

人工智能导论实验报告

张钰晖

2015011372, yuhui-zh15@mails.tsinghua.edu.cn, 185-3888-2881

目录

1	问题描述	2
2	模型选择	2
2.1	模型 1 (MNIST FOR EXPERT)	2
2.2	模型 2 (LeNet-5)	2
3	细节实现	3
4	实验结果	5
4.1	模型的选择	5
4.2	卷积层层数的选择	7
4.3	训练集的选择	7
4.4	最终模型与竞赛成绩	7
5	实验心得	8
6	后期工作	8
7	源代码	9

1 问题描述

日常生活中，我们经常需要在填写信件、银行开卡填写表单的时候手写大量的数字，如身份证号、手机号、邮编等等。事实上，专业人员在过去录入这些信息非常费时费力，而且还容易出现错误。现在请你设计一个手写体数字照片智能分类程序，省去人工识别 0-9 数字的麻烦。

在 Kaggle 竞赛平台上有 Digit Recognizer <https://www.kaggle.com/c/digit-recognizer> 的经典任务，并且提供了上万张手写体数字照片的灰度数据（可以借助 Matlab 还原照片）。目前，该任务已经被许多科研人员设计的人工智能程序完美解决，达到 100% 测试精度。请大家在熟悉 Kaggle 平台的使用之后，直接通过该平台完成 Digit Recognizer 的竞赛任务，与一线科学家一较高下；并将你在从事竞赛过程中的心得体会、实验流程，以及最佳测评结果截图写入实验报告，同时在附录中递交你的代码。

2 模型选择

根据 Tensorflow 官方教程中 MNIST FOR EXPERT 中的 CNN 部分，和 Lenet-5 的结构，以及相关论文文献，最终选择了模型。

一般卷积神经网络均为 2 层卷积层，然后连接 2-3 层全连接层，中间层用 Relu 函数激活，输出层用 Softmax 函数激活。

实验分别以两大类方案进行实验，并对每类方案进行了参数调节，两大类方案如下，参数调节见后文。

2.1 模型 1 (MNIST FOR EXPERT)

输入层：长度为 784 的向量，展平为 28x28。(28x28)

第一层：卷积层一。用 32 个卷积核进行 Padding=SAME 类型卷积，进行 2*2 最大池化，Relu 函数激活。(32@14x14)

第二层：卷积层二。用 64 个卷积核进行 Padding=SAME 类型卷积，进行 2*2 最大池化，Relu 函数激活。(64@7x7)

第三层：全连接层一。用 1024 个神经元与上一层进行全连接，Relu 函数激活。(1024)

输出层：全连接层二。用 10 个神经元与上一层进行全连接，Softmax 函数激活。(10)

2.2 模型 2 (LeNet-5)

输入层：长度为 784 的向量，展平为 28x28。(28x28)

第一层：卷积层一。用 6 个卷积核进行 Padding=VALID 类型卷积，进行 2*2 最大池化，Relu 函数激活。(6@12x12)

第二层：卷积层二。用 16 个卷积核进行 Padding=VALID 类型卷积，进行 2*2 最大池化，Relu 函数激活。(16@4x4)

第三层：全连接层一。用 120 个神经元与上一层进行全连接，Relu 函数激活。(120)

第四层：全连接层二。用 84 个神经元与上一层进行全连接，Relu 函数激活。(84)

输出层：全连接层三。用 10 个神经元与上一层进行全连接，Softmax 函数激活。(10)

下图展示了图片大小为 32x32 时的 LaNet-5 神经网络

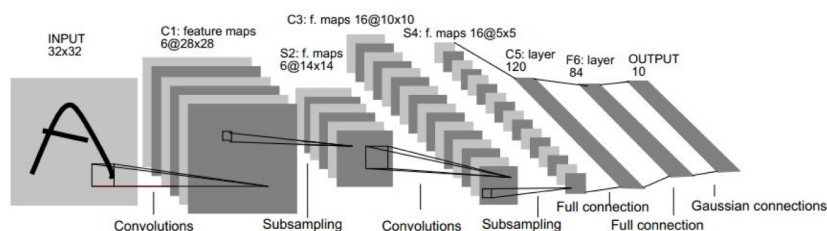


图 1: LaNet-5 神经网络

3 细节实现

作为开始，我们读进来所需要的数据。

train.csv 为训练集文件，包含 42000 行和 785 列。每一行代表一张手写的数字图片，第一列为对应数字的标签，剩下列为 784 个像素灰度值 (0255)，对应 28*28 的图片大小。

