人工智能大作业三

手写数字识别

沈磊
物理系 2015 级直博

一、任务描述

手写数字识别是图像识别领域，也是人工智能领域的一个典型的任务。这类的图像识别可以在很多应用场景解放生产力。比如：填写账单，订单时的电话，帐号，卡号等手写数字录入电脑;

上世纪末 MNIST 数据集发布了 60000 张带标签手写数字图片，10000 张不带标签的手写数字图片，引发了人们的广泛兴趣。时至今日，MNIST 数据集仍然是图像识别领域的经典案例，也常常被拿来测试新的图像识别模型。大家在 MNIST 数据集上应用各种方法，有线性分类器，K 最近邻方法，非线性分类器，支持向量机（SVMs），人工神经网络等。随着人工神经网络的发展，伴随着数据和算力的提升，深度神经网络得到快速发展。并在很多领域产生很震撼的效果。比如 Goggle 的 AlphaGo，AlexNet 图片分类器，智能推荐系统等等。

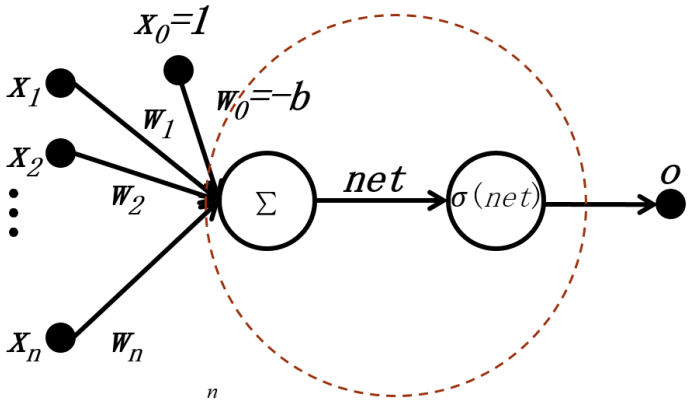
本次大作业是运用 Kaggle 竞赛网站上的手写数字数据集作为数据，利用所学知识自己搭建一套完整的深度神经网络系统，识别图像中的手写数字，并参与竞赛评分。

二、神经网络介绍

人类在识别数字的时候，是根据人以往“看过”的手写数字，根据经验来识别。其实这就是一个模型训练加预测的过程（假设你从来没有看过手写数字“7”，你可能不会认识“7”的手写体）。神经网络就是这样一种模型，通过大量的数据来学习其中的泛化特性，再根据学习到的泛化特性来识别和预测。

1、 线性神经元

神经网络的基础单元就是下图中的的线性神经元。



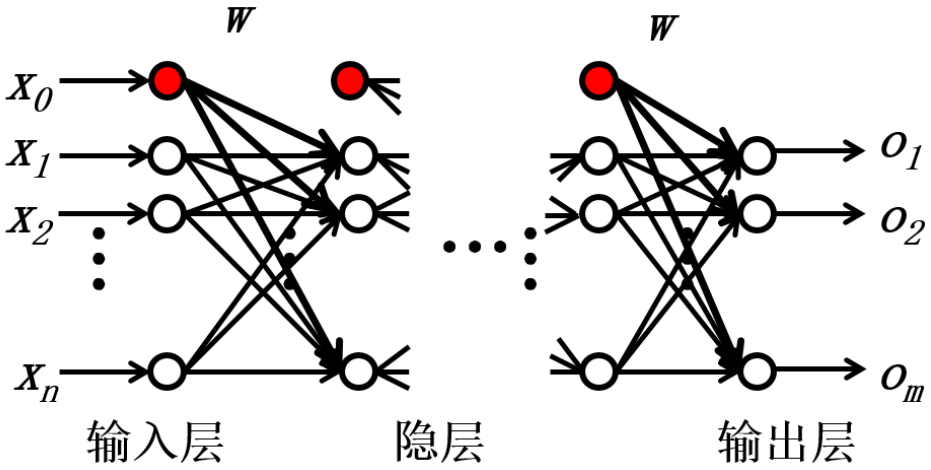
$$o = \sigma(\sum_{i=0}^{n} w_i x_i) = \sigma(\vec{w} \cdot \vec{x})$$

图一、线性神经元

该神经元输入是一维向量(x0,x1,x2,⋯,xn)，线性方程的权重是(w0,w1,w2,⋯,wn)，经过激活函数σ，得到输出 o。权重代表不用输入的重要性，这是靠学习学到的参数。

