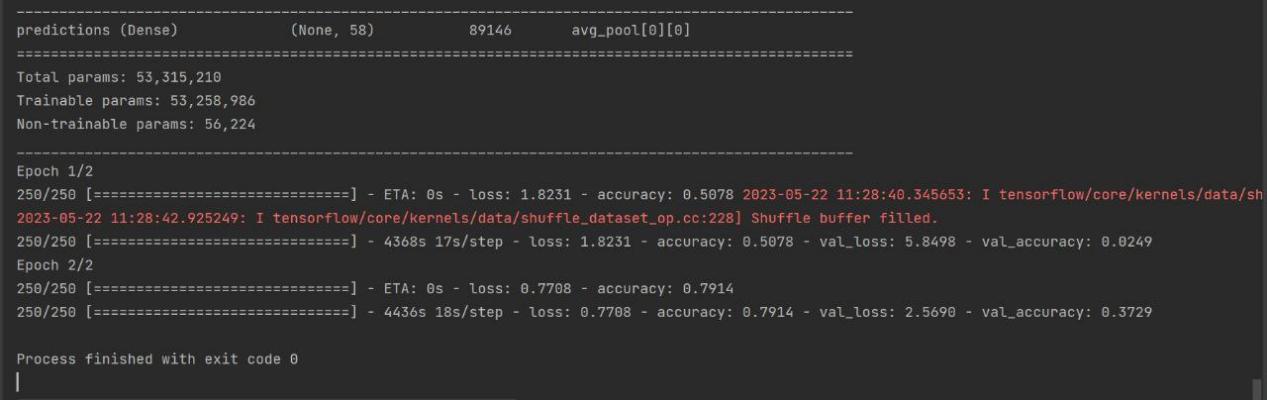
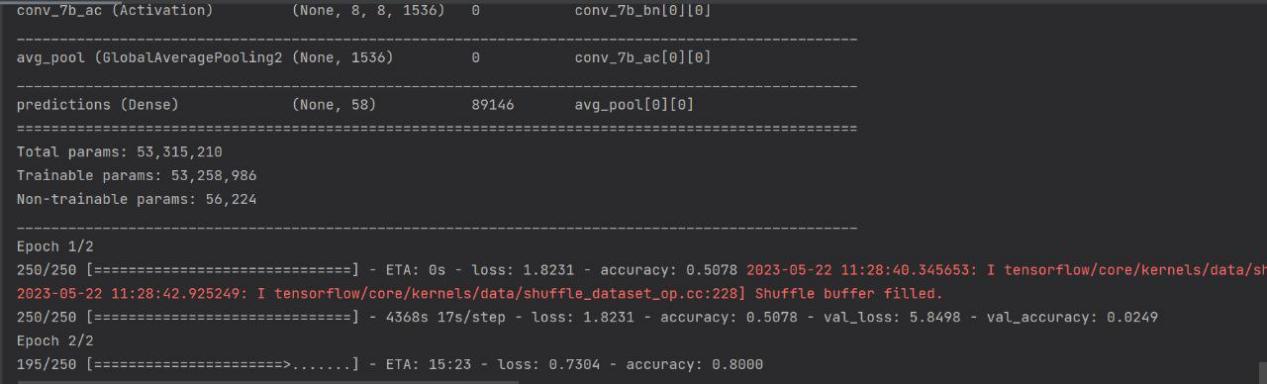
## 基于inception-resnet-v2的交通标识检测系统

