人机交互期末报告

姓名：赵阳　学号：2020632037　班级：周三 12

*所有代码在 drawboard 压缩文件夹中。同时已上传至 github：

https://github.com/Achenganggyel/Man-Machine-Pro

1　web 界面和移动界面设计的基本原则

1.1　web 界面设计基本原则-3C

web 页面遵循三条原则：简洁性、一致性和对比性。

其中，**简洁性**，意为设计不展现具体的物像和特征，而是意图和要求。更具体的说，建议使用简洁、醒目的图形；限制字体和所用颜色的数量；要求页面所有元素具有明确的含义和用途。

一致性，涵盖了页边距、文本间间距等留白形式的统一、图标风格的统一、色彩和风格的和谐等。通常，一个站点只使用一两种标准色。

对比性则应用于强调突出某些内容的情况下，它使内容容易便任何接受。比如，内容提要是蓝色而正文黑色，或者大标题的出现。

此外，为了更好的用户体验，有人将 web 端设计原则扩展如下。

- 以用户为中心：既考虑共性，也考虑差异性
- 一致性：内容和形式的一致性、风格的一致性
- 简洁与明确：网页层次化、显示字体和颜色的数目、减少浏览层次
- 体现特色
- 兼顾不同浏览器
- 明确的导航设计

1.2　移动界面设计基本原则

移动端界面设计需要在迅速响应的基础上，对页面的清晰度、美观度进行把控，尽可能使界面风格趋于一致，尊重用户使用习惯，不断优化界面布局，提升用户体验感。此处列举以下五点原则。

