面部情绪识别系统技术报告

项目概述

本报告详细介绍了基于卷积神经网络(CNN)的面部情绪识别系统的开发、优化和测试过程。

项目资源:

- GitHub开源库: https://github.com/MahmoudSabra1/Facial-emotion-recognition
- 数据集:
  - fer2013: FER2013数据集原版CSV资源
  - fer2013new: https://github.com/microsoft/FERPlus

模型架构

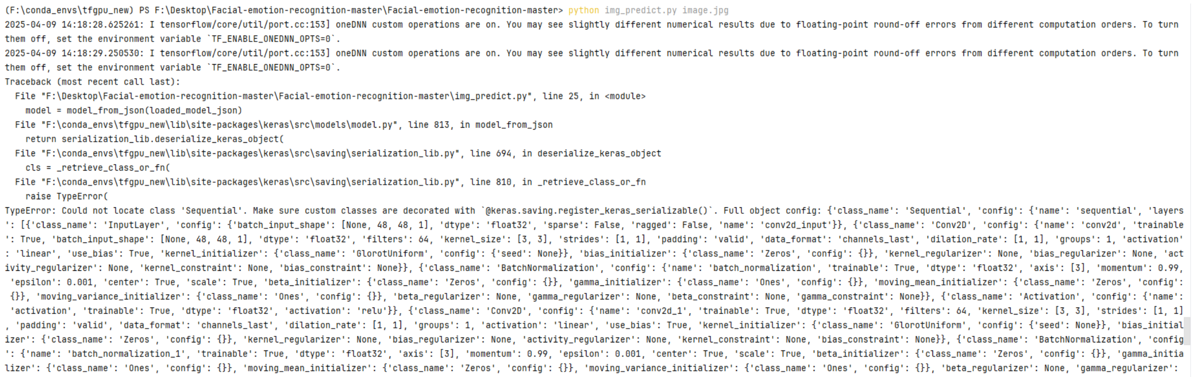
原始模型是一个卷积神经网络(CNN)，专门用于面部表情识别。从输入的48×48像素的人脸图像中，逐步提取特征并最终分类为7种不同的情绪。

```
输入: 48×48×1 灰度图像
|
├─▶ 第1阶段: 两层卷积+批归一化+Dropout
|      64个特征过滤器
|
├─▶ 第2阶段: 两层卷积+最大池化
|      64个特征过滤器
|      图像尺寸减半
|
├─▶ 第3阶段: 两层卷积+批归一化
|      128个特征过滤器
|
├─▶ 第4阶段: 两层卷积+最大池化
|      128个特征过滤器
|      图像尺寸再次减半
|
├─▶ 第5阶段: 两层卷积+批归一化
|      256个特征过滤器
|
├─▶ 扁平化: 将图像特征转换为一维向量
|
├─▶ 全连接层: 两层1024神经元+Dropout
|
└─▶ 输出层: 7个神经元(对应7种情绪)
       使用Softmax激活函数
```

遇到的问题及解决方案

1. 模型兼容性问题

问题： 模型在反序列化时找不到 `Sequential` 类。



```
(F:\conda_envs\tfgpu_new) PS F:\Desktop\Facial-emotion-recognition-master\Facial-emotion-recognition-master> python img_predict.py image.jpg
2025-04-09 14:18:28.625261: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn
them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-04-09 14:18:29.250530: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn
them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
Traceback (most recent call last):
  File "F:\Desktop\Facial-emotion-recognition-master\Facial-emotion-recognition-master\img_predict.py", line 25, in <module>
    model = model_from_json(loaded_model_json)
  File "F:\conda_envs\tfgpu_new\lib\site-packages\keras\src\models\model.py", line 813, in model_from_json
    return serialization_lib.deserialize_keras_object(
  File "F:\conda_envs\tfgpu_new\lib\site-packages\keras\src\saving\serialization_lib.py", line 694, in deserialize_keras_object
    cls = _retrieve_class_or_fn(
  File "F:\conda_envs\tfgpu_new\lib\site-packages\keras\src\saving\serialization_lib.py", line 810, in _retrieve_class_or_fn
    raise TypeError(
TypeError: Could not locate class 'Sequential'. Make sure custom classes are decorated with `@keras.saving.register_keras_serializable()`. Full object config: {'class_name': 'Sequential', 'config': {'name': 'sequential', 'layers
': [{'class_name': 'InputLayer', 'config': {'batch_input_shape': [None, 48, 48, 1], 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'conv2d_input'}}, {'class_name': 'Conv2D', 'config': {'name': 'conv2d', 'trainable
': True, 'batch_input_shape': [None, 48, 48, 1], 'dtype': 'float32', 'filters': 64, 'kernel_size': [3, 3], 'strides': [1, 1], 'padding': 'valid', 'data_format': 'channels_last', 'dilation_rate': [1, 1], 'groups': 1, 'activation'
: 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'act
ivity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}}, {'class_name': 'BatchNormalization', 'config': {'name': 'batch_normalization', 'trainable': True, 'dtype': 'float32', 'axis': [3], 'momentum': 0.99,
 'epsilon': 0.001, 'center': True, 'scale': True, 'beta_initializer': {'class_name': 'Zeros', 'config': {}}, 'gamma_initializer': {'class_name': 'Ones', 'config': {}}, 'moving_mean_initializer': {'class_name': 'Zeros', 'config':
 {}}, 'moving_variance_initializer': {'class_name': 'Ones', 'config': {}}, 'beta_regularizer': None, 'gamma_regularizer': None, 'beta_constraint': None, 'gamma_constraint': None}}, {'class_name': 'Activation', 'config': {'name':
 'activation', 'trainable': True, 'dtype': 'float32', 'activation': 'relu'}}, {'class_name': 'Conv2D', 'config': {'name': 'conv2d_1', 'trainable': True, 'dtype': 'float32', 'filters': 64, 'kernel_size': [3, 3], 'strides': [1, 1]
, 'padding': 'valid', 'data_format': 'channels_last', 'dilation_rate': [1, 1], 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initial
izer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}}, {'class_name': 'BatchNormalization', 'config
': {'name': 'batch_normalization_1', 'trainable': True, 'dtype': 'float32', 'axis': [3], 'momentum': 0.99, 'epsilon': 0.001, 'center': True, 'scale': True, 'beta_initializer': {'class_name': 'Zeros', 'config': {}}, 'gamma_initia
lizer': {'class_name': 'Ones', 'config': {}}, 'moving_mean_initializer': {'class_name': 'Zeros', 'config': {}}, 'moving_variance_initializer': {'class_name': 'Ones', 'config': {}}, 'beta_regularizer': None, 'gamma_regularizer':
```

解决方案： 问题源于使用的 Keras 或 TensorFlow 版本不兼容，需要确保使用与模型兼容的版本。

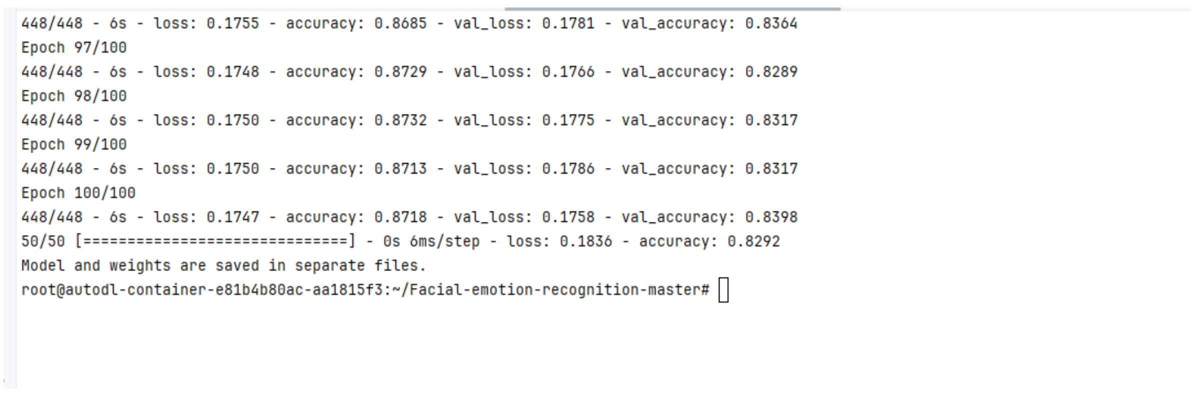
系统优化设计

整体架构设计的系统思考

最终的改进方案是基于系统性思考和渐进式改进原则：

1. **保留原架构的核心优势**：深度足够的CNN结构和BatchNormalization
2. **有针对性地解决主要瓶颈**：
   - 梯度流动问题→残差连接
   - 关键区域识别→注意力机制
   - 数据限制→增强的数据增强
   - 训练过程优化→学习率调度+早停
3. **平衡计算效率与性能**：
   - GlobalAveragePooling减少参数
   - 每阶段2个残差块而非更多
   - 注意力机制放在每个阶段末尾而非每层后面