2、多层感知机模型

由一层一层的神经网络全连接起来就是神经网络了，如下图，



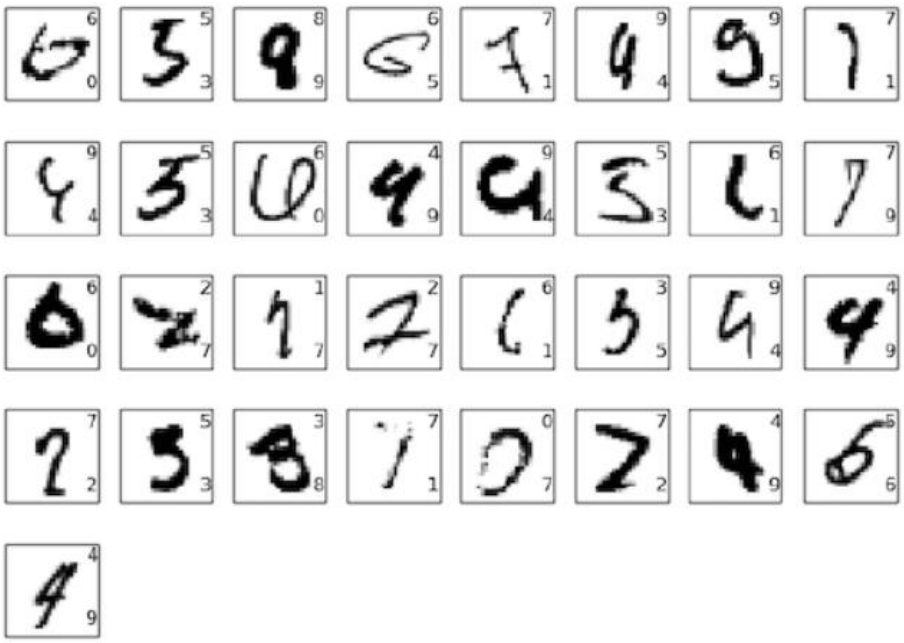
图二、神经网络

该图就是一个多层感知机（MLP）模型，有输入端、输出端和隐层组成，各层之间以全连接方式连接。MLP 是一种有效的分类器。

但是多层感知机有一个缺陷就是不能太多层，因为参数数量有$\sum_n \mathrm{N1} * \mathrm{N2} * \ldots * \mathrm{Nn}$，这个复杂度需要一些办法才能达到有效性。

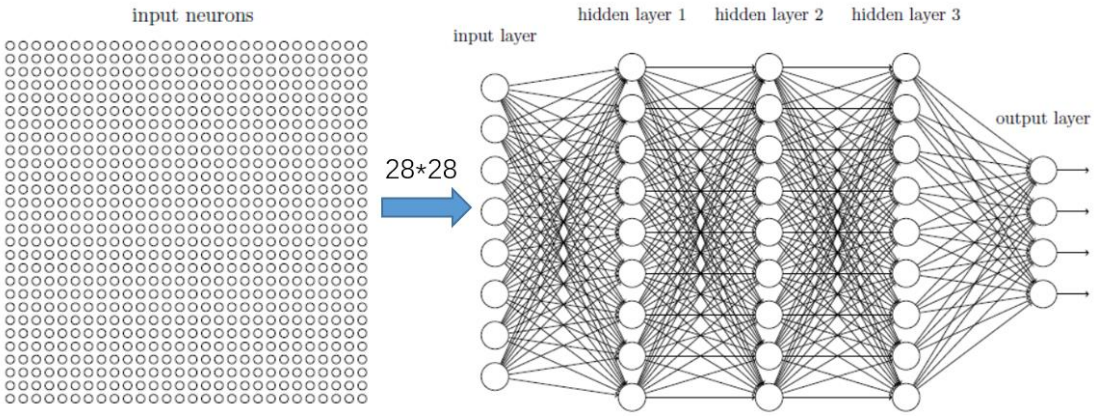
3、卷积神经网络

虽然多层感知机的神经网络很好，但是在实际问题中，求解往往需要多层神经网络，也就是深度神经网络，在这种情况下模型参数是指数增长（如上面分析）。

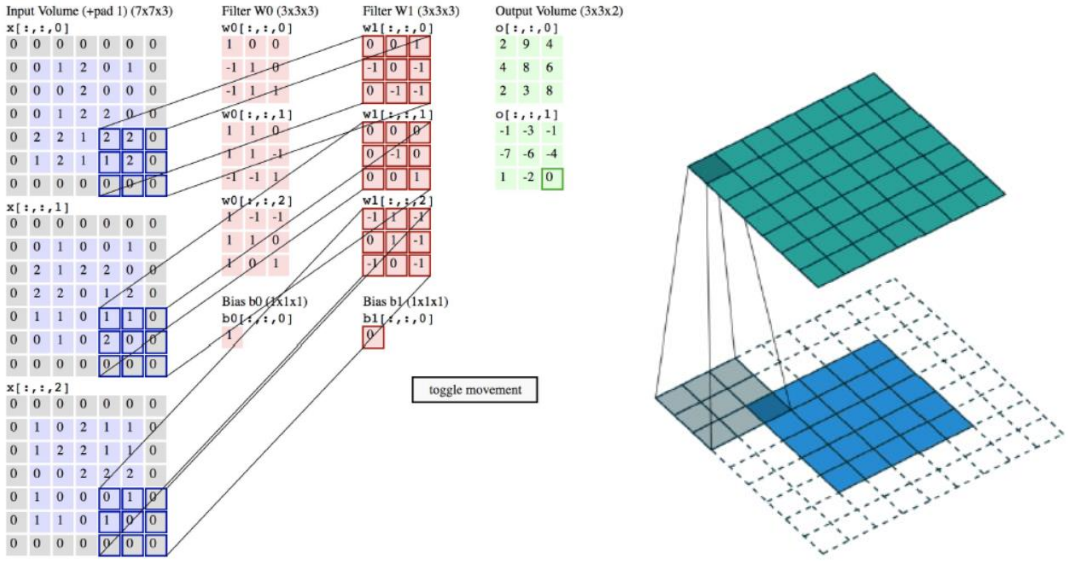


图三、MNIST 数据集

以 MNIST 数据集的手写数字识别为例，用多层感知机模型可以很好地识别数字（正确率可达 0.98）。但是仔细一想，多层感知机模型将图片的每一个像素点不差别的输入神经网络中，不考虑图片的空间结构，这是不那么合理的地方，所以就引入了卷积操作。



