第六部分

实战 TensorFlow 验证码识别

扫描二维码

试看/购买《TensorFlow 快速入门与实战》视频课程

第六部分 目录

准备模型开发环境

第三方依赖包

```
$ pip install Pillow captcha pydot flask
```

数据集生成

- Pillow
- captcha

模型可视化

- pydot

模型服务部署

- flask

Pillow (PIL Fork)

PIL(Python Imaging Library) 为 Python 解释器添加了图像处理功能。但是，在 2009 年发布 1.1.7 版本后，社区便停止更新和维护。

Pillow 是由 Alex Clark 及社区贡献者 一起开发和维护的一款分叉自 PIL 的图像工具库。至今，社区依然非常活跃，Pillow 仍在快速迭代。

Pillow提供广泛的文件格式支持，高效的内部表示和相当强大的图像处理功能。核心图像库旨在快速访问以几种基本像素格式存储的数据， 它应该为一般的图像处理工具提供坚实的基础。

https://github.com/python-pillow/Pillow

captcha

Catpcha 是一个生成图像和音频验证码的开源工具库。

```
from captcha.image import ImageCaptcha
from captcha.audio import AudioCaptcha

image = ImageCaptcha(fonts=['/path/A.ttf', '/path/B.ttf'])
data = image.generate('1234')
image.write('1234', 'out.png')

audio = AudioCaptcha(voicedir='/path/to/voices')
data = audio.generate('1234')
audio.write('1234', 'out.wav')
```

https://github.com/lepture/captcha

pydot

pydot 是用纯 Python 实现的 GraphViz 接口，支持使用 GraphViz 解析和存储 DOT语言（graph description language）。其主要依赖 pyparsing 和 GraphViz 这两个工具库。

pyparsing：仅用于加载DOT文件，在 pydot 安装期间自动安装。

GraphViz：将图形渲染为PDF，PNG，SVG等格式文件，需独立安装。

flask

flask 是一个基于 Werkzeug 和 jinja2 开发的 Python Web 应用程序框架，遵从 BSD 开源协议。它以一种简约的方式实现了框架核心，又保留了扩展性。

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return 'Hello, World!'
```

```
$ env FLASK_APP=hello.py flask run
 * Serving Flask app "hello"
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

https://github.com/pallets/flask

生成验证码数据集

验证码（CAPTCHA）简介

全自动区分计算机和人类的公开图灵测试（英语：Completely Automated Public Turing test to tell Computers and Humans Apart，简称CAPTCHA），俗称验证码，是**一种区分用户是计算机或人的公共全自动程序**。在CAPTCHA测试中，作为服务器的计算机会自动生成一个问题由用户来解答。这个问题可以由计算机生成并评判，但是必须只有人类才能解答。由于计算机无法解答CAPTCHA的问题，所以回答出问题的用户就可以被认为是人类。

一种常用的CAPTCHA测试是**让用户输入一个扭曲变形的图片上所显示的文字或数字**，扭曲变形是为了避免被光学字符识别（OCR, Optical Character Recognition）之类的计算机程序自动识别出图片上的文数字而失去效果。由于这个测试是由计算机来考人类，而不是标准图灵测试中那样由人类来考计算机，人们有时称CAPTCHA是一种**反向图灵测试**。

https://zh.wikipedia.org/wiki/captcha

验证码（CAPTCHA）破解

一些曾经或者正在使用中的验证码系统已被破解。

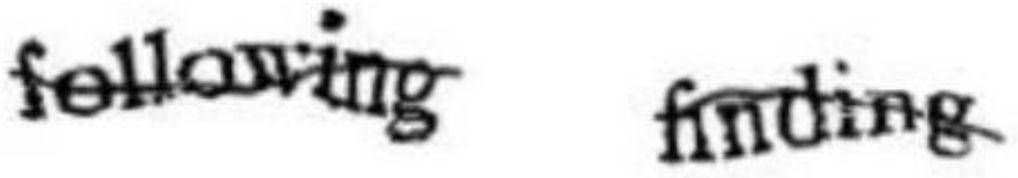
这包括Yahoo验证码的一个早期版本 EZ-Gimpy，PayPal使用的验证码，LiveJournal、phpBB使用的验证码，很多金融机构（主要是银行）使用的网银验证码以及很多其他网站使用的验证码。

俄罗斯的一个黑客组织使用一个自动识别软件在2006年破解了Yahoo的CAPTCHA。**准确率大概是15%，但是攻击者可以每天尝试10万次，相对来说成本很低**。而在2008年，Google的CAPTCHA也被俄罗斯黑客所破解。攻击者使用两台不同的计算机来调整破解进程，可能是用第二台计算机学习第一台对CAPTCHA的破解，或者是对成效进行监视。

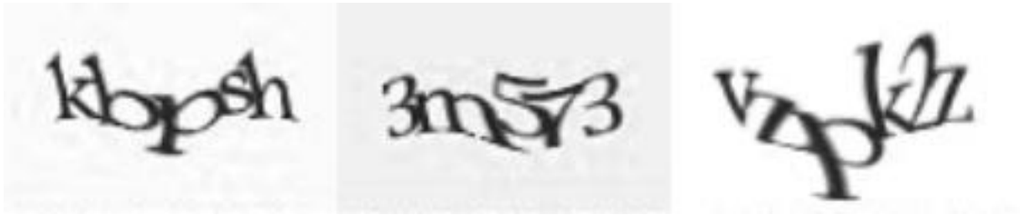
https://zh.wikipedia.org/wiki/captcha

验证码（CAPTCHA）演进

早期的Captcha验证码 "smwm" ，由EZ-Gimpy程序产生，使用扭曲的字母和背景颜色梯度



一种更现代的CAPTCHA，其不使用扭曲的背景及字母，而是增加一条曲线来使得图像分割（segmentation）更困难。



另一种增加图像分割难度的方法为将符号彼此拥挤在一起，但其也使得真人用户比较难以识别

要求用户识别图片的验证方式，本图为模拟12306网站的验证界面

https://zh.wikipedia.org/wiki/captcha

验证码（CAPTCHA）生成

使用 Pillow（PIL Fork） 和 captcha 库生成验证码图像：

`PIL.Image.open(fp, mode='r')` – 打开和识别输入的图像（文件）

`captcha.image.ImageCaptcha(width, height, )` – 创建 `ImageCaptcha` 实例

`captcha.image.ImageCaptcha.write('1234', 'out.png')` – 生成验证码并保存

`captcha.image.ImageCaptcha.generate('1234')` – 生成验证码图像




https://zh.wikipedia.org/wiki/captcha

Try it

输入与输出数据处理

输入数据处理

图像处理：RGB图 -> 灰度图 -> 规范化数据

```
array([[0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.32744314, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
```

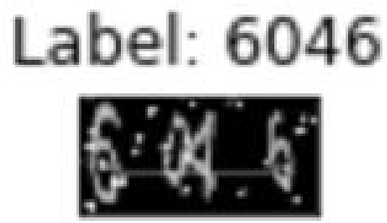
输入数据处理

适配 Keras 图像数据格式：“channels_frist” 或 “channels_last”

```
from keras import backend as K

def fit_keras_channels(batch, rows=CAPTCHA_HEIGHT, cols=CAPTCHA_WIDTH):
    if K.image_data_format() == 'channels_first':
        batch = batch.reshape(batch.shape[0], 1, rows, cols)
        input_shape = (1, rows, cols)
    else:
        batch = batch.reshape(batch.shape[0], rows, cols, 1)
        input_shape = (rows, cols, 1)

    return batch, input_shape
```

