手写数字识别

机器学习第三次

邹天舒

北京交通大学 | 指导老师 魏祥

目录

1. 摘要

卷积神经网络（Convolutional Neural Networks / CNNs / ConvNets）与普通神经网络非常相似，它们都由具有可学习的权重和偏置常量(biases)的神经元组成。每个神经元都接收一些输入，并做一些点积计算，输出是每个分类的分数，普通神经网络里的一些计算技巧到这里依旧适用。

所以哪里不同呢？卷积神经网络默认输入是图像，可以让我们把特定的性质编码入网络结构，使是我们的前馈函数更加有效率，并减少了大量参数。

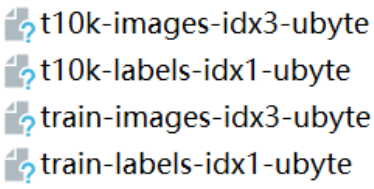
2. 需求分析

基于 Tensorflow.js 的在线手写数字识别
基本需求：
1. 三个 js 文件，分别完成：网络训练以及模型保存、模型加载及准确率测试、在线手写数字识别；
2. 模型测试准确率要高于 99.3%（尽量）；
3. 在线手写数字识别需要能够通过鼠标在画布中写入 0~9 数字，并进行实时识别，按空格键清除（下图示例）。测试需具有一定的准确性。

3. 实验过程

3.1 模型训练

在进行模型训练之前，我们需要加载数据集，将下载好的数据集放到项目目录中：



然后读取文件：

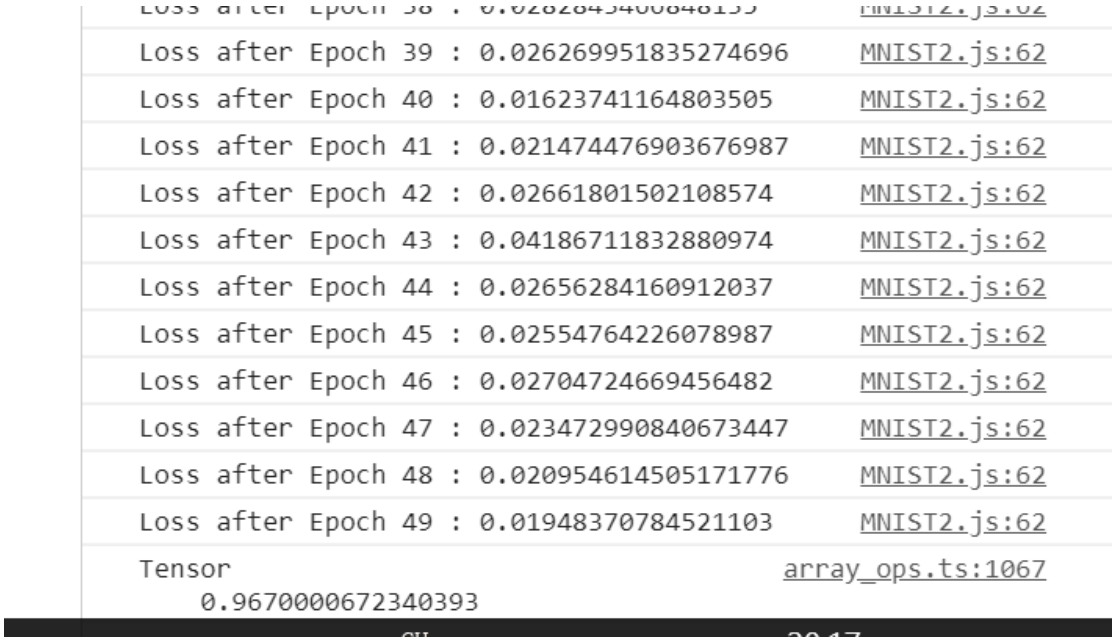
```
function loadMNIST(callback) {
  let mnist = {};
  let files = {
    train_images: 'train-images-idx3-ubyte',
    train_labels: 'train-labels-idx1-ubyte',
    test_images: 't10k-images-idx3-ubyte',
    test_labels: 't10k-labels-idx1-ubyte',
  };
  return Promise.all(Object.keys(files).map(async file => {
```

```
    mnist[file] = await loadFile(files[file])
  }))
    .then(() => callback(mnist));
}
```