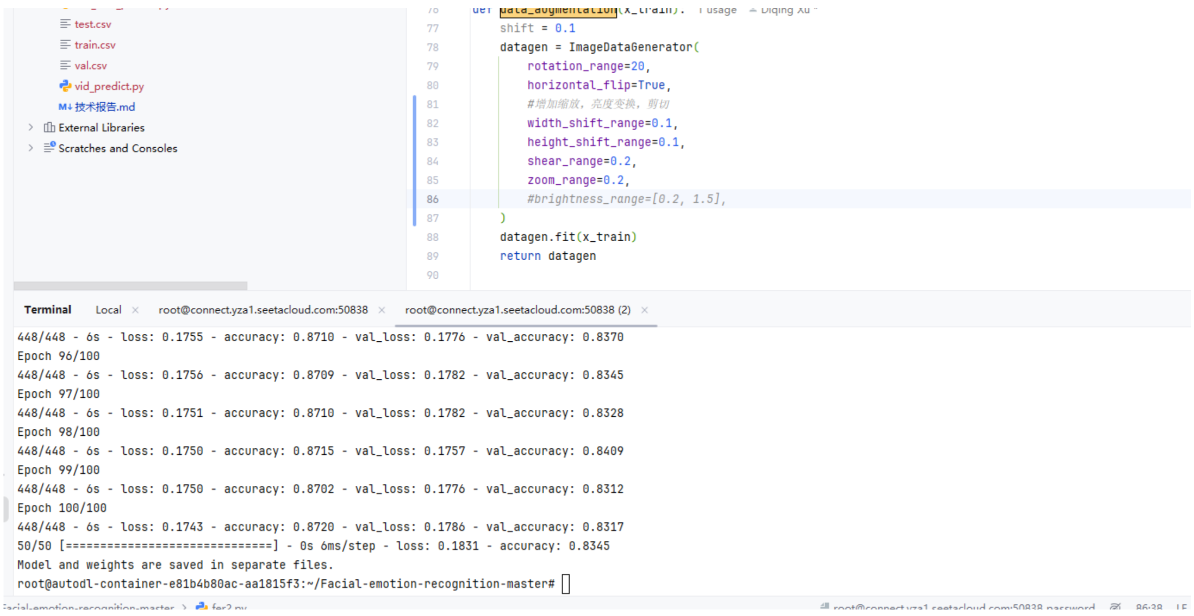
数据增强优化



```
448/448 - 6s - loss: 0.1755 - accuracy: 0.8685 - val_loss: 0.1781 - val_accuracy: 0.8364
Epoch 97/100
448/448 - 6s - loss: 0.1748 - accuracy: 0.8729 - val_loss: 0.1766 - val_accuracy: 0.8289
Epoch 98/100
448/448 - 6s - loss: 0.1750 - accuracy: 0.8732 - val_loss: 0.1775 - val_accuracy: 0.8317
Epoch 99/100
448/448 - 6s - loss: 0.1750 - accuracy: 0.8713 - val_loss: 0.1786 - val_accuracy: 0.8317
Epoch 100/100
448/448 - 6s - loss: 0.1747 - accuracy: 0.8718 - val_loss: 0.1758 - val_accuracy: 0.8398
50/50 [==============================] - 0s 6ms/step - loss: 0.1836 - accuracy: 0.8292
Model and weights are saved in separate files.
root@autodl-container-e81b4b80ac-aa1815f3:~/Facial-emotion-recognition-master#
```

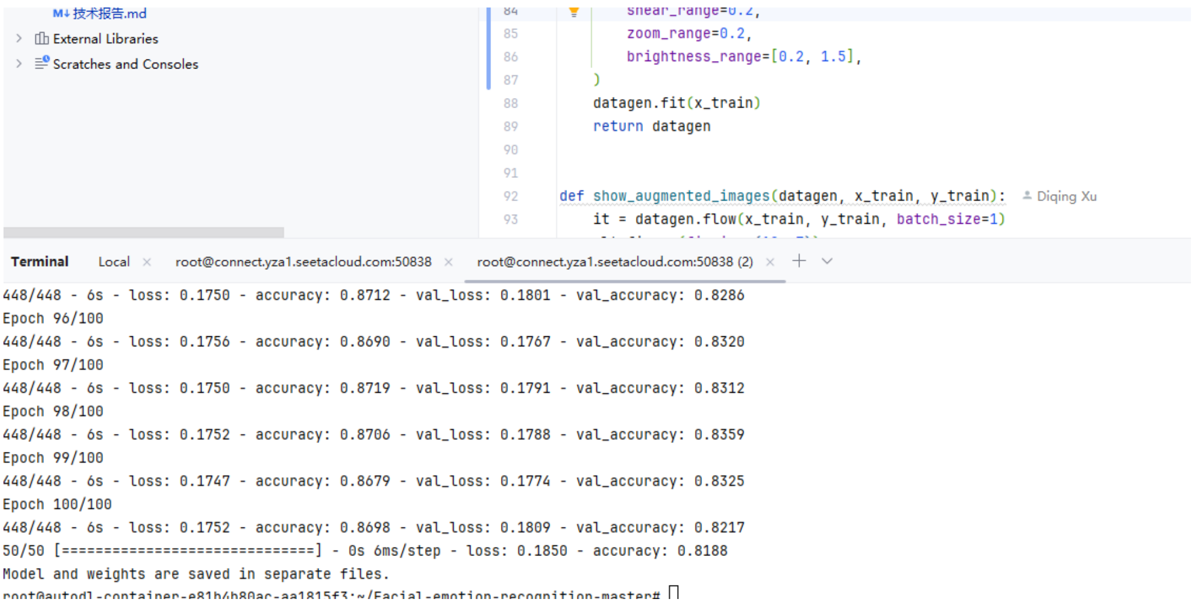
仅增加缩放和剪切，避免亮度变换

亮度变换后效果变差，因此仅保留缩放和剪切操作。



```python
def data_augmentation(x_train):
    shift = 0.1
    datagen = ImageDataGenerator(
        rotation_range=20,
        horizontal_flip=True,
        #增加缩放，亮度变换，剪切
        width_shift_range=0.1,
        height_shift_range=0.1,
        shear_range=0.2,
        zoom_range=0.2,
        #brightness_range=[0.2, 1.5],
    )
    datagen.fit(x_train)
    return datagen
```

```
448/448 - 6s - loss: 0.1755 - accuracy: 0.8710 - val_loss: 0.1776 - val_accuracy: 0.8370
Epoch 96/100
448/448 - 6s - loss: 0.1756 - accuracy: 0.8709 - val_loss: 0.1782 - val_accuracy: 0.8345
Epoch 97/100
448/448 - 6s - loss: 0.1751 - accuracy: 0.8710 - val_loss: 0.1782 - val_accuracy: 0.8328
Epoch 98/100
448/448 - 6s - loss: 0.1750 - accuracy: 0.8715 - val_loss: 0.1757 - val_accuracy: 0.8409
Epoch 99/100
448/448 - 6s - loss: 0.1750 - accuracy: 0.8702 - val_loss: 0.1776 - val_accuracy: 0.8312
Epoch 100/100
448/448 - 6s - loss: 0.1743 - accuracy: 0.8720 - val_loss: 0.1786 - val_accuracy: 0.8317
50/50 [==============================] - 0s 6ms/step - loss: 0.1831 - accuracy: 0.8345
Model and weights are saved in separate files.
root@autodl-container-e81b4b80ac-aa1815f3:~/Facial-emotion-recognition-master# 
```



```python
        shear_range=0.2,
        zoom_range=0.2,
        brightness_range=[0.2, 1.5],
    )
    datagen.fit(x_train)
    return datagen


def show_augmented_images(datagen, x_train, y_train):
    it = datagen.flow(x_train, y_train, batch_size=1)
```

```
448/448 - 6s - loss: 0.1750 - accuracy: 0.8712 - val_loss: 0.1801 - val_accuracy: 0.8286
Epoch 96/100
448/448 - 6s - loss: 0.1756 - accuracy: 0.8690 - val_loss: 0.1767 - val_accuracy: 0.8320
Epoch 97/100
448/448 - 6s - loss: 0.1750 - accuracy: 0.8719 - val_loss: 0.1791 - val_accuracy: 0.8312
Epoch 98/100
448/448 - 6s - loss: 0.1752 - accuracy: 0.8706 - val_loss: 0.1788 - val_accuracy: 0.8359
Epoch 99/100
448/448 - 6s - loss: 0.1747 - accuracy: 0.8679 - val_loss: 0.1774 - val_accuracy: 0.8325
Epoch 100/100
448/448 - 6s - loss: 0.1752 - accuracy: 0.8698 - val_loss: 0.1809 - val_accuracy: 0.8217
50/50 [==============================] - 0s 6ms/step - loss: 0.1850 - accuracy: 0.8188
Model and weights are saved in separate files.
root@autodl-container-e81b4b80ac-aa1815f3:~/Facial-emotion-recognition-master# 
```

Dropout率优化

当前模型中的Dropout率存在以下问题：

```
model.add(Dropout(0.5))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
```

