人智大作业 2 报告

2017011534 王思程 自 71

1.import 包

Import 了 tensorflow，pandas，numpy，matplolib（使用的是 tensorflow 的 2.0 版本）

使用了 tensorflow.keras.layers 中的 Conv2D, MaxPooling2D, Dropout, Dense, Flatten, ReLU, BatchNormalization, GlobalAveragePooling2D

2.读入数据并更改维度

用 numpy.load 读入.npy 文件，并修改维度为（xxx，28，28，1）数据类型改为 float

用 pandas.read_csv 读 csv 文件，并取出 label 那一列

3.画出来看一下数据长什么样

用修改维度前的 ndarray,隔 1000 取一张图出来显示，共显示 25 张，可以看到要分类的东西长什么样，并看到其有相当的椒盐噪声

4.数据处理

4.1normalization

都是 float 形式了，除以 255 即可

4.2shuffle

在 3 画图时就看到，原训练数据是同一种类的图片在一起连着的，有必要将其打乱以获得更好的效果。可以生成对应长度的 index，然后将其 shuffle，再将训练集的数据和标签按打乱后的 index 排，这样就保证数据和标签还是对应的

4.3one-hot 编码

标签原本是 0-9 的数，将其编码为 one-hot 形式。

其实也可以不改这个，那么分类时 loss 函数使用 Sparse Categorical Cross Entropy 而不是 categorical_corssentropy 了。

4.4 将数据进一步切分

我将训练集中 25000 个用于训练网络，5000 用于当测试集，这样是否过拟合自己就知道，没必要每次训练出来都交 kaggle，自己的 val_accuracy 就和最终交 kaggle 的结果差不多

5.模型

5.1 建立一些小模块

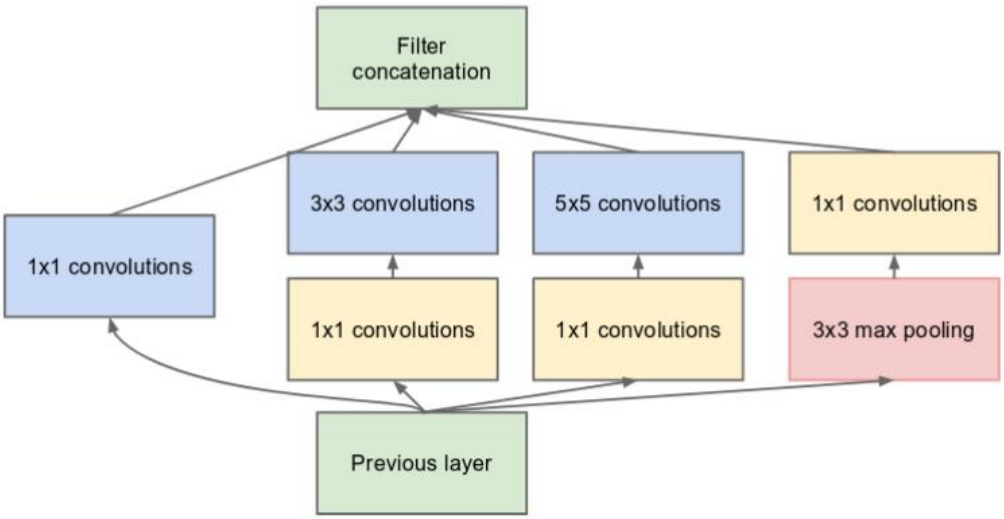
5.1.1 ConvBNRelu : conv + batch normalization + relu

一个 Conv2D 层，一个 BatchNormalization()，一个 relu()