test.csv 为测试集文件，包含 18000 行和 784 列。每一行为 784 个像素灰度值 (0255)，对应 28*28 的图片大小。

将训练集样本部分抽取出来作为验证集，以检测神经网络训练结果。

至此我们得到了训练集、验证集和测试集。

下图展示了其中一个训练样本。



图 2: 一个训练样本

建立神经网络结构，进行卷积、池化。下图展示了卷积和池化的原理。

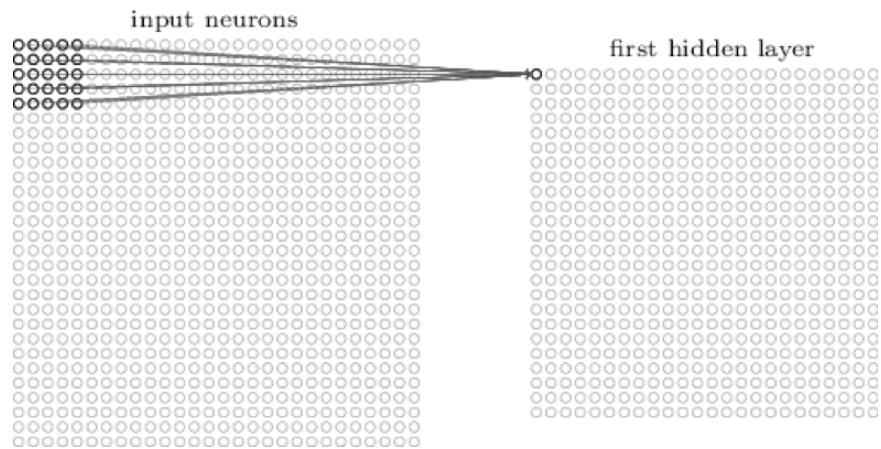


图 3: 卷积

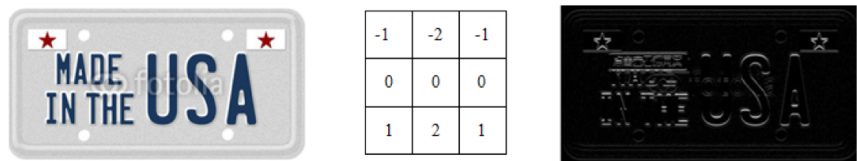


图 4: 卷积核

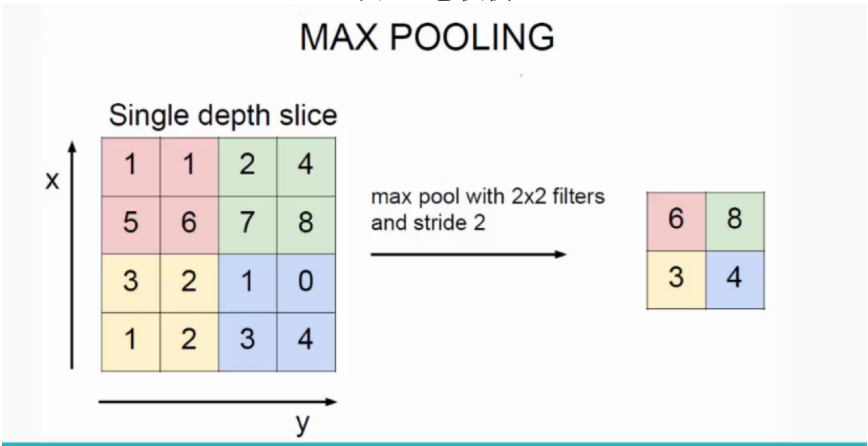


图 5: 池化

定义交叉熵 (Cross Entropy) 后，用改进的梯度下降法 (Adam Optimizer) 去降低交叉熵，训练 50000 次，每批次 (Batch) 投入 50 个训练数据。

因为所用神经网络神经元非常多，通过训练曲线可以看出，50000 次保证不至于欠拟合。同时为了防止过拟合，每 5000 次用验证集检验一次结果，如果结果优于上次结果，则保存本次结果，否则弃掉本次结果。

下图展示了某一次训练时的训练准确率曲线。

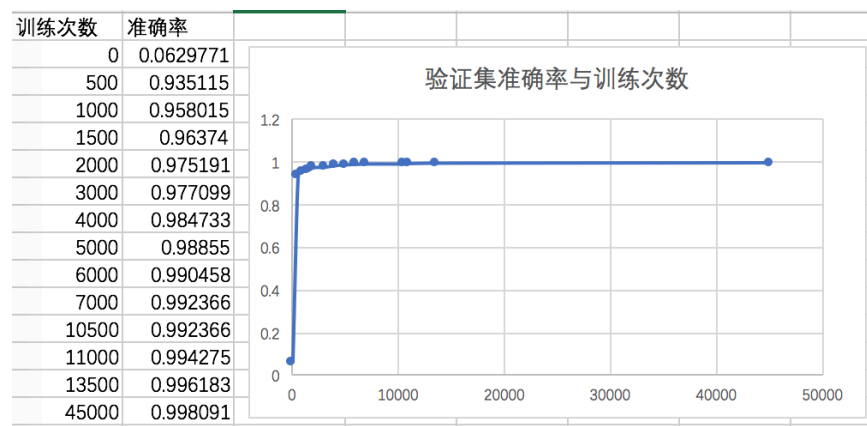


图 6: 训练准确率曲线

由训练曲线可以看出，该神经网络收敛非常快，在训练 500 次时验证集准确率便已经达到 93.5%，之后收敛逐渐变慢，最后趋于稳定。

最终可以看出，在训练集上，准确率为 99.809%，和实际提交结果 99.143% 相似，略低的原因可能是测试集噪声比较大。

为了提高准确率，我们可以加入更多的训练元素，比如著名的 MNIST 样本集结构和本次任务图像结果基本完全类似，引入 MNIST 样本集进行训练，同样训练 50000 次，准确率为 99.871%，截止写实验报告时，排名第 52。

4 实验结果

4.1 模型的选择

下文将对比模型 1 和模型 2 的结果，训练次数为 50000 次。

(1) 模型 1 (MNIST FOR EXPERT):

训练曲线如下：

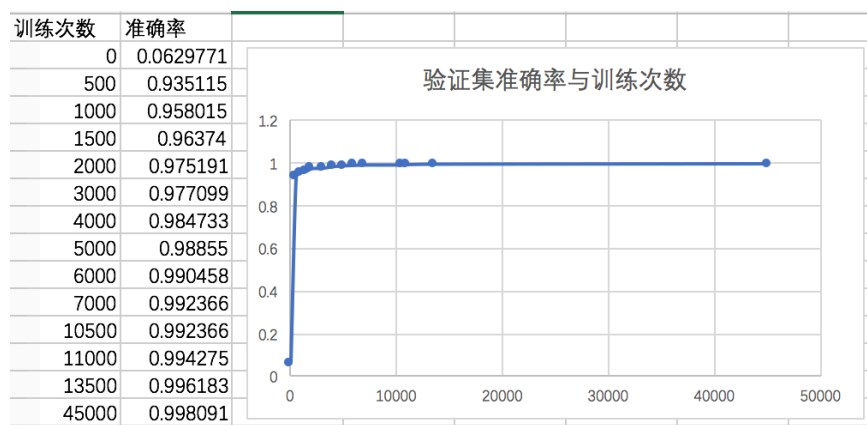


图 7: 模型 1 (MNIST FOR EXPERT) 训练准确率曲线

最终准确率 : 99.143%

(2) 模型 2 (LeNet-5) :

训练曲线如下 :

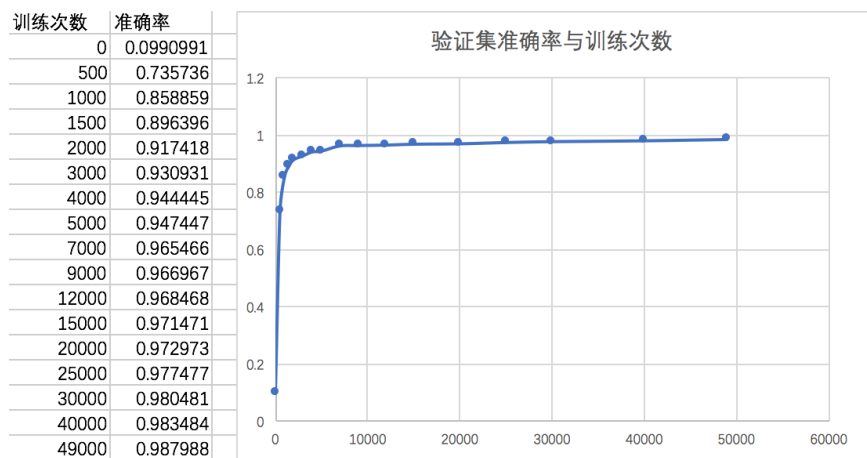


图 8: 模型 2 (LeNet-5) 训练准确率曲线

最终准确率 : 98.198%

由训练准确率曲线可以清楚地看出, 模型 1 不仅收敛更快, 而且最终准确率更高, 实际测试中, 模型 1 训练时间也较短, 故以下各测试均基于模型 1 进行测试。