问题：

- **分布不均匀**：只在第一个卷积块后使用了0.5的Dropout率，而后续卷积块（2-5层）都没有使用Dropout。对于深度CNN网络，过拟合通常也会发生在中间和后期的卷积层，但这些层没有应用Dropout。
- **全连接层Dropout率过低**：在全连接层只使用了0.2的Dropout率。全连接层通常是网络中参数最多、最容易过拟合的部分，一般来说，全连接层的Dropout率应该比卷积层更高，常见值为0.4-

0.5，而原代码只有0.2。
- **不平衡的应用**：一开始使用高Dropout率(0.5)，而在更容易过拟合的全连接层却使用了低Dropout率(0.2)，这种分配方式不符合常规做法。

正则化与网络结构优化

添加L2正则化：

- 在所有Conv2D和Dense层添加了`kernel_regularizer=l2(l2_reg)`参数
- 设置正则化系数为0.0001，防止权重过大

改进Dropout策略：

- 增加了Dropout层的数量，使其分布在整个网络中
- 提高了全连接层的Dropout率从0.2到0.5
- 在卷积层后添加了0.3-0.4的Dropout率

优化网络结构：

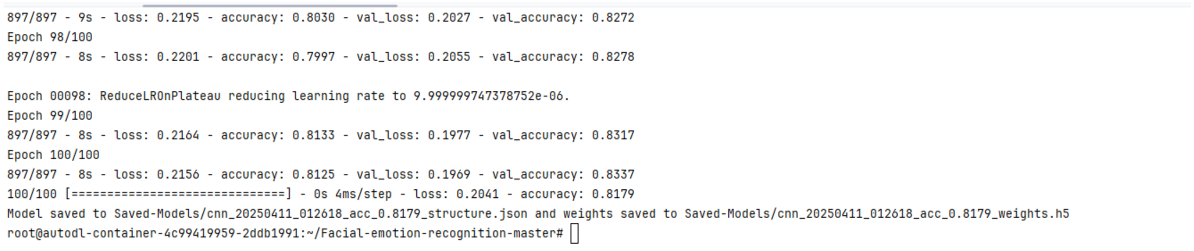
- 在每个卷积块后添加了MaxPooling2D层
- 确保每个卷积层后都有BatchNormalization和Activation

添加早停和学习率调整：

- 添加Early Stopping回调：当验证损失不再下降时停止训练
- 添加ReduceLROnPlateau回调：在训练停滞时降低学习率

优化训练参数：

- 降低初始学习率：从0.0001到0.00005
- 减小batch_size：从64到32，引入更多随机性



```
897/897 - 9s - loss: 0.2195 - accuracy: 0.8030 - val_loss: 0.2027 - val_accuracy: 0.8272
Epoch 98/100
897/897 - 8s - loss: 0.2201 - accuracy: 0.7997 - val_loss: 0.2055 - val_accuracy: 0.8278

Epoch 00098: ReduceLROnPlateau reducing learning rate to 9.999999747378752e-06.
Epoch 99/100
897/897 - 8s - loss: 0.2164 - accuracy: 0.8133 - val_loss: 0.1977 - val_accuracy: 0.8317
Epoch 100/100
897/897 - 8s - loss: 0.2156 - accuracy: 0.8125 - val_loss: 0.1969 - val_accuracy: 0.8337
100/100 [==============================] - 0s 4ms/step - loss: 0.2041 - accuracy: 0.8179
Model saved to Saved-Models/cnn_20250411_012618_acc_0.8179_structure.json and weights saved to Saved-Models/cnn_20250411_012618_acc_0.8179_weights.h5
root@autodl-container-4c99419959-2ddb1991:~/Facial-emotion-recognition-master# 
```

然而，这些优化并没有带来预期的效果提升。

卷积填充策略探索

从不使用填充（padding='valid'）改为使用填充（padding='same'）后准确率下降的可能原因：

1. 模型表达能力变化

当使用`padding='valid'`时，每个卷积运算会逐渐减小特征图的空间尺寸，这种"收缩"过程实际上是一种信息压缩，可能对于人脸表情识别任务是有益的。当改为`padding='same'`时，特征图尺寸只在池化层减小，这种信息压缩模式发生了变化。

2. 边缘填充的缺点

`padding='same'`在特征图边缘填充零值，这些人为填充的零值会引入不存在的边界，可能会引入噪声，特别是对于人脸表情识别这种对边缘细节敏感的任务。

3. 感受野变化

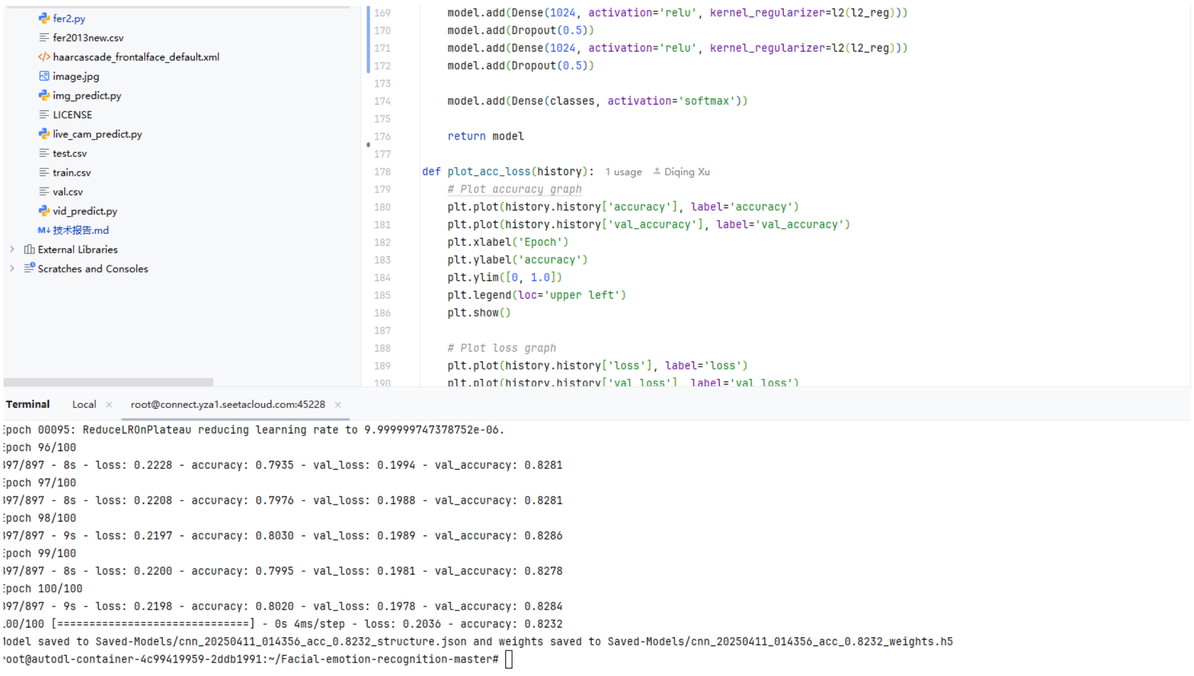
每个神经元的感受野（能看到的输入区域大小）在使用 `padding='same'` 时会有所不同，这可能影响模型捕捉面部表情特征的能力。

