码文件等方式耐心地解决了