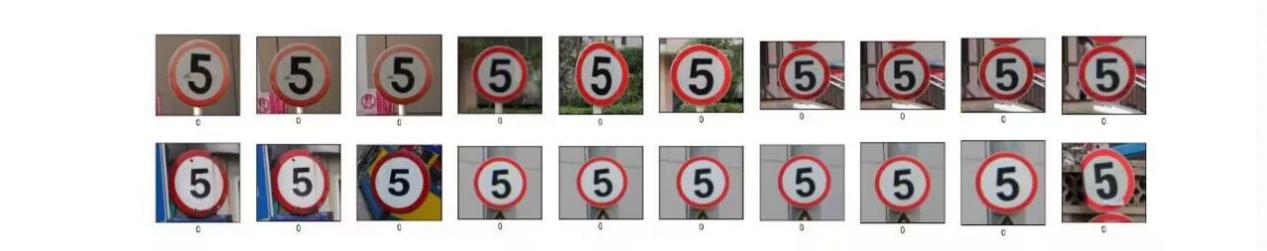
### 运行过程



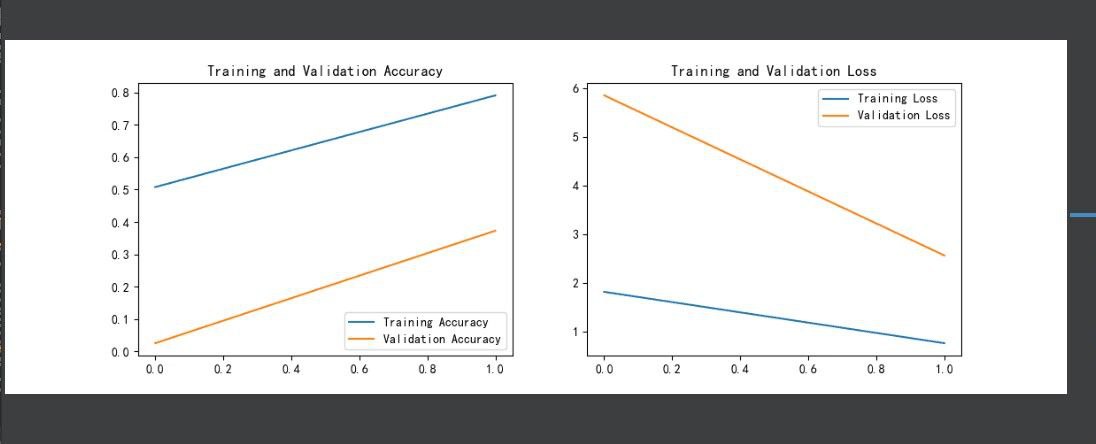


### 运行结果

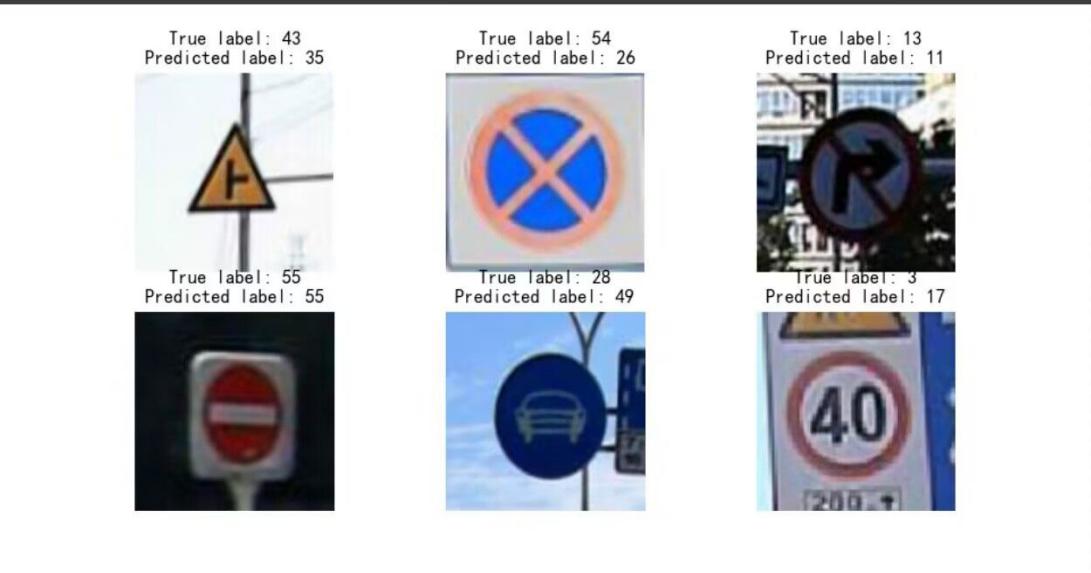
展示数据集：



画出精确度和损失度图像



实验结果：





### 代码

1. 可能需要的库

from tensorflow import keras  
from tensorflow.keras import layers,models  
import tensorflow as tf  
import matplotlib.pyplot as plt  
import pandas as pd  
import numpy as np  
import os,PIL,pathlib  
from tensorflow.keras import layers, models, Input  
from tensorflow.keras.models import Model  
from tensorflow.keras.layers import Conv2D, Dense, Flatten, Dropout,BatchNormalization, Activation  
from tensorflow.keras.layers import MaxPooling2D, AveragePooling2D,Concatenate, Lambda, GlobalAveragePooling2D  
from tensorflow.keras import backend as K