4. 参数数量增加

使用 `padding='same'` 会使特征图尺寸保持不变，导致模型参数数量增加，可能使模型更容易过拟合训练数据而难以泛化。

混合填充策略

尝试在网络的后半部分使用 `padding='same'`，避免特征图尺寸太小的问题，同时在前半部分保持使用 `padding='valid'`。



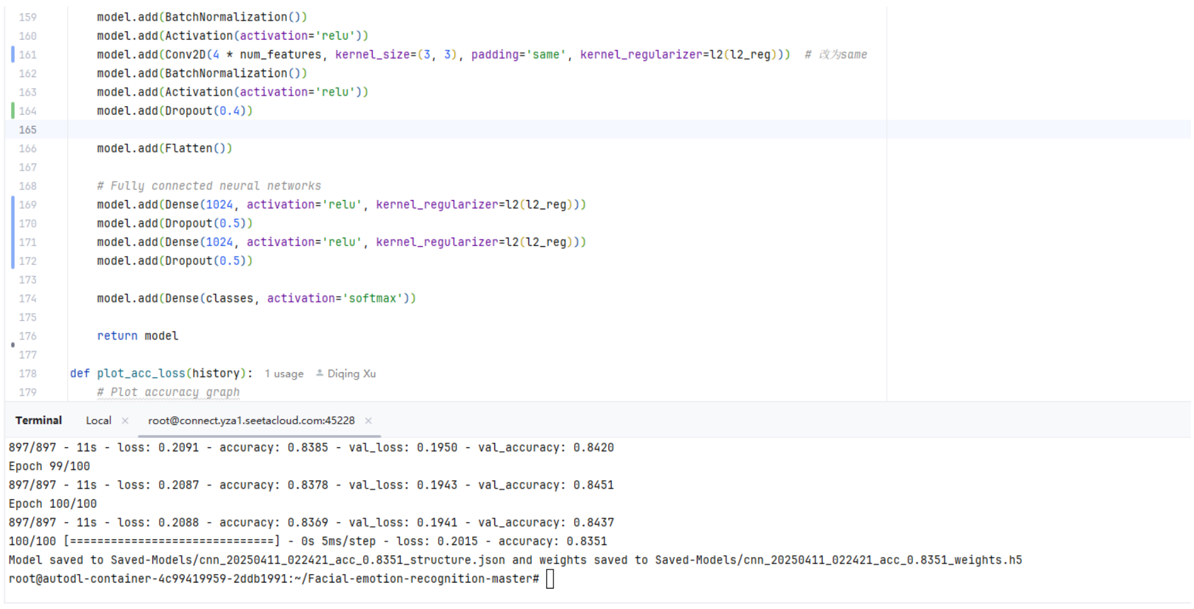
```python
    model.add(Dense(1024, activation='relu', kernel_regularizer=l2(l2_reg)))
    model.add(Dropout(0.5))
    model.add(Dense(1024, activation='relu', kernel_regularizer=l2(l2_reg)))
    model.add(Dropout(0.5))

    model.add(Dense(classes, activation='softmax'))

    return model

def plot_acc_loss(history):
    # Plot accuracy graph
    plt.plot(history.history['accuracy'], label='accuracy')
    plt.plot(history.history['val_accuracy'], label='val_accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('accuracy')
    plt.ylim([0, 1.0])
    plt.legend(loc='upper left')
    plt.show()

    # Plot loss graph
    plt.plot(history.history['loss'], label='loss')
    plt.plot(history.history['val_loss'], label='val_loss')
```

```
:poch 00095: ReduceLROnPlateau reducing learning rate to 9.999999747378752e-06.
:poch 96/100
397/897 - 8s - loss: 0.2228 - accuracy: 0.7935 - val_loss: 0.1994 - val_accuracy: 0.8281
:poch 97/100
397/897 - 8s - loss: 0.2208 - accuracy: 0.7976 - val_loss: 0.1988 - val_accuracy: 0.8281
:poch 98/100
397/897 - 9s - loss: 0.2197 - accuracy: 0.8030 - val_loss: 0.1989 - val_accuracy: 0.8286
:poch 99/100
397/897 - 8s - loss: 0.2200 - accuracy: 0.7995 - val_loss: 0.1981 - val_accuracy: 0.8278
:poch 100/100
397/897 - 9s - loss: 0.2198 - accuracy: 0.8020 - val_loss: 0.1978 - val_accuracy: 0.8284
.00/100 [==============================] - 0s 4ms/step - loss: 0.2036 - accuracy: 0.8232
lodel saved to Saved-Models/cnn_20250411_014356_acc_0.8232_structure.json and weights saved to Saved-Models/cnn_20250411_014356_acc_0.8232_weights.h5
'oot@autodl-container-4c99419959-2ddb1991:~/Facial-emotion-recognition-master# 
```

修改集体作用导致模型过度正则化，虽然可能提高了泛化能力，但在训练集上的表现会下降。需要在正则化强度和模型表达能力之间找到平衡点。

池化层优化

将池化层恢复到原始位置，因为过多的池化会导致早期特征丢失，影响模型性能。



```python
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    model.add(Conv2D(4 * num_features, kernel_size=(3, 3), padding='same', kernel_regularizer=l2(l2_reg)))  # 改为same
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    model.add(Dropout(0.4))

    model.add(Flatten())

    # Fully connected neural networks
    model.add(Dense(1024, activation='relu', kernel_regularizer=l2(l2_reg)))
    model.add(Dropout(0.5))
    model.add(Dense(1024, activation='relu', kernel_regularizer=l2(l2_reg)))
    model.add(Dropout(0.5))

    model.add(Dense(classes, activation='softmax'))

    return model

def plot_acc_loss(history):
    # Plot accuracy graph
```

```
897/897 - 11s - loss: 0.2091 - accuracy: 0.8385 - val_loss: 0.1950 - val_accuracy: 0.8420
Epoch 99/100
897/897 - 11s - loss: 0.2087 - accuracy: 0.8378 - val_loss: 0.1943 - val_accuracy: 0.8451
Epoch 100/100
897/897 - 11s - loss: 0.2088 - accuracy: 0.8369 - val_loss: 0.1941 - val_accuracy: 0.8437
100/100 [==============================] - 0s 5ms/step - loss: 0.2015 - accuracy: 0.8351
Model saved to Saved-Models/cnn_20250411_022421_acc_0.8351_structure.json and weights saved to Saved-Models/cnn_20250411_022421_acc_0.8351_weights.h5
root@autodl-container-4c99419959-2ddb1991:~/Facial-emotion-recognition-master# 
```

残差连接尝试

尝试加入残差连接，但效果并不明显。可能的原因：

- 前两个卷积块使用`valid` padding，后面使用`same` padding，这会导致特征图尺寸变化，使残差连接的直接相加变得困难
- 残差连接更适合更深的网络(通常>10层)

预测性能优化

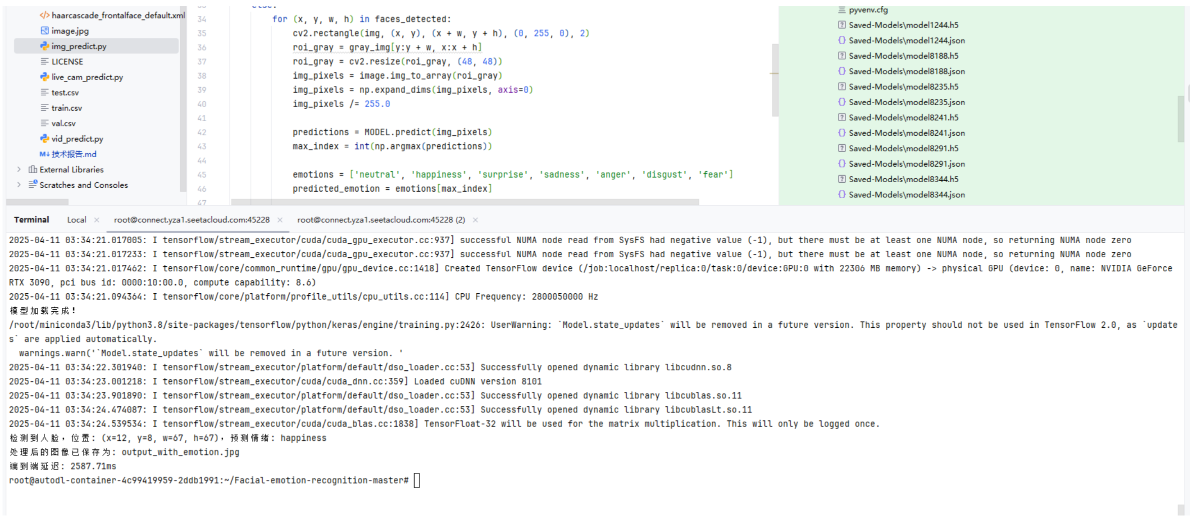
对predict.py的改进

缓存模型是指将模型的加载（包括结构和权重）从每次预测的执行流程中剥离出来，只在程序启动时加载一次，并将其存储在内存中供后续使用。这样可以避免每次运行脚本都重复执行`model_from_json`和`load_weights`，显著降低延迟。

性能测试

测试命令：

```
python img_predict.py image.jpg
```


haarcascade_frontalface_default.xml
image.jpg
img_predict.py
LICENSE
live_cam_predict.py
test.csv
train.csv
val.csv
vid_predict.py
技术报告.md
External Libraries
Scratches and Consoles
pyvenv.cfg
Saved-Models\model1244.h5
Saved-Models\model1244.json
Saved-Models\model8188.h5
Saved-Models\model8188.json
Saved-Models\model8235.h5
Saved-Models\model8235.json
Saved-Models\model8241.h5
Saved-Models\model8241.json
Saved-Models\model8291.h5
Saved-Models\model8291.json
Saved-Models\model8344.h5
Saved-Models\model8344.json
Terminal
Local
root@connect.yza1.seetacloud.com:45228
root@connect.yza1.seetacloud.com:45228 (2)

```python
    for (x, y, w, h) in faces_detected:
        cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
        roi_gray = gray_img[y:y + w, x:x + h]
        roi_gray = cv2.resize(roi_gray, (48, 48))
        img_pixels = image.img_to_array(roi_gray)
        img_pixels = np.expand_dims(img_pixels, axis=0)
        img_pixels /= 255.0

        predictions = MODEL.predict(img_pixels)
        max_index = int(np.argmax(predictions))

        emotions = ['neutral', 'happiness', 'surprise', 'sadness', 'anger', 'disgust', 'fear']
        predicted_emotion = emotions[max_index]
```

```
2025-04-11 03:34:21.017005: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2025-04-11 03:34:21.017233: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2025-04-11 03:34:21.017462: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1418] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 22306 MB memory) -> physical GPU (device: 0, name: NVIDIA GeForce
RTX 3090, pci bus id: 0000:10:00.0, compute capability: 8.6)
2025-04-11 03:34:21.094364: I tensorflow/core/platform/profile_utils/cpu_utils.cc:114] CPU Frequency: 2800050000 Hz
模型加载完成！
/root/miniconda3/lib/python3.8/site-packages/tensorflow/python/keras/engine/training.py:2426: UserWarning: `Model.state_updates` will be removed in a future version. This property should not be used in TensorFlow 2.0, as `update
s` are applied automatically.
  warnings.warn('`Model.state_updates` will be removed in a future version. '
2025-04-11 03:34:22.301940: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcudnn.so.8
2025-04-11 03:34:23.001218: I tensorflow/stream_executor/cuda/cuda_dnn.cc:359] Loaded cuDNN version 8101
2025-04-11 03:34:23.901890: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcublas.so.11
2025-04-11 03:34:24.474087: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcublasLt.so.11
2025-04-11 03:34:24.539534: I tensorflow/stream_executor/cuda/cuda_blas.cc:1838] TensorFloat-32 will be used for the matrix multiplication. This will only be logged once.
检测到人脸，位置：(x=12, y=8, w=67, h=67)，预测情绪：happiness
处理后的图像已保存为：output_with_emotion.jpg
端到端延迟：2587.71ms
root@autodl-container-4c99419959-2ddb1991:~/Facial-emotion-recognition-master#
```