输出数据处理

One-hot 编码：验证码转向量



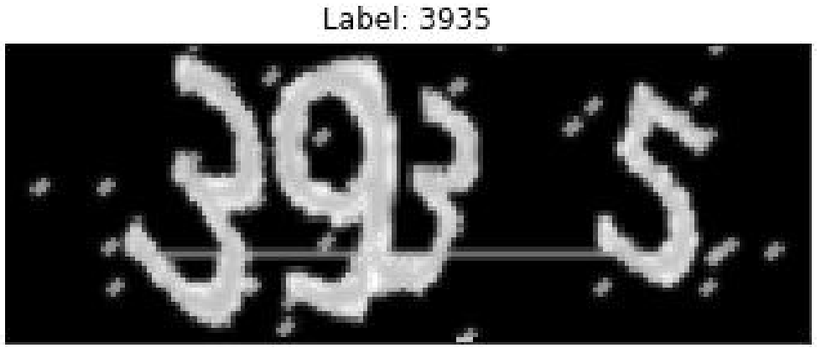
```
array([0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
       1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 1., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 1., 0., 0., 0.])
```

输出数据处理

解码：模型输出向量转验证码

```
array([[2.0792404e-10, 4.3756086e-07, 3.1140310e-10, 9.9823320e-01,
        5.1135743e-15, 3.7417038e-05, 1.0556480e-08, 9.0933657e-13,
        2.7573466e-07, 1.7286760e-03, 1.1030550e-07, 1.1852034e-07,
        7.9457263e-10, 3.4533365e-09, 6.6065012e-14, 2.8996323e-05,
        7.6345885e-13, 3.1817032e-16, 3.9540555e-05, 9.9993122e-01,
        5.3814397e-13, 1.2061575e-10, 1.6408040e-03, 9.9833637e-01,
        6.5149628e-08, 5.2246549e-12, 1.1365444e-08, 9.5700288e-12,
        2.2725430e-05, 5.2195204e-10, 3.2457771e-13, 2.1413280e-07,
        7.3547295e-14, 4.4094882e-06, 3.8390007e-07, 9.9230206e-01,
        6.4467136e-03, 3.9224533e-11, 1.2461344e-03, 1.1253484e-07]],
      dtype=np.float32)
```