4.2 卷积层层数的选择

以下测试均基于模型 1 (MNIST FOR EXPERT), 仅改变卷积层层数, 训练次数为 50000 次。

- (1) 卷积层层数为 1 时, 训练 10000 步时准确率为 98.4733%, 最终准确率 99.0458%。
- (2) 卷积层层数为 2 时, 即为原始模型, 训练 10000 步时准确率为 99.2366%, 最终准确率 99.8091%。
- (3) 卷积层层数为 3 时, 训练 10000 步时准确率为 99.0458%, 最终准确率 99.4275%。

三者准确率随训练次数收敛速度近乎相等, 但是训练速度 1 层快于 2 层快于 3 层, 消耗资源 1 层少于 2 层少于 3 层。

可见, 卷积层层数为 2 为最合适的参数。层数减小训练速度加快、消耗资源减少, 但准确率不高; 层数增加训练速度减慢, 消耗资源增加, 而且准确率也并没有提高。

全连接层一般也选择 2-3 层, 减少或增加层数可预测结果应该类似, 由于时间有限, 不再进行实验。

4.3 训练集的选择

以下测试均基于模型 1 (MNIST FOR EXPERT), 卷积层层数为 2, 仅改变训练集, 训练次数为 50000 次。

- (1) 采用原始训练集时, 验证集准确率为 99.8071%, Kaggle 测试最终准确率为 99.143%, 排名约为 350 名。
- (2) 采用 MNIST 训练集, 验证集准确率为 99.6183%, Kaggle 测试最终准确率为 99.871%, 排名为 52 名。

4.4 最终模型与竞赛成绩

最终采用模型 1 (MNIST FOR EXPERT), 卷积层层数为 2, 训练次数为 50000 次, 两种训练集结果如图所示:

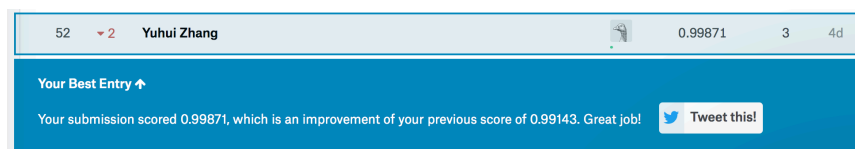


图 9: 排名和提交次数

Submission and Description	Public Score	Use for Final Score
result.csv 4 days ago by zyh2015011372 add submission details	0.99871	<input type="checkbox"/>
result.csv 4 days ago by zyh2015011372 add submission details	0.99143	<input type="checkbox"/>

图 10: 不同数据集训练下的准确率

准确率最高时提交次数：3 次

最终提交次数：4 次

5 实验心得

通过本次实验，我大大加深了对卷积神经网络的理解，初步掌握了神经网络的参数选择技巧，也初步学习了 Tensorflow 框架的使用方法，进入了高级机器学习的新世界。同时也学习了 LaTeX 排版技巧。

在这次实验中，我充分感受到了神经网络的神奇之处，通过这样的连接，识别准确率竟然可以达到 99.871%，接近 100%，而且是在噪声很高的数据集上进行的测试，实际表现应该会更好。

但是这只是一个入门，神经网络的高级技巧和参数设置技巧我还没有完全掌握，还需要进一步的练习。

值得思考的是，LeNet-5 是上课讲的专门针对手写字母识别的卷积神经网络，我本以为效果会很好，结果却并不够理想，可能是卷积核和神经元太少，没有充分提取图像的特征。