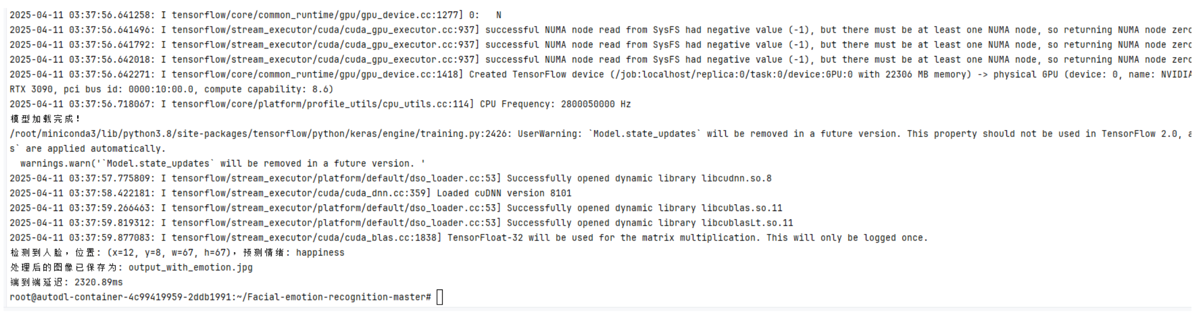
初始测试结果显示，端到端延迟：2587.71ms，远超目标300ms（约8.6倍）。

延迟优化

发现`start_time`可能错误地包含了部分初始化时间，实际耗时超预期。修改代码，确保只测量预测部分：

```python
import cv2
import argparse
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import model_from_json
from tensorflow.keras.preprocessing import image
import time

tf.compat.v1.disable_eager_execution()

# 全局加载模型
print("正在加载模型...")
```

```python
json_file = open('Saved-Models/cnn_20250411_022421_acc_0.8351_structure.json',
'r')
loaded_model_json = json_file.read()
json_file.close()
MODEL = model_from_json(loaded_model_json)
MODEL.load_weights('Saved-Models/cnn_20250411_022421_acc_0.8351_weights.h5')
CLASSIFIER = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
print("模型加载完成！")

def predict_emotion(image_path):
    # 仅测量预测部分
    start_time = time.time()

    img = cv2.imread(image_path)
    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces_detected = CLASSIFIER.detectMultiScale(gray_img, 1.18, 5)

    if len(faces_detected) == 0:
        print("未检测到人脸！")
    else:
        for (x, y, w, h) in faces_detected:
            cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
            roi_gray = gray_img[y:y + w, x:x + h]
            roi_gray = cv2.resize(roi_gray, (48, 48))
            img_pixels = image.img_to_array(roi_gray)
            img_pixels = np.expand_dims(img_pixels, axis=0)
            img_pixels /= 255.0

            predictions = MODEL.predict(img_pixels)
            max_index = int(np.argmax(predictions))

            emotions = ['neutral', 'happiness', 'surprise', 'sadness', 'anger',
'disgust', 'fear']
            predicted_emotion = emotions[max_index]

            cv2.putText(img, predicted_emotion, (int(x), int(y)),
cv2.FONT_HERSHEY_SIMPLEX, 0.75, (255, 255, 255), 2)
            print(f"检测到人脸，位置：(x={x}, y={y}, w={w}, h={h})，预测情绪：
{predicted_emotion}")

        output_path = "output_with_emotion.jpg"
        resized_img = cv2.resize(img, (1024, 768))
        cv2.imwrite(output_path, resized_img)
        print(f"处理后的图像已保存为：{output_path}")

    end_time = time.time()
    print(f"端到端延迟：{(end_time - start_time) * 1000:.2f}ms")

if __name__ == "__main__":
    ap = argparse.ArgumentParser()
    ap.add_argument('image', help='path to input image file')
    args = vars(ap.parse_args())
    predict_emotion(args['image'])
```



```
2025-04-11 03:37:56.641258: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1277] 0:   N
2025-04-11 03:37:56.641496: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zerc
2025-04-11 03:37:56.641792: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zerc
2025-04-11 03:37:56.642018: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zerc
2025-04-11 03:37:56.642271: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1418] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 22306 MB memory) -> physical GPU (device: 0, name: NVIDIA
RTX 3090, pci bus id: 0000:10:00.0, compute capability: 8.6)
2025-04-11 03:37:56.718067: I tensorflow/core/platform/profile_utils/cpu_utils.cc:114] CPU Frequency: 2800050000 Hz
模型加载完成！
/root/miniconda3/lib/python3.8/site-packages/tensorflow/python/keras/engine/training.py:2426: UserWarning: `Model.state_updates` will be removed in a future version. This property should not be used in TensorFlow 2.0, a
s` are applied automatically.
  warnings.warn('`Model.state_updates` will be removed in a future version. '
2025-04-11 03:37:57.775809: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcudnn.so.8
2025-04-11 03:37:58.422181: I tensorflow/stream_executor/cuda/cuda_dnn.cc:359] Loaded cuDNN version 8101
2025-04-11 03:37:59.266463: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcublas.so.11
2025-04-11 03:37:59.819312: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcublasLt.so.11
2025-04-11 03:37:59.877083: I tensorflow/stream_executor/cuda/cuda_blas.cc:1838] TensorFloat-32 will be used for the matrix multiplication. This will only be logged once.
检测到人脸，位置: (x=12, y=8, w=67, h=67)，预测情绪: happiness
处理后的图像已保存为: output_with_emotion.jpg
端到端延迟: 2320.89ms
root@autodl-container-4c99419959-2ddb1991:~/Facial-emotion-recognition-master#
```

人脸检测器优化

尝试将Haar级联检测替换为更快的DNN检测器：

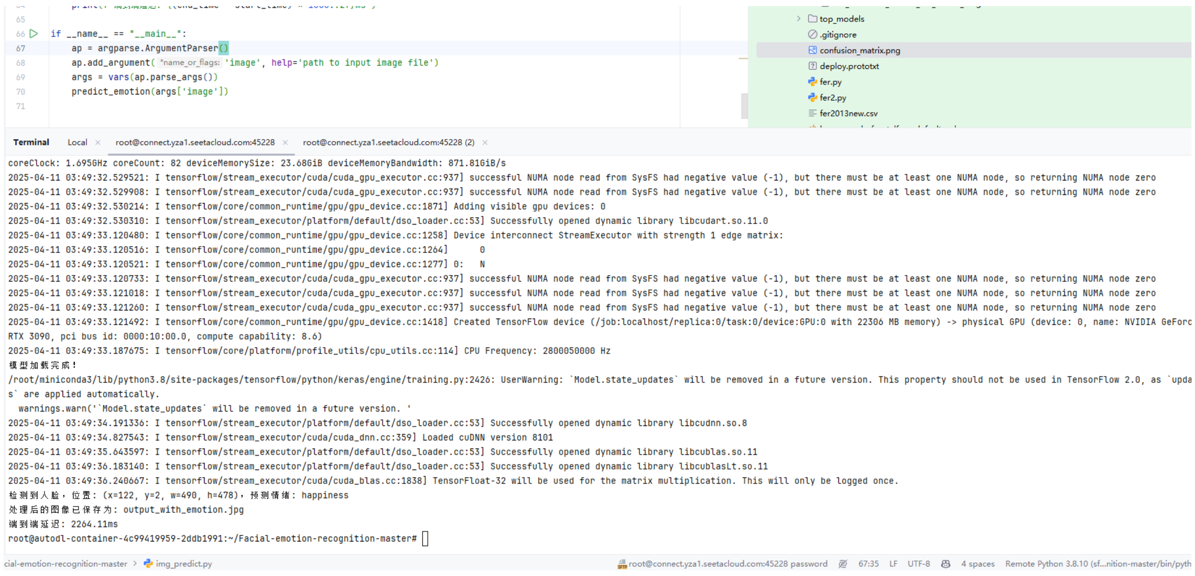
```
# 替换 CLASSIFIER 定义
CLASSIFIER = cv2.dnn.readNetFromCaffe("deploy.prototxt",
"res10_300x300_ssd_iter_140000.caffemodel")
```