argmax

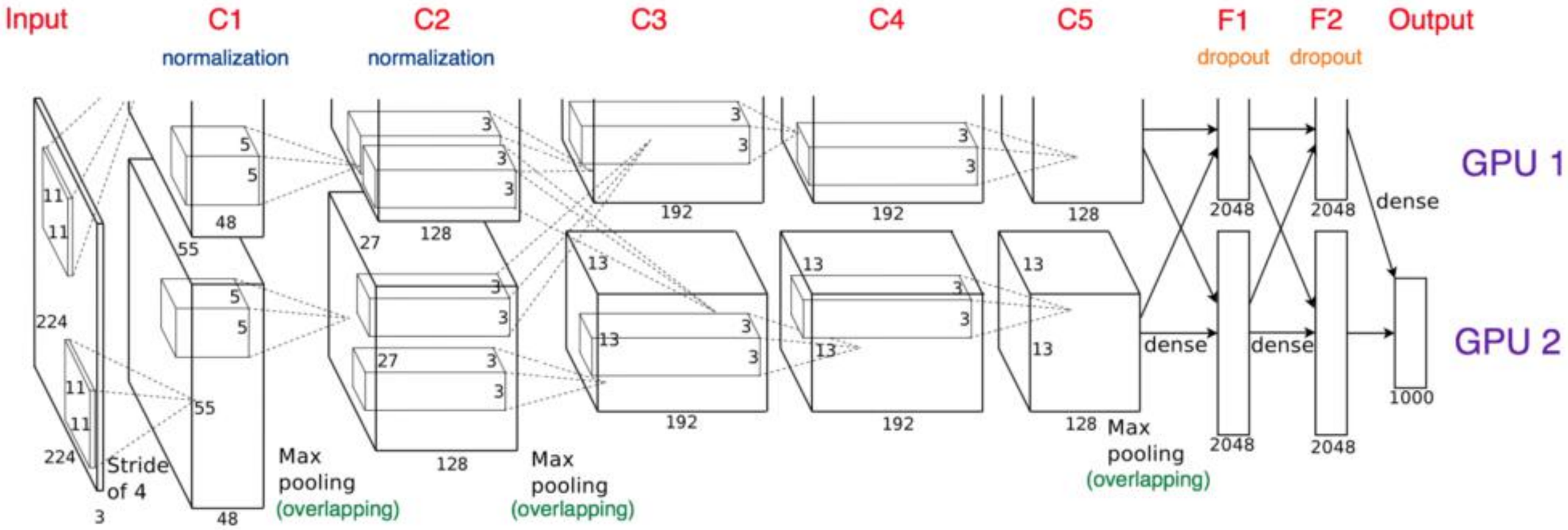


Try it

模型结构设计

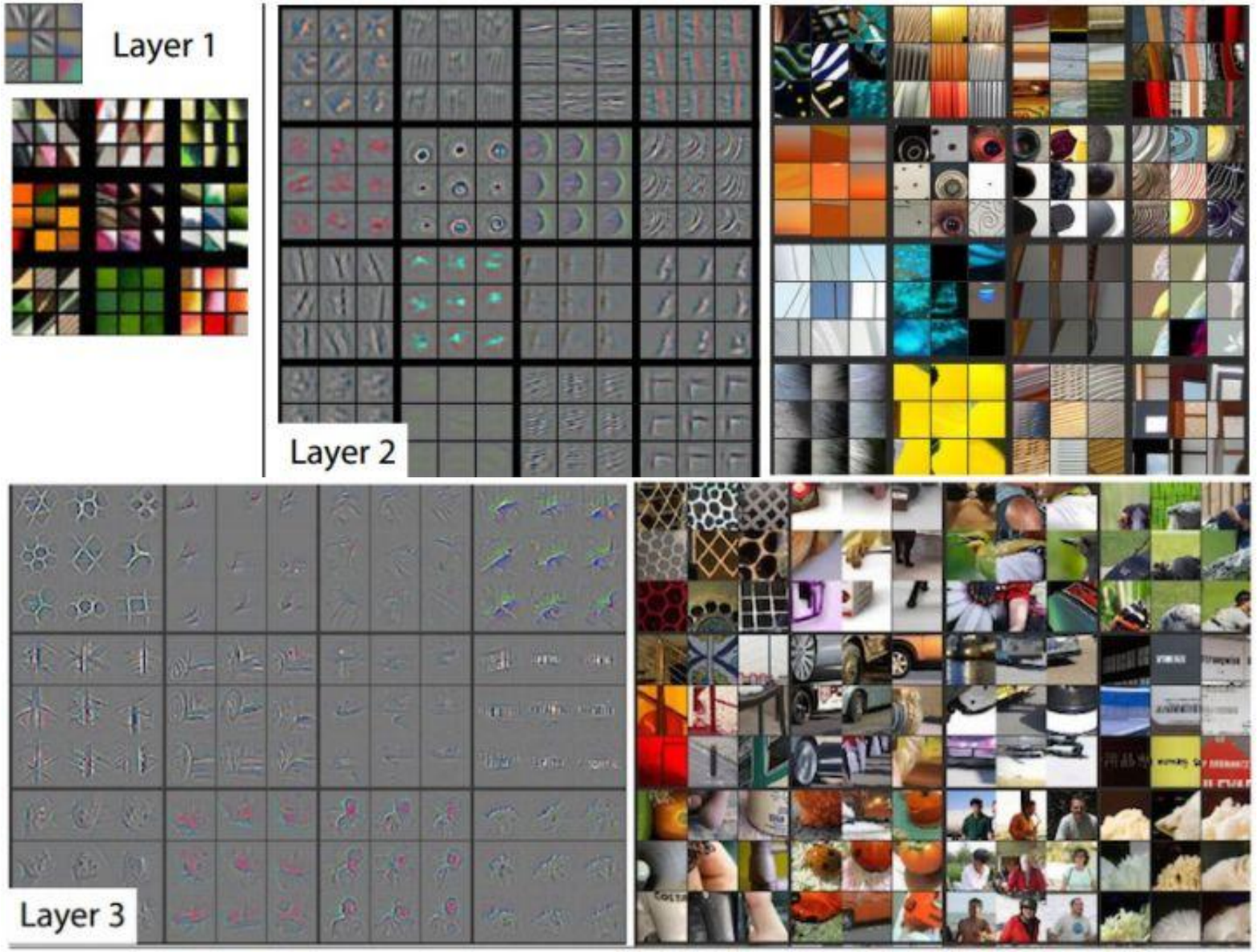
分类问题

图像分类模型 AlexNet

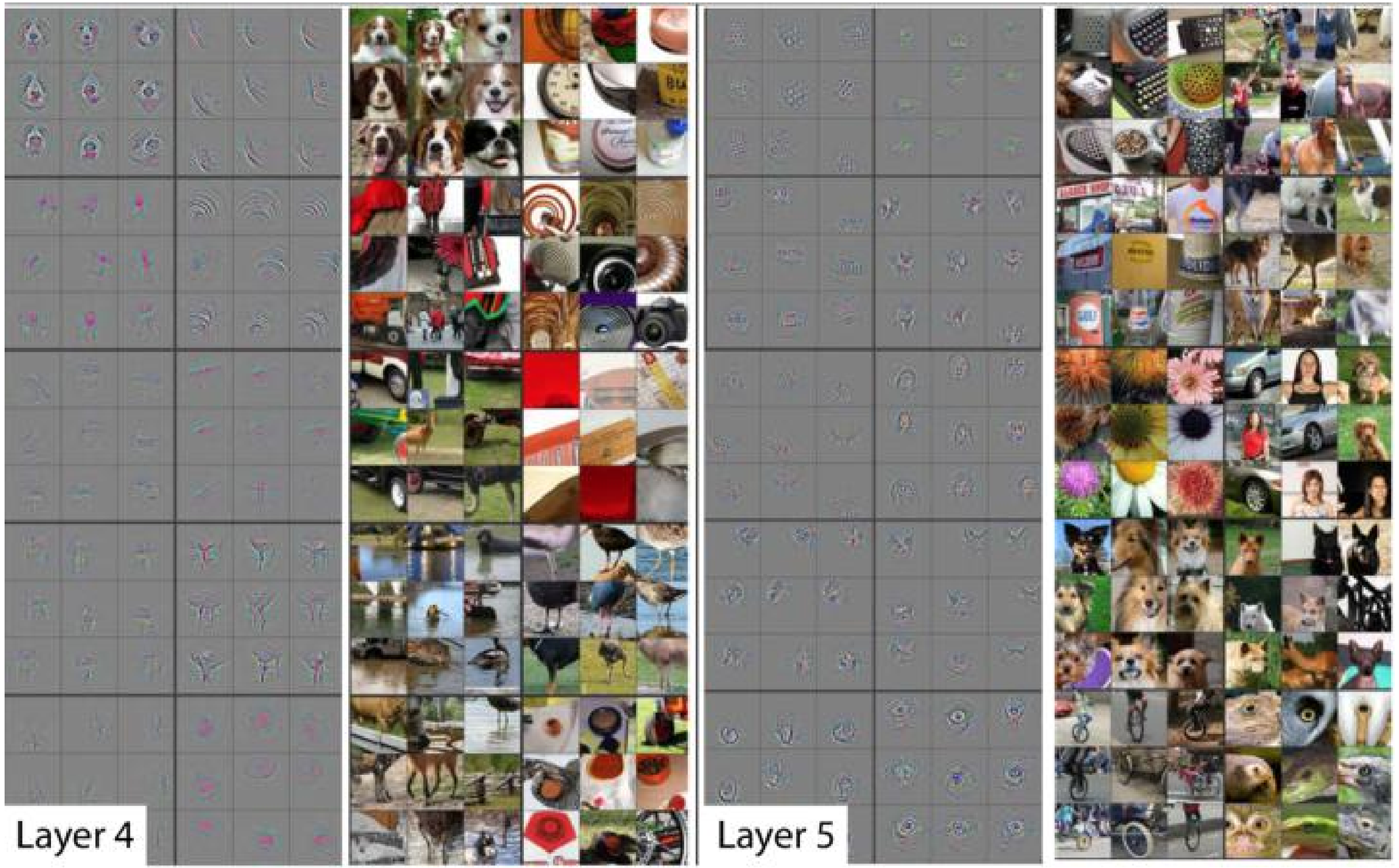


Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classi cation with deep convolutional neural networks. In: NIPS (2012)