另外遗憾的是，由于期末复习时间太紧张，没有时间学习其余的神经网络，也没有时间对比训练结果，比如 LSTM，希望暑假有机会可以进一步尝试。

6 后期工作

我认为可以通过 ([1]) ([2]) ([3]) ([4]) ([5]) ([6]) ([7]) ([8]) ([9]) ([10]) ([11]) ([12]) ([13]) ([14]) ([15]) ([16]) ([17]) ([18]) ([19]) ([20]) ([21]) ([22]) ([23]) ([24]) ([25]) ([26]) ([27]) ([28]) ([29]) ([30]) ([31]) ([32]) ([33]) ([34]) ([35]) ([36]) ([37]) ([38]) ([39]) ([40]) ([41]) ([42]) ([43]) ([44]) ([45]) ([46]) ([47]) ([48]) ([49]) ([50]) ([51]) ([52]) ([53]) ([54]) ([55]) ([56]) ([57]) ([58]) ([59]) ([60]) ([61]) ([62]) ([63]) ([64]) ([65]) ([66]) ([67]) ([68]) ([69]) ([70]) ([71]) ([72]) ([73]) ([74]) ([75]) ([76]) ([77]) ([78]) ([79]) ([80]) ([81]) ([82]) ([83]) ([84]) ([85]) ([86]) ([87]) ([88]) ([89]) ([90]) ([91]) ([92]) ([93]) ([94]) ([95]) ([96]) ([97]) ([98]) ([99]) ([100]) ([101]) ([102]) ([103]) ([104]) ([105]) ([106]) ([107]) ([108]) ([109]) ([110]) ([111]) ([112]) ([113]) ([114]) ([115]) ([116]) ([117]) ([118]) ([119]) ([120]) ([121]) ([122]) ([123]) ([124]) ([125]) ([126]) ([127]) ([128]) ([129]) ([130]) ([131]) ([132]) ([133]) ([134]) ([135]) ([136]) ([137]) ([138]) ([139]) ([140]) ([141]) ([142]) ([143]) ([144]) ([145]) ([146]) ([147]) ([148]) ([149]) ([150]) ([151]) ([152]) ([153]) ([154]) ([155]) ([156]) ([157]) ([158]) ([159]) ([160]) ([161]) ([162]) ([163]) ([164]) ([165]) ([166]) ([167]) ([168]) ([169]) ([170]) ([171]) ([172]) ([173]) ([174]) ([175]) ([176]) ([177]) ([178]) ([179]) ([180]) ([181]) ([182]) ([183]) ([184]) ([185]) ([186]) ([187]) ([188]) ([189]) ([190]) ([191]) ([192]) ([193]) ([194]) ([195]) ([196]) ([197]) ([198]) ([199]) ([200]) ([201]) ([202]) ([203]) ([204]) ([205]) ([206]) ([207]) ([208]) ([209]) ([210]) ([211]) ([212]) ([213]) ([214]) ([215]) ([216]) ([217]) ([218]) ([219]) ([220]) ([221]) ([222]) ([223]) ([224]) ([225]) ([226]) ([227]) ([228]) ([229]) ([230]) ([231]) ([232]) ([233]) ([234]) ([235]) ([236]) ([237]) ([238]) ([239]) ([240]) ([241]) ([242]) ([243]) ([244]) ([245]) ([246]) ([247]) ([248]) ([249]) ([250]) ([251]) ([252]) ([253]) ([254]) ([255]) ([256]) ([257]) ([258]) ([259]) ([260]) ([261]) ([262]) ([263]) ([264]) ([265]) ([266]) ([267]) ([268]) ([269]) ([270]) ([271]) ([272]) ([273]) ([274]) ([275]) ([276]) ([277]) ([278]) ([279]) ([280]) ([281]) ([282]) ([283]) ([284]) ([285]) ([286]) ([287]) ([288]) ([289]) ([290]) ([291]) ([292]) ([293]) ([294]) ([295]) ([296]) ([297]) ([298]) ([299]) ([300]) ([301]) ([302]) ([303]) ([304]) ([305]) ([306]) ([307]) ([308]) ([309]) ([310]) ([311]) ([312]) ([313]) ([314]) ([315]) ([316]) ([317]) ([318]) ([319]) ([320]) ([321]) ([322]) ([323]) ([324]) ([325]) ([326]) ([327]) ([328]) ([329]) ([330]) ([331]) ([332]) ([333]) ([334]) ([335]) ([336]) ([337]) ([338]) ([339]) ([340]) ([341]) ([342]) ([343]) ([344]) ([345]) ([346]) ([347]) ([348]) ([349]) ([350]) ([351]) ([352]) ([353]) ([354]) ([355]) ([356]) ([357]) ([358]) ([359]) ([360]) ([361]) ([362]) ([363]) ([364]) ([365]) ([366]) ([367]) ([368]) ([369]) ([370]) ([371]) ([372]) ([373]) ([374]) ([375]) ([376]) ([377]) ([378]) ([379]) ([380]) ([381]) ([382]) ([383]) ([384]) ([385]) ([386]) ([387]) ([388]) ([389]) ([390]) ([391]) ([392]) ([393]) ([394]) ([395]) ([396]) ([397]) ([398]) ([399]) ([400]) ([401]) ([402]) ([403]) ([404]) ([405]) ([406]) ([407]) ([408]) ([409]) ([410]) ([411]) ([412]) ([413]) ([414]) ([415]) ([416]) ([417]) ([418]) ([419]) ([420]) ([421]) ([422]) ([423]) ([424]) ([425]) ([426]) ([427]) ([428]) ([429]) ([430]) ([431]) ([432]) ([433]) ([434]) ([435]) ([436]) ([437]) ([438]) ([439]) ([440]) ([441]) ([442]) ([443]) ([444]) ([445]) ([446]) ([447]) ([448]) ([449]) ([450]) ([451]) ([452]) ([453]) ([454]) ([455]) ([456]) ([457]) ([458]) ([459]) ([460]) ([461]) ([462]) ([463]) ([464]) ([465]) ([466]) ([467]) ([468]) ([469]) ([470]) ([471]) ([472]) ([473]) ([474]) ([475]) ([476]) ([477]) ([478]) ([479]) ([480]) ([481]) ([482]) ([483]) ([484]) ([485]) ([486]) ([487]) ([488]) ([489]) ([490]) ([491]) ([492]) ([493]) ([494]) ([495]) ([496]) ([497]) ([498]) ([499]) ([500]) ([501]) ([502]) ([503]) ([504]) ([505]) ([506]) ([507]) ([508]) ([509]) ([510]) ([511]) ([512]) ([513]) ([514]) ([515]) ([516]) ([517]) ([518]) ([519]) ([520]) ([521]) ([522]) ([523]) ([524]) ([525]) ([526]) ([527]) ([528]) ([529]) ([530]) ([531]) ([532]) ([533]) ([534]) ([535]) ([536]) ([537]) ([538]) ([539]) ([540]) ([541]) ([542]) ([543]) ([544]) ([545]) ([546]) ([547]) ([548]) ([549]) ([550]) ([551]) ([552]) ([553]) ([554]) ([555]) ([556]) ([557]) ([558]) ([559]) ([560]) ([561]) ([562]) ([563]) ([564]) ([565]) ([566]) ([567]) ([568]) ([569]) ([570]) ([571]) ([572]) ([573]) ([574]) ([575]) ([576]) ([577]) ([578]) ([579]) ([580]) ([581]) ([582]) ([583]) ([584]) ([585]) ([586]) ([587]) ([588]) ([589]) ([590]) ([591]) ([592]) ([593]) ([594]) ([595]) ([596]) ([597]) ([598]) ([599]) ([600]) ([601]) ([602]) ([603]) ([604]) ([605]) ([606]) ([607]) ([608]) ([609]) ([610]) ([611]) ([612]) ([613]) ([614]) ([615]) ([616]) ([617]) ([618]) ([619]) ([620]) ([621]) ([622]) ([623]) ([624]) ([625]) ([626]) ([627]) ([628]) ([629]) ([630]) ([631]) ([632]) ([633]) ([634]) ([635]) ([636]) ([637]) ([638]) ([639]) ([640]) ([641]) ([642]) ([643]) ([644]) ([645]) ([646]) ([647]) ([648]) ([649]) ([650]) ([651]) ([652]) ([653]) ([654]) ([655]) ([656]) ([657]) ([658]) ([659]) ([660]) ([661]) ([662]) ([663]) ([664]) ([665]) ([666]) ([667]) ([668]) ([669]) ([670]) ([671]) ([672]) ([673]) ([674]) ([675]) ([676]) ([677]) ([678]) ([679]) ([680]) ([681]) ([682]) ([683]) ([684]) ([685]) ([686]) ([687]) ([688]) ([689]) ([690]) ([691]) ([692]) ([693]) ([694]) ([695]) ([696]) ([697]) ([698]) ([699]) ([700]) ([701]) ([702]) ([703]) ([704]) ([705]) ([706]) ([707]) ([708]) ([709]) ([710]) ([711]) ([712]) ([713]) ([714]) ([715]) ([716]) ([717]) ([718]) ([719]) ([720]) ([721]) ([722]) ([723]) ([724]) ([725]) ([726]) ([727]) ([728]) ([729]) ([730