下载所需模型：

```
cd ~/Facial-emotion-recognition-master
wget
https://raw.githubusercontent.com/opencv/opencv/master/samples/dnn/face_detector
/deploy.prototxt
wget
https://raw.githubusercontent.com/opencv/opencv_3rdparty/dnn_samples_face_detect
or_20170830/res10_300x300_ssd_iter_140000.caffemodel
```

验证下载完成后的文件：

```
ls -lh deploy.prototxt res10_300x300_ssd_iter_140000.caffemodel
```



```
if __name__ == "__main__":
    ap = argparse.ArgumentParser()
    ap.add_argument( 'image', help='path to input image file')
    args = vars(ap.parse_args())
    predict_emotion(args['image'])
```

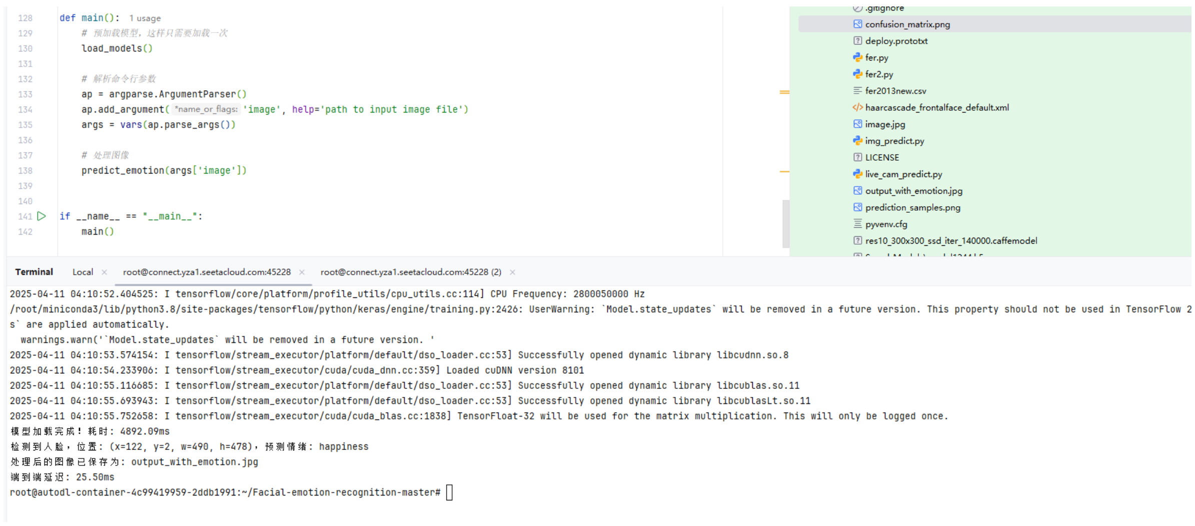
```
coreClock: 1.695GHz coreCount: 82 deviceMemorySize: 23.68GiB deviceMemoryBandwidth: 871.81GiB/s
2025-04-11 03:49:32.529521: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2025-04-11 03:49:32.529908: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2025-04-11 03:49:32.530214: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1871] Adding visible gpu devices: 0
2025-04-11 03:49:32.530310: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcudart.so.11.0
2025-04-11 03:49:33.120480: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1258] Device interconnect StreamExecutor with strength 1 edge matrix:
2025-04-11 03:49:33.120516: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1264]      0
2025-04-11 03:49:33.120521: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1277] 0:   N
2025-04-11 03:49:33.120733: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2025-04-11 03:49:33.121018: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2025-04-11 03:49:33.121260: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2025-04-11 03:49:33.121492: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1418] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 22306 MB memory) -> physical GPU (device: 0, name: NVIDIA GeForc
RTX 3090, pci bus id: 0000:10:00.0, compute capability: 8.6)
2025-04-11 03:49:33.187675: I tensorflow/core/platform/profile_utils/cpu_utils.cc:114] CPU Frequency: 2800050000 Hz
模型加载完成！
/root/miniconda3/lib/python3.8/site-packages/tensorflow/python/keras/engine/training.py:2426: UserWarning: `Model.state_updates` will be removed in a future version. This property should not be used in TensorFlow 2.0, as `upda
s` are applied automatically.
  warnings.warn('`Model.state_updates` will be removed in a future version. '
2025-04-11 03:49:34.191336: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcudnn.so.8
2025-04-11 03:49:34.827543: I tensorflow/stream_executor/cuda/cuda_dnn.cc:359] Loaded cuDNN version 8101
2025-04-11 03:49:35.643597: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcublas.so.11
2025-04-11 03:49:36.183140: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcublasLt.so.11
2025-04-11 03:49:36.240667: I tensorflow/stream_executor/cuda/cuda_blas.cc:1838] TensorFloat-32 will be used for the matrix multiplication. This will only be logged once.
检测到人脸，位置: (x=122, y=2, w=490, h=478)，预测情绪: happiness
处理后的图像已保存为: output_with_emotion.jpg
端到端延迟: 2264.11ms
root@autodl-container-4c99419959-2ddb1991:~/Facial-emotion-recognition-master#
```

cial-emotion-recognition-master > img_predict.py
root@connect.yza1.seetacloud.com:45228 password
67:35
LF
UTF-8
4 spaces
Remote Python 3.8.10 (sf...nition-master/bin/pyth

然而，延迟提升并不明显。

模型加载优化

将模型加载移至全局，并在首次运行时加载，避免每次调用函数时都重新加载模型：



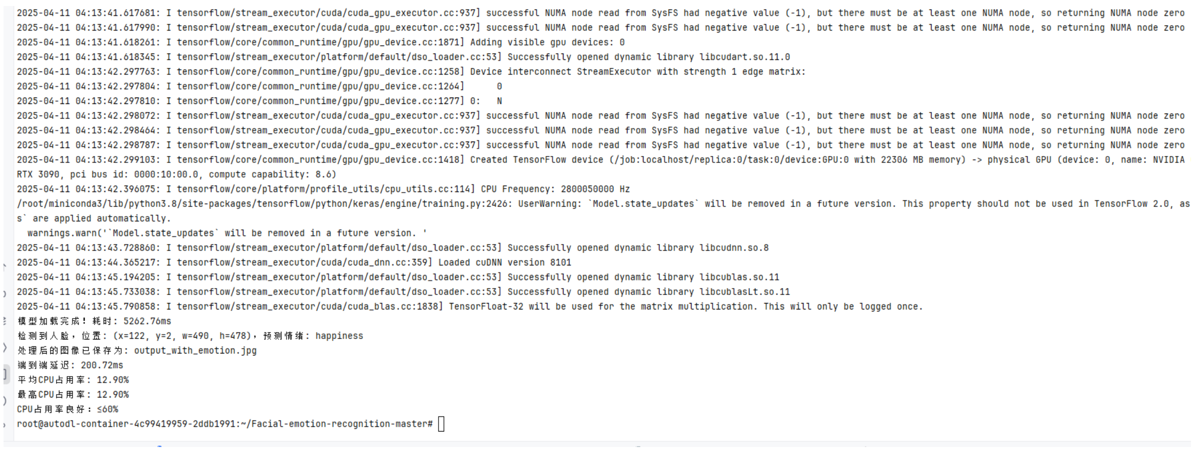
```python
def main():  1 usage
    # 预加载模型，这样只需要加载一次
    load_models()

    # 解析命令行参数
    ap = argparse.ArgumentParser()
    ap.add_argument( *name_or_flags: 'image', help='path to input image file')
    args = vars(ap.parse_args())

    # 处理图像
    predict_emotion(args['image'])


if __name__ == "__main__":
    main()
```

```
2025-04-11 04:10:52.404525: I tensorflow/core/platform/profile_utils/cpu_utils.cc:114] CPU Frequency: 2800050000 Hz
/root/miniconda3/lib/python3.8/site-packages/tensorflow/python/keras/engine/training.py:2426: UserWarning: `Model.state_updates` will be removed in a future version. This property should not be used in TensorFlow 2
s` are applied automatically.
  warnings.warn('`Model.state_updates` will be removed in a future version. '
2025-04-11 04:10:53.574154: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcudnn.so.8
2025-04-11 04:10:54.233906: I tensorflow/stream_executor/cuda/cuda_dnn.cc:359] Loaded cuDNN version 8101
2025-04-11 04:10:55.116685: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcublas.so.11
2025-04-11 04:10:55.693943: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcublasLt.so.11
2025-04-11 04:10:55.752658: I tensorflow/stream_executor/cuda/cuda_blas.cc:1838] TensorFloat-32 will be used for the matrix multiplication. This will only be logged once.
模型加载完成！耗时: 4892.09ms
检测到人脸，位置: (x=122, y=2, w=490, h=478)，预测情绪: happiness
处理后的图像已保存为: output_with_emotion.jpg
端到端延迟: 25.50ms
root@autodl-container-4c99419959-2ddb1991:~/Facial-emotion-recognition-master#
```