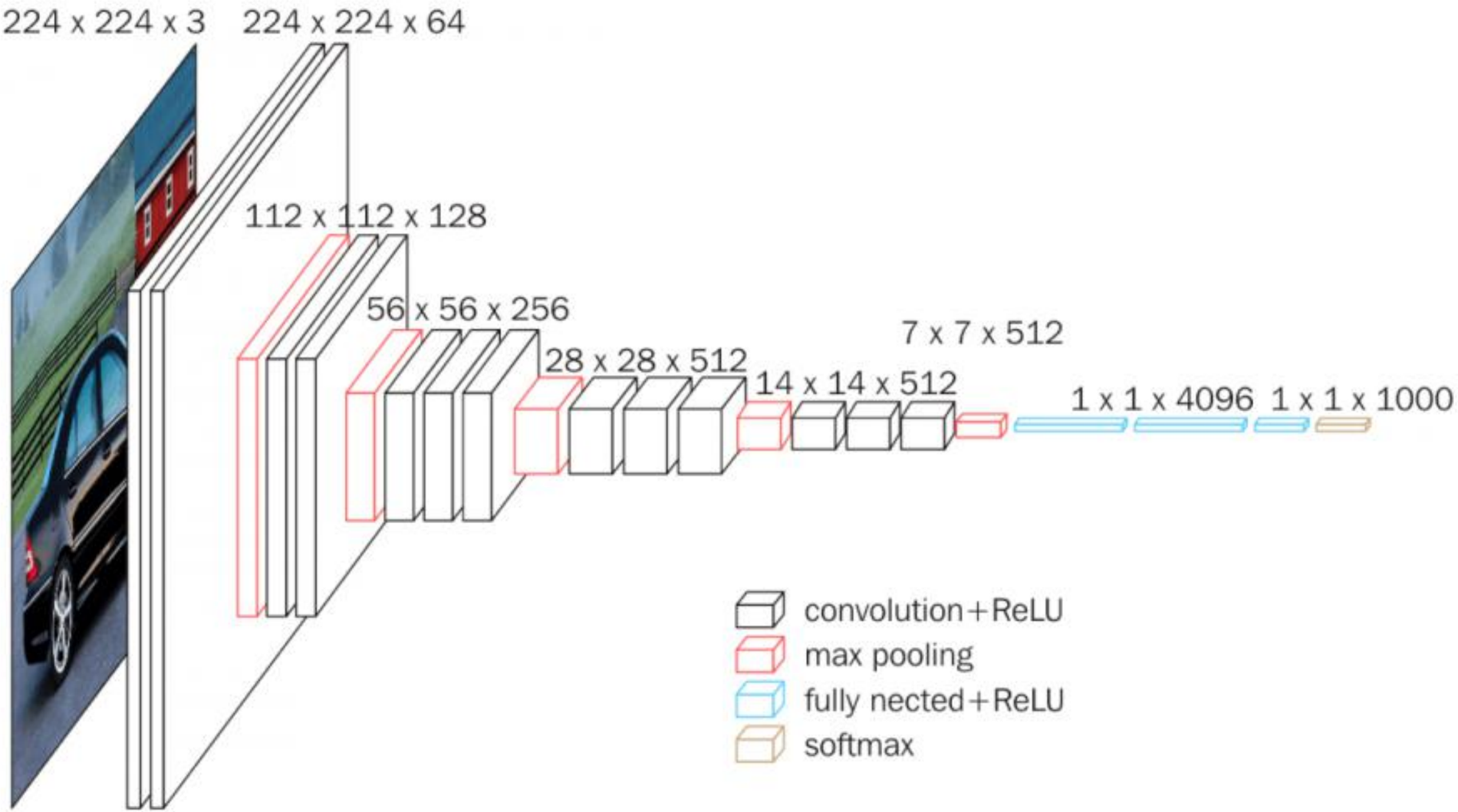
使用卷积进行特征提取



使用卷积进行特征提取

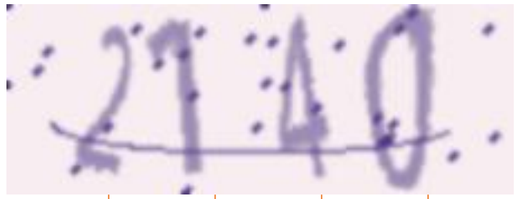


图像分类模型 VGG-16



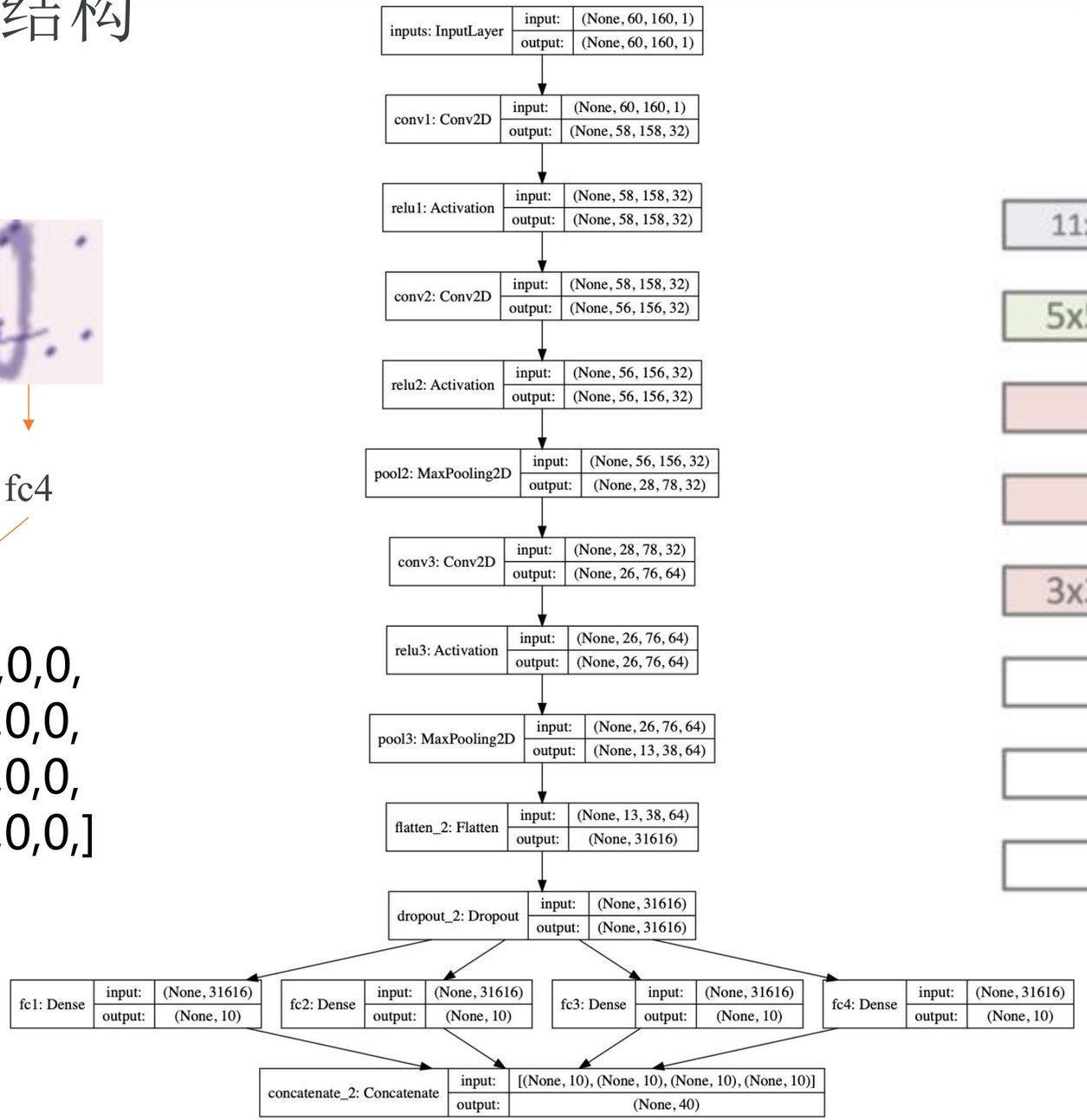
Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

验证码识别模型结构

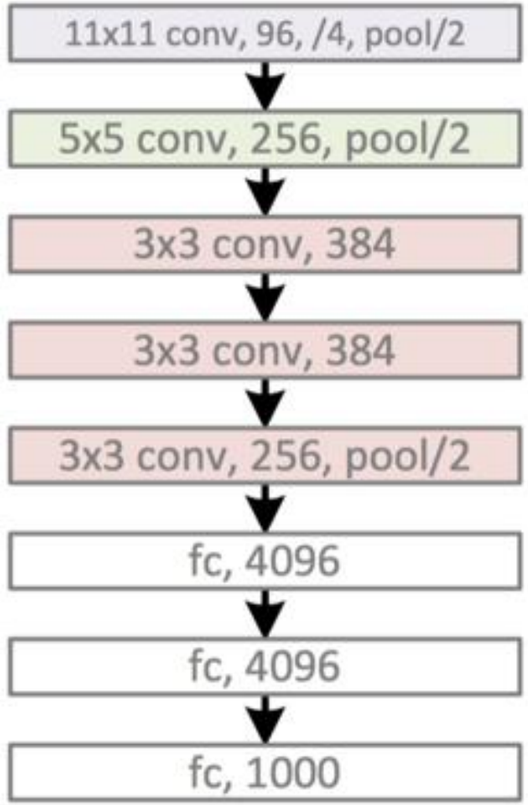


fc1 fc2 fc3 fc4

[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,
1,0,0,0,0,0,0,0,0,0,]



AlexNet



验证码识别模型实现

```
# 输入层
inputs = Input(shape = input_shape, name = "inputs")

# 第1层卷积
conv1 = Conv2D(32, (3, 3), name = "conv1")(inputs)
relu1 = Activation('relu', name="relu1")(conv1)

# 第2层卷积
conv2 = Conv2D(32, (3, 3), name = "conv2")(relu1)
relu2 = Activation('relu', name="relu2")(conv2)
pool2 = MaxPooling2D(pool_size=(2,2), padding='same', name="pool2")(relu2)

# 第3层卷积
conv3 = Conv2D(64, (3, 3), name = "conv3")(pool2)
relu3 = Activation('relu', name="relu3")(conv3)
pool3 = MaxPooling2D(pool_size=(2,2), padding='same', name="pool3")(relu3)

# 将 Pooled feature map 摊平后输入全连接网络
x = Flatten()(pool3)

# Dropout
x = Dropout(0.25)(x)

# 4个全连接层分别做10分类，分别对应4个字符。
x = [Dense(10, activation='softmax', name='fc%d'%(i+1))(x) for i in range(4)]

# 4个字符向量拼接在一起，与标签向量形式一致，作为模型输出。
outs = Concatenate()(x)

# 定义模型的输入与输出
model = Model(inputs=inputs, outputs=outs)
model.compile(optimizer=OPT, loss=LOSS, metrics=['accuracy'])
```