1. 加载和预处理图片

gpus = tf.config.list\_physical\_devices("GPU")  
if gpus:  
 tf.config.experimental.set\_memory\_growth(gpus[0], True)  
 tf.config.set\_visible\_devices([gpus[0]],"GPU")  
  
plt.rcParams['font.sans-serif'] = ['SimHei']  
#用来显示中文标签  
plt.rcParams['axes.unicode\_minus'] = False  
#用来显示正负号  
np.random.seed(1)  
tf.random.set\_seed(1)  
  
# 导入图片数据  
pictures\_dir = "14\_traffic\_sign/images"  
pictures\_dir = pathlib.Path(pictures\_dir)  
  
# 导入训练数据的图片路径名及标签  
train = pd.read\_csv("/home/erfenjiao/14\_traffic/14\_traffic\_sign/annotations.csv")  
# 查看数据  
image\_count = len(list(pictures\_dir.glob('\*.png')))  
print("图片总数为：",image\_count)

3 数据集的加载

def preprocess\_image(image):  
 image = tf.image.decode\_jpeg(image, channels=3) # 编码解码处理  
 image = tf.image.resize(image, [299,299]) # 图片调整  
 return image/255.0 # 归一化处理  
#加载数据路径  
def load\_and\_preprocess\_image(path):  
 image = tf.io.read\_file(path)  
 return preprocess\_image(image)  
# tf.image.decode\_jpeg(image, channels=3): 这一行代码将 JPEG 格式的图像解码为 RGB 格式，即 3 个通道。  
  
AUTOTUNE = tf.data.experimental.AUTOTUNE  
common\_paths = "/home/erfenjiao/14\_traffic/14\_traffic\_sign/images/"  
  
#训练数据的标签  
train\_image\_label = [i for i in train["category"]]  
train\_label\_ds = tf.data.Dataset.from\_tensor\_slices(train\_image\_label)  
  
#训练数据的路径  
train\_image\_paths = [ common\_paths+i for i in train["file\_name"]]  
train\_path\_ds = tf.data.Dataset.from\_tensor\_slices(train\_image\_paths)  
train\_image\_ds = train\_path\_ds.map(load\_and\_preprocess\_image, num\_parallel\_calls=AUTOTUNE)  
#将图片与标签进行打包  
image\_label\_ds = tf.data.Dataset.zip((train\_image\_ds,  
train\_label\_ds))  
  
#显示图片和标签  
plt.figure(figsize=(20, 4))  
for i in range(20):  
 plt.subplot(2, 10, i + 1)  
 plt.xticks([])  
 plt.yticks([])  
 plt.grid(False)  
 images = plt.imread(train\_image\_paths[i])  
 plt.imshow(images)  
 plt.xlabel(train\_image\_label[i])  
plt.show()

4配置数据集

BATCH\_SIZE = 20  
# 将训练数据集拆分成训练集与验证集  
  
#shuffle() ; 打乱数据  
# prefetch() ：预取数据，加速运行，  
# cache() ：将数据集缓存到内存当中，加速运行  
train\_ds = image\_label\_ds.take(5000).shuffle(1000) # 前 1500 个 batch  
val\_ds = image\_label\_ds.skip(5000).shuffle(1000) # 跳过前 1500，选取后面的  
train\_ds = train\_ds.batch(BATCH\_SIZE)  
train\_ds = train\_ds.prefetch(buffer\_size=AUTOTUNE)  
val\_ds = val\_ds.batch(BATCH\_SIZE)  
val\_ds = val\_ds.prefetch(buffer\_size=AUTOTUNE)  
for image\_batch, labels\_batch in train\_ds:  
 print(image\_batch.shape)  
 print(labels\_batch.shape)  
 break  
#再次查看数据，确认是否被打乱  
plt.figure(figsize=(8, 8))  
for images, labels in train\_ds.take(1):  
 for i in range(6):  
 ax = plt.subplot(4, 3, i + 1)  
 plt.imshow(images[i])  
 plt.title(labels[i].numpy()) # 使用.numpy()将张量转换为  
 plt.axis("off")  
plt.show()

