一、卷积量化推理计算：

上周先学习了师兄们之前手动进行卷积量化推理计算的代码。

1.1 输入图片：

image.shape ：（3，416，416）

img_tensor_min: 0.0

img_tensor_max: 1.0

由 $S = \frac{r_{max} - r_{min}}{q_{max} - q_{min}}$ $Z = round\left(q_{max} - \frac{r_{max}}{S}\right)$

求得：

s_img_tensor：0.00392156862745098

z_img_tensor： -0.0

由 $q = round\left(\frac{r}{S} + Z\right)$

求得：q_after_img

q_after_img.shape：(3, 416, 416)

1.2 输入卷积 conv

conv.shape：(16, 3, 3, 3)

convWithBN

将 bn 折叠进卷积核 并将卷积核量化

gamma：（16，）

moving_variance：（16，）

eps = 0.0001

beta：（16，）

moving_mean：（16，）

由
```
scaleFactor = gamma/((moving_variance+eps)**(1/2))
```

求得：scaleFactor.shape：（16，）

scaleFactor.reshape(-1) #转换为一串，去除行列

由
```
bias = [beta - scaleFactor*moving_mean]
```

求得：bias：（16，）

1.3 卷积核量化

```
for i in range(16):
    current_conv = conv[i,:,:,:]    #取出每一个卷积核
    current_scaleFactor = scaleFactor[i]
    kernelWithBN = current_conv*current_scaleFactor

    求出其 min，max，s，z
    s_kernelWithBN_tensors.append(s_kernelWithBN_tensor)
    z_kernelWithBN_tensors.append(z_kernelWithBN_tensor)#把 16 个卷积核的量化 s，z 值合并存储
```

由 $q = \text{round}\left(\frac{r}{s} + Z\right)$

求得：q_kernelWithBN_tensor

after_q_kernel.append(q_kernelWithBN_tensor) #把 16 个卷积核的量化结果合并

1.4 手动卷积

（得到 convWithBN 的卷积输出）

（1）生成单层卷积：

my_convWithBN = torch.nn.Conv2d(3,16,3,stride=1,padding=1,bias = False)

（2）权重 reshape 到(16,3,3,3)：

convWithBN_weights = torch.Tensor(after_q_kernel-z_kernelWithBN_tensors.
reshape(16,1,1,1)).reshape(16,3,3,3)

（3）将权重赋值给卷积：

my_convWithBN.weight = torch.nn.Parameter(convWithBN_weights)

（4）得到卷积输出：

output_convWithBN = my_convWithBN.forward(torch.Tensor(q_after_img-
z_img_tensor).reshape(1,3,416,416))

（5）卷积后加 bias：

after_convWithBN = output_convWithBN.reshape(1,16,416,416).detach().numpy() +
np.round(bias/(s_kernelWithBN_tensors*s_img_tensor)).reshape(1,16,1,1)

1.5 量化卷积结果

由 round($M = \frac{S_1 S_2}{S_3}$ *卷积输出+z)

得：q_result：(1, 16, 416, 416)

1.6 经过 relu

由 $r = S(q - Z)$ 得 r

my_relu = torch.nn.LeakyReLU(0.1)　　#单层 relu

r_after_relu = my_relu(r)

再量化

1.7 经过 maxpooling

由 $r = S(q - Z)$ 得 r

my_relu = torch.nn.LeakyReLU(0.1)　　#单层 relu

r_after_relu = my_relu(r)　　#relu 后的结果传给 maxpooling

my_maxpooling =torch.nn.MaxPool2d(2)

r_after_maxpooling = my_maxpooling(r_after_relu)

再量化

二、反量化进度

2.1 提取 tflite 文件中量化后权重的 q 值

```
k = 1
weight_q =  [0 for x in range(0, 13)]
for i in weight_location:
    print("conv"+str(k))
    #weight_q[k-1] = str(inter.get_tensor(after_invoke_tensor_details[i]['index']))
    weight_q[k-1] = inter.get_tensor(after_invoke_tensor_details[i]['index'])
    k += 1
```

2.2 提取量化后权重的 S 值和 Z 值

```
input_location = np.array([30, 33,36,39,42,45,48,50,52,59,61,52,65])
weight_location = np.array([4,  6, 8,10,12,14,16,18,20,22,24,26,28])
bias_location   = np.array([5,  7, 9,11,13,15,17,19,21,23,25,27,29])
output_location = np.array([31,34,37,40,43,46,49,51,53,60,62,64,66])

weight_s =  [0 for x in range(0, 13)]
weight_z =  [0 for x in range(0, 13)]
k = 1
for i in weight_location:
    print("conv"+str(k))
    #weight_s[k-1] = str(interpreter.get_tensor_details()[i]['quantization_parameters']['scales'])
    #weight_z[k-1] = str(interpreter.get_tensor_details()[i]['quantization_parameters']['zero_points'])
    weight_s[k-1] = interpreter.get_tensor_details()[i]['quantization_parameters']['scales']
    weight_z[k-1] = interpreter.get_tensor_details()[i]['quantization_parameters']['zero_points']
    k += 1
```

2.3

weight_q.shape()：(16, 3, 3, 3)　(32, 3, 3, 16) .....

weight_s.shape()：(16,)　(32,)　.....

weight_z.shape()：(16,)　(32,)　.....

