期中任务	神经网络回归模型	组别
1	Machine Learning	#3

机器学习报告

小组#3

小组成员：帅灿宇 20354240

万俊麟 20354123

廖逸霖 20354079

任务完成概况：

1 使用多种超参数组合训练模型得到最佳组合：

Table: Best Hyper-parameters

Hyper-parameter	max_iter	solver	Learning_rate	early-stopping	Alpha
	2000	Sgd	‘adaptive’	False	10^{-5}
Hyper-parameter	tol	momentum	Hidden_layer_sizes	learning_rate_init	
	10^{-8}	0.95	（130, 110, 50）	0.0012	

2 使用得到的最佳超参数组合训练模型得到神经隐层尺寸为（130， 110， 50）的神经网络分类模型。

3 描述模型训练算法

4 模型训练结果：

Table： Result（test mape 由训练集 9:1 划分获得的测试集测得）

	Test MAPE Mean	Train MAPE Mean	Test MSE Mean	Train MSE Mean	Estimator Loss
Test1	0.106378	0.013793	9.939504	0.193662	0.193662
Test2	0.113266	0.014569	10.055687	0.251675	0.251675
Test3	0.117263	0.016853	12.942200	0.321466	0.321466
Test4	0.124841	0.016232	13.354804	0.290457	0.290457
Test5	0.108457	0.018222	10.394702	0.367046	0.367046