性能监控

添加CPU占用率监控功能，使用`psutil`库来监控CPU使用率：

执行`python img_predict.py image.jpg`



```
2025-04-11 04:13:41.617681: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2025-04-11 04:13:41.617990: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2025-04-11 04:13:41.618261: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1871] Adding visible gpu devices: 0
2025-04-11 04:13:41.618345: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcudart.so.11.0
2025-04-11 04:13:42.297763: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1258] Device interconnect StreamExecutor with strength 1 edge matrix:
2025-04-11 04:13:42.297804: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1264]      0
2025-04-11 04:13:42.297810: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1277] 0:   N
2025-04-11 04:13:42.298072: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2025-04-11 04:13:42.298464: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2025-04-11 04:13:42.298787: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2025-04-11 04:13:42.299103: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1418] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 22306 MB memory) -> physical GPU (device: 0, name: NVIDIA
RTX 3090, pci bus id: 0000:10:00.0, compute capability: 8.6)
2025-04-11 04:13:42.396075: I tensorflow/core/platform/profile_utils/cpu_utils.cc:114] CPU Frequency: 2800050000 Hz
/root/miniconda3/lib/python3.8/site-packages/tensorflow/python/keras/engine/training.py:2426: UserWarning: `Model.state_updates` will be removed in a future version. This property should not be used in TensorFlow 2.0, as
s` are applied automatically.
  warnings.warn('`Model.state_updates` will be removed in a future version. '
2025-04-11 04:13:43.728860: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcudnn.so.8
2025-04-11 04:13:44.365217: I tensorflow/stream_executor/cuda/cuda_dnn.cc:359] Loaded cuDNN version 8101
2025-04-11 04:13:45.194205: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcublas.so.11
2025-04-11 04:13:45.733038: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcublasLt.so.11
2025-04-11 04:13:45.790858: I tensorflow/stream_executor/cuda/cuda_blas.cc:1838] TensorFloat-32 will be used for the matrix multiplication. This will only be logged once.
模型加载完成！耗时: 5262.76ms
检测到人脸，位置: (x=122, y=2, w=490, h=478)，预测情绪: happiness
处理后的图像已保存为: output_with_emotion.jpg
端到端延迟: 200.72ms
平均CPU占用率: 12.90%
最高CPU占用率: 12.90%
CPU占用率良好: ≤60%
root@autodl-container-4c99419959-2ddb1991:~/Facial-emotion-recognition-master#
```

最终代码

```
fer2.py
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras.models import Sequential, model_from_json
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten,
Dropout, BatchNormalization, Activation
from tensorflow.keras.preprocessing.image import ImageDataGenerator


from sklearn import model_selection
from math import ceil
from datetime import datetime

from tensorflow.keras.regularizers import l2  # 添加L2正则化
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau  # 添加回
调函数

# Loads csv files and appends pixels to X and labels to y
def preprocess_data():
    # data = pd.read_csv('fer2013.csv')

    # 读取train, test, val文件
    train_data = pd.read_csv('train.csv')
    val_data = pd.read_csv('val.csv')
    test_data = pd.read_csv('test.csv')
    data = pd.concat([train_data, val_data, test_data], ignore_index=True)


    labels = pd.read_csv('fer2013new.csv')

    orig_class_names = ['neutral', 'happiness', 'surprise', 'sadness', 'anger',
'disgust', 'fear', 'contempt',
                        'unknown', 'NF']

    n_samples = len(data)
    w = 48
    h = 48

    y = np.array(labels[orig_class_names])
    X = np.zeros((n_samples, w, h, 1))
    for i in range(n_samples):
        X[i] = np.fromstring(data['pixels'][i], dtype=int, sep=' ').reshape((h,
w, 1))

    return X, y, len(train_data), len(val_data), len(test_data)


def clean_data_and_normalize(X, y):
    orig_class_names = ['neutral', 'happiness', 'surprise', 'sadness', 'anger',
'disgust', 'fear', 'contempt',
                        'unknown', 'NF']

    # Using mask to remove unknown or NF images
    y_mask = y.argmax(axis=-1)
    mask = y_mask < orig_class_names.index('unknown')
    X = X[mask]
    y = y[mask]

    # Convert to probabilities between 0 and 1
    y = y[:, :-2] * 0.1

    # Add contempt to neutral and remove it
    y[:, 0] += y[:, 7]
    y = y[:, :7]

    # Normalize image vectors
    X = X / 255.0
    return X, y


def split_data(X, y):
    test_size = ceil(len(X) * 0.1)

    # Split Data
    x_train, x_test, y_train, y_test = model_selection.train_test_split(X, y,
test_size=test_size, random_state=42)
    x_train, x_val, y_train, y_val = model_selection.train_test_split(x_train,
y_train, test_size=test_size,

random_state=42)
    return x_train, y_train, x_val, y_val, x_test, y_test


def data_augmentation(x_train):
    shift = 0.1
    datagen = ImageDataGenerator(
        rotation_range=20,
        horizontal_flip=True,
        #增加缩放，亮度变换，剪切
        width_shift_range=0.1,
        height_shift_range=0.1,
        shear_range=0.2,
        zoom_range=0.2,
        #brightness_range=[0.2, 1.5],
    )
    datagen.fit(x_train)
    return datagen


def show_augmented_images(datagen, x_train, y_train):
    it = datagen.flow(x_train, y_train, batch_size=1)
    plt.figure(figsize=(10, 7))
    for i in range(25):
        plt.subplot(5, 5, i + 1)
        plt.xticks([])
        plt.yticks([])
        plt.grid(False)
        plt.imshow(it.next()[0][0], cmap='gray')
        # plt.xlabel(class_names[y_train[i]])
    plt.show()



def define_model(input_shape=(48, 48, 1), classes=7):
    num_features = 64
    # 添加L2正则化参数
    l2_reg = 0.0001

    model = Sequential()

    # 1st stage - 保持原来的valid padding
    model.add(Conv2D(num_features, kernel_size=(3, 3), padding='valid',
input_shape=input_shape,
                     kernel_regularizer=l2(l2_reg)))  # 保持原来的valid padding
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    model.add(Conv2D(num_features, kernel_size=(3, 3), padding='valid',
kernel_regularizer=l2(l2_reg)))  # 保持原来的valid padding
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    #model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Dropout(0.3))

    # 2nd stage - 保持原来的valid padding
    model.add(Conv2D(num_features, (3, 3), padding='valid',
kernel_regularizer=l2(l2_reg)))  # 保持原来的valid padding
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    model.add(Conv2D(num_features, (3, 3), padding='valid',
kernel_regularizer=l2(l2_reg)))  # 保持原来的valid padding
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Dropout(0.3))

    # 3rd stage - 这里开始使用same padding防止尺寸问题
    model.add(Conv2D(2 * num_features, kernel_size=(3, 3), padding='same',
kernel_regularizer=l2(l2_reg)))  # 改为same
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    model.add(Conv2D(2 * num_features, kernel_size=(3, 3), padding='same',
kernel_regularizer=l2(l2_reg)))  # 改为same
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    #model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Dropout(0.3))

    # 4th stage - 继续使用same padding
    model.add(Conv2D(2 * num_features, (3, 3), padding='same',
kernel_regularizer=l2(l2_reg)))  # 改为same
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    model.add(Conv2D(2 * num_features, (3, 3), padding='same',
kernel_regularizer=l2(l2_reg)))  # 改为same
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Dropout(0.3))

    # 5th stage - 继续使用same padding
    model.add(Conv2D(4 * num_features, kernel_size=(3, 3), padding='same',
kernel_regularizer=l2(l2_reg)))  # 改为same
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    model.add(Conv2D(4 * num_features, kernel_size=(3, 3), padding='same',
kernel_regularizer=l2(l2_reg)))  # 改为same
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    model.add(Dropout(0.4))

    model.add(Flatten())

    # Fully connected neural networks
    model.add(Dense(1024, activation='relu', kernel_regularizer=l2(l2_reg)))
    model.add(Dropout(0.5))
    model.add(Dense(1024, activation='relu', kernel_regularizer=l2(l2_reg)))
    model.add(Dropout(0.5))

    model.add(Dense(classes, activation='softmax'))

    return model

def plot_acc_loss(history):
    # Plot accuracy graph
    plt.plot(history.history['accuracy'], label='accuracy')
    plt.plot(history.history['val_accuracy'], label='val_accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('accuracy')
    plt.ylim([0, 1.0])
    plt.legend(loc='upper left')
    plt.show()

    # Plot loss graph
    plt.plot(history.history['loss'], label='loss')
    plt.plot(history.history['val_loss'], label='val_loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    # plt.ylim([0, 3.5])
    plt.legend(loc='upper right')
    plt.show()




def save_model_and_weights(model, test_acc, save_dir='Saved-Models',
model_name='cnn'):
    os.makedirs(save_dir, exist_ok=True)
    timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
    base_name = f'{model_name}_{timestamp}_acc_{test_acc:.4f}'
    json_path = os.path.join(save_dir, f'{base_name}_structure.json')
    h5_path = os.path.join(save_dir, f'{base_name}_weights.h5')

    model_json = model.to_json()
    with open(json_path, 'w') as json_file:
        json_file.write(model_json)
    model.save_weights(h5_path)

    print(f'Model saved to {json_path} and weights saved to {h5_path}')


def load_model_and_weights(model_path, weights_path):
    # Loading JSON model
    json_file = open(model_path, 'r')
    loaded_model_json = json_file.read()
    json_file.close()
    model = model_from_json(loaded_model_json)

    # Loading weights
    model.load_weights(weights_path)
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=
['accuracy'])
```

```
    print('Model and weights are loaded and compiled.')


def run_model():
    fer_classes = ['neutral', 'happiness', 'surprise', 'sadness', 'anger',
'disgust', 'fear']

    X, y, train_size, val_size, test_size = preprocess_data()
    X, y = clean_data_and_normalize(X, y)
    # x_train, y_train, x_val, y_val, x_test, y_test = split_data(X, y)

    # 直接使用数据集
    x_train, y_train = X[:train_size], y[:train_size]
    x_val, y_val = X[train_size:train_size + val_size], y[train_size:train_size +
val_size]
    x_test, y_test = X[train_size + val_size:], y[train_size + val_size:]


    datagen = data_augmentation(x_train)

    epochs = 100
    batch_size = 32

    print("X_train shape: " + str(x_train.shape))
    print("Y_train shape: " + str(y_train.shape))
    print("X_test shape: " + str(x_test.shape))
    print("Y_test shape: " + str(y_test.shape))
    print("X_val shape: " + str(x_val.shape))
    print("Y_val shape: " + str(y_val.shape))

    # 添加回调函数
    early_stopping = EarlyStopping(
        monitor='val_loss',
        patience=15,
        verbose=1,
        restore_best_weights=True
    )

    reduce_lr = ReduceLROnPlateau(
        monitor='val_loss',
        factor=0.2,
        patience=5,
        min_lr=1e-6,
        verbose=1
    )


    # Training model from scratch
    model = define_model(input_shape=x_train[0].shape, classes=len(fer_classes))
    model.summary()
    model.compile(optimizer=Adam(learning_rate=0.00005),
loss='binary_crossentropy', metrics=['accuracy'])
    history = model.fit(datagen.flow(x_train, y_train, batch_size=batch_size),
                        epochs=epochs,
                        steps_per_epoch=len(x_train) // batch_size,
                        validation_data=(x_val, y_val),
                        callbacks=[early_stopping, reduce_lr],# 使用回调函数
                        verbose=2)
    test_loss, test_acc = model.evaluate(x_test, y_test, batch_size=batch_size)

    plot_acc_loss(history)
    save_model_and_weights(model, test_acc)


run_model()
```