目标：由 `r=S(q-Z)` 得 weight_r

对 weight_z 进行维度扩展：

```
for i in range(13):
    for j in range(3):
        weight_z[i] = np.expand_dims(weight_z[i], axis=j+1)
```

以 conv2 为例计算 weight_r，其余层同理：（数值未达到预期）

```
# conv2
weight_dr =  np.zeros((32, 3, 3, 16))
print(weight_s[1].shape)
print(weight_q[1].shape)
print(weight_z[1].shape)

for j in range (32):
    #weight_dr[0][j] = weight_s[0][j] * (weight_q[0] - weight_z[0])[j,:,:,:]
    weight_dr[j,:,:,:] = weight_s[1][j] * (weight_q[1] - weight_z[1])[j,:,:,:]
print(weight_dr.shape)
weight_r[1] = weight_dr
weight_r = np.array(weight_r)
print(weight_r.shape)
```

转置进行维度调整：

```
for i in range(0,13):
    weight_r[i] = np.transpose(weight_r[i], (1, 2, 3, 0))
    print(weight_r[i].shape)
```

替换原.h5 模型文件的权重，生成新的模型文件：

（以 conv2 为例，其余层同理）

```
# conv2
print("conv2:")
print(weight_r[1].shape)
print(f['model_weights']['conv2d_1']['conv2d_1']['kernel:0'][:].shape)
f['model_weights']['conv2d_1']['conv2d_1']['kernel:0'][:] = weight_r[1]
```

预测验证：

```python
elif predict_model=='dir_predict':
    #----------------------------------
    #拿到所有图片 通过detect image 检测
    #----------------------------------
    imgs=os.listdir(dir_img_input)
    for img_name in tqdm(imgs):
        if img_name.lower().endswith(('.bmp', '.dib', '.png', '.jpg', '.jpeg', '.pbm', '.pgm', '.ppm', '.tif', '.tiff')):
            image_path  = os.path.join(dir_img_input, img_name)
            image       = Image.open(image_path)
            r_image     = yolo.detect_image(image)
            if not os.path.exists(dir_save_path):
                os.makedirs(dir_save_path)
            r_image.save(os.path.join(dir_save_path, img_name))
```

总结：
本周对反量化的整体流程进行了一遍，但计算出的 weight_r 与原.h5 文件中的数值仍有差异，下周检查计算问题的原因。验证方面，本周只进行了可视化预测，推理计算出的模型未收敛，下周计划计算 mAP 的值。

三、多模态认知

3.1 方向总结：
（1）多模态检索：过滤模态特有的语义信息，匹配互通与模态无关的语义一致性信息
~~（2）多视图聚类-智慧医疗~~
（3）多模态智能描述与生成：融合绑定不同模态之间的语义互补信息
视频描述
跨模态多传感器数据融合
跨媒体场景理解

多模态数据任务还有一个重要的应用领域是自动化系统，包括机器人、自动驾驶等。通过多种传感器（红外摄像机、激光雷达、全球位置测定系统）收集到的异构数据用于训练并完成一系列复杂任务，包括定位和绘图，场景理解，运动规划，以及驾驶员状态识别。

3.2 方法认知：语义挖掘、绑定、解耦、推理
（1）单一模态语义挖掘：多空间蒸馏学习
（2）双模：模态内、模态间信息联合
（3）多模态语义一致性学习：（以文本为核心）多媒体数据动态增长
（4）非完整数据的语义表达：上下文自编码器、自监督学习

通用语义挖掘（语义一致性表达）
模型蒸馏技术、跨模态知识迁移
单模态：多个教师模型
多模态：模态信息解耦学习，以文本为中介

关联学习
多模态语义融通：为目标检测与识别、跨媒体描述与生成、多模态检索、多视图聚类等跨媒体技术提供技术支撑

通用泛化模型、协同计算：单一媒体的高层语义挖掘（单个网络完成多种任务）、多媒体语义一致性学习

更多模态：
Affect Computing (情感计算)：使用语音、视觉(人脸表情)、文本信息、心电、脑电等模态进行情感识别。
Medical Image：不同医疗图像模态如 CT、MRI、PET
RGB-D 模态：RGB 图与深度图

1、多模态表征学习（研究如何在不同模态数据存在异质性的情况下最好地利用其共性和特性）
包括：
（1）Joint(联合，也称为单塔结构)：
融合多个输入模态 x1,x2 获得多模态表征 xm = f(x1, ..., xn)进而使用 xm 完成某种预测任务。
网络优化目标：某种预测任务的性能。
适用多模态输入：视听语音识别，VQA，情感识别