]) ([731]) ([732]) ([733]) ([734]) ([735]) ([736]) ([737]) ([738]) ([739]) ([740]) ([741]) ([742]) ([743]) ([744]) ([745]) ([746]) ([747]) ([748]) ([749]) ([750]) ([751]) ([752]) ([753]) ([754]) ([755]) ([756]) ([757]) ([758]) ([759]) ([760]) ([761]) ([762]) ([763]) ([764]) ([765]) ([766]) ([767]) ([768]) ([769]) ([770]) ([771]) ([772]) ([773]) ([774]) ([775]) ([776]) ([777]) ([778]) ([779]) ([780]) ([781]) ([782]) ([783]) ([784]) ([785]) ([786]) ([787]) ([788]) ([789]) ([790]) ([791]) ([792]) ([793]) ([794]) ([795]) ([796]) ([797]) ([798]) ([799]) ([800]) ([801]) ([802]) ([803]) ([804]) ([805]) ([806]) ([807]) ([808]) ([809]) ([810]) ([811]) ([812]) ([813]) ([814]) ([815]) ([816]) ([817]) ([818]) ([819]) ([820]) ([821]) ([822]) ([823]) ([824]) ([825]) ([826]) ([827]) ([828]) ([829]) ([830]) ([831]) ([832]) ([833]) ([834]) ([835]) ([836]) ([837]) ([838]) ([839]) ([840]) ([841]) ([842]) ([843]) ([844]) ([845]) ([846]) ([847]) ([848]) ([849]) ([850]) ([851]) ([852]) ([853]) ([854]) ([855]) ([856]) ([857]) ([858]) ([859]) ([860]) ([861]) ([862]) ([863]) ([864]) ([865]) ([866]) ([867]) ([868]) ([869]) ([870]) ([871]) ([872]) ([873]) ([874]) ([875]) ([876]) ([877]) ([878]) ([879]) ([880]) ([881]) ([882]) ([883]) ([884]) ([885]) ([886]) ([887]) ([888]) ([889]) ([890]) ([891]) ([892]) ([893]) ([894]) ([895]) ([896]) ([897]) ([898]) ([899]) ([900]) ([901]) ([902]) ([903]) ([904]) ([905]) ([906]) ([907]) ([908]) ([909]) ([910]) ([911]) ([912]) ([913]) ([914]) ([915]) ([916]) ([917]) ([918]) ([919]) ([920]) ([921]) ([922]) ([923]) ([924]) ([925]) ([926]) ([927]) ([928]) ([929]) ([930]) ([931]) ([932]) ([933]) ([934]) ([935]) ([936]) ([937]) ([938]) ([939]) ([940]) ([941]) ([942]) ([943]) ([944]) ([945]) ([946]) ([947]) ([948]) ([949]) ([950]) ([951]) ([952]) ([953]) ([954]) ([955]) ([956]) ([957]) ([958]) ([959]) ([960]) ([961]) ([962]) ([963]) ([964]) ([965]) ([966]) ([967]) ([968]) ([969]) ([970]) ([971]) ([972]) ([973]) ([974]) ([975]) ([976]) ([977]) ([978]) ([979]) ([980]) ([981]) ([982]) ([983]) ([984]) ([985]) ([986]) ([987]) ([988]) ([989]) ([990]) ([991]) ([992]) ([993]) ([994]) ([995]) ([996]) ([997]) ([998]) ([999]) ([1000]) ([1001]) ([1002]) ([1003]) ([1004]) ([1005]) ([1006]) ([1007]) ([1008]) ([1009]) ([1010]) ([1011]) ([1012]) ([1013]) ([1014]) ([1015]) ([1016]) ([1017]) ([1018]) ([1019]) ([1020]) ([1021]) ([1022]) ([1023]) ([1024]) ([1025]) ([1026]) ([1027]) ([1028]) ([1029]) ([1030]) ([1031]) ([1032]) ([1033]) ([1034]) ([1035]) ([1036]) ([1037]) ([1038]) ([1039]) ([1040]) ([1041]) ([1042]) ([1043]) ([1044]) ([1045]) ([1046]) ([1047]) ([1048]) ([1049]) ([1050]) ([1051]) ([1052]) ([1053]) ([1054]) ([1055]) ([1056]) ([1057]) ([1058]) ([1059]) ([1060]) ([1061]) ([1062]) ([1063]) ([1064]) ([1065]) ([1066]) ([1067]) ([1068]) ([1069]) ([1070]) ([1071]) ([1072]) ([1073]) ([1074]) ([1075]) ([1076]) ([1077]) ([1078]) ([1079]) ([1080]) ([1081]) ([1082]) ([1083]) ([1084]) ([1085]) ([1086]) ([1087]) ([1088]) ([1089]) ([1090]) ([1091]) ([1092]) ([1093]) ([1094]) ([1095]) ([1096]) ([1097]) ([1098]) ([1099]) ([1100]) ([1101]) ([1102]) ([1103]) ([1104]) ([1105]) ([1106]) ([1107]) ([1108]) ([1109]) ([1110]) ([1111]) ([1112]) ([1113]) ([1114]) ([1115]) ([1116]) ([1117]) ([1118]) ([1119]) ([1120]) ([1121]) ([1122]) ([1123]) ([1124]) ([1125]) ([1126]) ([1127]) ([1128]) ([1129]) ([1130]) ([1131]) ([1132]) ([1133]) ([1134]) ([1135]) ([1136]) ([1137]) ([1138]) ([1139]) ([1140]) ([1141]) ([1142]) ([1143]) ([1144]) ([1145]) ([1146]) ([1147]) ([1148]) ([1149]) ([1150]) ([1151]) ([1152]) ([1153]) ([1154]) ([1155]) ([1156]) ([1157]) ([1158]) ([1159]) ([1160]) ([1161]) ([1162]) ([1163]) ([1164]) ([1165]) ([1166]) ([1167]) ([1168]) ([1169]) ([1170]) ([1171]) ([1172]) ([1173]) ([1174]) ([1175]) ([1176]) ([1177]) ([1178]) ([1179]) ([1180]) ([1181]) ([1182]) ([1183]) ([1184]) ([1185]) ([1186]) ([1187]) ([1188]) ([1189]) ([1190]) ([1191]) ([1192]) ([1193]) ([1194]) ([1195]) ([1196]) ([1197]) ([1198]) ([1199]) ([1200]) ([1201]) ([1202]) ([1203]) ([1204]) ([1205]) ([1206]) ([1207]) ([1208]) ([1209]) ([1210]) ([1211]) ([1212]) ([1213]) ([1214]) ([1215]) ([1216]) ([1217]) ([1218]) ([1219]) ([1220]) ([1221]) ([1222]) ([1223]) ([1224]) ([1225]) ([1226]) ([1227]) ([1228]) ([1229]) ([1230]) ([1231]) ([1232]) ([1233]) ([1234]) ([1235]) ([1236]) ([1237]) ([1238]) ([1239]) ([1240]) ([1241]) ([1242]) ([1243]) ([1244]) ([1245]) ([1246]) ([1247]) ([1248]) ([1249]) ([1250]) ([1251]) ([1252]) ([1253]) ([1254]) ([1255]) ([1256]) ([1257]) ([1258]) ([1259]) ([1260]) ([1261]) ([1262]) ([1263]) ([1264]) ([1265]) ([1266]) ([1267]) ([1268]) ([1269]) ([1270]) ([1271]) ([1272]) ([1273]) ([1274]) ([1275]) ([1276]) ([1277]) ([1278]) ([1279]) ([1280]) ([1281]) ([1282]) ([1283]) ([1284]) ([1285]) ([1286]) ([1287]) ([1288]) ([1289]) ([1290]) ([1291]) ([1292]) ([1293]) ([1294]) ([1295]) ([1296]) ([1297]) ([1298]) ([1299]) ([1300]) ([1301]) ([1302]) ([1303]) ([1304]) ([1305]) ([1306]) ([1307]) ([1308]) ([1309]) ([1310]) ([1311]) ([1312]) ([1313]) ([1314]) ([1315]) ([1316]) ([1317]) ([1318]) ([1319]) ([1320]) ([1321]) ([1322]) ([1323]) ([1324]) ([1325]) ([1326]) ([1327]) ([1328]) ([1329]) ([1330]) ([1331]) ([1332]) ([1333]) ([1334]) ([1335]) ([1336]) ([1337]) ([1338]) ([1339]) ([1340]) ([1341]) ([1342]) ([1343]) ([1344]) ([1345]) ([1346]) ([1347]) ([1348]) ([1349]) ([1350]) ([1351]) ([1352]) ([1353]) ([1354]) ([1355]) ([1356]) ([1357]) ([1358]) ([1359]) ([1360]) ([1361]) ([1362]) ([1363]) ([1364]) ([1365]) ([1366]) ([1367]) ([1368]) ([1369]) ([1370]) ([1371]) ([1372]) ([1373]) ([1374]) ([1375]) ([1376]) ([1377]) ([1378]) ([1379]) ([1380]) ([1381]) ([1382]) ([1383]) ([1384]) ([1385]) ([1386]) ([1387]) ([1388]) ([1389]) ([1390]) ([1391]) ([1392]) ([1393]) ([1394]) ([1395]) ([1396]) ([1397]) ([1398]) ([1399]) ([1400]) ([1401]) ([1402]) ([1403]) ([1404]) ([1405]) ([1406