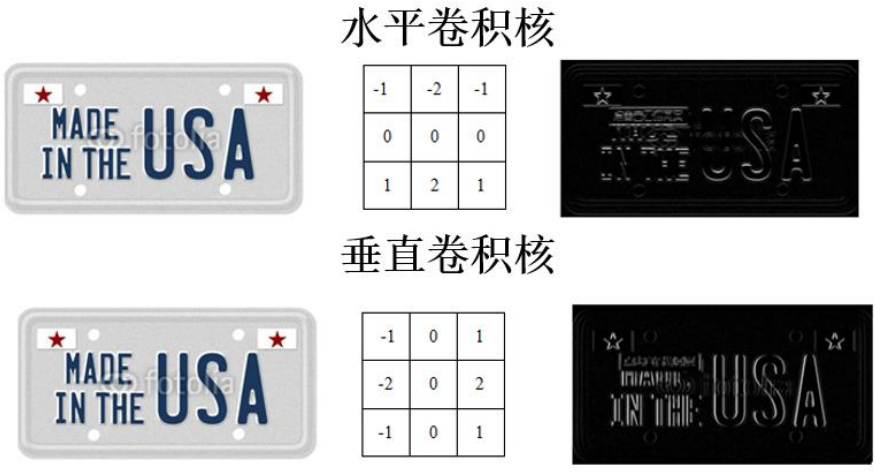
图四、多层感知机模型



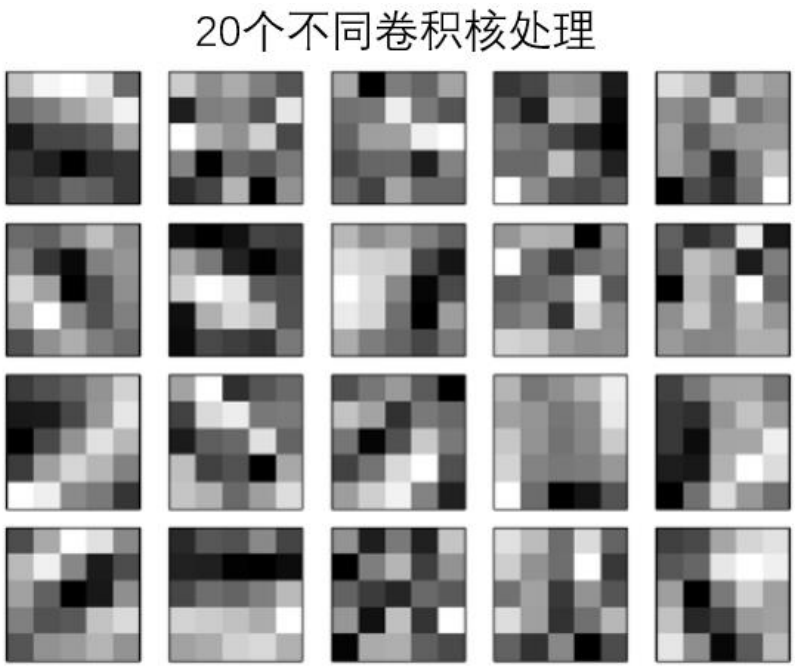
图五、卷积操作

图片就是一组二维矩阵，按照上图的卷积核蠕动的操作，就是图像数据的卷积。卷积操作中有两个重要概念就是局部感受野和贡献权重。在一个卷积核一次卷积操作中，卷积的输出矩阵中每个像素只“感受”输入矩阵的局部，在每次卷积操作中卷积核的权重不变。

卷积操作有什么作用呢？看下图



（a）



（b）

图六、卷积操作提取特征

正如多层感知机所不具备的，卷积操作对图片的空间结构非常敏感，对图片的特征特区非常有效。

假设图像是 n*n 大小的，通过 m*m 卷积核操作后，图像就变成了(n-m+1)*(n-m+1)，数据量可以少许下降，有利于后面神经网络训练。但是数据量还是比较大，于是有一种取样操作称为混合（pooling），可以使数据成倍减小并保留卷积留下来的特征，常用的是 Max-Pooling。

现在图像识别领域常用的方法是先通过卷积操作提取图像特征，再通过全连接神经网络进行分类。

三、实验过程

1、 编程环境及数据预处理

本次大作业所用编程语言是 Python3.6，模型搭建应用 Python 版本的 tensorflow

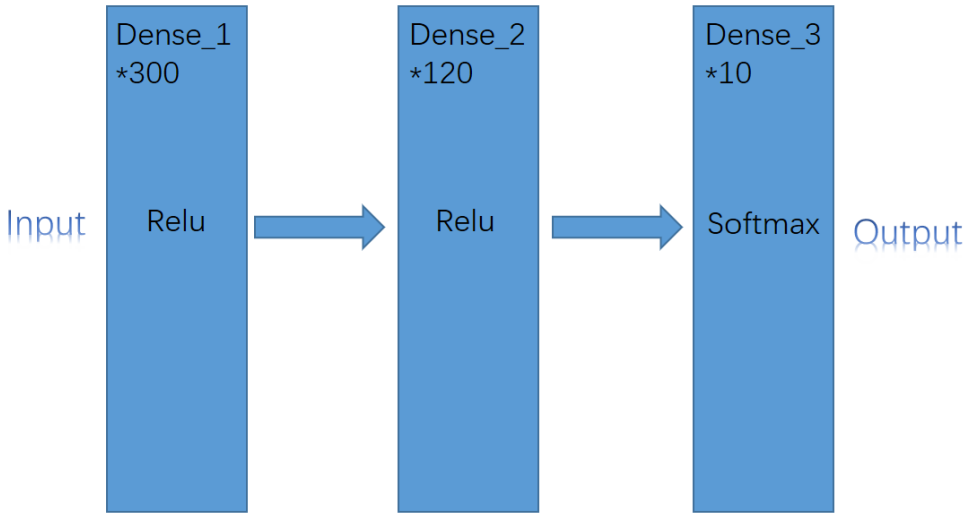
（1.12.0）及其高级 api keras。模型监控运用 tensorboard。