（2）Coordinated(协作，双塔结构)：
建模多模态之间数据的相关性，将多个模态映射到写作空间，表示为：f(x1)~g(x2)，~表示协作关系。
网络的优化目标：这种协作关系(通常是相似性，即最小化 cosine 距离等度量)。
适用单一模态输入：跨模态检索

基于 Transformer 的多模态预训练模型：
Joint 结构：LXMERT，Oscar，UNITER
Coordinated 结构：CLIP，BriVL

2、跨模态翻译：（将一种模态转化为另一种模态）将源模态映射到目标模态
（1）模板法：借助词典进行翻译，词典：训练集中的数据对{(x1,y1), ... ,(xN,yN)}，给定测试样本 x^，在词典中检索最匹配的翻译结果 yi。
检索分为：单模态检索、跨模态检索
单模态检索：首先找到与 x^最相似的 xi，然后获得 xi 对应的 yi
多模态检索：直接在{y1, ... ,yN}集合中检索与 x^最相似的 yi

（2）生成式模型：抛弃词典，直接生成目标模态的数据。
1）基于语法模板：（非端到端）人为设定多个针对目标模态的语法模版，将模型的预测结果插入模版中作为翻译结果。如：图像描述：who did what to whom in a place.通过不同类型的目标/属性/场景检测器可以获得 who, what, whom, place 等具体单词，进而完成翻译。

2）编码-解码器：（端到端）首先将源模态的数据编码为隐特征 z，后续被解码器用于生成目标模态。
3）连续性生成：（端到端）针对源模态与目标模态都为流数据且在时间上严格对齐的任务。

3、跨模态对齐（寻找不同模态元素之间的直接联系）（挖掘细粒度交互，有可解释性）

挖掘多模态数据的子元素之间的关联性。

（1）显示对齐

（2）隐式对齐：基于注意力机制的对齐

4、多模态融合（输入包含多个模态）

（1）模型无关的融合策略

1）前融合：在模型的浅层(或输入层)将多个模态的特征拼接起来

2）后融合：独立训练多个模型，在预测层(最后一层)进行融合。

3）混合融合：同时结合前融合和后融合，以及在模型中间层进行特征交互。

（2）基于模型的融合策略

1）多核学习：类似 SVM。SVM 通过核函数将输入特征映射到高维空间，使线性不可分问题在高维空间可分。在处理多个输入时，多核处理多个模态特征，使得每个模态都找到其最佳核函数。

2）基于概率图模型利用隐马尔可夫模型或贝叶斯网络建模数据的联合概率分布(生成式)或条件概率(判别式)。

3）基于神经网络的融合。使用 LSTM、卷积层、注意力层、门机制、双线性融合等设计序列数据或图像数据的复杂交互。

5、协同学习（在不同模态、模态的表示预测模型之间传递知识。）

3.3 应用场景：

应用场合	表示	翻译	对齐	融合	协同学习
语音识别					
~~视听语音识别~~	√		√	√	√
事件监测					
动作分类	√			√	√
多媒体事件监测	√			√	√
情感 情绪					
情感 情绪识别	√		√	√	√
情感 情绪合成	√	√			
媒体描述					
~~图像描述~~	√	√	√		√
视频描述	√	√	√	√	√
视觉问答	√		√	√	√
媒体总结	√	√		√	
多媒体检索					
跨模态检索	√	√	√		√
跨模态哈希	√				√
多媒体生成					
(视觉)语音和声音合成	√	√			
图像和场景生成	√	√			

3.4 计划看的论文

3D 目标检测：

LiDAR R-CNN: An Efficient and Universal 3D Object Detector

https://arxiv.org/abs/2103.15297

视频实例分割：

End-to-End Video Instance Segmentation with Transformers

https://arxiv.org/abs/2011.14503

视频目标分割：

Learning to Recommend Frame for Interactive Video Object Segmentation in the Wild

https://arxiv.org/pdf/2103.10391.pdf

视频检索：

On Semantic Similarity in Video Retrieval(视频检索中的语义相似度)

https://arxiv.org/abs/2103.10095

视频动作识别：

Recognizing Actions in Videos from Unseen Viewpoints(从看不见的角度识别视频中的动作)

https://arxiv.org/abs/2103.16516

No frame left behind: Full Video Action Recognition(没有残影：完整的视频动作识别)

https://arxiv.org/abs/2103.15395

视频字幕：

VX2TEXT: End-to-End Learning of Video-Based Text Generation From Multimodal Inputs(基于视频的文本生成的端到端学习来自多模式输入)

https://arxiv.org/pdf/2101.12059.pdf

视觉推理/视觉问答(Visual Reasoning/VQA)：

Transformation Driven Visual Reasoning(转型驱动的视觉推理)

https://arxiv.org/pdf/2011.13160.pdf

场景图理解(Scene Graph Understanding)：

Bidirectional Projection Network for Cross Dimension Scene Understanding(双向投影网络，用于跨维度场景理解)

https://arxiv.org/abs/2103.14326