*(参考了*https://blog.csdn.net/abc13526222160/article/details/95472241*)*

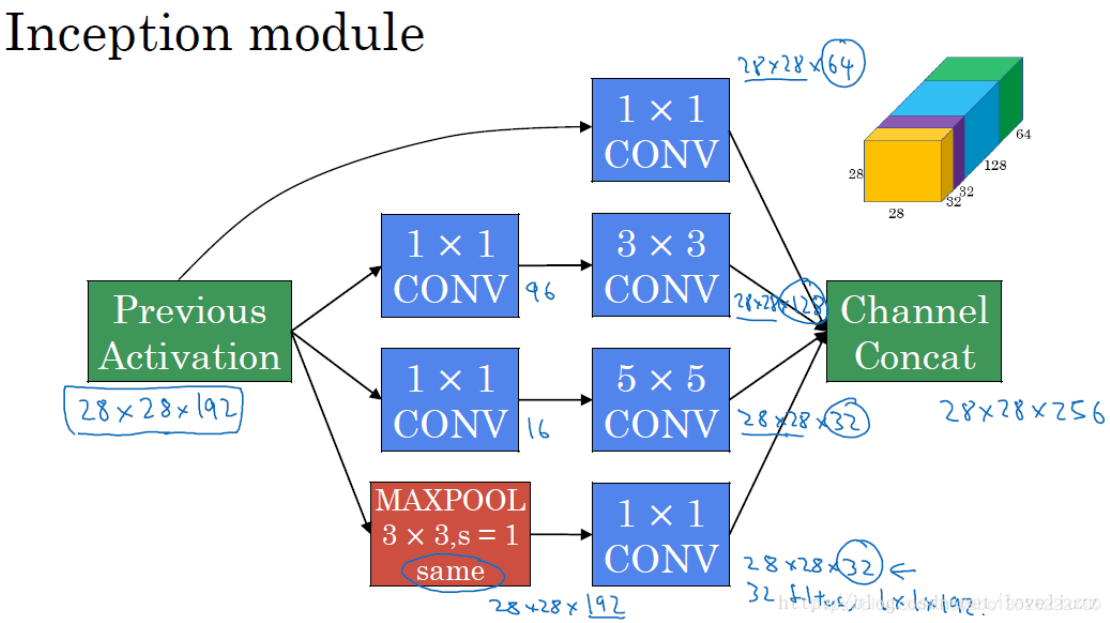
将其作为基本单元替代 Con2D 有一些好处，一是收敛速度更快了：因为不在处于 sigmoid 函数的饱和区，梯度信息会更大。二是更加鲁棒：更不容易梯度消失，让 learning rate 可以更容易选择

5.1.2 Inception 层



(b) Inception module with dimension reductions

基本按上图实现的*（图来自 Going deeper with convolutions 这篇论文）*



但是 channel 数有所不同，按照上图这个比例写的，其中上图的 32 作为一个基本单元，可以改变其大小*（图来自 https://blog.csdn.net/abc13526222160/article/details/95472241）*

Strides 设计的可以自己控制，因为这 4 个计算完要在页的维度垒叠在一起 concat(…,axis=3)，所以要保证各自计算完后 width 和 height 一样，所以右三个

两层的，第一层 strides=1，第二层 strides=给定的 strides，左边那个就 strides=给定的 strides。

其中 5*5 卷积核本来计算量很大，但在前面加一个 1*1 卷积核进行降维，可以减少计算压力。

追求可解释性的话，这个单元很好用的原因就是它有各种卷积和池化层，相当于让机器训练自己看那种效果更好，而不是人来指定具体哪一种

5.2 设计 Model

5.2.1 普通的卷积池化后加全连接层的网络

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 64)        640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 32)        51232
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 7, 7, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 1568)              0
_________________________________________________________________
dense (Dense)                (None, 128)               200832
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 253,994
Trainable params: 253,994
Non-trainable params: 0
```

感觉可提升空间不大，试过各种参数，最后 val_accuracy 也就 80-86 之间吧。

5.2.2 用了两个 InceptionBlk，不同步长