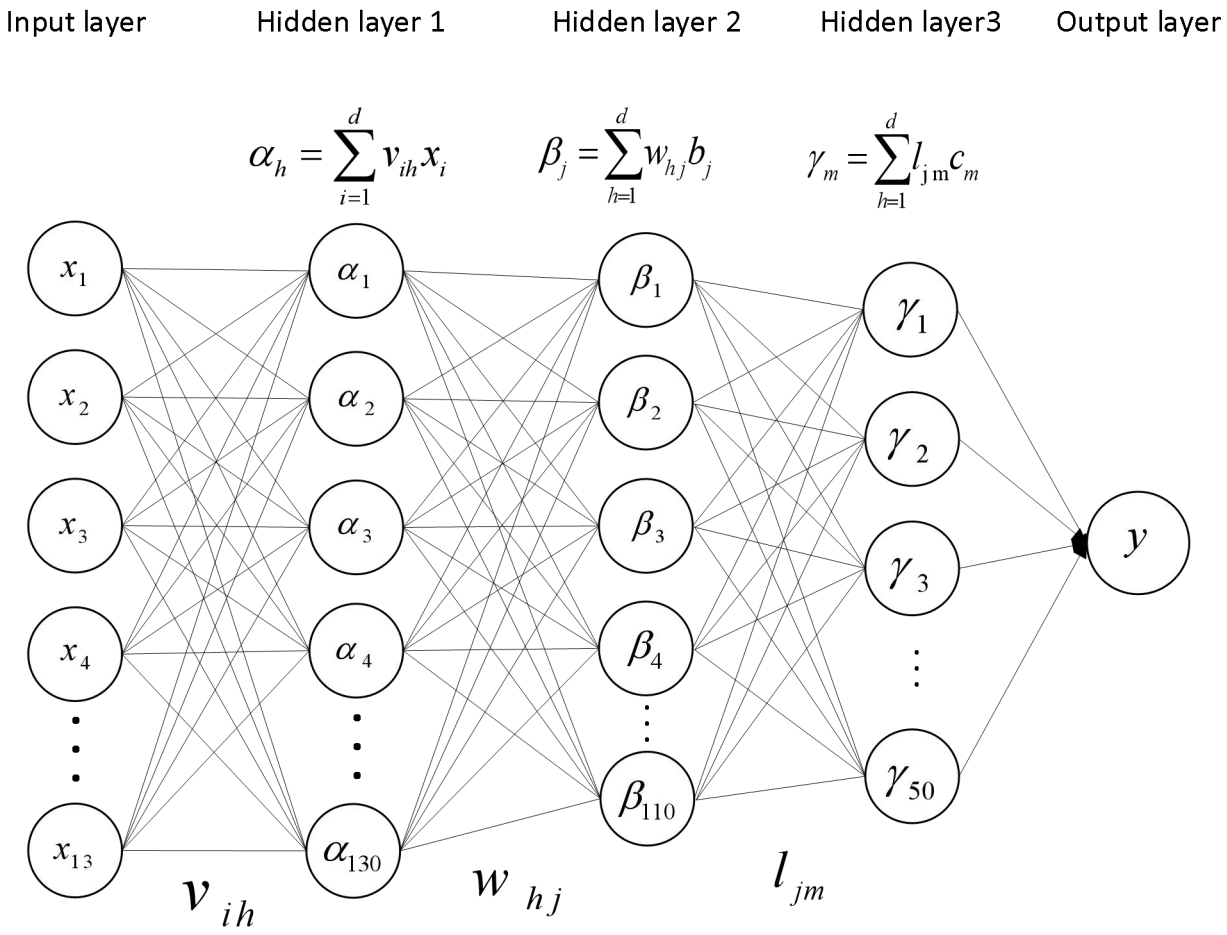
Best Test MAPE: 0.015692（训练集作为测试集测得）

目录

1 神经网络模型

我们经过多种超参数的组合进行训练，最后得到的最佳模型为 3 层隐层神经元个数分别为（130,110,50）的神经网络模型。

对数据处理后训练得到的回归器神经网络模型如下：



2 模型训练算法描述

2.1.1 随机梯度下降法（固定学习率）

分析可知，超平面优化问题即找到最佳的参数组合(ω^*, b^*)，使得均方误差 MSE 最小，

由随机梯度下降法求解最佳参数的算法步骤为：

算法随机梯度下降(SGD)在第 k 个训练迭代的更新

Require: 学习率 α_k

Require: 初始参数 θ

while 停止准则未满足 do

从训练集中采包含 m 个样本 $\{x^{(1)},...,x^{(m)}\}$ 的小批量，其中 $x^{(i)}$ 对应目标为 $y^{(i)}$。

计算梯度估计: $\hat{g} \leftarrow +\frac{1}{m}\nabla_\theta \sum_i L(f(x^{(i)};\theta),y^{(i)})$

应用更新: $\theta \leftarrow \theta - \alpha\hat{g}$

end

问题：

我们发现，使用固定学习率的随机梯度下降算法存在一定问题,对于某些参数，通过算法已经优化到了极小值附近，但是有的参数仍然有着很大的梯度。如果学习率太小，则梯度很大的参数会有一个很慢的收敛速度；如果学习率太大，则参数可能会出现不稳定的情况。

策略：

使用自适应学习率算法，即对每个参与训练的参数设置不同的学习率，在整个学习过程中通过一些算法自动适应这些参数的学习率，以保证模型训练过程中算法迭代不会因学习率过小而收敛过慢，也不因学习率过大而出现不稳定。

2.1.2 随机梯度下降法（带 Nesterov 动量）

动量算法描述（Momentum 算法）

Require: 学习率 ϵ，动量参数 α

Require: 初始参数 $\boldsymbol{\theta}$，初始速度 $\boldsymbol{v}$

while 没有达到停止准则 **do**

从训练集中采包含 m 个样本 $\{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(m)}\}$ 的小批量，对应目标为 $\boldsymbol{y}^{(i)}$。

计算梯度估计：$\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}),\boldsymbol{y}^{(i)})$

计算速度更新：$\boldsymbol{v} \leftarrow \alpha\boldsymbol{v} - \epsilon\boldsymbol{g}$

应用更新：$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$

end while

原始 SGD 每次更新的步长只是梯度乘以学习率；现在，步长还取决于历史梯度序列的大小和排列；当许多连续的梯度指向相同的方向时，步长会被不断增大；动量算法使用带有动量的梯度，而不是当前梯度来对 w 进行更新。相当于每次在进行参数更新的时候，都会将之前的速度考虑进来，每个参数在各方向上的移动幅度不仅取决于当前的梯度，还取决于过去各个梯度在各个方向上是否一致，如果一个梯度一直沿着当前方向进行更新，那么每次更新的幅度就越来越大，如果一个梯度在一个方向上不断变化，那么其更新幅度就会被衰减，这样我们就可以使用一个较大的学习率，使得收敛更快，同时梯度比较大的方向就会因为动量的关系每次更新的幅度减少。