我们为模型添加如下的网络进行训练：

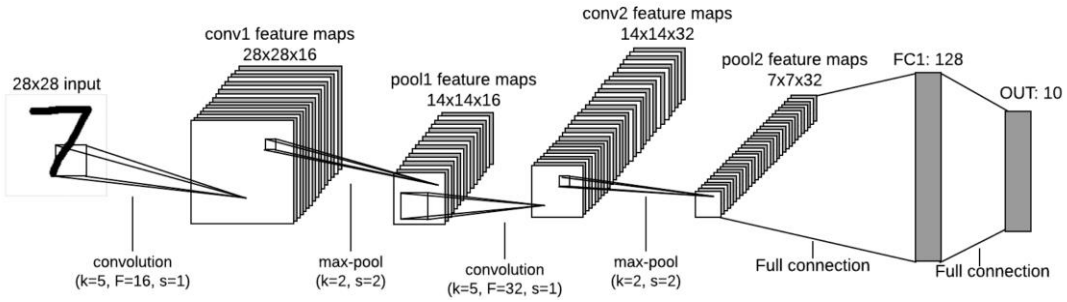
```
model.add(tf.layers.dense({
      units: 128,
      inputShape: [784],
      activation: 'relu'
   }));

model.add(tf.layers.dense({
      units: 64,
      activation: 'relu'
   }));

model.add(tf.layers.dense({
      units: 64,
      activation: 'relu'
   }));

model.add(tf.layers.dense({
      units: 10,
   }));
```