模型损失函数设计

交叉熵（Cross-Entropy, CE）

我们使用交叉熵作为该模型的损失函数。

虽然 Categorical / Binary CE 是更常用的损失函数，不过他们都是 CE 的变体。

CE 定义如下：

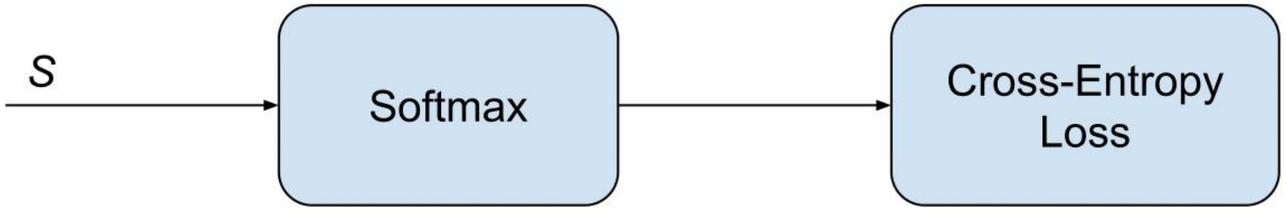
$$CE = -\sum_{i}^{C} t_i log(s_i)$$

对于二分类问题 (C‘=2)，CE 定义如下：

$$CE = -\sum_{i=1}^{C'=2} t_i log(s_i) = -t_1 log(s_1) + (1 - t_1) log(1 - s_1)$$

Categorical CE Loss（Softmax Loss）

常用于输出为 One-hot 向量的多类别分类（Multi-Class Classification）模型。



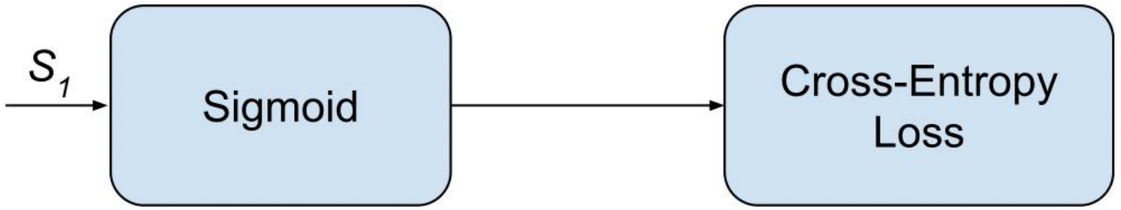
$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

$$CE = -log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right)$$

Binary CE Loss（Sigmoid CE Loss）

与 Softmax Loss 不同，Binary CE Loss 对于每个向量分量（class）都是独立的，这意味着每个向量分量计算的**损失不受其他分量的影响**。

因此，它常被用于多标签分类（Multi-label classification）模型。



$$f(s_i) = \frac{1}{1 + e^{s_i}}$$

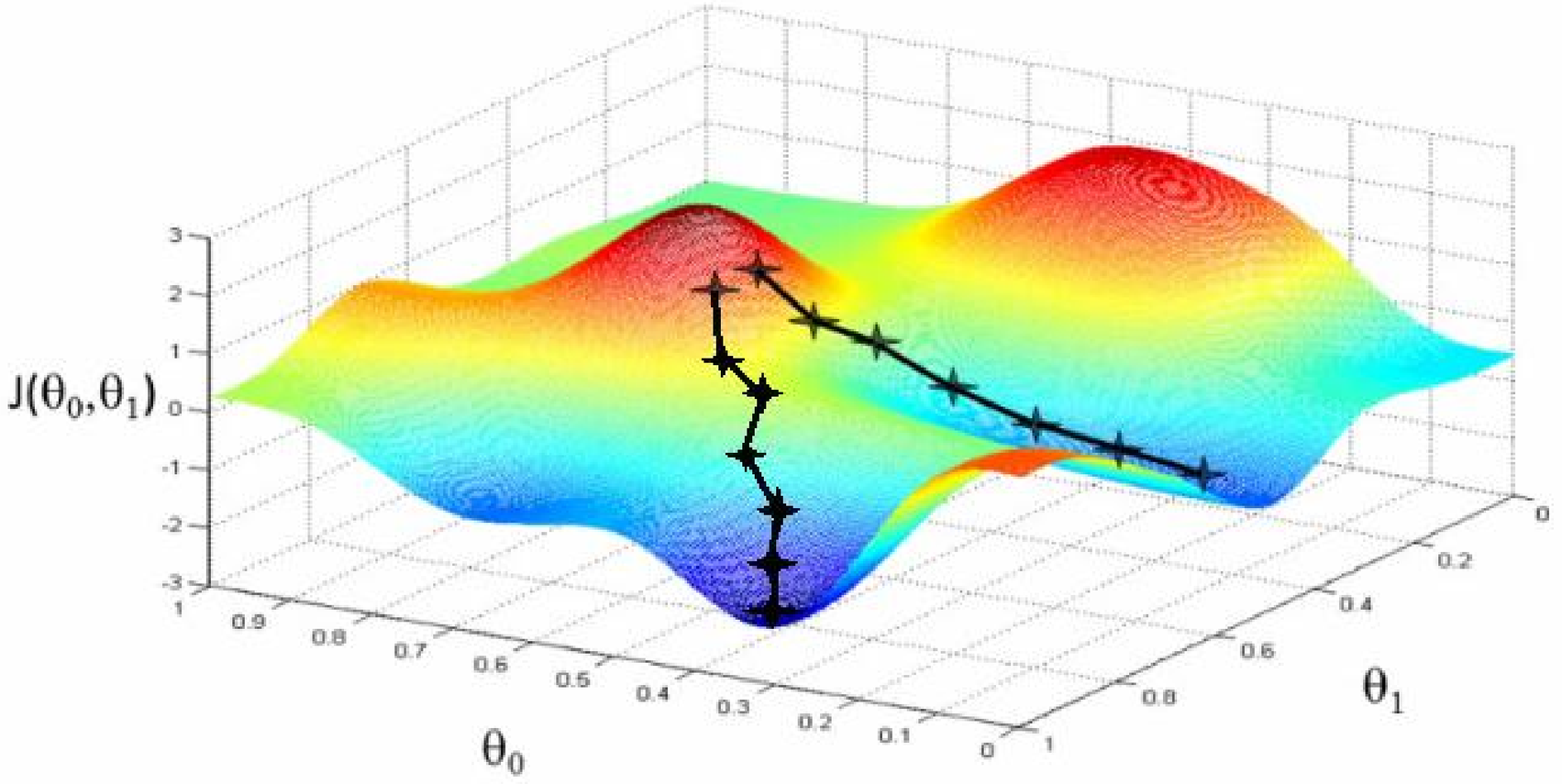
$$CE = -t_1 log(f(s_1)) + (1 - t_1) log(1 - f(s_1))$$

$$CE = \begin{cases} -log(s_1) & if \quad t_1 = 1 \\ -log(1 - s_1) & if \quad t_1 = 0 \end{cases}$$

