step1:pytorch训练流程

安装库

```
pip install pytorch_quantization
```

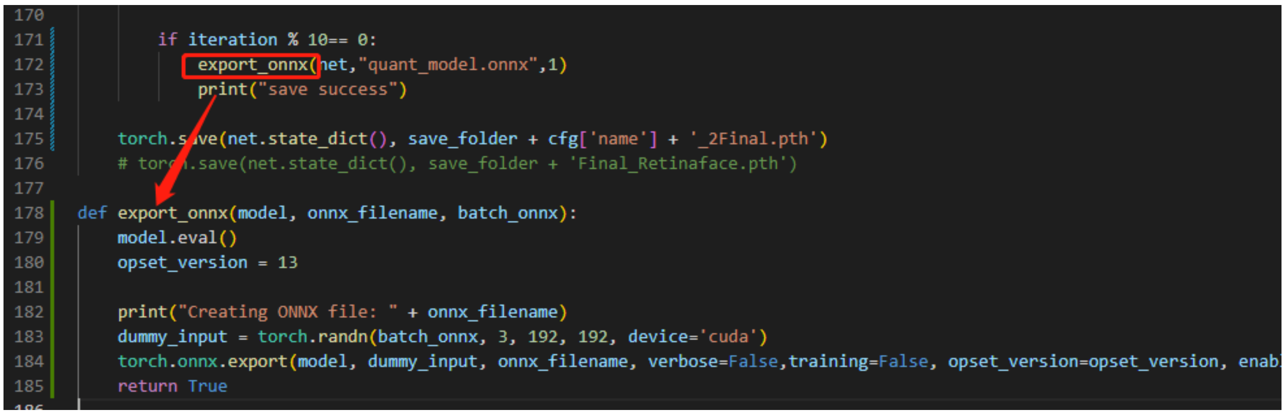
train.py中导入库

```

from pytorch_quantization import nn as quant_nn
logging.set_verbosity(logging.FATAL)  # Disable logging as they are too noisy in notebook
from pytorch_quantization import quant_modules
quant_nn.TensorQuantizer.use_fb_fake_quant = True

quant_modules.initialize()
```

模型就正常训练

step2:onnx导出

量化导出onnx文件，如下图

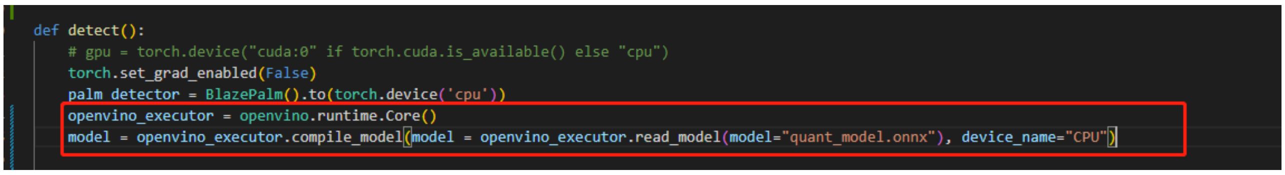


```
            if iteration % 10== 0:
                export_onnx(net,"quant_model.onnx",1)
                print("save success")

        torch.save(net.state_dict(), save_folder + cfg['name'] + '_2Final.pth')
        # torch.save(net.state_dict(), save_folder + 'Final_Retinaface.pth')

def export_onnx(model, onnx_filename, batch_onnx):
    model.eval()
    opset_version = 13

    print("Creating ONNX file: " + onnx_filename)
    dummy_input = torch.randn(batch_onnx, 3, 192, 192, device='cuda')
    torch.onnx.export(model, dummy_input, onnx_filename, verbose=False,training=False, opset_version=opset_version, enab
    return True
```

```
def export_onnx(model, onnx_filename, batch_onnx):
    model.eval()
    opset_version = 13

    dummy_input = torch.randn(batch_onnx, 3, 192, 192, device='cuda') #TODO: switch input dims by model
    torch.onnx.export(model, dummy_input, onnx_filename, verbose=False,training=False, opset_version=opset_version, enable_onnx_checker=False, do_constant_folding=False)
    return True
```

step3:openvino推理调用

网络声明



```
def detect():
    # gpu = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
    torch.set_grad_enabled(False)
    palm detector = BlazePalm().to(torch.device('cpu'))
    openvino_executor = openvino.runtime.Core()
    model = openvino_executor.compile_model(model = openvino_executor.read_model(model="quant_model.onnx"), device_name="CPU")
```