```
Model: "sequential_197"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_223 (Conv2D)          (None, 28, 28, 192)       1920
_________________________________________________________________
batch_normalization_175 (Bat (None, 28, 28, 192)       768
_________________________________________________________________
re_lu_175 (ReLU)             (None, 28, 28, 192)       0
_________________________________________________________________
inception_blk_29 (InceptionB (None, 14, 14, 256)       165168
_________________________________________________________________
inception_blk_30 (InceptionB (None, 14, 14, 256)       178480
_________________________________________________________________
dropout_4 (Dropout)          (None, 14, 14, 256)       0
_________________________________________________________________
global_average_pooling2d_9 ( (None, 256)               0
_________________________________________________________________
dense_11 (Dense)             (None, 10)                2570
=================================================================
Total params: 348,906
Trainable params: 347,050
Non-trainable params: 1,856
```

其中，最后没有使用 Flatten 后加一个 Dense，而是用一个 global_average_pooling2d 层。若使用 Dense 层则参数过多训练很慢，用 GAP（global average pooling）按作者的话说"is easier to interpret and less prone to overfitting than traditional fully connected layers"（*论文 Network In Network*）就更不容易过拟合，确实这次只切分后只有 25000 个训练样本，不是很多，要多注意过拟合问题。

下图是采用传统 Flatten + Dense 层的，参数数量比用 GAP 多了近 20 倍。

```
Model: "sequential_223"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_249 (Conv2D)          (None, 28, 28, 192)       1920
_________________________________________________________________
batch_normalization_201 (Bat (None, 28, 28, 192)       768
_________________________________________________________________
re_lu_201 (ReLU)             (None, 28, 28, 192)       0
_________________________________________________________________
inception_blk_33 (InceptionB (None, 14, 14, 256)       165168
_________________________________________________________________
inception_blk_34 (InceptionB (None, 14, 14, 256)       178480
_________________________________________________________________
dropout_6 (Dropout)          (None, 14, 14, 256)       0
_________________________________________________________________
flatten_2 (Flatten)          (None, 50176)             0
_________________________________________________________________
dense_13 (Dense)             (None, 128)               6422656
_________________________________________________________________
dense_14 (Dense)             (None, 10)                1290
=================================================================
Total params: 6,770,282
Trainable params: 6,768,426
Non-trainable params: 1,856
```

5.2.3 实现 **GoogLeNet** 缩短版

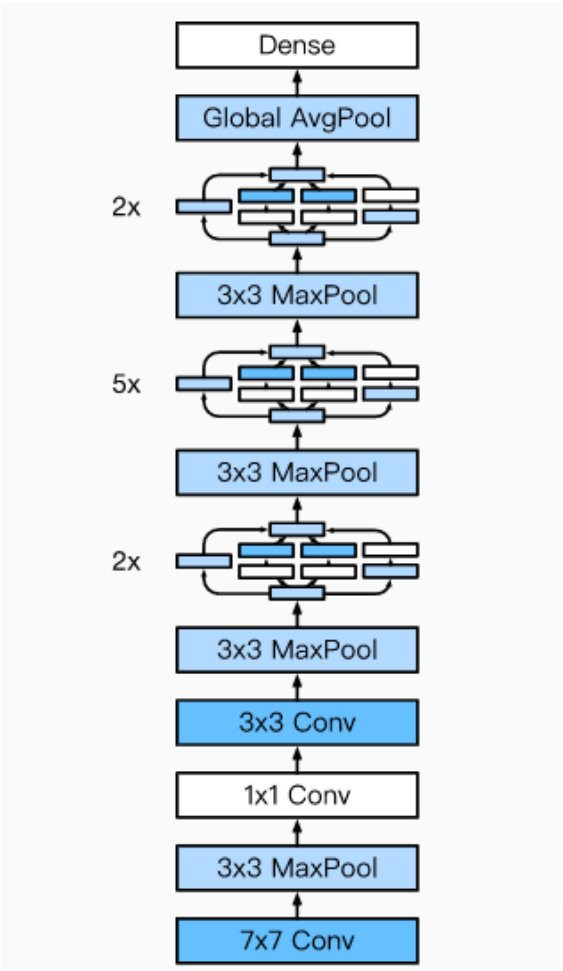


Fig. 7.4.2 Full GoogLeNet Model

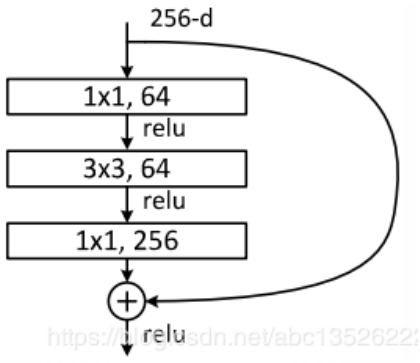
```
sequential0 output shape:    (1, 64, 24, 24)
sequential1 output shape:    (1, 192, 12, 12)
sequential2 output shape:    (1, 480, 6, 6)
sequential3 output shape:    (1, 832, 3, 3)
sequential4 output shape:    (1, 1024, 1, 1)
dense0 output shape:         (1, 10)
```

上图是 https://d2l.ai/chapter_convolutional-modern/googlenet.html 中给出的一个版本的 GoogLeNet 模型，这个太深了，我自己给它剪短了点，结果如下