一、响应迅速原则

移动端界面的响应速度不仅仅是开发人员要考虑的问题，也是界面设计人员要考虑的，因为开发人员的操作是以设计人员制定的总体方案进行的。

二、清晰性原则

简洁大方的界面风格，条理清晰，主题明确，功能具有较强的指向性，用户使用起来一目了然，可以为用户提供更好的体验感。

三、尊重用户习惯原则

设计师在进行移动端界面设计的时候不要一味地标新立异，因为有些界面的功能是用户已经使用习惯了的，一旦改变，用户可能会对界面不适应，不利于良好的用户体验。

四、界面风格一致原则

用户通过手机端的移动界面可以看到更多电脑端看不到的细节，如果手机端界面设计的风格和框架不够统一，很容易给用户一种混乱的感觉。

五、美观度原则

爱美之心，人皆有之。自古以来，人们都在追求美观。

2 画板页面的设计

考虑到适配性和兼容性等问题，仅开发 web 端的画板页面。

2.1 概念设计

本页面主要分为 3 大部分：文件及设置区，功能区和画布区。

文件及设置区涵盖传统的画板的文件相关配置，包括保存、撤销等。

功能区则为直线、输入文字、绘画形状等，根据初步设定，途中从上至下分别为直线、箭头、虚线、输入文字、矩形、圆形工具。

而画布区是绘画时的区域。

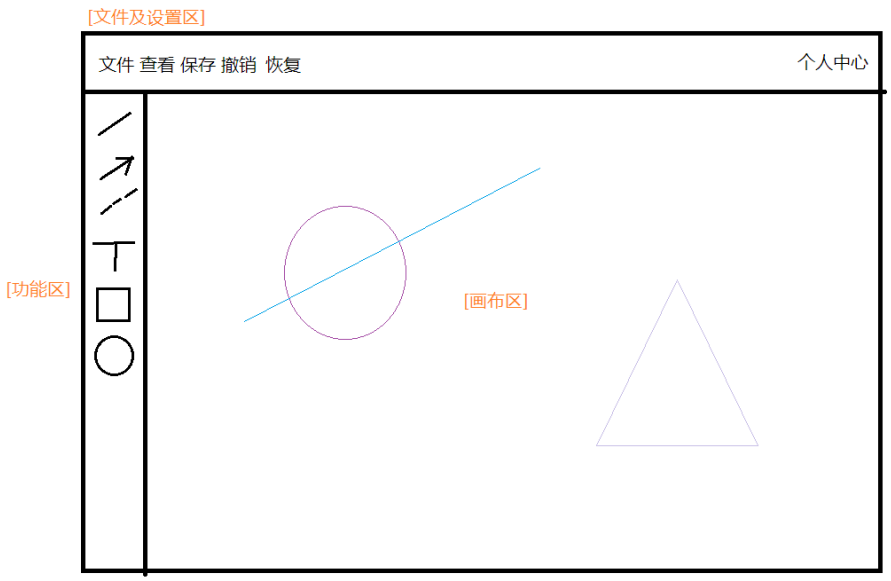


图 1-画板页面布局

其中，功能区和画布区的预期实现参考下图。

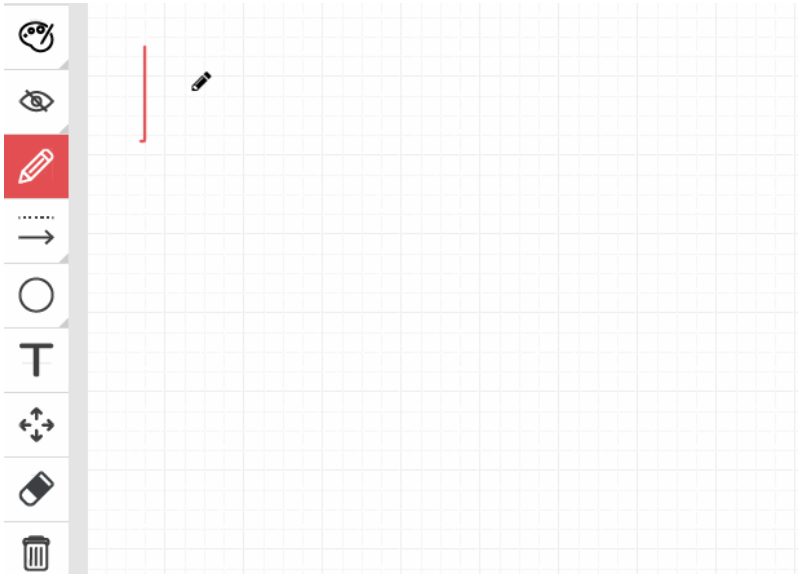


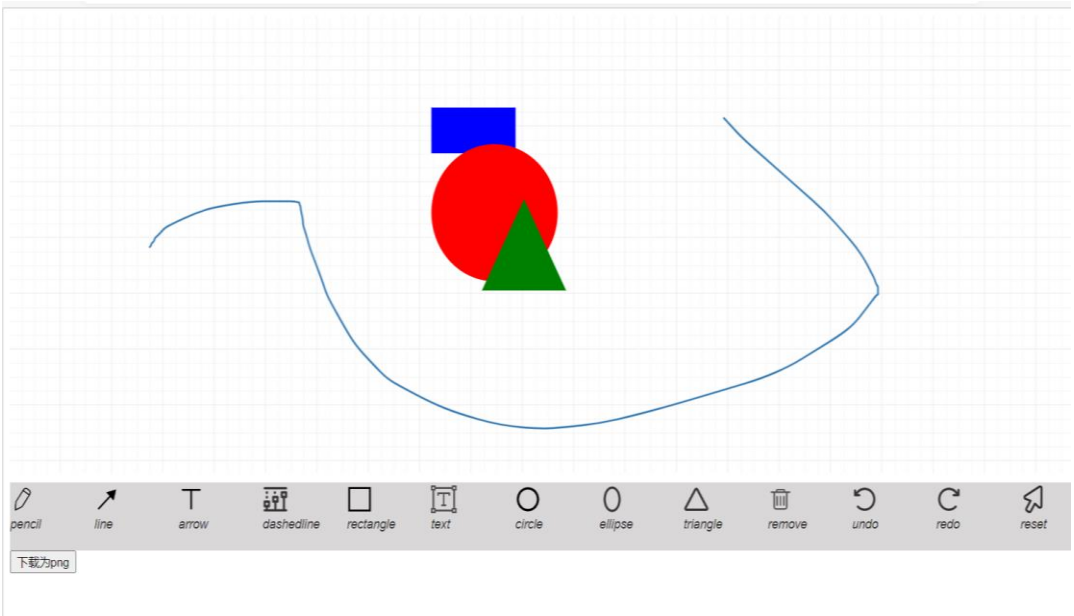
图 2-画板功能和画布区的预期实现

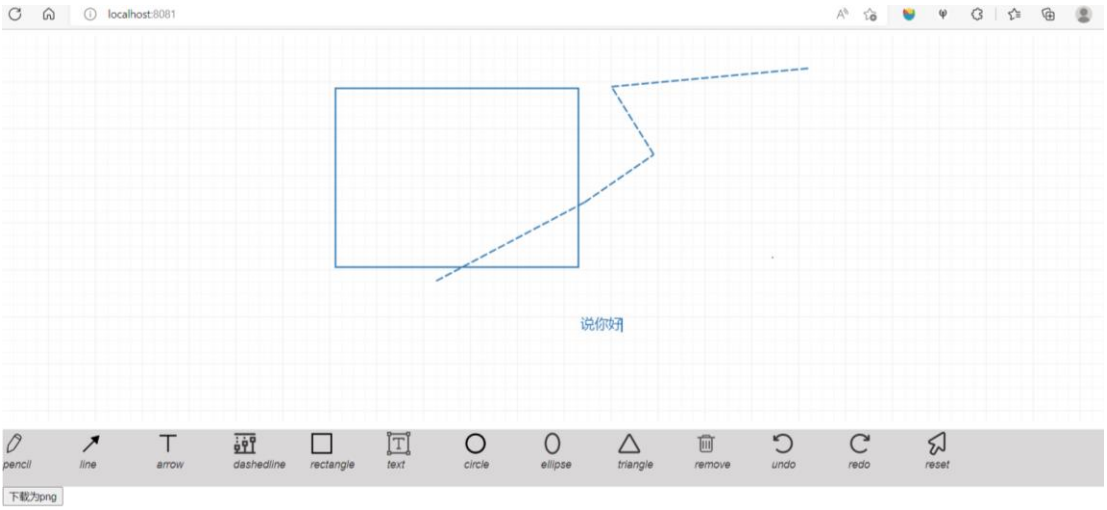
2.2 具体实现 vue3+fabric.js