Try it

模型训练过程分析

模型训练过程



学习率（Learning rate）

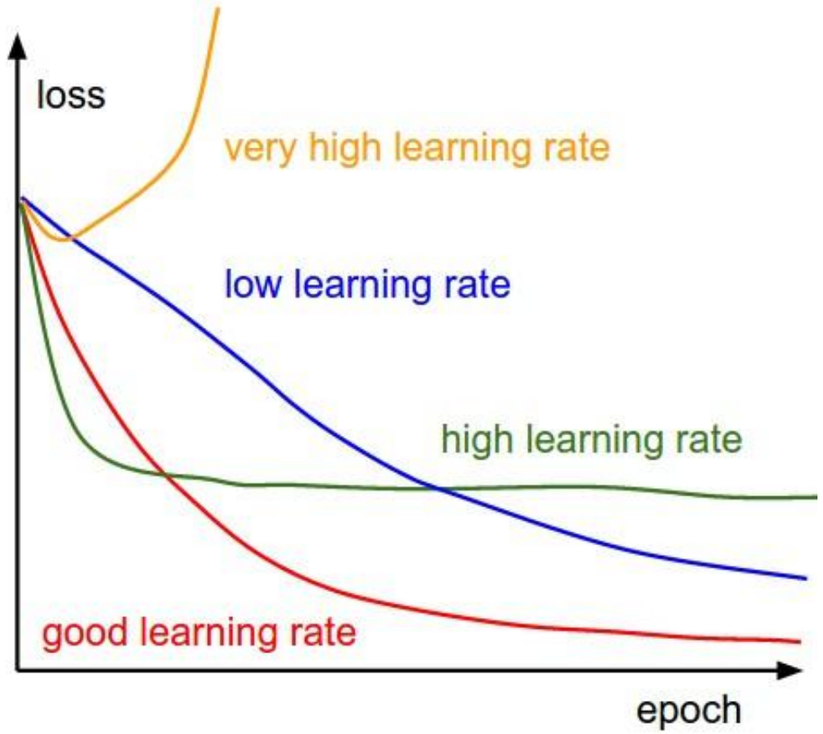
学习率与损失值变化（模型收敛速度）直接相关。

何时加大学习率

- 训练初期，损失值一直没什么波动

何时减小学习率

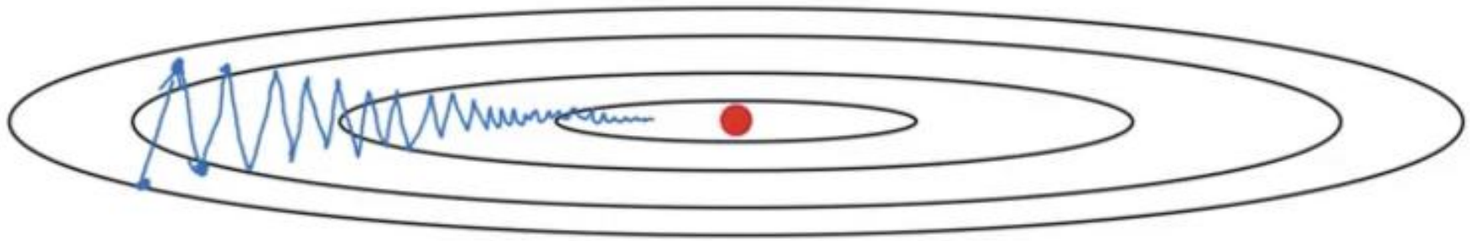
- 训练初期，损失值直接爆炸或者 NAN
- 损失值先开始速降，后平稳多时
- 训练后期，损失值反复上下波动



优化器介绍：SGD（Stochastic Gradient Descent）

$$g_t = \nabla_\theta J(\theta)$$

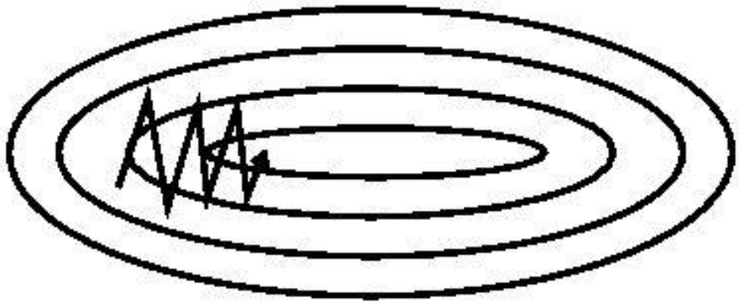
$$\theta_{i+1} = \theta_t - \eta g_t$$



优化器介绍：SGD-M（Momentum）

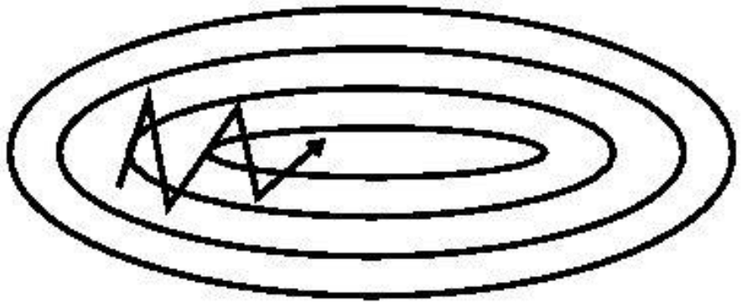
SGD 在遇到沟壑时容易陷入震荡。为此，可以为其引入动量（Momentum），加速 SGD 在正确方向的下降并抑制震荡。

$$m_t = \eta g_t$$



SGD

$$m_t = \gamma m_{t-1} + \eta g_t$$



SGD with Momentum

优化器介绍：Adagrad – RMSprop – Adam

$$v_t = \mathrm{diag}(\sum_{i=1}^{t} g_{i,1}^2, \sum_{i=1}^{t} g_{i,2}^2, \cdots, \sum_{i=1}^{t} g_{i,d}^2)$$

Adagrad

（引入二阶动量）

$$v_t = \gamma v_{t-1} + (1-\gamma) \cdot \mathrm{diag}(g_t^2)$$

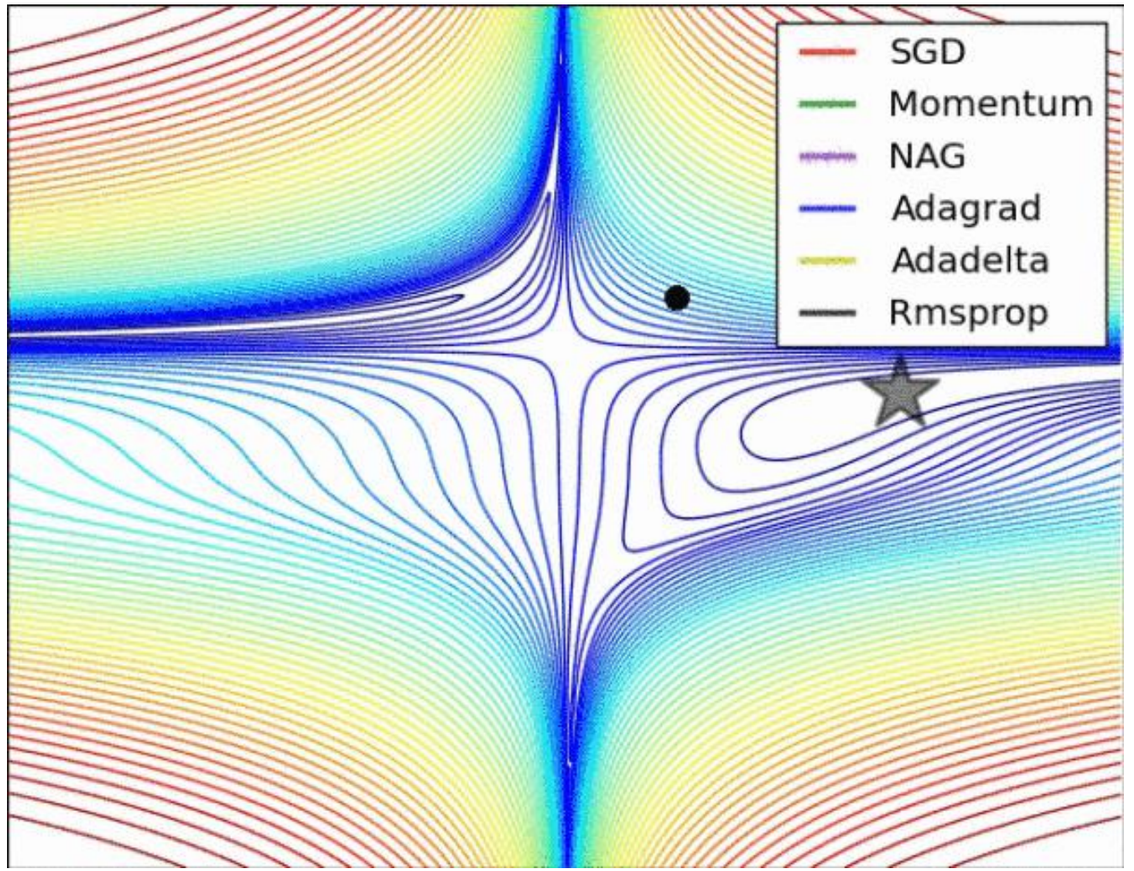
RMSprop

（二阶动量 指数移动平均）

$$m_t = \eta[\beta_1 m_{t-1} + (1-\beta_1) g_t]$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) \cdot \mathrm{diag}(g_t^2)$$

