卷积神经网络实现MNIST分类器

姓名：胡畅

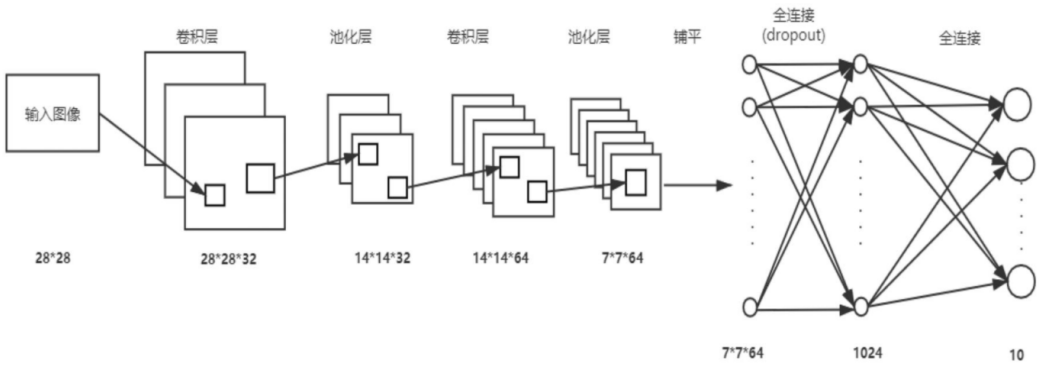
班级：大数据1801

学号：2018317230105

实验要求

用`tensorflow`框架实现卷积神经网络。网络结构为：`输入层->卷积层->池化层->激活函数->卷积层->池化层->激活函数->全连接->输出`

具体网络结构如下图所示：



实验方案

本次实验是想用卷积神经网络实现`MNIST`数据集的分类，`tensorflow2`相比起`tensorflow1`接口更加简洁，优化的更好，所以本次实验使用`tensorflow2`，这样可以更加简洁的构建本次网络。

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
model = tf.keras.models.Sequential() # 得到一个模型序列
model.add(keras.layers.Conv2D(filters=32, kernel_size=(3, 3),
padding="SAME", strides=1, input_shape=[28, 28, 1])) # 卷积层
model.add(keras.layers.MaxPool2D(pool_size=(2, 2), strides=2,
padding="SAME")) # 池化层
model.add(keras.layers.ReLU()) # ReLU激活
model.add(keras.layers.Conv2D(filters=64, kernel_size=(3, 3),
padding="SAME", strides=1))
```