使用 fabric.js（封装 canvas 的 js 库）实现画板功能。考虑到功能较少，没有必要使用 element-ui，所以将 sidebar 和 header 的内容整合到下方长条中。

最终画板支持自由绘制、直线、箭头、虚线、长方形、文字、圆形、椭圆、三角形此 9 种绘图选项及删除、撤销、重做、清空 4 种功能。

此处展示 2 张效果截屏。第 1 张是曲线自由绘制/手绘和初始的三个图形，第 2 张是矩形、虚线和文字显示。





3 手写数字识别

3.1 MVC 原则

在人机交互领域中，MVC 原则指通过 model，view，controller 三个方面描述应用，并通过三者的交互使功能得以运转。其出现不仅实现了功能模块和显示模块的分离，同时它还提高了应用系统的可维护性、可扩展性、可移植性和组件的可复用性。

model：模型是应用程序的主体部分。模型表示业务数据，或者业务逻辑。包括算法模型

view：应用程序中用户界面相关的部分，是用户看到并与之交互的界面

controller：根据用户的输入，控制用户界面数据显示和更新 model 对象状态

3.2 框架设计

MVC 结构	文件和功能
Model	./model/model.py 神经网络设计
View	./templates/index.html 页面设计 ./static/index.js 事件监听函数， ./static/style.css 样式定义
Controller	./app.py 路由和输入获取的控制 ./elevate.py 将预测的图片保存在./static/output.png；./train.py 使用 mnist 数据集训练网络，如果不存在模型，则调用 model 文件夹下的网络 model.py 新建模型并将模型保存为 model/mnist.pth