带 Nesterov 动量算法描述

Require: 全局学习率 ϵ，衰减速率 ρ，动量系数 α

Require: 初始参数 $\boldsymbol{\theta}$，初始参数 $\boldsymbol{v}$

初始化累积变量 $\boldsymbol{r} = 0$

while 没有达到停止准则 **do**

从训练集中采包含 m 个样本 $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ 的小批量，对应目标为 $\boldsymbol{y}^{(i)}$。

计算临时更新：$\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta} + \alpha \boldsymbol{v}$

计算梯度：$\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\tilde{\boldsymbol{\theta}}} \sum_i L(f(\boldsymbol{x}^{(i)}; \tilde{\boldsymbol{\theta}}), \boldsymbol{y}^{(i)})$

累积梯度：$\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1 - \rho) \boldsymbol{g} \odot \boldsymbol{g}$

计算速度更新：$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \frac{\epsilon}{\sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$ （$\frac{1}{\sqrt{\boldsymbol{r}}}$ 逐元素应用）

应用更新：$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$

end while

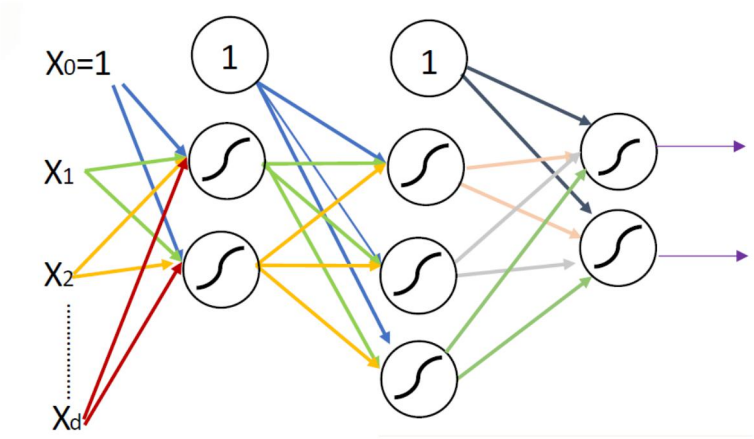
Nesterov 动量可以解释为往标准动量方法中添加了一个修正因子，带 Nesterov 动量的梯度下降算法把梯度计算放在对参数施加当前速度之后。这个“提前量”的设计让算法有了对前方环境“预判”的能力，与 Momentum 算法唯一区别就是，计算梯度的不同，Nesterov 先用当前的速度 v 更新一遍参数，在用更新的临时参数计算梯度。相当于添加了矫正因子的 Momentum 算法。

2.2 神经网络权重优化

BP 算法

算法基本思想：

BP 算法（误差逆传播算法）利用输出后的误差来估计输出层的前一层的误差，再用这个误差估计更前一层的误差，如此一层一层地反传下去，从而获得所有其它各层的误差估计。再根据估计的误差值对网络进行训练，直至得到一个较好的网络模型。