```
model.add(keras.layers.MaxPool2D(pool_size=(2, 2), strides=2,
padding="SAME"))
model.add(keras.layers.ReLU())
model.add(keras.layers.Flatten()) # 将矩阵变成一维数据
model.add(keras.layers.Dense(1024)) # 全连接层，输出维度是1024维
model.add(keras.layers.Dense(10,
activation=keras.layers.Softmax())) # 全连接层，输出维度是10维，在经过
sofmax激活
model.compile(optimizer=keras.optimizers.Adam(lr=1e-3),
loss=tf.losses.SparseCategoricalCrossentropy(), metrics=
["accuracy"])
# 下面是开始训练过程
history = model.fit(x = np.expand_dims(train_dataset, axis=3), y
= train_label, batch_size=512,epochs=30, validation_data=
(np.expand_dims(test_dataset, axis=3), test_label)) # history中保
存实验中的正确率和损失等信息，之后会利用这些信息进行后续的分析工作
```

首先了解一下`tensorflow2`里面卷积核的大小和步长对卷积过程中图片大小的变化如下：

$$
\begin{gathered}
input = MXN \\
\text{下面对于一个维度} M \text{进行计算} \\
stride = F \\
kernel\_size = K \\
if\ padding =" VALID " \\
out\_size = \left\lceil \frac{M - K + 1}{F} \right\rceil \\
if\ padding =" SAME " \\
out\_size = \left\lceil \frac{M}{F} \right\rceil
\end{gathered}
$$

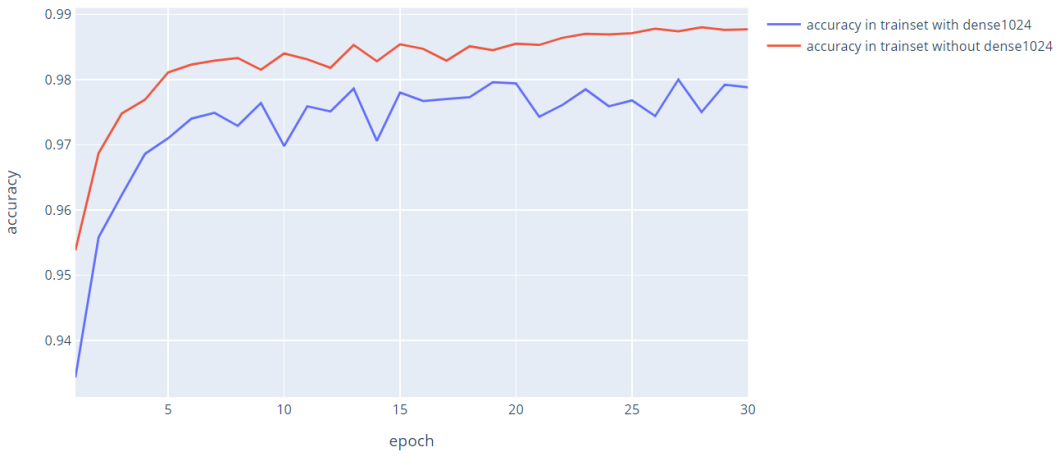
因此上述模型的网络参数大小如下图所示

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_________________________________________________________________
re_lu (ReLU)                 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
re_lu_1 (ReLU)               (None, 7, 7, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 3136)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              3212288
_________________________________________________________________
dense_1 (Dense)              (None, 10)                10250
=================================================================
Total params: 3,241,354
Trainable params: 3,241,354
Non-trainable params: 0
_________________________________________________________________
```

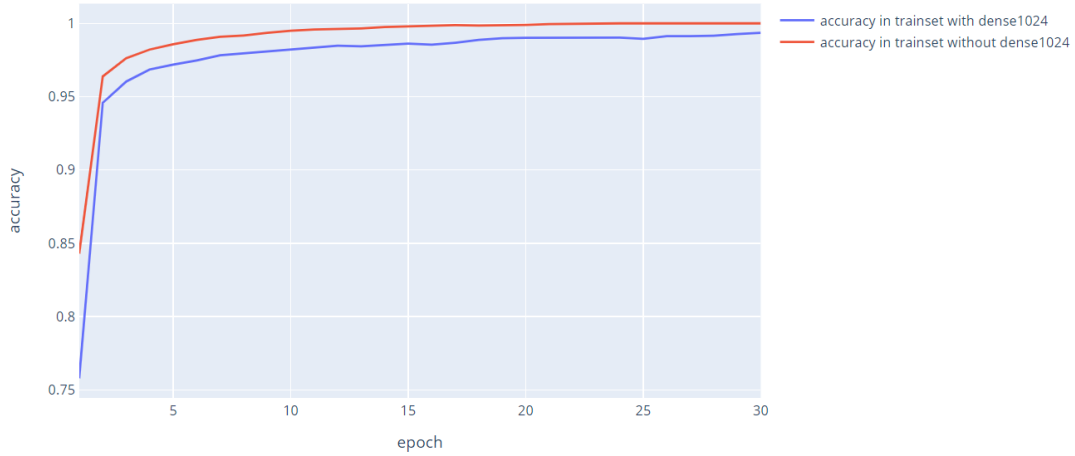
其实中间`Flatten->dense(1024)`这一层可以省去，直接到最后一个`dense`层即可，中间再加一层`dense(1024)`层极大的增加了网络参数，会导致训练时间增加，也可能导致过拟合的问题。