]) ([1407]) ([1408]) ([1409]) ([1410]) ([1411]) ([1412]) ([1413]) ([1414]) ([1415]) ([1416]) ([1417]) ([1418]) ([1419]) ([1420]) ([1421]) ([1422]) ([1423]) ([1424]) ([1425]) ([1426]) ([1427]) ([1428]) ([1429]) ([1430]) ([1431]) ([1432]) ([1433]) ([1434]) ([1435]) ([1436]) ([1437]) ([1438]) ([1439]) ([1440]) ([1441]) ([1442]) ([1443]) ([1444]) ([1445]) ([1446]) ([1447]) ([1448]) ([1449]) ([1450]) ([1451]) ([1452]) ([1453]) ([1454]) ([1455]) ([1456]) ([1457]) ([1458]) ([1459]) ([1460]) ([1461]) ([1462]) ([1463]) ([1464]) ([1465]) ([1466]) ([1467]) ([1468]) ([1469]) ([1470]) ([1471]) ([1472]) ([1473]) ([1474]) ([1475]) ([1476]) ([1477]) ([1478]) ([1479]) ([1480]) ([1481]) ([1482]) ([1483]) ([1484]) ([1485]) ([1486]) ([1487]) ([1488]) ([1489]) ([1490]) ([1491]) ([1492]) ([1493]) ([1494]) ([1495]) ([1496]) ([1497]) ([1498]) ([1499]) ([1500]) ([1501]) ([1502]) ([1503]) ([1504]) ([1505]) ([1506]) ([1507]) ([1508]) ([1509]) ([1510]) ([1511]) ([1512]) ([1513]) ([1514]) ([1515]) ([1516]) ([1517]) ([1518]) ([1519]) ([1520]) ([1521]) ([1522]) ([1523]) ([1524]) ([1525]) ([1526]) ([1527]) ([1528]) ([1529]) ([1530]) ([1531]) ([1532]) ([1533]) ([1534]) ([1535]) ([1536]) ([1537]) ([1538]) ([1539]) ([1540]) ([1541]) ([1542]) ([1543]) ([1544]) ([1545]) ([1546]) ([1547]) ([1548]) ([1549]) ([1550]) ([1551]) ([1552]) ([1553]) ([1554]) ([1555]) ([1556]) ([1557]) ([1558]) ([1559]) ([1560]) ([1561]) ([1562]) ([1563]) ([1564]) ([1565]) ([1566]) ([1567]) ([1568]) ([1569]) ([1570]) ([1571]) ([1572]) ([1573]) ([1574]) ([1575]) ([1576]) ([1577]) ([1578]) ([1579]) ([1580]) ([1581]) ([1582]) ([1583]) ([1584]) ([1585]) ([1586]) ([1587]) ([1588]) ([1589]) ([1590]) ([1591]) ([1592]) ([1593]) ([1594]) ([1595]) ([1596]) ([1597]) ([1598]) ([1599]) ([1600]) ([1601]) ([1602]) ([1603]) ([1604]) ([1605]) ([1606]) ([1607]) ([1608]) ([1609]) ([1610]) ([1611]) ([1612]) ([1613]) ([1614]) ([1615]) ([1616]) ([1617]) ([1618]) ([1619]) ([1620]) ([1621]) ([1622]) ([1623]) ([1624]) ([1625]) ([1626]) ([1627]) ([1628]) ([1629]) ([1630]) ([1631]) ([1632]) ([1633]) ([1634]) ([1635]) ([1636]) ([1637]) ([1638]) ([1639]) ([1640]) ([1641]) ([1642]) ([1643]) ([1644]) ([1645]) ([1646]) ([1647]) ([1648]) ([1649]) ([1650]) ([1651]) ([1652]) ([1653]) ([1654]) ([1655]) ([1656]) ([1657]) ([1658]) ([1659]) ([1660]) ([1661]) ([1662]) ([1663]) ([1664]) ([1665]) ([1666]) ([1667]) ([1668]) ([1669]) ([1670]) ([1671]) ([1672]) ([1673]) ([1674]) ([1675]) ([1676]) ([1677]) ([1678]) ([1679]) ([1680]) ([1681]) ([1682]) ([1683]) ([1684]) ([1685]) ([1686]) ([1687]) ([1688]) ([1689]) ([1690]) ([1691]) ([1692]) ([1693]) ([1694]) ([1695]) ([1696]) ([1697]) ([1698]) ([1699]) ([1700]) ([1701]) ([1702]) ([1703]) ([1704]) ([1705]) ([1706]) ([1707]) ([1708]) ([1709]) ([1710]) ([1711]) ([1712]) ([1713]) ([1714]) ([1715]) ([1716]) ([1717]) ([1718]) ([1719]) ([1720]) ([1721]) ([1722]) ([1723]) ([1724]) ([1725]) ([1726]) ([1727]) ([1728]) ([1729]) ([1730]) ([1731]) ([1732]) ([1733]) ([1734]) ([1735]) ([1736]) ([1737]) ([1738]) ([1739]) ([1740]) ([1741]) ([1742]) ([1743]) ([1744]) ([1745]) ([1746]) ([1747]) ([1748]) ([1749]) ([1750]) ([1751]) ([1752]) ([1753]) ([1754]) ([1755]) ([1756]) ([1757]) ([1758]) ([1759]) ([1760]) ([1761]) ([1762]) ([1763]) ([1764]) ([1765]) ([1766]) ([1767]) ([1768]) ([1769]) ([1770]) ([1771]) ([1772]) ([1773]) ([1774]) ([1775]) ([1776]) ([1777]) ([1778]) ([1779]) ([1780]) ([1781]) ([1782]) ([1783]) ([1784]) ([1785]) ([1786]) ([1787]) ([1788]) ([1789]) ([1790]) ([1791]) ([1792]) ([1793]) ([1794]) ([1795]) ([1796]) ([1797]) ([1798]) ([1799]) ([1800]) ([1801]) ([1802]) ([1803]) ([1804]) ([1805]) ([1806]) ([1807]) ([1808]) ([1809]) ([1810]) ([1811]) ([1812]) ([1813]) ([1814]) ([1815]) ([1816]) ([1817]) ([1818]) ([1819]) ([1820]) ([1821]) ([1822]) ([1823]) ([1824]) ([1825]) ([1826]) ([1827]) ([1828]) ([1829]) ([1830]) ([1831]) ([1832]) ([1833]) ([1834]) ([1835]) ([1836]) ([1837]) ([1838]) ([1839]) ([1840]) ([1841]) ([1842]) ([1843]) ([1844]) ([1845]) ([1846]) ([1847]) ([1848]) ([1849]) ([1850]) ([1851]) ([1852]) ([1853]) ([1854]) ([1855]) ([1856]) ([1857]) ([1858]) ([1859]) ([1860]) ([1861]) ([1862]) ([1863]) ([1864]) ([1865]) ([1866]) ([1867]) ([1868]) ([1869]) ([1870]) ([1871]) ([1872]) ([1873]) ([1874]) ([1875]) ([1876]) ([1877]) ([1878]) ([1879]) ([1880]) ([1881]) ([1882]) ([1883]) ([1884]) ([1885]) ([1886]) ([1887]) ([1888]) ([1889]) ([1890]) ([1891]) ([1892]) ([1893]) ([1894]) ([1895]) ([1896]) ([1897]) ([1898]) ([1899]) ([1900]) ([1901]) ([1902]) ([1903]) ([1904]) ([1905]) ([1906]) ([1907]) ([1908]) ([1909]) ([1910]) ([1911]) ([1912]) ([1913]) ([1914]) ([1915]) ([1916]) ([1917]) ([1918]) ([1919]) ([1920]) ([1921]) ([1922]) ([1923]) ([1924]) ([1925]) ([1926]) ([1927]) ([1928]) ([1929]) ([1930]) ([1931]) ([1932]) ([1933]) ([1934]) ([1935]) ([1936]) ([1937]) ([1938]) ([1939]) ([1940]) ([1941]) ([1942]) ([1943]) ([1944]) ([1945]) ([1946]) ([1947]) ([1948]) ([1949]) ([1950]) ([1951]) ([1952]) ([1953]) ([1954]) ([1955]) ([1956]) ([1957]) ([1958]) ([1959]) ([1960]) ([1961]) ([1962]) ([1963]) ([1964]) ([1965]) ([1966]) ([1967]) ([1968]) ([1969]) ([1970]) ([1971]) ([1972]) ([1973]) ([1974]) ([1975]) ([1976]) ([1977]) ([1978]) ([1979]) ([1980]) ([1981]) ([1982]) ([1983]) ([1984]) ([1985]) ([1986]) ([1987]) ([1988]) ([1989]) ([1990]) ([1991]) ([1992]) ([1993]) ([1994]) ([1995]) ([1996]) ([1997]) ([1998]) ([1999]) ([2000]) ([2001]) ([2002]) ([2003]) ([2004]) ([2005]) ([2006]) ([2007]) ([2008]) ([2009]) ([2010]) ([2011]) ([2012]) ([2013]) ([2014]) ([2015]) ([2016]) ([2017]) ([2018]) ([2019]) ([2020]) ([2021]) ([2022]) ([2023]) ([2024]) ([2025]) ([2026]) ([2027]) ([2028]) ([2029]) ([2030]) ([2031]) ([2032]) ([2033]) ([2034]) ([2035]) ([2036]) ([2037]) ([2038]) ([2039]) ([2040]) ([2041]) ([2042]) ([2043]) ([2044]) ([2045]) ([2046]) ([2047]) ([2048]) ([2049]) ([2050]) ([2051]) ([205