3.3 flask(BP network) 实现

3.3.1 model 模型构建

Model 类的定义包含 7 个层级（如下图，model/model/.py），首先是二维卷积及 ReLu 激活函数，然后是最大池化层，之后在新的维度上重复前两个过程，最后是三个 Linear 及激活函数，输出 10 种数字的分别预测概率。同时，训练神经网络（train.py）时采用误差反向传播来提升模型的准确度，且在每轮训练时将梯度置零，以避免梯度的混合积累。

```
from torch.nn import Module
from torch import nn

class Model(Module):

    def __init__(self):
        super(Model, self).__init__()
        self.conv1 = nn.Conv2d(1, 6, 5)
        self.relu1 = nn.ReLU()
        self.pool1 = nn.MaxPool2d(2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.relu2 = nn.ReLU()
        self.pool2 = nn.MaxPool2d(2)
        self.fc1 = nn.Linear(256, 120)
        self.relu3 = nn.ReLU()
        self.fc2 = nn.Linear(120, 84)
        self.relu4 = nn.ReLU()
        self.fc3 = nn.Linear(84, 10)
        self.relu5 = nn.ReLU()

    def forward(self, x):
        y = self.conv1(x)
        y = self.relu1(y)
        y = self.pool1(y)
        y = self.conv2(y)
        y = self.relu2(y)
        y = self.pool2(y)
        y = y.view(y.shape[0], -1)
        y = self.fc1(y)
        y = self.relu3(y)
        y = self.fc2(y)
        y = self.relu4(y)
        y = self.fc3(y)
        y = self.relu5(y)
        return y
```

3.3.2 view 交互效果

本模块建立在 html5 提供的 canvas 画布上。首先在黑色矩形中绘制手写数字

点击预测后出现结果（下左），点击清除后清空绘图区域（下右）

3.3.3 controller 代码一览

controller 使用了 flask 框架，整合 model 和 view 模块。当项目运行时，app.py，elevate.py 和 train.py 都会启动。首先渲染 templates/index.html 页面并开启 static/index.js 的静态网页监听事件，按下预测按钮后将 canvas 图片保存为 static/output.png，并通过 POST 类型的接口交换信息，等待 app.py 的/predict/相关函数分析结果返回给 templates/index.html 使其输出结果。下图是 app.py 的部分代码，包含路由设定和页面交互，此处展示的是获取手写字体后的处理。

```
@app.route('/predict/', methods=['Get', 'POST'])
def preditc():
    global net
    parseImage(request.get_data())
    '''预测'''
    data_transform = transforms.Compose([transforms.ToTensor(), ])
    root = 'static/output.png'
    img = Image.open(root)
    img = img.resize((28,28))
    img = img.convert('L')
    img = data_transform(img)
    img = torch.unsqueeze(img, dim=0)  # 输入要与model对应
    predict_y = net(img.float()).detach()
    predict_ys = np.argmax(predict_y, axis=-1)
    ans = predict_ys.item()
    print(predict_y)
    print(predict_y.numpy().squeeze()[ans])
    return jsonify(ans)


def get_visit_info(code=0):
    response = {}
    response['code'] = code
    return response


def parseImage(imgData):
    imgStr = re.search(b'base64,(.*)', imgData).group(1)
    with open('./static/output.png', 'wb') as output:
        output.write(base64.decodebytes(imgStr))


if __name__ == '__main__':
```

这里展示部分静态网页的代码，包含画布中鼠标拖拽、鼠标按下和松开。

```
canvas.addEventListener("mousemove", function (e) {
    lastMouse.x = Mouse.x;
    lastMouse.y = Mouse.y;
    Mouse.x = e.pageX - this.offsetLeft - 15;
    Mouse.y = e.pageY - this.offsetTop - 15;
}, false);

canvas.addEventListener("mousedown", function (e) {
    canvas.addEventListener("mousemove", onPaint, false);
}, false);

canvas.addEventListener("mouseup", function () {
    canvas.removeEventListener("mousemove", onPaint, false);
}, false);
```

4　多个数字同时识别

4.1　算法设计

项目直接新建了面向验证码的模型，其优势在于网络结构简单（尤其较于语义分割）且准确率提升，劣势则为没有很好的和单数字识别模型与 flask 结合。提前使用 keras 图像预处理将数字分开，每个数字单独训练、经过使用了 8 层结构的网络。数字输入后首先是两个 Conv2D 二维卷积层，然后经过一层最大池化层后，通过 Dropout 层删除 25%的神经元再展平（Fltatten 层），之后经历两层全连接神经网络夹一层 Dropout 输出。