实验结果

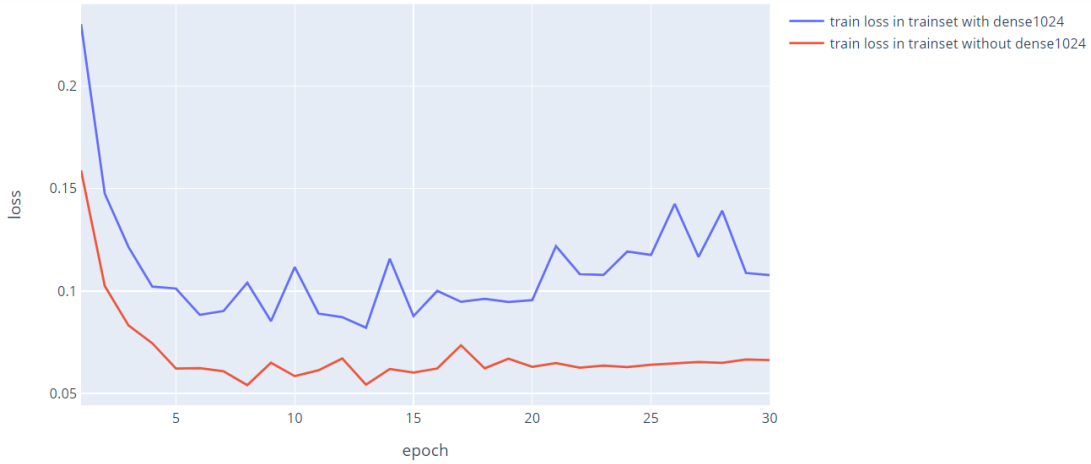
- 在测试集上的正确率



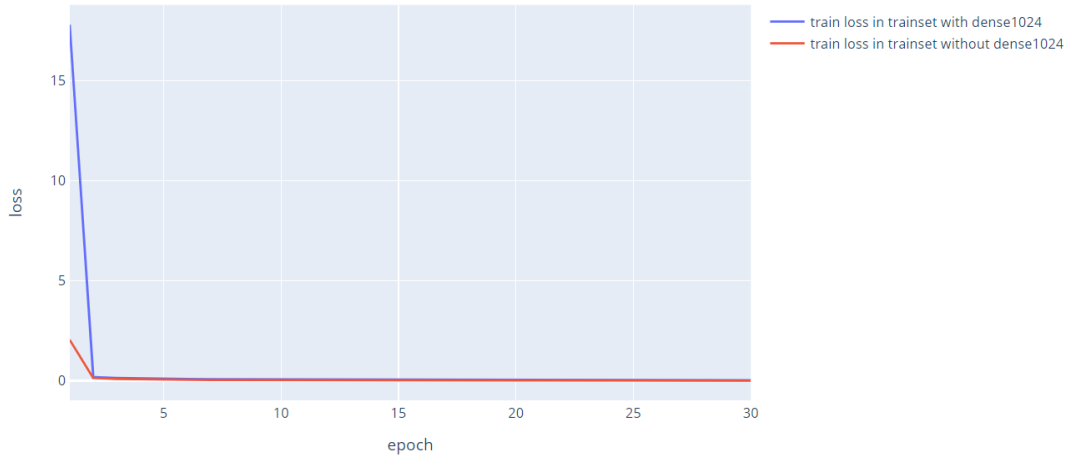
- 在训练集上的正确率



- 在测试集上的损失



- 在训练集上的损失



由上面的含有`Dense1024`的网络，不管是在测试集还是验证集上的正确率都要比不含`Dense1024`网络的正确率要低，而且对于损失来说，在测试集上含有`Dense1024`层的网络比不含`Dense1024`层的网络损失要更大一点，因此从实验结果上来看从网络结构中提出`Dense1024`层，网络的整体表现会更好，原因上来看是因为网络参数的增加导致训练变得更加困难，而不是因为数据集的大小问题，因为即使加了`Dense1024`层，整体的正确率仍然很高。从在训练集上看，增加了`Dense1024`层的

网络达不到`100%`的正确率，可能是进入到了局部最优值，但是这种可能性也比较小，因为优化器的选择是一样的，而且正确率很可观。

Source code & pretrained model are available at https://github.com/Chang-Le Hung/Deep-Learning-Tutorials

实验总结

本次实验代码比上一次代码简单很多，因为只需要自己构造网络结构即可，数据的传递全部由框架来完成。框架可以帮助开发者和研究人员快速进行工程实现，程序员只需要关注算法本身，不需要去管更底层的具体实现，非常方便。