1. 定义 Inception-ResNetV2 模型

def inception\_resnet\_block(x, scale, block\_type, block\_idx,activation='relu'):  
 if block\_type == 'block35':  
 branch\_0 = conv2d\_bn(x, 32, 1)  
 branch\_1 = conv2d\_bn(x, 32, 1)  
 branch\_1 = conv2d\_bn(branch\_1, 32, 3)  
 branch\_2 = conv2d\_bn(x, 32, 1)  
 branch\_2 = conv2d\_bn(branch\_2, 48, 3)  
 branch\_2 = conv2d\_bn(branch\_2, 64, 3)  
 branches = [branch\_0, branch\_1, branch\_2]  
 elif block\_type == 'block17':  
 branch\_0 = conv2d\_bn(x, 192, 1)  
 branch\_1 = conv2d\_bn(x, 128, 1)  
 branch\_1 = conv2d\_bn(branch\_1, 160, [1, 7])  
 branch\_1 = conv2d\_bn(branch\_1, 192, [7, 1])  
 branches = [branch\_0, branch\_1]  
 elif block\_type == 'block8':  
 branch\_0 = conv2d\_bn(x, 192, 1)  
 branch\_1 = conv2d\_bn(x, 192, 1)  
 branch\_1 = conv2d\_bn(branch\_1, 224, [1, 3])  
 branch\_1 = conv2d\_bn(branch\_1, 256, [3, 1])  
 branches = [branch\_0, branch\_1]  
 else:  
 raise ValueError('Unknown Inception-ResNet block type. '  
 'Expects "block35", "block17" or "block8", '  
 'but got: ' + str(block\_type))  
 block\_name = block\_type + '\_' + str(block\_idx)  
 mixed = Concatenate(name=block\_name + '\_mixed')(branches)  
 up = conv2d\_bn(mixed, K.int\_shape(x)[3], 1, activation=None,  
use\_bias=True, name=block\_name + '\_conv')  
 x = Lambda(lambda inputs, scale: inputs[0] + inputs[1] \* scale,  
 output\_shape=K.int\_shape(x)[1:],  
 arguments={'scale': scale},  
 name=block\_name)([x, up])  
 if activation is not None:  
 x = Activation(activation, name=block\_name + '\_ac')(x)  
 return x  
  
# InceptionResNetV2() 函数：这是主要的模型构建函数，它首先定义了输入  
# 层（Input），然后通过一系列的操作构建了网络的各个部分，最后定义了输出  
# 层（Dense），并将这些层组合成一个完整的模型。  
def InceptionResNetV2(input\_shape=[299, 299, 3], classes=1000):  
 inputs = Input(shape=input\_shape)  
 #Stem block  
 x = conv2d\_bn(inputs, 32, 3, strides=2, padding='valid')  
 x = conv2d\_bn(x, 32, 3, padding='valid')  
 x = conv2d\_bn(x, 64, 3)  
 x = MaxPooling2D(3, strides=2)(x)  
 x = conv2d\_bn(x, 80, 1, padding='valid')  
 x = conv2d\_bn(x, 192, 3, padding='valid')  
 x = MaxPooling2D(3, strides=2)(x)  
 # Mixed 5b (Inception-A block)  
 branch\_0 = conv2d\_bn(x, 96, 1)  
 branch\_1 = conv2d\_bn(x, 48, 1)  
 branch\_1 = conv2d\_bn(branch\_1, 64, 5)  
 branch\_2 = conv2d\_bn(x, 64, 1)  
 branch\_2 = conv2d\_bn(branch\_2, 96, 3)  
 branch\_2 = conv2d\_bn(branch\_2, 96, 3)  
 branch\_pool = AveragePooling2D(3, strides=1, padding='same')(x)  
 branch\_pool = conv2d\_bn(branch\_pool, 64, 1)  
 branches = [branch\_0, branch\_1, branch\_2, branch\_pool]  
 x = Concatenate(name='mixed\_5b')(branches)  
 # 10 次 Inception-ResNet-A block  
 for block\_idx in range(1, 11):  
 x = inception\_resnet\_block(x, scale=0.17, block\_type='block35', block\_idx=block\_idx)  
 # Reduction-A block  
 branch\_0 = conv2d\_bn(x, 384, 3, strides=2, padding='valid')  
 branch\_1 = conv2d\_bn(x, 256, 1)  
 branch\_1 = conv2d\_bn(branch\_1, 256, 3)  
 branch\_1 = conv2d\_bn(branch\_1, 384, 3, strides=2,padding='valid')  
 branch\_pool = MaxPooling2D(3, strides=2, padding='valid')(x)  
 branches = [branch\_0, branch\_1, branch\_pool]  
 x = Concatenate(name='mixed\_6a')(branches)  
 # 20 次 Inception-ResNet-B block  
 for block\_idx in range(1, 21):  
 x = inception\_resnet\_block(x, scale=0.1,  
block\_type='block17', block\_idx=block\_idx)  
 # Reduction-B block  
 branch\_0 = conv2d\_bn(x, 256, 1)  
 branch\_0 = conv2d\_bn(branch\_0, 384, 3, strides=2,padding='valid')  
 branch\_1 = conv2d\_bn(x, 256, 1)  
 branch\_1 = conv2d\_bn(branch\_1, 288, 3, strides=2,  
padding='valid')  
 branch\_2 = conv2d\_bn(x, 256, 1)  
 branch\_2 = conv2d\_bn(branch\_2, 288, 3)  
 branch\_2 = conv2d\_bn(branch\_2, 320, 3, strides=2,  
padding='valid')  
 branch\_pool = MaxPooling2D(3, strides=2, padding='valid')(x)  
 branches = [branch\_0, branch\_1, branch\_2, branch\_pool]  
 x = Concatenate(name='mixed\_7a')(branches)  
 # 10 次 Inception-ResNet-C block  
 for block\_idx in range(1, 10):  
 x = inception\_resnet\_block(x, scale=0.2, block\_type='block8',  
block\_idx=block\_idx)  
 x = inception\_resnet\_block(x, scale=1., activation=None,  
block\_type='block8', block\_idx=10)  
 x = conv2d\_bn(x, 1536, 1, name='conv\_7b')  
 x = GlobalAveragePooling2D(name='avg\_pool')(x)  
 x = Dense(classes, activation='softmax', name='predictions')(x)  
 # 创建模型  
 model = Model(inputs, x, name='inception\_resnet\_v2')  
 return model  
model = InceptionResNetV2([299, 299, 3], 58)  
model.summary()