算法原理（在假设有 t 层神经元的情况下）：

① 针对输入网络的数据集 $\{(\boldsymbol{x}_1,\boldsymbol{y}_1),(\boldsymbol{x}_2,\boldsymbol{y}_2),\cdots(\boldsymbol{x}_m,\boldsymbol{y}_m)\}$，根据原定的初始网络模型经过激活函数计算出初始的输出值 $\hat{y}_n = g(z_n^{(t)})$，并计算初始误差 E_m。本问题中损失函数使用均方误差。

$$E_m = \frac{1}{2}\sum_{n=1}^{dt}(\hat{y}_n^m - y_n^m)^2$$

② 采用广义的感知机学习规则，基于梯度下降的策略，BP 算法以目标的负梯度方向对权重进行调整：

$$\omega_{kn}^{(t)} = \omega_{kn}^{(t)} - \eta\delta_n^{(t)}x_k^{(t-1)}$$

其中 η 为学习率，代表每一轮迭代的学习步长，一般在 0-1 之间。

对于 δ_n^t，有：

$$\delta_n^t = (\hat{y}_n^m - y_n^m)\bullet y_n^m \bullet (1-\hat{y}_n^m)\bullet x_k^{(t-1)}$$

③ 从第 t 层神经元的权重不断前推，直至修改所有层的权重，同时动态修改 δ_n^t 值经过整一个阶段后，误差（损失值）会有所下降。

④ 不断重复②③两步，使误差（损失值）不断下降至所规定的误差范围 ε 内。

3 模型训练过程

3.1 超参数确定过程

模型训练过程中的超参数包括：

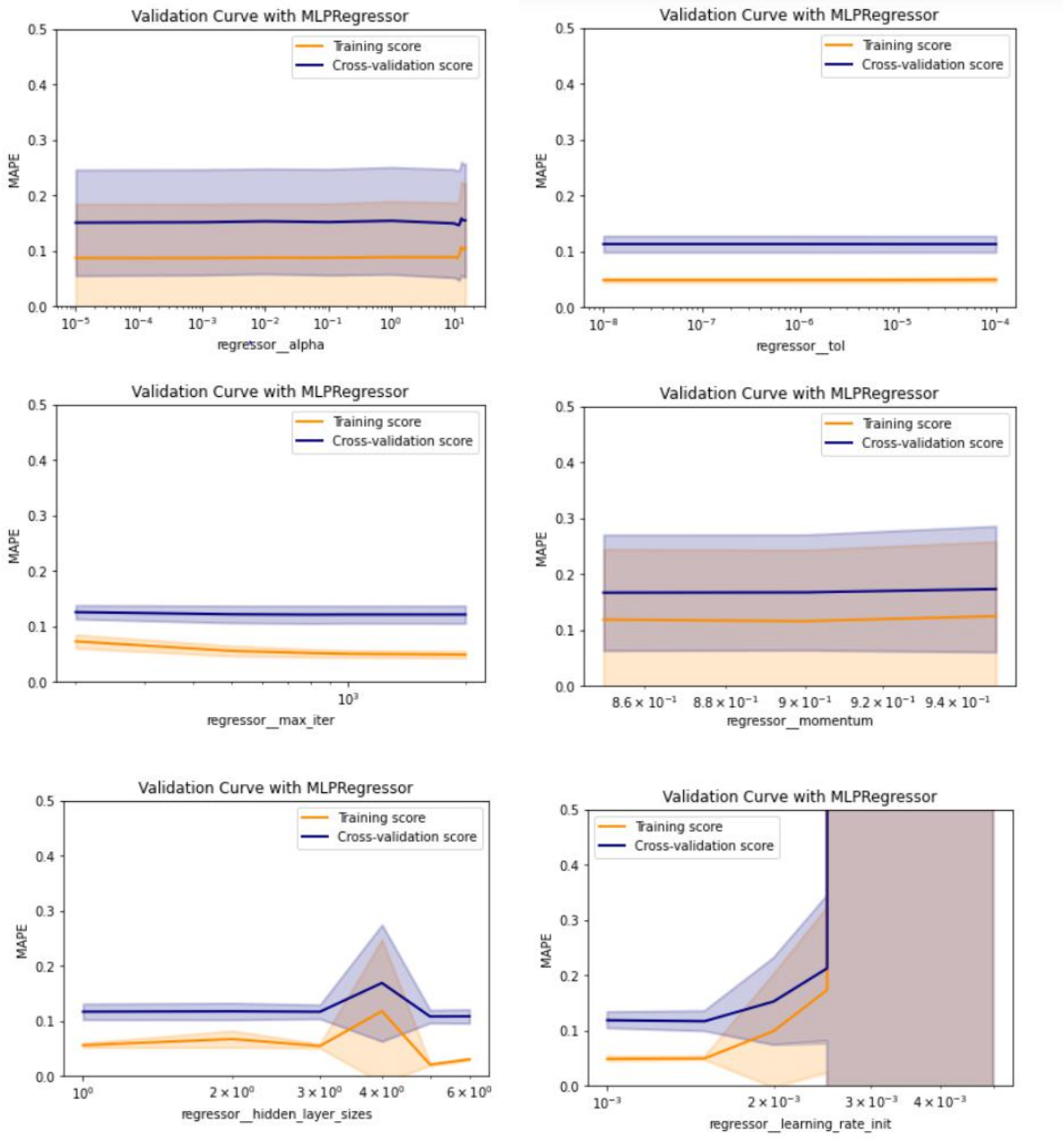
Table: Hyper-parameter

max_iter	最大迭代次数
solver	权重优化的求解算法，包括{‘lbfgs’, ‘sgd, ‘adamm’}三种算法
Learning_rate	‘adaptve’，自适应学习率
early-stopping	当验证评分没有改善时，是否使用提前停止来终止培训。
alpha	L2（欧氏距离）正则化惩罚参数