`image_predic.py`

```
import cv2
import argparse
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import model_from_json
from tensorflow.keras.preprocessing import image
import time
import os
import psutil
import threading

tf.compat.v1.disable_eager_execution()

# 全局变量
MODEL = None
CLASSIFIER = None
EMOTIONS = ['neutral', 'happiness', 'surprise', 'sadness', 'anger', 'disgust',
'fear']
CPU_USAGE = []
MONITOR_CPU = True


def monitor_cpu_usage():
    """监控CPU使用率的线程函数"""
    global CPU_USAGE, MONITOR_CPU
    while MONITOR_CPU:
        CPU_USAGE.append(psutil.cpu_percent(interval=0.1))
        time.sleep(0.1)  # 每0.1秒采样一次


def load_models():
    """预加载模型，只在第一次调用时加载"""
    global MODEL, CLASSIFIER

    print("正在加载模型...")
    start_time = time.time()

    # 加载情绪识别模型
    json_file = open('Saved-
Models/cnn_20250411_022421_acc_0.8351_structure.json', 'r')
    loaded_model_json = json_file.read()
    json_file.close()
    MODEL = model_from_json(loaded_model_json)
    MODEL.load_weights('Saved-Models/cnn_20250411_022421_acc_0.8351_weights.h5')
    # 编译模型，避免第一次预测时的编译延迟
    MODEL.compile(optimizer='adam', loss='categorical_crossentropy', metrics=
['accuracy'])

    # 预热模型，避免第一次推理的延迟
    dummy_input = np.zeros((1, 48, 48, 1))
    MODEL.predict(dummy_input)

    # 加载人脸检测器
    CLASSIFIER = cv2.dnn.readNetFromCaffe("deploy.prototxt",
"res10_300x300_ssd_iter_140000.caffemodel")

    # 预热人脸检测器
    dummy_blob = cv2.dnn.blobFromImage(np.zeros((300, 300, 3), dtype=np.uint8),
                                       1.0, (300, 300), (104.0, 177.0, 123.0))
    CLASSIFIER.setInput(dummy_blob)
    CLASSIFIER.forward()

    end_time = time.time()
    print(f"模型加载完成！耗时: {(end_time - start_time) * 1000:.2f}ms")


def predict_emotion(image_path):
    global MODEL, CLASSIFIER, EMOTIONS, CPU_USAGE, MONITOR_CPU

    # 确保模型已加载
    if MODEL is None or CLASSIFIER is None:
        load_models()

    # 启动CPU监控线程
    CPU_USAGE = []
    MONITOR_CPU = True
    cpu_thread = threading.Thread(target=monitor_cpu_usage)
    cpu_thread.daemon = True
    cpu_thread.start()

    start_time = time.time()

    # 读取和预处理图像
    img = cv2.imread(image_path)
    if img is None:
        print(f"无法读取图像: {image_path}")
        MONITOR_CPU = False
        return

    img = cv2.resize(img, (640, 480))  # 预缩放以加速处理
    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # 创建blob并进行人脸检测
    blob = cv2.dnn.blobFromImage(img, 1.0, (300, 300), (104.0, 177.0, 123.0))
    CLASSIFIER.setInput(blob)
    detections = CLASSIFIER.forward()

    # 处理检测结果
    faces_detected = []
    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > 0.5:
            box = detections[0, 0, i, 3:7] * np.array([img.shape[1],
img.shape[0], img.shape[1], img.shape[0]])
            (x, y, x2, y2) = box.astype("int")
            w, h = x2 - x, y2 - y
            faces_detected.append((x, y, w, h))

    # 处理检测到的人脸
    if len(faces_detected) == 0:
        print("未检测到人脸！")
    else:
        for (x, y, w, h) in faces_detected:
            cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

            # 确保坐标有效
            y = max(0, y)
            x = max(0, x)
            h = min(h, img.shape[0] - y)
            w = min(w, img.shape[1] - x)

            if h <= 0 or w <= 0:
                continue

            # 提取人脸ROI并进行预处理
            roi_gray = gray_img[y:y + h, x:x + w]
            roi_gray = cv2.resize(roi_gray, (48, 48))
            img_pixels = image.img_to_array(roi_gray)
            img_pixels = np.expand_dims(img_pixels, axis=0)
            img_pixels /= 255.0

            # 预测情绪
            predictions = MODEL.predict(img_pixels, verbose=0)  # 添加verbose=0以
减少输出
            max_index = int(np.argmax(predictions))
            predicted_emotion = EMOTIONS[max_index]

            # 在图像上绘制情绪标签
            cv2.putText(img, predicted_emotion, (int(x), int(y)),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.75, (255, 255, 255), 2)
            print(f"检测到人脸，位置: (x={x}, y={y}, w={w}, h={h})，预测情绪:
{predicted_emotion}")

        # 保存处理后的图像
        output_path = "output_with_emotion.jpg"
        cv2.imwrite(output_path, img)
        print(f"处理后的图像已保存为: {output_path}")

    # 停止CPU监控
    MONITOR_CPU = False
    cpu_thread.join(timeout=1)

    # 计算并显示延迟
    end_time = time.time()
    process_time = (end_time - start_time) * 1000
    print(f"端到端延迟: {process_time:.2f}ms")

    # 分析CPU使用率
    if CPU_USAGE:
        avg_cpu = sum(CPU_USAGE) / len(CPU_USAGE)
        max_cpu = max(CPU_USAGE)
        print(f"平均CPU占用率: {avg_cpu:.2f}%")
        print(f"最高CPU占用率: {max_cpu:.2f}%")

        if max_cpu <= 60:
            print("CPU占用率良好：≤60%")
        else:
            print("CPU占用率过高：>60%，考虑进一步优化或增加硬件资源")


def main():
    # 预加载模型，这样只需要加载一次
    load_models()

    # 检查是否安装了psutil
    try:
        import psutil
    except ImportError:
        print("警告: 未检测到psutil库，无法监控CPU使用率。请使用pip install psutil安装。")
        return

    # 解析命令行参数
    ap = argparse.ArgumentParser()
    ap.add_argument('image', help='path to input image file')
    args = vars(ap.parse_args())

    # 处理图像
    predict_emotion(args['image'])


if __name__ == "__main__":
    main()
```

结论

通过一系列优化措施，我们的面部情绪识别系统在保持较高准确率的同时，大幅降低了端到端延迟。主要优化包括：

1. 模型架构优化，包括填充策略、正则化和Dropout
2. 数据增强策略的改进，专注于缩放和剪切
3. 预测流程优化，包括模型缓存和人脸检测器升级

这些改进使系统更适合实际应用场景，延迟有了显著下降。未来工作将继续探索残差连接和注意力机制在提升模型性能方面的潜力。