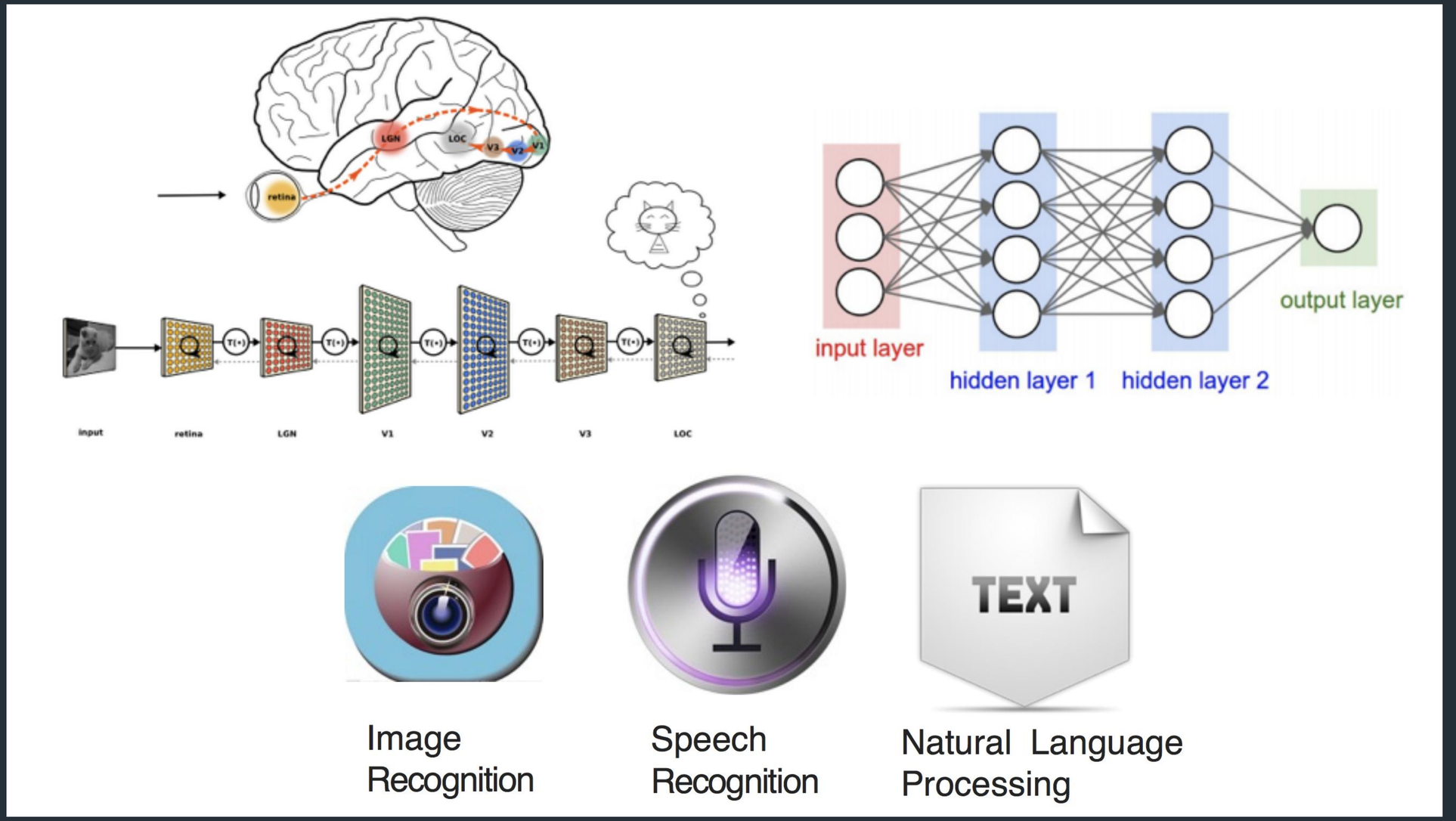
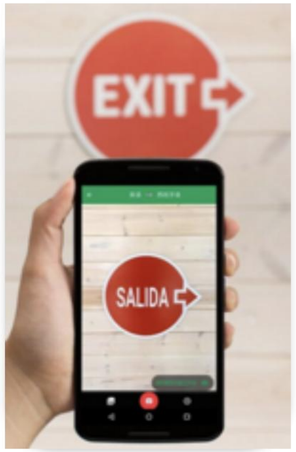
TENSORFLOW下的构建高性能神经网络模型的最佳实践