learning_rate_init	初始学习率，控制权重更新步长
tol	权重优化容忍度，容差优化
momentum	梯度下降更新的动量
Hidden_layer_sizes	隐层神经层尺寸

对于超参数的设置，我们先设置多种超参数组合进行网格搜索，通过比对训练结果的情况后确定超参数设置的大致范围，再在该范围内对参数进行细调（细调详细过程在 jupyter notebook 中）：

Hyper-parameter	
max_iter	[200, 500, 800]
solver	{‘sgd’, ‘lgbfs’}
Learning_rate	‘adaptive’
early_stopping	{‘True’, ‘False’}
alpha	10^k, $(-5 \le k \le 1)$
learning_rate_init	$k*10^{-3}$, $(1 \le k \le 15)$
tol	10^k, $(-8 \le k \le -4)$
momentum	[0.85, 0.9, 0.95]
Hidden_layer_sizes	(13, 3), (9, 5), (13, 11), (13, 11, 5), (130, 110, 50)

超参数对模型训练结果的 mape 值影响如下：



通过所有超参数组合的训练，我们得到最佳的超参数组合如下：

Table: Best Hyper-parameters

Hyper-parameter	max_iter	solver	Learning_rate	early-stopping	Alpha
	2000	Sgd	‘adaptive’	False	10^{-5}
Hyper-parameter	tol	momentum	Hidden_layer_sizes	learning_rate_init	
	10^{-8}	0.95	（130, 110, 50）	0.0012	

3.2 模型训练结果

五次五折交叉验证结果：

得到的最佳训练超参数组合后，用同样的参数训练模型，对数据进行五折交叉验证处理划分训练集以及测试集，分别测试得到五次五折训练结果的模型对应平均绝对百分误差mape值