4.2　验证码实现效果

训练网络后，准确率为 98.0%

```
New model created.
2023-01-06 16:35:33.510347: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Op
n Passes are enabled (registered 2)
Epoch 1/10
75/75 [==============================] - 2s 18ms/step - loss: 2.3166 - accuracy: 0.1289 - val_loss: 2.2877 - va
y: 0.1000
Epoch 2/10
75/75 [==============================] - 1s 15ms/step - loss: 2.2042 - accuracy: 0.2556 - val_loss: 2.0592 - va
y: 0.4667
Epoch 3/10
75/75 [==============================] - 1s 15ms/step - loss: 1.6626 - accuracy: 0.4578 - val_loss: 1.3213 - va
y: 0.5733
Epoch 4/10
75/75 [==============================] - 1s 15ms/step - loss: 1.0082 - accuracy: 0.6622 - val_loss: 0.6961 - va
y: 0.8467
Epoch 5/10
75/75 [==============================] - 1s 15ms/step - loss: 0.6029 - accuracy: 0.8222 - val_loss: 0.2829 - va
y: 0.9267
Epoch 6/10
75/75 [==============================] - 1s 15ms/step - loss: 0.3443 - accuracy: 0.8867 - val_loss: 0.2142 - va
y: 0.9467
Epoch 7/10
75/75 [==============================] - 1s 15ms/step - loss: 0.2186 - accuracy: 0.9244 - val_loss: 0.1025 - va
y: 0.9800
Epoch 8/10
75/75 [==============================] - 1s 15ms/step - loss: 0.2621 - accuracy: 0.9133 - val_loss: 0.0909 - va
y: 0.9867
Epoch 9/10
75/75 [==============================] - 1s 15ms/step - loss: 0.1575 - accuracy: 0.9533 - val_loss: 0.1063 - va
y: 0.9733
Epoch 10/10
75/75 [==============================] - 1s 15ms/step - loss: 0.1059 - accuracy: 0.9644 - val_loss: 0.0647 - va
y: 0.9800
Test loss: 0.064740851521492
Test accuracy: 0.9800000190734863
```

测试以下验证码的识别效果，左上为原图，右上是结果，下方是过程

结果：337188

```
(pytorch) PS D:\desktop\人机\RecognizeMN> python predict.py testing/337188.png
2023-01-06 16:42:21.614690: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use
s:  AUX AUX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-01-06 16:42:21.894120: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimization Passes are enabled (registered 2)
Traceback (most recent call last):
  File "D:\desktop\人机\RecognizeMN\predict.py", line 42, in <module>
    result_class = model.predict_classes(np.array([x_list[i]]), verbose=0)
AttributeError: 'Sequential' object has no attribute 'predict_classes'
(pytorch) PS D:\desktop\人机\RecognizeMN> python predict.py testing/337188.png
2023-01-06 16:48:52.046811: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use
s:  AUX AUX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-01-06 16:48:52.351730: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimization Passes are enabled (registered 2)
Digit 1: Confidence=> [ 0.000023746  0.000130344  0.06114048   0.930505455  0.000171114  0.004937804  0.000149147  0.001114249  0.001398405  0.000429255]    Predict=> 3
Digit 2: Confidence=> [ 0.000006940  0.000160905  0.000089376  0.955846548  0.000083375  0.036063157  0.000035555  0.002480668  0.005225806  0.000007600]    Predict=> 3
Digit 3: Confidence=> [ 0.000014626  0.000305469  0.001579018  0.000760435  0.000005077  0.000040342  0.000000557  0.997009099  0.000072913  0.000132994]    Predict=> 7
Digit 4: Confidence=> [ 0.000001173  0.562320948  0.000074082  0.001525513  0.433910102  0.000773359  0.000995919  0.000146382  0.000064881  0.000186814]    Predict=> 1
Digit 5: Confidence=> [ 0.000014130  0.000009774  0.000005469  0.002108892  0.000094106  0.002927705  0.001174556  0.000001606  0.993659794  0.000004080]    Predict=> 8
Digit 6: Confidence=> [ 0.000000226  0.000000690  0.000000218  0.000104256  0.000071529  0.000039602  0.000163537  0.000000034  0.999619246  0.000000614]    Predict=> 8
Predicted varification code: [3, 3, 7, 1, 8, 8]
(pytorch) PS D:\desktop\人机\RecognizeMN>
```