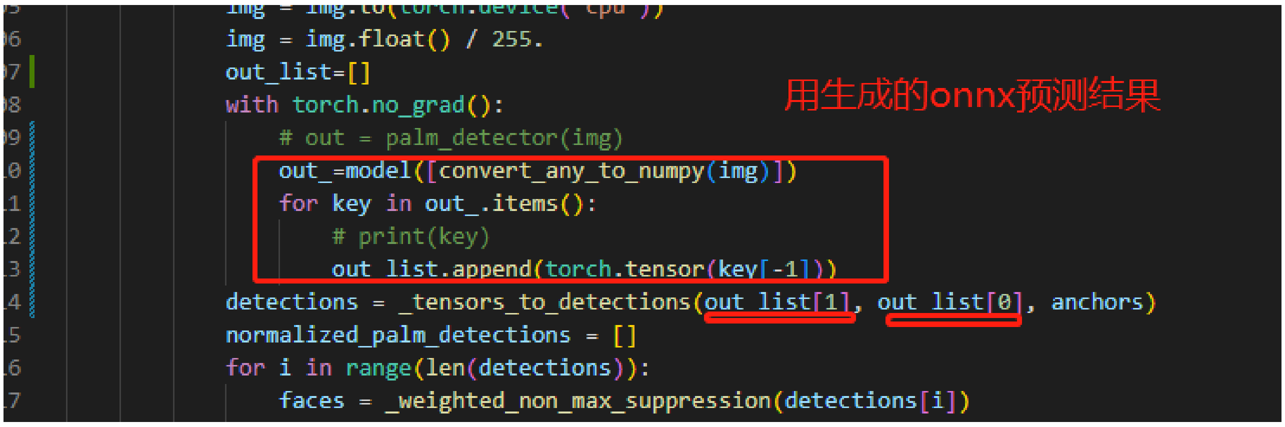
```
    import openvino.runtime
    openvino_executor = openvino.runtime.Core()
    model = openvino_executor.compile_model(model = openvino_executor.read_model(model="quant_model.onnx"), device_name="CPU")
```

加入需要使用的工具函数

```
from typing import Union
def convert_any_to_numpy(
    x: Union[torch.Tensor, np.ndarray, int, float, list, tuple],
    accept_none: bool=True) -> np.ndarray:
    if x is None and accept_none: return None
    if x is None and not accept_none: raise ValueError('Trying to convert an empty value.')
    if isinstance(x, np.ndarray): return x
    elif isinstance(x, int) or isinstance(x, float): return np.array([x, ])
    elif isinstance(x, torch.Tensor):
        if x.numel() == 0 and accept_none: return None
        if x.numel() == 0 and not accept_none: raise ValueError('Trying to convert an empty value.')
        if x.numel() >= 1: return x.detach().cpu().numpy()
    elif isinstance(x, list) or isinstance(x, tuple):
        return np.array(x)
    else:
        raise TypeError(f'input value {x}({type(x)}) can not be converted as numpy type.')
```

替换原始的model，生成结果即可

```python
out_=model([convert_any_to_numpy(img)])
                for key in out_.items():
                    # print(key)
                    out_list.append(torch.tensor(key[-1]))
```



```python
img = img.to(torch.device('cpu'))
img = img.float() / 255.
out_list=[]
with torch.no_grad():
    # out = palm_detector(img)
    out_=model([convert_any_to_numpy(img)])
    for key in out_.items():
        # print(key)
        out_list.append(torch.tensor(key[-1]))
detections = _tensors_to_detections(out_list[1], out_list[0], anchors)
normalized_palm_detections = []
for i in range(len(detections)):
    faces = _weighted_non_max_suppression(detections[i])
```