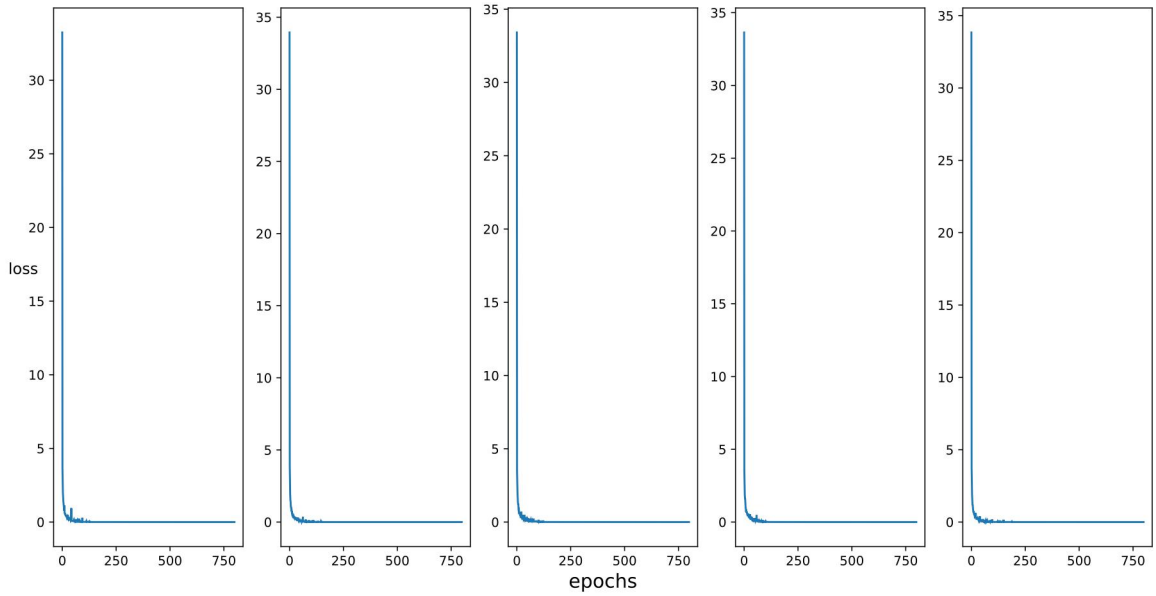
$$mape = \left(\frac{100}{n}\right)\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \qquad (A_t: 真实值，\ F_t: 预测值)$$

Table： Result（test mape 由训练集 9:1 划分获得的测试集测得）

	Test MAPE Mean	Train MAPE Mean	Test MSE Mean	Train MSE Mean	Estimator Loss
Test1	0.106378	0.013793	9.939504	0.193662	0.193662
Test2	0.113266	0.014569	10.055687	0.251675	0.251675
Test3	0.117263	0.016853	12.942200	0.321466	0.321466
Test4	0.124841	0.016232	13.354804	0.290457	0.290457
Test5	0.108457	0.018222	10.394702	0.367046	0.367046

Best Test MAPE: 0.015692（训练集作为测试集测得）

得到均方损失MSE随迭代次数增加变化情况如下：



可以看出，随迭代次数增加，模型的均方损失 MSE 减小，在迭代次数接近 500 次时，均方损失 MSE 变化趋于平缓。

5 模型训练收获

使用神经网络处理回归问题，在选择超参数组合时，考虑使用多种算法、迭代次数、动量算法的动量参数等参数组合，经过多种参数组合的拟合选出最佳超参数。

我们经过多种超参数的组合进行训练，最后得到的最佳模型的隐层神经元尺寸为 2 层神经元个数分别为（13，11）的神经层，但其平均绝对百分误差 mape 值大于 10%，因此，我们考虑分别增加神经元层数至三层，尺寸为（130，110，50），最终得到的神经网络模型的平均绝对百分误差 mape 值降低到了 6%，相比拟合程度更佳。但要想得到更优的模型，我们选择增加隐层神经元的个数，而神经元层数依旧为两层，两层隐层神经元的个数为（130，110，50），最后得出的神经网络模型回归误差 mape 值降低到小于 1.8%，拟合程度又优化了很多。

故当处理回归问题时，要得出拟合度最佳的神经网络模型，可以考虑适当增加隐层神经元的个数，也可以考虑增加神经元个数，但还需要考虑过拟合的情况，因此增加到合适的尺寸就不必继续增大神经元尺寸。

设置超参数 alpha 时要按数据多少设，和学习率大小也有关，学习率不能太大也不能太小，sgd 在 fine-tuned 精细调整情况下效果比 adam 好，但是 adam 不用太细调超参数就能获得比较好的效果，网络尺寸好坏只能靠交叉验证看出来，但是大尺寸训练成本远高于小尺寸而效果未必会改善太多，所以要做出权衡。