```
Model: "sequential_292"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_320 (Conv2D)          (None, 22, 22, 64)        3200
_________________________________________________________________
max_pooling2d_77 (MaxPooling (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_321 (Conv2D)          (None, 10, 10, 64)        4160
_________________________________________________________________
conv2d_322 (Conv2D)          (None, 8, 8, 192)         110784
_________________________________________________________________
max_pooling2d_78 (MaxPooling (None, 4, 4, 192)         0
_________________________________________________________________
inception_blk_44 (InceptionB (None, 4, 4, 256)         165168
_________________________________________________________________
inception_blk_45 (InceptionB (None, 4, 4, 272)         197829
_________________________________________________________________
inception_blk_46 (InceptionB (None, 4, 4, 288)         221886
_________________________________________________________________
inception_blk_47 (InceptionB (None, 4, 4, 304)         247323
_________________________________________________________________
inception_blk_48 (InceptionB (None, 4, 4, 320)         274140
_________________________________________________________________
max_pooling2d_84 (MaxPooling (None, 1, 1, 320)         0
_________________________________________________________________
inception_blk_49 (InceptionB (None, 1, 1, 256)         191792
_________________________________________________________________
inception_blk_50 (InceptionB (None, 1, 1, 512)         603744
_________________________________________________________________
global_average_pooling2d_12  (None, 512)               0
_________________________________________________________________
dense_17 (Dense)             (None, 10)                5130
=================================================================
Total params: 2,025,156
Trainable params: 2,018,808
Non-trainable params: 6,348
```

5.2.3 ResNet

5.2.3.1 Bottleneck Residual Block



基本的残差模块，由于有残差所以输入输出的通道数一样。

且有两个 1*1 大大缩小参数数量，使得可以层数很深。

5.2.3.2 Res Block

先来个 ConvBNRelu，然后不同通道数的 Bottleneck Residual Block 叠起来，最后再一个全连接层分类。

5.2.3.3 Res Net

选择 64，256，256，512 通道的 Bottleneck Residual Block 有几个，垒叠出来就是 ResNet。

```
Model: "res_net_27"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential_168 (Sequential)  multiple                  896
_________________________________________________________________
sequential_169 (Sequential)  multiple                  9664
_________________________________________________________________
sequential_170 (Sequential)  multiple                  61376
_________________________________________________________________
sequential_172 (Sequential)  multiple                  311040
_________________________________________________________________
sequential_174 (Sequential)  multiple                  381184
_________________________________________________________________
global_average_pooling2d_17  multiple                  0
_________________________________________________________________
dense_20 (Dense)             multiple                  5130
=================================================================
Total params: 769,290
Trainable params: 763,018
Non-trainable params: 6,272
_________________________________________________________________
```

5.3 编译 Model

优化器选择的 adam

Loss 函数选择的 categorical_crossentropy

Metrics 选择的 accuracy

5.4 训练 Model

BATCH_SIZE = 32。Epochs 初始设置为 50，然后在 callbacks 里用 earlingstopping 监视 val_accuracy，耐心为 2，不再增长时停止训练。validation_data 就是最开始切出来的那 5000 个样本。

6.预测和保存

就预测给的测试集样本，然后改成要求的 DataFrame 格式保存为 csv 文件交到 Kaggle 上就行了。