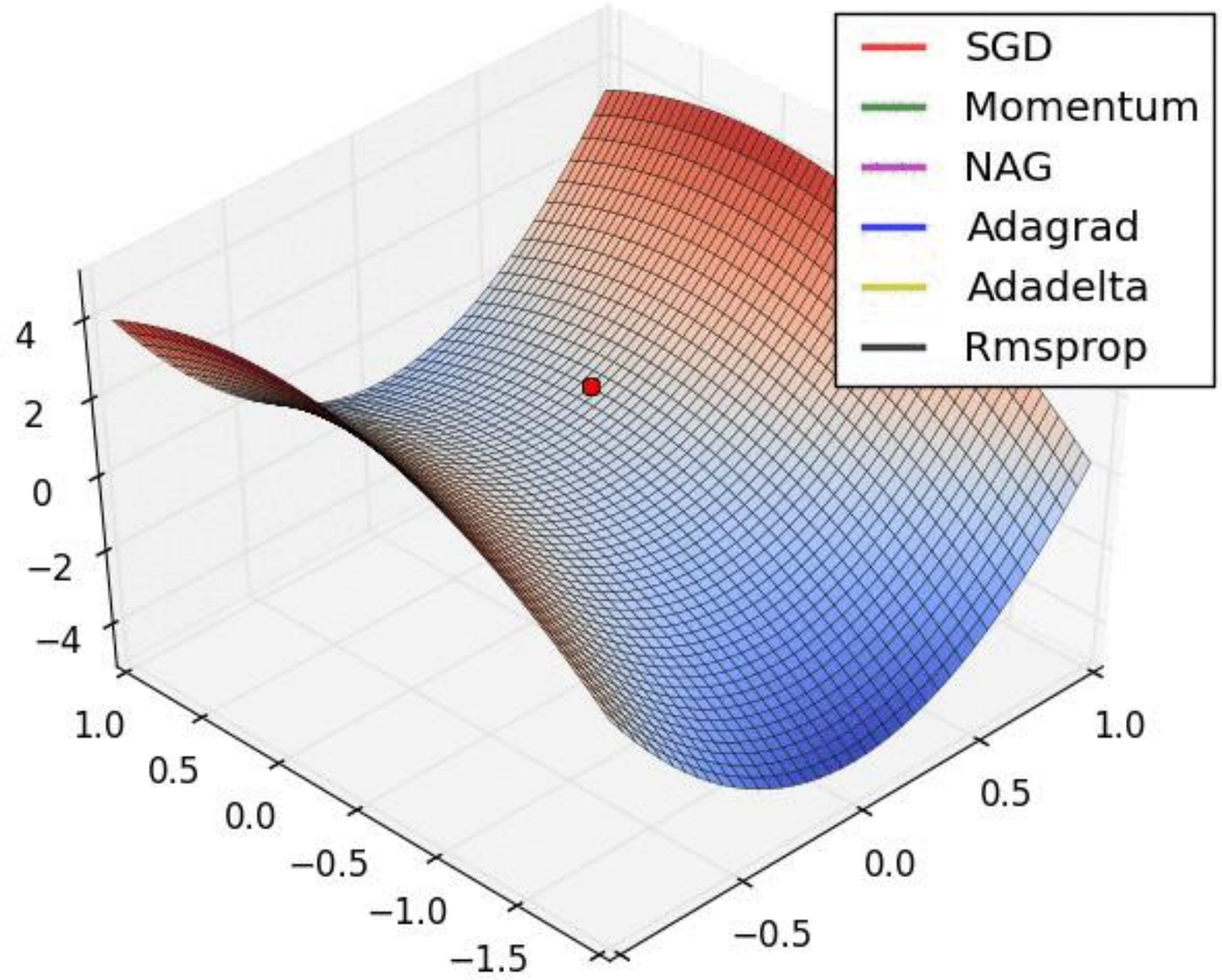
Adam

（一/二阶动量 指数移动平均）

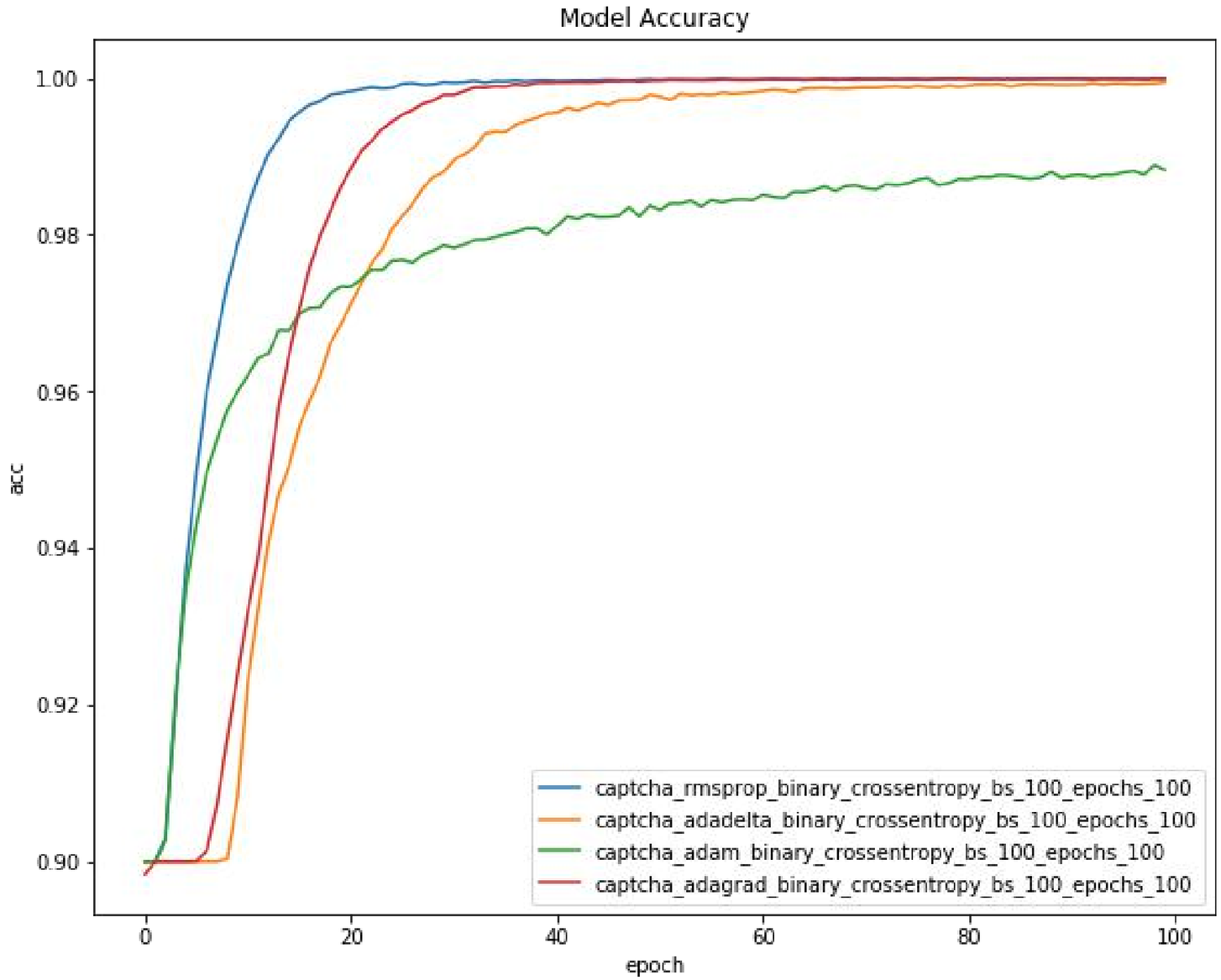


优化器对比：损失面等高线图

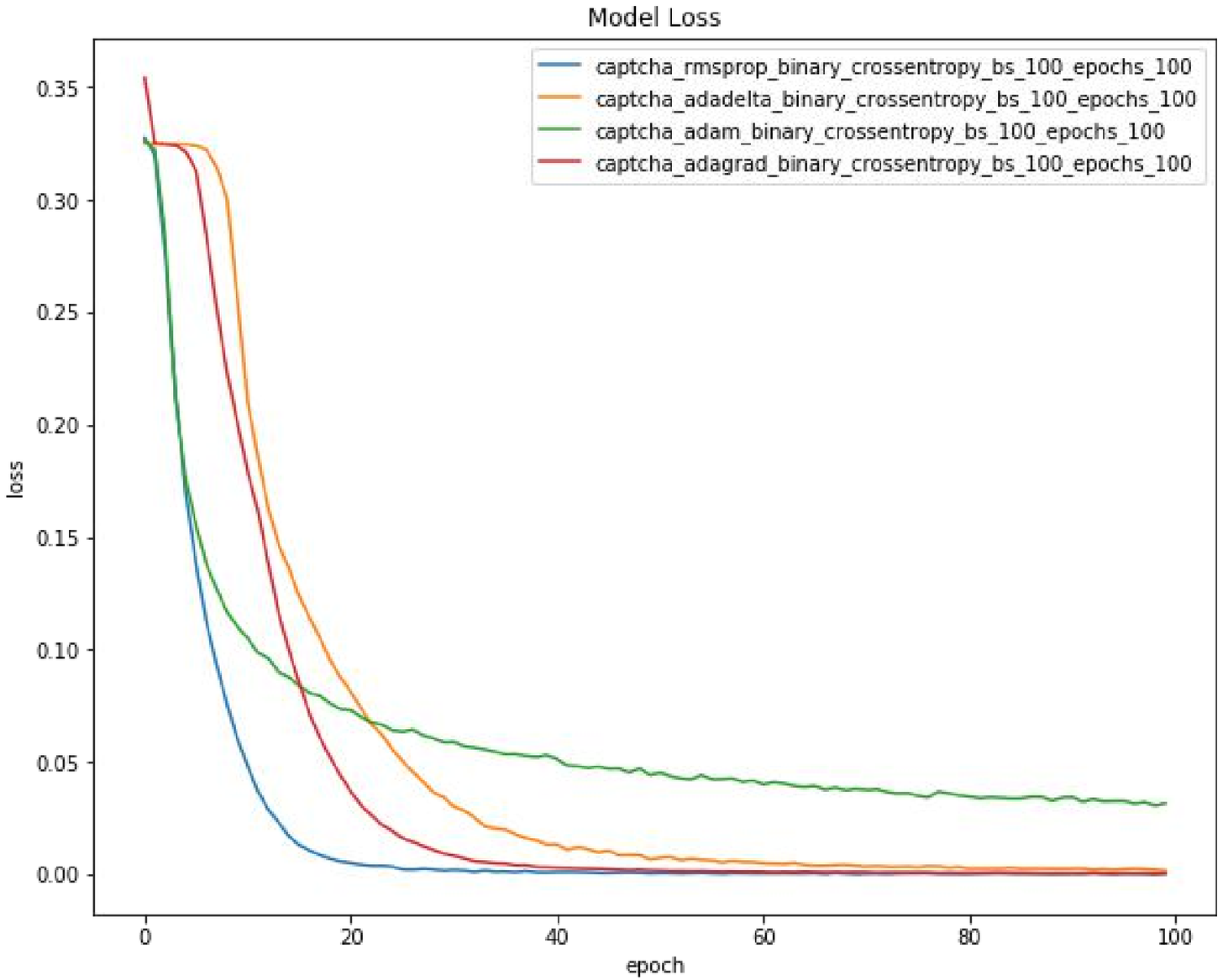
优化器对比：鞍点


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5

优化器对比： 验证码识别模型



优化器对比： 验证码识别模型



Try it

模型部署与效果演示

数据-模型-服务流水线

使用 Flask 快速搭建 验证码识别服务

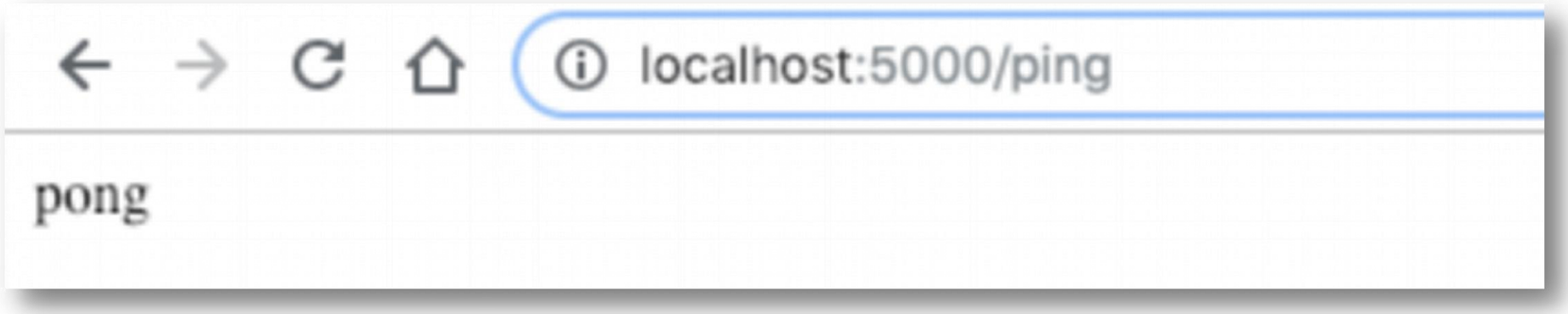
```
app = Flask(__name__) # 创建 Flask 实例

# 测试 URL
@app.route('/ping', methods=['GET', 'POST'])
def hello_world():
    return 'pong'

# 验证码识别 URL
@app.route('/predict', methods=['POST'])
def predict(model=model, graph=graph):
    response = {'success': False, 'prediction': '', 'debug': 'error'}
    received_image= False
    if request.method == 'POST':
        if request.files.get('image'): # 图像文件
            image = request.files['image'].read()
            received_image = True
            response['debug'] = 'get image'
        elif request.get_json(): # base64 编码的图像文件
            encoded_image = request.get_json()['image']
            image = base64.b64decode(encoded_image)
            received_image = True
            response['debug'] = 'get json'
        if received_image:
            image = np.array(Image.open(BytesIO(image)))
            image = rgb2gray(image).reshape(1, 60, 160, 1).astype('float32') / 255
            with graph.as_default():
                pred = model.predict(image)