从 Kaggle 网站得到的数据有 42000 训练集和 28000 的测试样本。考虑到训练过程中需要验证集。我手动按照 9：1 分出 4200 个数据作为验证集，其余数据作为训练集。

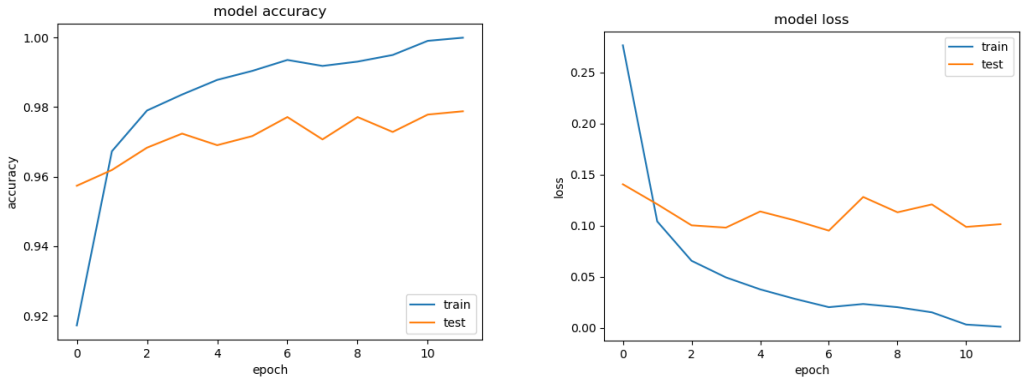
2、 模型搭建

（1）、多层感知机（MLP）

我首先设置了两个全连接层（Relu），和一个输出层（Softmax）,模型如图所示：



（a）结构图



（b）训练过程图

图七，MLP 第一版

经过调参之后，发现效果并不是太好。经过思考，考虑到拟合过程中振荡厉害，所以加上一些限制条件：

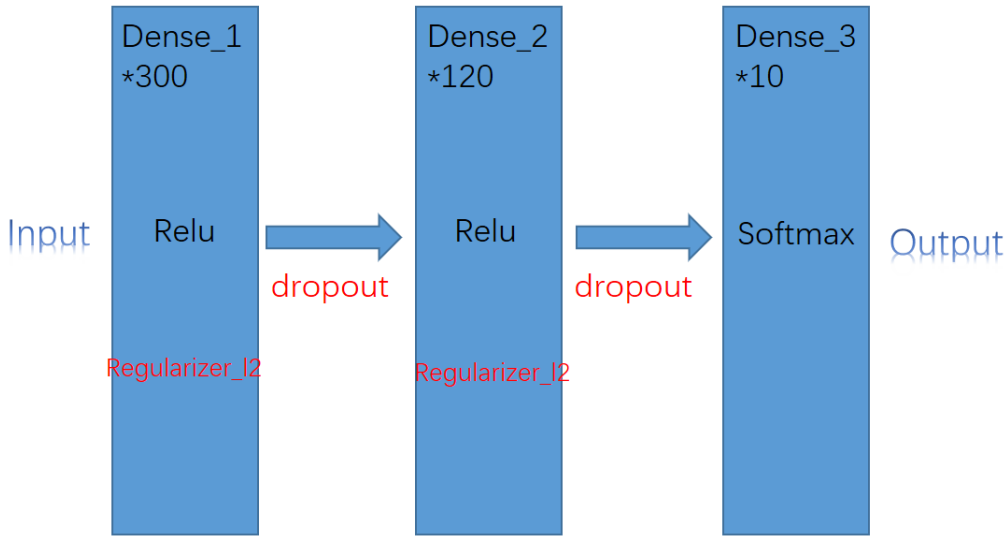
A、 tf.keras.callbacks.ReduceLROnPlateau()加入学习率动态调整，以提高学习速度和避免局部最后

B、

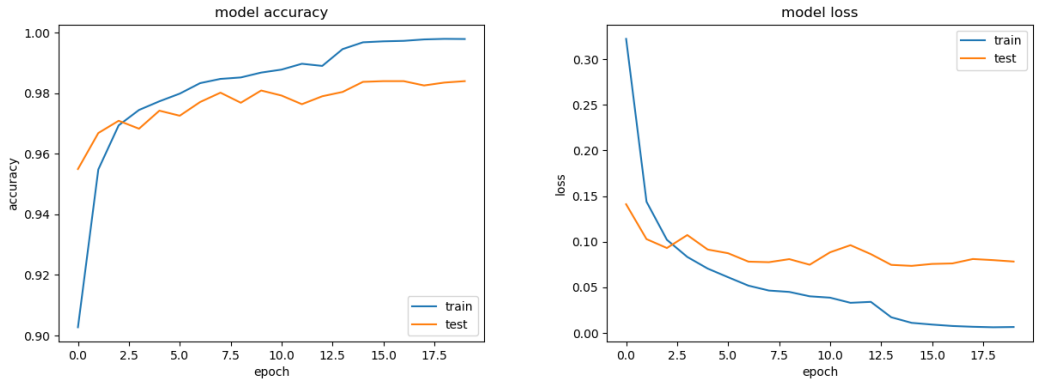
```
model.add(tf.keras.layers.Dense(300, activation=tf.nn.relu, bias_regularizer=tf.keras.regularizers.l2(0.01), input_shape=(784,), name='dense_1'))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(120, activation=tf.nn.relu, bias_regularizer=tf.keras.regularizers.l2(0.01), name='dense_2'))  # <- [None, 150]
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax, name='dense_3'))  # <- [None, 10]
# 注：每个层都自定义名称，用于tensorboard观察
```