运行出来的准确度约是 0.967，运行截图如下：



3.2 模型优化

那么为了提高模型训练的精确度，我们可以采用 CNN 神经网络，如下图所示：

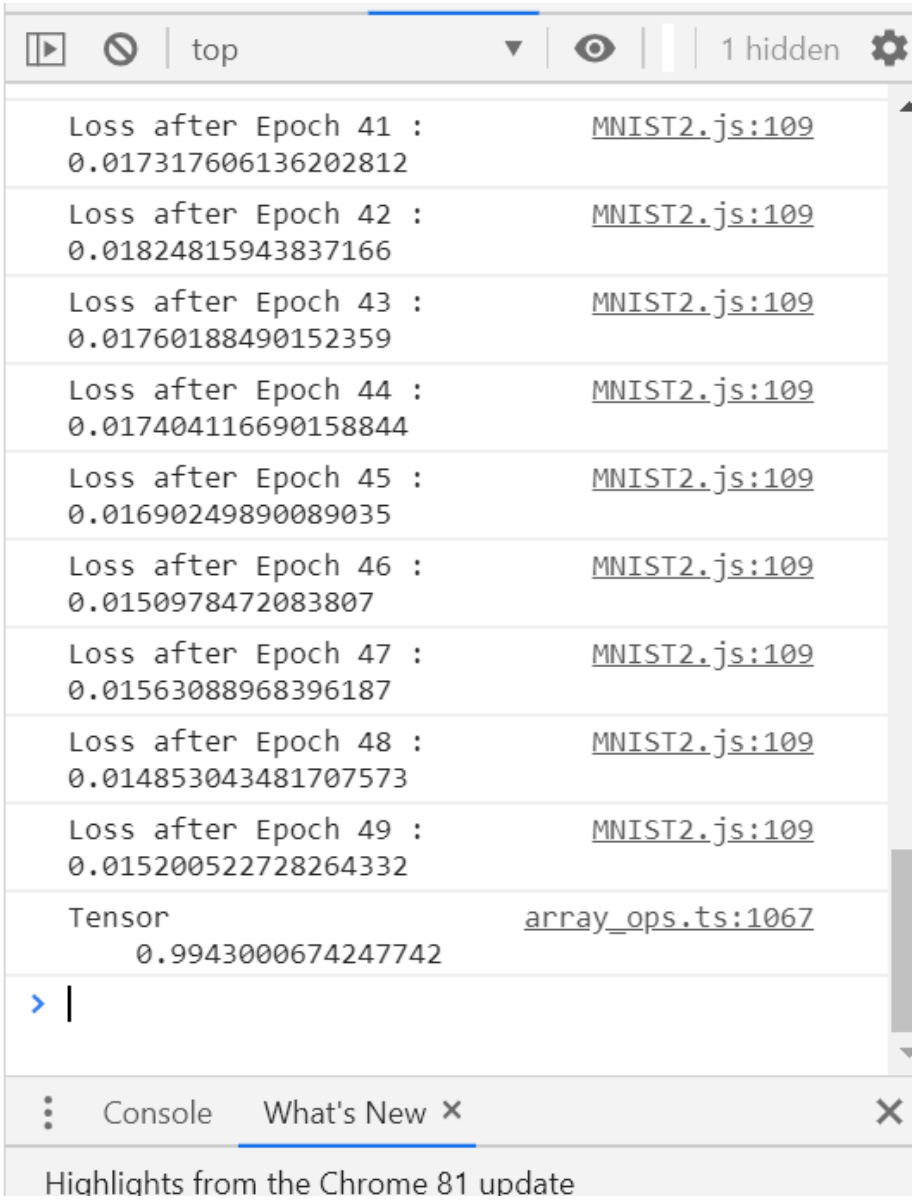


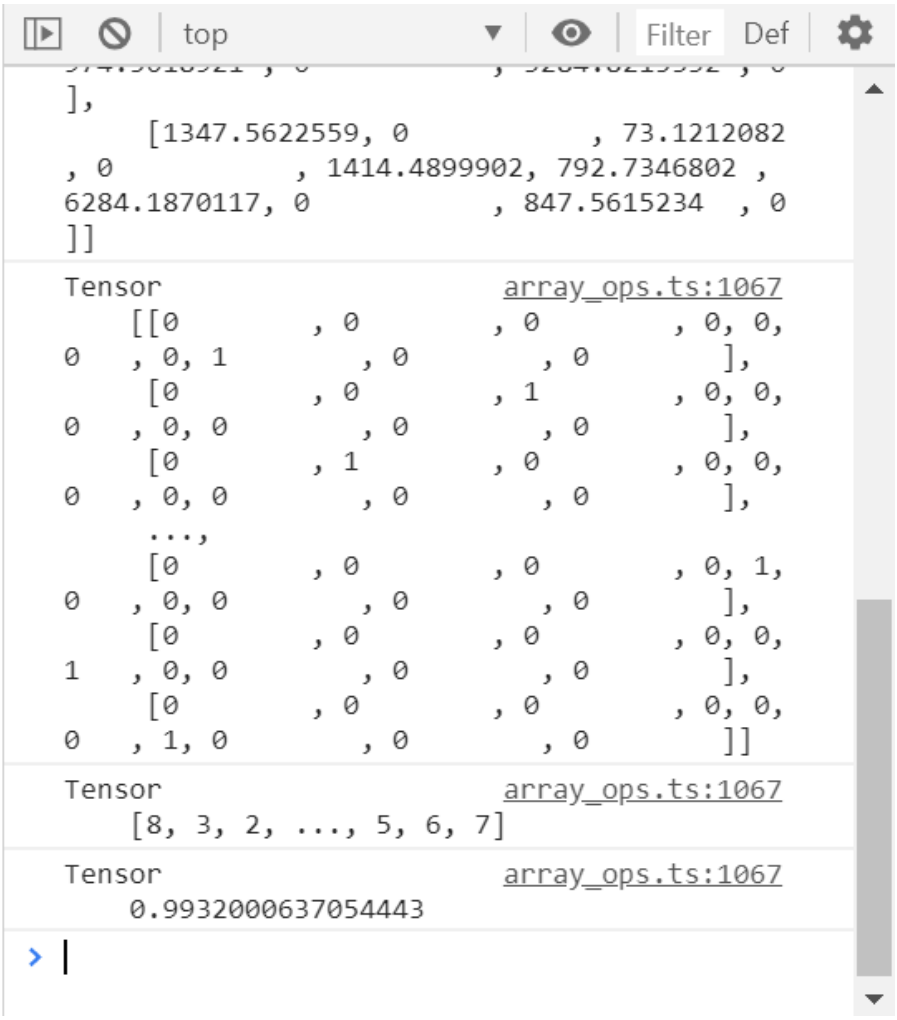
为了更加提高训练的准确度，可以添加 dropout 来进行正则化。

```
model.add(tf.layers.conv2d({
   inputShape: [28, 28, 1],
   kernelSize: 5,
   filters: 16,
   strides: 1,
   activation: 'relu',
   kernelInitializer: 'varianceScaling'
}));
model.add(tf.layers.maxPooling2d({
   poolSize: [2, 2],
   strides: [2, 2]
}));
model.add(tf.layers.conv2d({
   kernelSize: 5,
   filters: 32,
   strides: 1,
   activation: 'relu',
   kernelInitializer: 'varianceScaling'
}));
model.add(tf.layers.maxPooling2d({
   poolSize: [2, 2],
   strides: [2, 2]
}));
model.add(tf.layers.dropout({
   rate:0.5,
}));
model.add(tf.layers.flatten());
model.add(tf.layers.dense({
   units: 128,
   activation: 'relu'
}));
```

```
model.add(tf.layers.dense({
   units: 10,
   activation: 'relu'
}));
```