结果：702203

```
(pytorch) PS D:\desktop\人机\RecognizeMN> python predict.py testing/702203.png
2023-01-06 16:56:50.122525: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use
s:  AUX AUX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-01-06 16:56:50.450002: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimization Passes are enabled (registered 2)
Digit 1: Confidence=> [ 0.000001563  0.000145633  0.000425857  0.000053309  0.000000235  0.000000020  0.000000016  0.999371827  0.000000225  0.000001295]    Predict=> 7
Digit 2: Confidence=> [ 0.998435557  0.000000009  0.000000103  0.000002040  0.000000113  0.000004913  0.001368602  0.000000019  0.000000917  0.000187619]    Predict=> 0
Digit 3: Confidence=> [ 0.000000377  0.000000134  0.999960899  0.000017779  0.000000003  0.000002050  0.000000000  0.000004699  0.000001342  0.000012708]    Predict=> 2
Digit 4: Confidence=> [ 0.000000768  0.000000327  0.999948978  0.000019943  0.000000002  0.000000682  0.000000000  0.000023956  0.000000226  0.000005105]    Predict=> 2
Digit 5: Confidence=> [ 0.998528600  0.000000006  0.000000459  0.000002595  0.000000203  0.000005478  0.001307522  0.000000038  0.000001261  0.000153834]    Predict=> 0
Digit 6: Confidence=> [ 0.000003360  0.002714094  0.000196141  0.952116489  0.000165996  0.040565968  0.000037742  0.000135187  0.004025122  0.000039874]    Predict=> 3
Predicted varification code: [7, 0, 2, 2, 0, 3]
```

4.3 具体代码

RecognizeMN 文件夹下，包含 model 文件夹，保存网络已训练的模型；testing 和 training 分别是测试集图片和训练集图片；train.py 是网络的训练部分、构建网络，predict.py 是使用已有的神经网络识别验证码的结果。

此处列 train.py

```
import numpy as np
import os
from sklearn.model_selection import train_test_split
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import models
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img

epochs = 10
img_rows = None
img_cols = None
digits_in_img = 6
x_list = list()
y_list = list()
x_train = list()
y_train = list()
x_test = list()
y_test = list()

def split_digits_in_img(img_array, x_list, y_list):
    for i in range(digits_in_img):
        step = img_cols // digits_in_img
        x_list.append(img_array[:, i * step:(i + 1) * step] / 255)
        y_list.append(img_filename[i])

# load all img filenames
img_filenames = os.listdir('training')

# load images as arrays
for img_filename in img_filenames:
    if '.png' not in img_filename:
        continue
    img = load_img('training/{0}'.format(img_filename),
color_mode='grayscale')
    img_array = img_to_array(img)
    img_rows, img_cols, _ = img_array.shape
    split_digits_in_img(img_array, x_list, y_list)
```

```
y_list = keras.utils.to_categorical(y_list, num_classes=10)

# split data into training set and testing set
x_train, x_test, y_train, y_test = train_test_split(x_list, y_list)

# model
if os.path.isfile('model/cnn_model.h5'):
    # recreate the exact same model purely from the file if exist
    model = models.load_model('model/cnn_model.h5')
    print('Model loaded from file.')
else:
    # otherwise, create a new cnn model
    model = models.Sequential()
    model.add(layers.Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=(img_rows, img_cols // digits_in_img, 1)))
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D(pool_size=(2, 2)))
    model.add(layers.Dropout(rate=0.25))
    model.add(layers.Flatten())
    model.add(layers.Dense(128, activation='relu'))
    model.add(layers.Dropout(rate=0.5))
    model.add(layers.Dense(10, activation='softmax'))
    print('New model created.')

model.compile(loss=keras.losses.categorical_crossentropy,
optimizer=keras.optimizers.Adam(), metrics=['accuracy'])

# start training
model.fit(np.array(x_train), np.array(y_train), batch_size=digits_in_img,
epochs=epochs, verbose=1, validation_data=(np.array(x_test),
np.array(y_test)))

# evaluate model
loss, accuracy = model.evaluate(np.array(x_test), np.array(y_test),
verbose=0)
print('Test loss:', loss)
print('Test accuracy:', accuracy)

# save the model
model.save('model/cnn_model.h5')
```