加上 dropout 机制和正则化，以避免参数拟合的振荡和过拟合

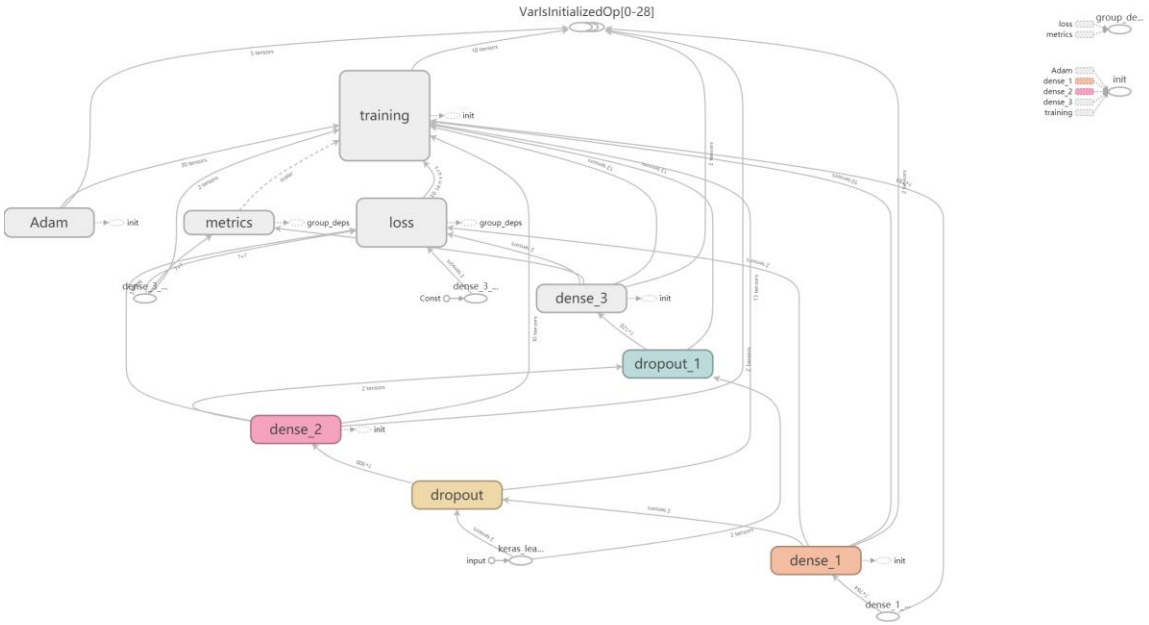
通过优化之后得到下面模型：



(a) 结构图



(b) 训练过程图



(c) Tensorboard 结构图

图八、MLP 第二版

模型分析：

A、在前面两个 Dense 层加了 dropout 和正则化的 MLP，训练过程参数振荡明显减弱，拟合曲线更加平滑，表现比之前好。

B、经过调解神经元个数，发现第一层 300 个，第二层 120 个，否则 loss 衰减很慢，而且降不下来。

C、性能表现良好：手动划分的验证集准确率达到 0.9840。预测测试集在 Kaggle 官网给出的准确率是 0.98114，如图：

Submission and Description	Public Score
MLP_keras_submit_v2_0.9840.csv 2 hours ago by sl2015311081 add submission details	0.98114