1. 设置学习率，优化器并且编译模型

initial\_learning\_rate = 1e-4  
lr\_schedule = tf.keras.optimizers.schedules.ExponentialDecay(  
 initial\_learning\_rate,  
 decay\_steps=100,  
 decay\_rate=0.96, # lr 经过一次衰减就会变成 decay\_rate\*lr  
 staircase=True)  
# 将指数衰减学习率送入优化器  
optimizer = tf.keras.optimizers.Adam(learning\_rate=lr\_schedule)  
# 有了模型，在训练之前，还需要进行一些设置  
model.compile(optimizer=optimizer, loss='sparse\_categorical\_crossentropy',metrics=['accuracy'])  
epochs = 2  
history = model.fit(  
 train\_ds,  
 validation\_data=val\_ds,  
 epochs=epochs  
)

1. 训练模型

history = model.fit(  
train\_ds,  
validation\_data=val\_ds,  
epochs=epochs  
)

1. 模型评估

cc = history.history['accuracy']  
val\_acc = history.history['val\_accuracy']  
loss = history.history['loss']  
val\_loss = history.history['val\_loss']  
epochs\_range = range(epochs)  
plt.figure(figsize=(12, 4))  
plt.subplot(1, 2, 1)  
plt.plot(epochs\_range, acc, label='Training Accuracy')  
plt.plot(epochs\_range, val\_acc, label='Validation Accuracy')  
plt.legend(loc='lower right')  
plt.title('Training and Validation Accuracy')  
plt.subplot(1, 2, 2)  
plt.plot(epochs\_range, loss, label='Training Loss')  
plt.plot(epochs\_range, val\_loss, label='Validation Loss')  
plt.legend(loc='upper right')  
plt.title('Training and Validation Loss')  
plt.show()  
# 保存模型  
model.save('model/14\_model.h5')

9 模型测试

#采用加载的模型（new\_model）来看预测结果  
  
new\_model = keras.models.load\_model('/home/erfenjiao/14\_traffic/14\_model.h5')  
plt.figure(figsize=(10, 5)) # 图形的宽为10高为5  
  
#模型预测  
for images, labels in val\_ds.take(1):  
 for i in range(6):  
 ax = plt.subplot(2, 3, i + 1)  
  
 # 显示图片  
 plt.imshow(images[i])  
  
 # 需要给图片增加一个维度  
 img\_array = tf.expand\_dims(images[i], 0)  
  
 # 使用模型预测路标  
 predictions = new\_model.predict(img\_array)  
 plt.title(np.argmax(predictions))  
  
 plt.axis("off")