3.2.1

```
import onnxruntime as ort
import numpy as np
import scipy.special
from PIL import Image


# 预处理图像
def preprocess_image(image, resize_size=256, crop_size=224, mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]):
  image = image.resize((resize_size, resize_size), Image.BILINEAR)
  w, h = image.size
  left = (w - crop_size) / 2
  top = (h - crop_size) / 2
  image = image.crop((left, top, left + crop_size, top + crop_size))
  image = np.array(image).astype(np.float32)
  image = image / 255.0
  image = (image - mean) / std
  image = np.transpose(image, (2, 0, 1))
  image = image.reshape((1,) + image.shape)
  return image


# 模型加载 2分
session = ort.InferenceSession('resnet.onnx')


# 加载类别标签
labels_path = 'labels.txt'
with open(labels_path) as f:
  labels = [line.strip() for line in f.readlines()]


# 获取模型输入和输出的名称
input_name = session.get_inputs()[0].name
output_name = session.get_outputs()[0].name


# 加载图片 2分
image = Image.open('img_test.jpg').convert('RGB')
```

```
# 预处理图片 2分
processed_image = preprocess_image(image)


# 确保输入数据是 float32 类型
processed_image = processed_image.astype(np.float32)


# 进行图片识别 2分
output = session.run([output_name], {input_name: processed_image})[0]


# 应用 softmax 函数获取概率 2分
probabilities = scipy.special.softmax(output, axis=-1)


# 获取最高的5个概率和对应的类别索引 2分
top5_idx = np.argsort(probabilities[0])[-5:][::-1]
top5_prob = probabilities[0][top5_idx]


# 打印结果
print("Top 5 predicted classes:")
for i in range(5):
  print(f"{i+1}: {labels[top5_idx[i]]} - Probability: {top5_prob[i]}")
```

```
Top 5 predicted classes:
1: tusker - Probability: 0.931233286857605
2: Indian elephant - Probability: 0.06785939633846283
3: African elephant - Probability: 0.0009034110116772354
4: gorilla - Probability: 4.78091294553451e-07
5: water buffalo - Probability: 3.9817274455344887e-07
```

3.2.2

```python
import onnxruntime
import numpy as np
from PIL import Image


# 加载ONNX模型  2分
ort_session = onnxruntime.InferenceSession("mnist.onnx")


# 加载图像 2分
image = Image.open("img_test.png").convert('L')  # 转为灰度图


#图像预处理
image = image.resize((28, 28))  # 调整大小为MNIST模型的输入尺寸1分
image_array = np.asarray(image, dtype=np.float32)  # 转为numpy数组1分
image_array = np.expand_dims(image_array, axis=0)  # 添加batch维度1分
image_array = np.expand_dims(image_array, axis=0)  # 添加通道维度1分（灰度图）


# 执行推理 2分
ort_inputs = {ort_session.get_inputs()[0].name: image_array}
ort_outs = ort_session.run(None, ort_inputs)


# 获取预测结果 2分
predicted_class = np.argmax(ort_outs[0])


# 输出预测结果
print(f"Predicted class: {predicted_class}")
```

```
Predicted class: 8
```

3.2.3

```python
# 导入必要的库
import onnx
import numpy as np
from PIL import Image
import onnxruntime as ort


# 定义预处理函数，用于将图片转换为模型所需的输入格式
def preprocess(image_path):
    input_shape = (1, 1, 64, 64)    # 模型输入期望的形状，这里是 (N, C, H, W)，N=batch size, C=channels, H=height, W=width
    img = Image.open(image_path).convert('L')    # 打开图像文件并将其转换为灰度图  1分
    img = img.resize((64, 64), Image.ANTIALIAS)    # 调整图像大小到模型输入所需的尺寸
    img_data = np.array(img, dtype=np.float32)    # 将PIL图像对象转换为numpy数组，并确保数据类型是float32
    # 调整数组的形状以匹配模型输入的形状
    img_data = np.expand_dims(img_data, axis=0)  # 添加 batch 维度
    img_data = np.expand_dims(img_data, axis=1)  # 添加 channel 维度
    assert img_data.shape == input_shape, f"Expected shape {input_shape}, but got {img_data.shape}"    # 确保最终的形状与模型输入要求的形状一致
    return img_data    # 返回预处理后的图像数据


# 定义情感类别与数字标签的映射表 2分
emotion_table = {'neutral':0, 'happiness':1, 'surprise':2, 'sadness':3, 'anger':4, 'disgust':5, 'fear':6, 'contempt':7}


# 加载模型 2分
ort_session = ort.InferenceSession('emotion-ferplus.onnx')    # 使用onnxruntime创建一个会话，用于加载并运行模型


# 加载本地图片并进行预处理 2分
input_data = preprocess('img_test.png')


# 准备输入数据，确保其符合模型输入的要求
ort_inputs = {ort_session.get_inputs()[0].name: input_data}    # ort_session.get_inputs()[0].name 是获取模型的第一个输入的名字


# 运行模型，进行预测 2分
ort_outs = ort_session.run(None, ort_inputs)


# 解码模型输出，找到预测概率最高的情感类别 2分
predicted_label = np.argmax(ort_outs[0])


# 根据预测的标签找到对应的情感名称 2分
predicted_emotion = list(emotion_table.keys())[predicted_label]


# 输出预测的情感
print(f"Predicted emotion: {predicted_emotion}")
```

```
Predicted emotion: surprise
```

3.2.4

```python
import onnxruntime as ort
import numpy as np
import scipy.special
from PIL import Image


# 预处理图像
def preprocess_image(image, resize_size=256, crop_size=224, mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]):
    image = image.resize((resize_size, resize_size), Image.BILINEAR)
    w, h = image.size
    left = (w - crop_size) / 2
    top = (h - crop_size) / 2
    image = image.crop((left, top, left + crop_size, top + crop_size))
    image = np.array(image).astype(np.float32)
    image = image / 255.0
    image = (image - mean) / std
    image = np.transpose(image, (2, 0, 1))
    image = image.reshape((1,) + image.shape)
    return image


# 加载模型  2分
session = ort.InferenceSession('flower-detection.onnx')


# 加载类别标签 2分
with open('labels.txt') as f:
    labels = [line.strip() for line in f.readlines()]


# 获取模型输入和输出的名称
input_name = session.get_inputs()[0].name
output_name = session.get_outputs()[0].name
```

```python
# 加载图片  2分
image = Image.open('flower_test.png').convert('RGB')


# 预处理图片  2分
processed_image = preprocess_image(image)


# 确保输入数据是 float32 类型
processed_image = processed_image.astype(np.float32)


# 进行图片识别  2分
output = session.run([output_name], {input_name: processed_image})[0]


# 应用 softmax 函数获取识别分类后的准确率  2分
accuracy = scipy.special.softmax(output, axis=-1)


# 获取预测的类别索引
predicted_idx = np.argmax(accuracy)


# 获取预测的准确值（转换为百分比）
prob_percentage = accuracy[0, predicted_idx] * 100


# 获取预测的类别标签
predicted_label = labels[predicted_idx]


# 输出预测结果，包含百分比形式的概率
print(f"Predicted class: {predicted_label}, Accuracy: {prob_percentage:.2f}%")
```

```
Predicted class: daisy, Accuracy: 96.41%
```

3.2.5