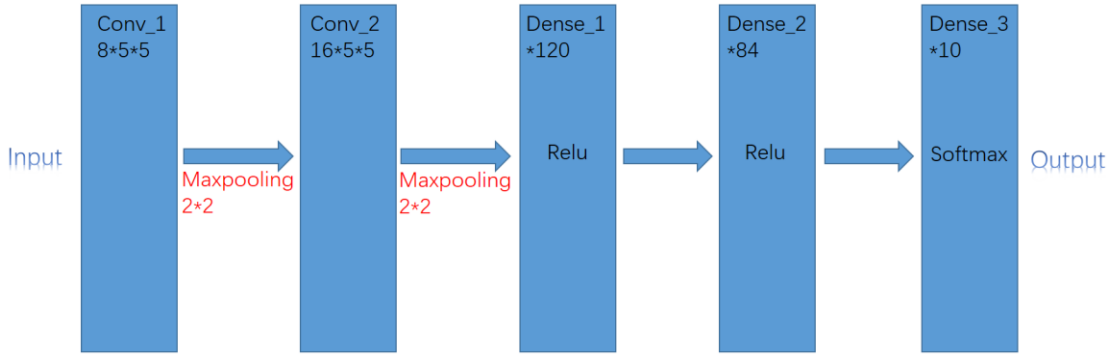
图九、Kaggle 网站对 MLP_v2 的评分

（2）、卷积神经网络

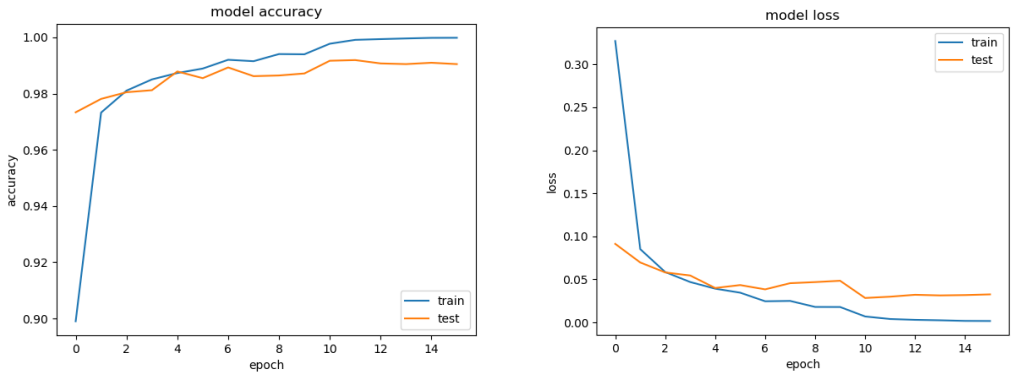
由之前背景介绍可以知道，在图像识别中应用卷积神经网络（CNN）可以很好的提取特片的特征，得到更好的模型。

（i）第一版 CNN

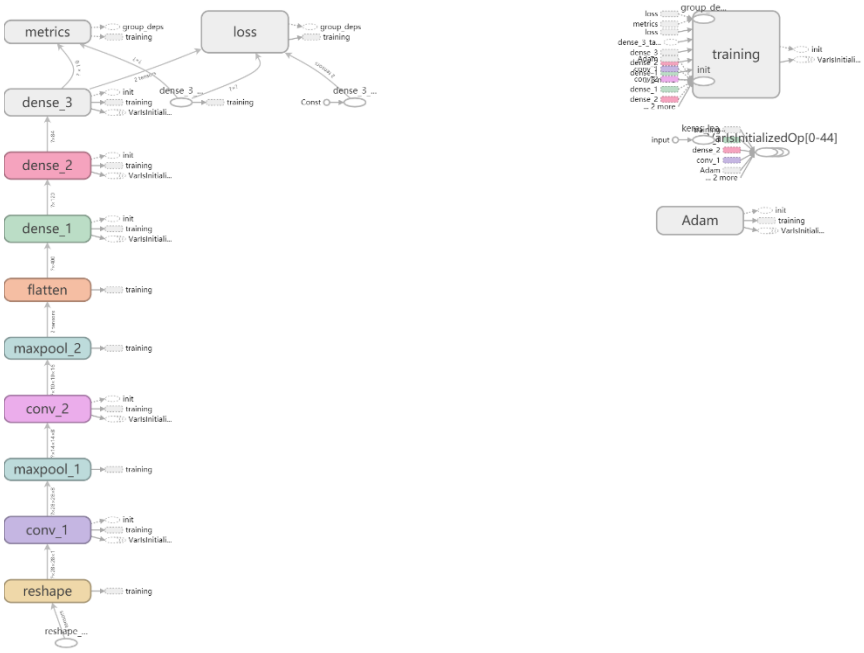
开始我和 MLP 一样只搭建了两层卷积层（分别是 8*5*5 和 16*5*5），两层 Pooling 层（2*2），两层 Relu 的 Dense（分别是 120，84）和最后一层 Softmax 的 Dense（10），CNN 第一版结构图如下：



（a）模型结构图



（b）训练过程图



（d）tensorboard 可视化结构

图十、第一版 CNN

模型分析：

A、该模型比 MLP 模型有了很大的提升，从训练过程图可以看出，比 MLP 收敛地更好，振荡更小。训练集和测试集差距更小。

B、模型效果也比 MLP 表现更好，经过 Kaggle 网站评价，分数达到 0.98828。

CNN_keras_submit.csv 0.98828

20 hours ago by sl2015311081

Conv8,Conv16,ratio0.1,

C、经过一系列调参，可以看到过程 loss 和 accuracy 的振荡依然比较大。

（ii）、第二版 CNN

模型改进：

其实上述这个模型还是比较简陋的，为了更好地改进模型的性能，我又引入了其他东西。

```
model.add(tf.keras.layers.Reshape((28, 28, 1), input_shape=(784,), name='reshape'))   # <- [None, 28, 28, 1]
model.add(tf.keras.layers.Conv2D(filters=8, kernel_size=(5, 5), padding='same', activation=tf.nn.relu, name='conv_1'))  # <- [None, 28,
model.add(tf.keras.layers.MaxPool2D(pool_size=(2, 2), strides=(2, 2), padding='valid', name='maxpool_1'))  # <- [None, 14, 14, 8]
model.add(tf.keras.layers.Conv2D(filters=16, kernel_size=(5, 5), strides=(1, 1), padding='valid', activation=tf.nn.relu, name='conv_2'))
model.add(tf.keras.layers.MaxPool2D(pool_size=(2, 2), strides=(2, 2), padding='valid', name='maxpool_2'))  # <- [None, 5, 5, 16]
model.add(tf.keras.layers.Flatten(name='flatten'))   # <- [None, 400]
model.add(tf.keras.layers.Dense(120, activation=tf.nn.relu, bias_regularizer=tf.keras.regularizers.l2(0.01), name='dense_1'))  # <- [Non
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(84, activation=tf.nn.relu, bias_regularizer=tf.keras.regularizers.l2(0.01), name='dense_2'))  # <- [None
# model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax, name='dense_3'))  # <- [None, 10]
# 注：每个层都自定义名称，方便tensorboard观察
```

```
callbacks = [tf.keras.callbacks.EarlyStopping(monitor='val_loss', min_delta=0.0001, patience=4),
             tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=1, min_lr=0.000015 * 0.2),
             tf.keras.callbacks.TensorBoard(log_dir='logs', histogram_freq=0)
             ]
```