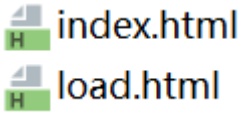
通过修改网络模型，我们将训练的准确率提高到了 99.3%以上，运行截图如下：




top
Filter
Def
],
[1347.5622559, 0 , 73.1212082
, 0 , 1414.4899902, 792.7346802 ,
6284.1870117, 0 , 847.5615234 , 0
]]
Tensor array_ops.ts:1067
[[0 , 0 , 0 , 0, 0,
0 , 0, 1 , 0 , 0],
[0 , 0 , 1 , 0, 0,
0 , 0, 0 , 0 , 0],
[0 , 1 , 0 , 0, 0,
0 , 0, 0 , 0 , 0],
...,
[0 , 0 , 0 , 0, 1,
0 , 0, 0 , 0 , 0],
[0 , 0 , 0 , 0, 0,
1 , 0, 0 , 0 , 0],
[0 , 0 , 0 , 0, 0,
0 , 1, 0 , 0 , 0]]
Tensor array_ops.ts:1067
[8, 3, 2, ..., 5, 6, 7]
Tensor array_ops.ts:1067
0.9932000637054443

3.3 识别结果

3.4 目录结构



首先有两个 html 文件，index.html 是用来进行模型训练保存的，load.html 是用来加载模型然后根据训练的模型进行识别用户的手写数字的。

```
<script src="mnist.js"></script>
<script src="MNIST2.js"></script>
```

Index.html 中导入图上两个 js 文件，一个用来读取数据集一个用来训练模型

```
<script src="mnist.js"></script>
<script src="result.js"></script>
```

load.html 中导入如上图两个文件，除了读取数据集的 js 文件，result 是用来加载训练模型然后进行手写数字识别的。

4. 总结

经过这次实验，尤其是在进行模型的优化实验的时候，深刻的体会到了，CNN 神经网络运用，以及 dropout 的应用，如何运用适合的模型对具体的数据集进行训练一步步提高模型的准确率。