            response['prediction'] = response['prediction'] + vec2text(pred)
            response['success'] = True
            response['debug'] = 'predicted'
    else:
        response['debug'] = 'No Post'
    return jsonify(response)
```

使用 Flask 启动 验证码识别服务

```
$ export FLASK_ENV=development && flask run --host=0.0.0.0
```

```
(py3) Django:~/tf-course/notebook-examples/chapter-6  $ export FLASK_ENV=development && flask run --host=0.0.0.0
 * Environment: development
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 217-274-263
Using TensorFlow backend.
2019-02-22 00:29:38.128433: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that th
Using TensorFlow backend.
2019-02-22 00:29:38.714408: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that th
```

打开浏览器访问测试 URL（http://localhost:5000/ping）



访问 验证码识别服务

```
$ curl -X POST -F image=@2140.png 'http://localhost:5000/predict'
```

```
Django:~  $ curl -X POST -F image=@2140.png 'http://localhost:5000/predict'
{
  "debug": "predicted",
  "prediction": "2140",
  "success": true
}
Django:~  $ curl -X POST -F image=@1459.png 'http://localhost:5000/predict'
{
  "debug": "predicted",
  "prediction": "1459",
  "success": true
}
Django:~  $ curl -X POST -F image=@6598.png 'http://localhost:5000/predict'
{
  "debug": "predicted",
  "prediction": "8598",
  "success": true
}
```

2140.png

1459.png

6598.png

Try it

扫描二维码

试看/购买《TensorFlow 快速入门与实战》视频课程