- (1) 对训练集进行处理, 例如适当旋转一定角度, 适当放大或缩小, 从而获得更多的训练数据, 使得训练更为充分。
- (2) 对测试集进行处理, 例如去噪声, 比如有的测试数据有一个黑点, 可以采用深度优先搜索和设置阈值的方法得以实现。
- (3) 采用 RNN、LSTM 等不同的神经网络进行训练与测试, 对比结果。

7 源代码

运行方式 :

将文件 train.csv 和 test.cs 拷贝至 Python 程序所在目录, 在命令行中输入 python main.py 即可。

输出结果为文件 result.csv, 同时会保存训练出的模型, 可以直接使用 Tensorflow 中的模块读取训练好的模型。

```
#coding=utf-8
import numpy
import csv
import tensorflow as tf

train_pixel = [] # 训练集
train_label = [] # 训练标签
test_pixel = [] # 测试集
train_count = -2 # 训练集大小
validation_count = -2 # 验证集大小
test_count = -2 # 测试集大小
now = 0 # 全局计数器

# 加载训练集
def load_train_data():
    global train_pixel, train_label, train_count, validation_count
    # 从train.csv文件读取数据, train.csv每行为785个整数, 第1个为标签, 后面784个为对应像素灰度(0-255)
    csvfile = file('train.csv', 'rb')
    reader = csv.reader(csvfile)
    for line in reader:
        train_count = train_count + 1
        if (train_count == -1):
            continue
        # 将标签转换为对应的10维向量
        train_label.append(numpy.zeros(10))
        train_label[train_count][int(line[0])] = 1.0
        train_pixel.append([])
        for i in xrange(1, len(line)):
```

```

        train_pixel[train_count].append(float(line[i]) / 255) # 灰度归一化
    csvfile.close()
    # 从训练集中抽取部分作为验证集
    validation_count = int(0.0125 * train_count);
    train_count -= validation_count;

# 加载测试集
def load_test_data():
    global test_pixel, test_count
    # 从test.csv文件读取数据, test.csv每行为784个整数, 为对应像素灰度(0-255)
    csvfile = file('test.csv', 'rb')
    reader = csv.reader(csvfile)
    for line in reader:
        test_count = test_count + 1
        if (test_count == -1):
            continue
        test_pixel.append([])
        for i in xrange(0, len(line)):
            test_pixel[test_count].append(float(line[i]) / 255) # 灰度归一化
    csvfile.close()

# 保存结果
def save_result(result):
    # 保存至result.csv文件
    csvfile = file('result.csv', 'wb')
    writer = csv.writer(csvfile)
    # csv文件表头
    writer.writerow(['ImageId', 'Label'])
    # 每行保存序号(从1开始)和对应结果
    for i in xrange(0, len(result)):
        writer.writerow([i + 1, result[i]])
    csvfile.close()

# 获取下一批训练集, 大小为num
def next_batch(num):
    global now
    label = numpy.array(train_label[now: now + num])
    pixel = numpy.array(train_pixel[now: now + num])
    now = now + num
    # 如果训练集全部用完, 重新开始用
    if (now > train_count): now = now - train_count
    return (pixel, label)

# 获取验证集
def validation_set():
    label = numpy.array(train_label[train_count: train_count + validation_count])
    pixel = numpy.array(train_pixel[train_count: train_count + validation_count])