A、为了减弱过拟合现象，在全连接层后加上 dropout 机制，可以更好地找到数据的泛化性质。

B、在全连接层内加上运用 L2 正则化，可以很好地减弱训练时，参数的振荡，收敛性能更

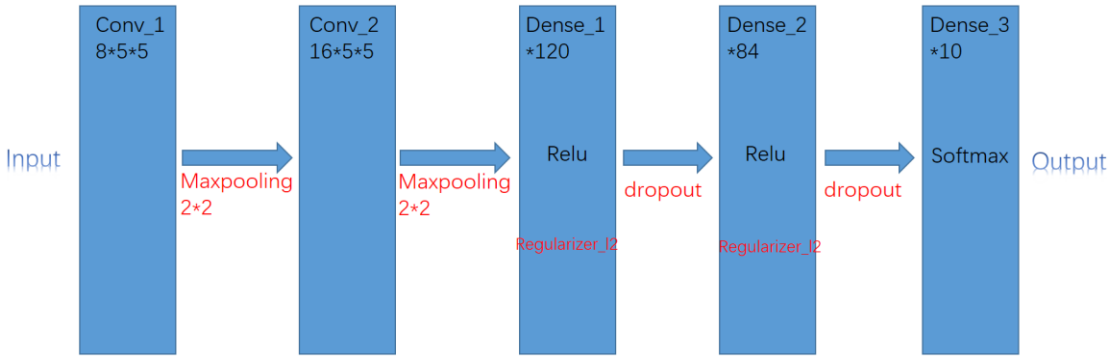
好。

C、运用多种 keras 训练的回调函数，可以很好地控制在适当的时候停止训练，以防止过拟合。

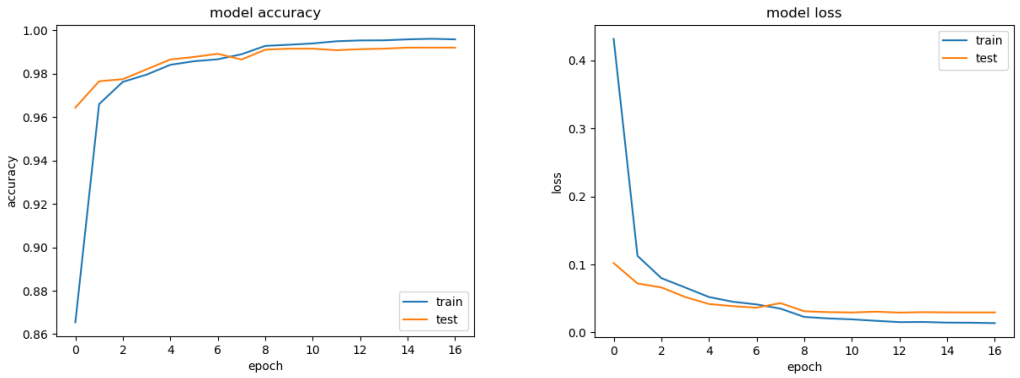
D、通过设置学习率动态变化，可以很好地提升训练速度，也比较好地避免落入局部最优解中。

E、验证集分离比提升到原训练集的 0.2，可以更好地用于验证。

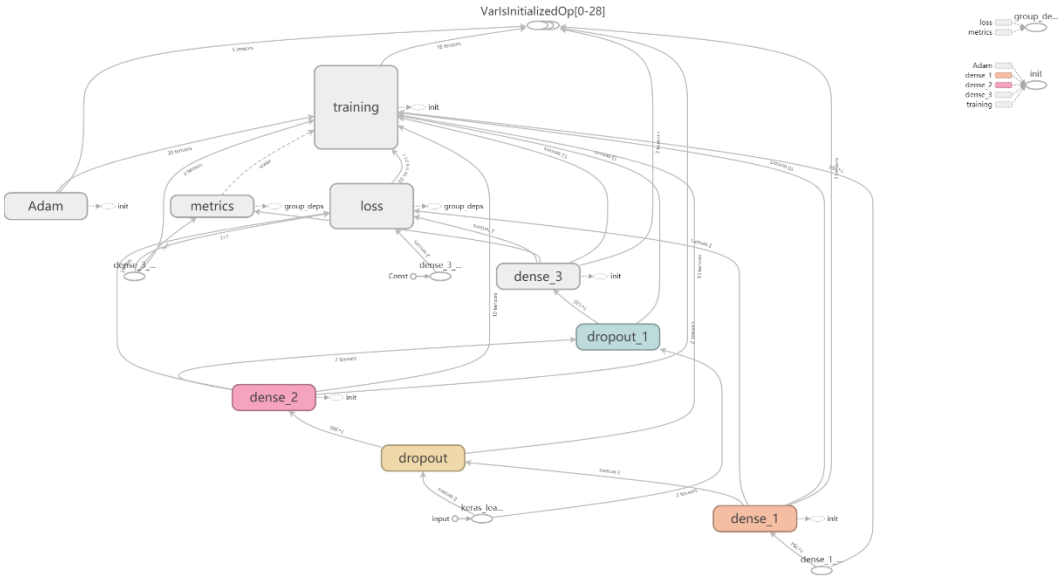
以下是第二版 CNN 模型概况：



（a）第二版 CNN 结构图



（b）训练过程图



（c）tensorboard 可视化结构图

图十一、第二版 CNN

模型分析：

A、从上面的结果可以看出，加上 dropout 和正则化的第二版 CNN，训练具有非常好的收敛性。从 loss 和 accuracy 看，训练集和测试集收敛地都非常好。趋势很平缓。

B、模型用 Kaggle 评分达到 0.99028，为我所有模型最高得分

Submission and Description	Public Score
CNN_keras_submit_drop1_regu2_r0.2_0.9919_120_84.csv 5 minutes ago by sl2015311081 add submission details	0.99028

3、 实验思考

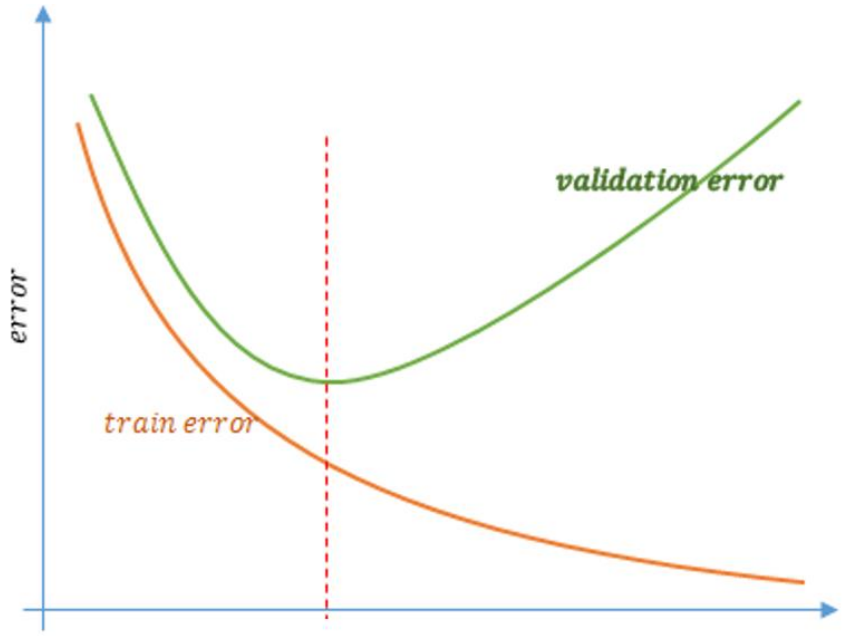
（1）、总体模型来说 CNN 要强于 MLP，而且 CNN 比较容易达到比 MLP 更高的分数。因为 CNN 网络能更好的抓住图片的整体特征。

（2）、由于模型选择等原因，在训练的时候容易造成过拟合问题。Dropout 机制和正则化机制，可以很好的抑制参数随着训练的振荡，可以很好的避免过拟合。

（3）、不同模型的选择比如神经元个数，batch 大小，学习率等，会影响模型训练的性能。

（4）、CNN 能够更好地提取出图片的特征，模型深度可以做的比 MLP 更深，模型有效性也更好。

（5）、验证集用来控制训练过程如图所示：



图十二、过拟合训练过程

我们假设训练集和验证集是独立同分布的。在训练过程中，由于优化器会一直朝着训练集方向拟合，但是训练集毕竟数量有限，最终会偏移它本身的统计属性。用一个独立同分布的验证集可以用来检测模型是不是矫枉过正，是不是偏移他本身的统计属性。

在训练过程中，我们会在竖直红色虚线的地方地址训练，从而可以避免过拟合。但是由于实际训练过程中，validation data 的 error 不是平滑的，它是振荡的 V 型曲线，所以我们会有一个“容忍度”的设定，来看看验证集是不是真的显示过拟合。在本实验中我利用早

停法，给模型训练设置监管，我的容忍 epoch 设的是 4

```
callbacks = [tf.keras.callbacks.EarlyStopping(monitor='val_loss', min_delta=0.0001, patience=4),
             tf.keras.callbacks.ReduceLR0nPlateau(monitor='val_loss', factor=0.2, patience=1, min_lr=0.000015 * 0.2),
             tf.keras.callbacks.TensorBoard(log_dir='logs', histogram_freq=0)
             ]
```

（6）、从训练集分离出验证集也是一个头疼的问题（问题的根源是数据量不够）。验证集分少了，会导致验证集不能很好体现统计性。验证集分多了，导致数据不够，会导致魔性训练效果变差。经过多次调参，我把验证集和训练集比例设成 1：5。

四、实验结果

1、 MLP 最好分数是 0.98114，提交一次。

Submission and Description	Public Score
MLP_keras_submit_v2_0.9840.csv 2 hours ago by sl2015311081 add submission details	0.98114

2、 CNN 最好分数是 0.99085，提交 5 次。

Submission and Description	Public Score
CNN_keras_submit_drop2_regu2_0.9924_110_100.csv 9 minutes ago by sl2015311081 add submission details	0.99085

Submission and Description	Public Score
CNN_keras_submit_drop1_regu2_r0.2_0.9919_120_84.csv 5 minutes ago by sl2015311081 add submission details	0.99028