```

```

    return (pixel, label)

# 随机初始化大小为shape的神经元
def weight_variable(shape):
    initial = tf.truncated_normal(shape, stddev = 0.1)
    return tf.Variable(initial)

# 生成大小为shape的偏置变量
def bias_variable(shape):
    initial = tf.constant(0.1, shape = shape)
    return tf.Variable(initial)

# 卷积核
def conv2d(x, W):
    return tf.nn.conv2d(x, W, strides = [1, 1, 1, 1], padding = 'SAME')

# 2x2最大池化
def max_pool(x):
    return tf.nn.max_pool(x, ksize = [1, 2, 2, 1], strides = [1, 2, 2, 1], padding = 'SAME')

# 训练神经网络
def train_nn():
    sess = tf.InteractiveSession() # 建立session

    x = tf.placeholder(tf.float32, shape = [None, 784]) # 神经网络输入
    y_ = tf.placeholder(tf.float32, shape = [None, 10]) # 神经网络输出
    x_image = tf.reshape(x, [-1, 28, 28, 1]) # 将784像素重置为28x28像素

    W_conv1 = weight_variable([5, 5, 1, 32]) # 第一层卷积层
    b_conv1 = bias_variable([32]) # 第一层偏置
    h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1) # 第一层卷积, relu激活
    h_pool1 = max_pool(h_conv1) # 第一层2x2池化

    W_conv2 = weight_variable([5, 5, 32, 64]) # 第二层卷积层
    b_conv2 = bias_variable([64]) # 第二层偏置
    h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2) # 第二层卷积, relu激活
    h_pool2 = max_pool(h_conv2) # 第二层2x2池化

    h_pool2_flat = tf.reshape(h_pool2, [-1, 7 * 7 * 64]) # 将第二层重置shape

    W_fc1 = weight_variable([7 * 7 * 64, 1024]) # 第三层全连接层
    b_fc1 = bias_variable([1024]) # 第三层偏置
    h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1) # 第三层计算, relu激活

    keep_prob = tf.placeholder(tf.float32) # 保持概率
    h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob) # 适度丢弃

```

```

W_fc2 = weight_variable([1024, 10]) # 第四层全连接层
b_fc2 = bias_variable([10]) # 第四层偏置
y_conv = tf.matmul(h_fc1_drop, W_fc2) + b_fc2 # 第四层计算, 保留原始计算结果
'''

# LaNet-5模型, 如果换用模型直接把上述代码替换为下段代码即可

# 第一卷积层
W_conv1 = weight_variable([5, 5, 1, 6])
b_conv1 = bias_variable([6])
h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)

# 第二卷积层
W_conv2 = weight_variable([5, 5, 6, 16])
b_conv2 = bias_variable([16])
h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)

# 第一全连接层
W_fc1 = weight_variable([4 * 4 * 16, 120])
b_fc1 = bias_variable([120])
h_pool2_flat = tf.reshape(h_pool2, [-1, 4 * 4 * 16])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
keep_prob = tf.placeholder(tf.float32)
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

# 第二全连接层
W_fc2 = weight_variable([120, 84])
b_fc2 = bias_variable([84])
h_fc2 = tf.nn.relu(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
h_fc2_drop = tf.nn.dropout(h_fc2, keep_prob)

# 第三全连接层
W_fc3 = weight_variable([84, 10])
b_fc3 = bias_variable([10])
y_conv = tf.matmul(h_fc2_drop, W_fc3) + b_fc3
'''

cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels = y_, logits =
    y_conv)) # 交叉熵计算
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy) # 最小化交叉熵
prediction = tf.argmax(y_conv, 1) # 预测结果
correct_prediction = tf.equal(prediction, tf.argmax(y_, 1)) # 结果是否正确
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32)) # 准确率

sess.run(tf.global_variables_initializer()) # 初始化所有变量

```

```

saver = tf.train.Saver() # 参数保存/加载器

with tf.device('/cpu:0'): # 指定在某设备上运行 (如CPU、显卡)
    best_accuracy = 0.0 # 在验证集最高准确率
    best_num = 0 # 对应最高准确率的训练次数
    for i in range(1000): # 训练50000次
        batch = next_batch(50) # 每次提供50个训练样本
        # 每训练500次用验证集测试准确率, 若高于最高准确率, 记录并保存结果
        if i % 500 == 0:
            [validation_pixel, validation_label] = validation_set()
            train_accuracy = accuracy.eval(feed_dict = {x: validation_pixel, y_:
                validation_label, keep_prob: 1.0})
            if (train_accuracy >= best_accuracy):
                best_accuracy = train_accuracy
                best_num = i
                saver.save(sess, './model')
            print("step %d, training accuracy %g, best accuracy %g, best num %d" % (i,
                train_accuracy, best_accuracy, best_num))
        train_step.run(feed_dict = {x: batch[0], y_: batch[1], keep_prob: 0.5}) # 训练

    load_path = saver.restore(sess, './model') # 加载训练最好结果
    result = [] # 结果向量
    # 每次记录500个结果, 避免消耗过多资源 (尤其是使用显卡时)
    i = 0
    while (i <= test_count):
        result[i: i + 500] = prediction.eval(feed_dict = {x: test_pixel[i: i + 500], keep_prob
            : 1.0})
        i += 500
    save_result(result) # 保存结果文件

# 主程序
def main():
    load_train_data() # 加载训练集
    load_test_data() # 加载测试集
    train_nn() # 训练神经网络

if __name__ == '__main__':
    main()

```

参考文献

- [1] 人工智能导论, 马少平, 2017.
- [2] Python 机器学习及实践: 从零开始通往 Kagle 竞赛之路, 范森, 李超, 2017.