《TensorFlow技术解析与实战》作者 李嘉璇

Deep Learning: Next Wave of AI



Deep Learning的应用

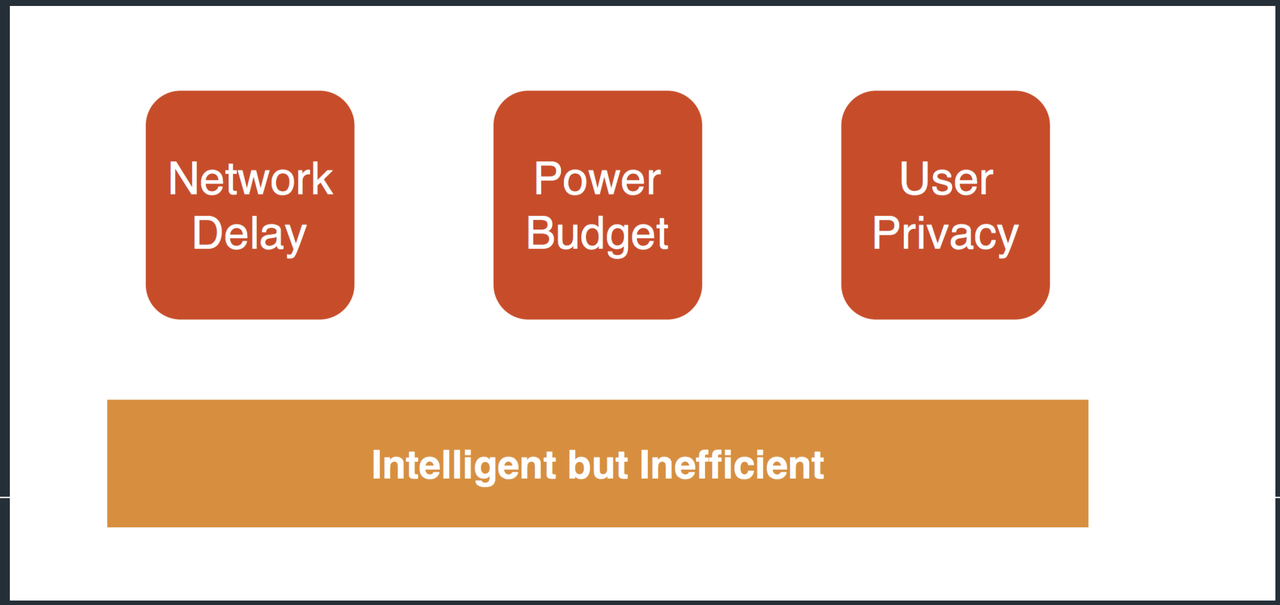


神经网络模型的运行方式及问题

- 在移动端或者嵌入式设备上应用深度学习，有两种方式：
  - 一是将模型运行在云端服务器上，向服务器发送请求，接收服务器响应；
  - 二是在本地运行模型。

- 在本地移动端运行存在的问题：
  - App developers suffers from the model size
  - Hardware engineer suffers from the model size

- 在云端服务器运行存在的问题：



如何处理及解决

- 一般来说，采用后者的方式，也就是在PC上训练好一个模型，然后将其放在移动端上进行预测。这就衍生出了很多加速计算的方向，其中重要的两个方向是对内存空间和速度的优化。
  - 一是精简模型，既可以节省内存空间，也可以加快计算速度；
  - 二是加快框架的执行速度，影响框架执行速度主要有两方面的因素，即模型的复杂度和每一步的计算速度。

- 精简模型主要是使用更低的权重精度，如量化（quantization）或权重剪枝（weight pruning）。剪枝是指减少权重的连接，把所有权值连接低于一个阈值的连接从网络里移除。

- 加速框架的执行速度一般不会影响模型的参数，是试图优化矩阵之间的通用乘法（GEMM）运算，因此会同时影响卷积层（卷积层的计算是先对数据进行im2col运算，再进行GEMM运算）和全连接层。

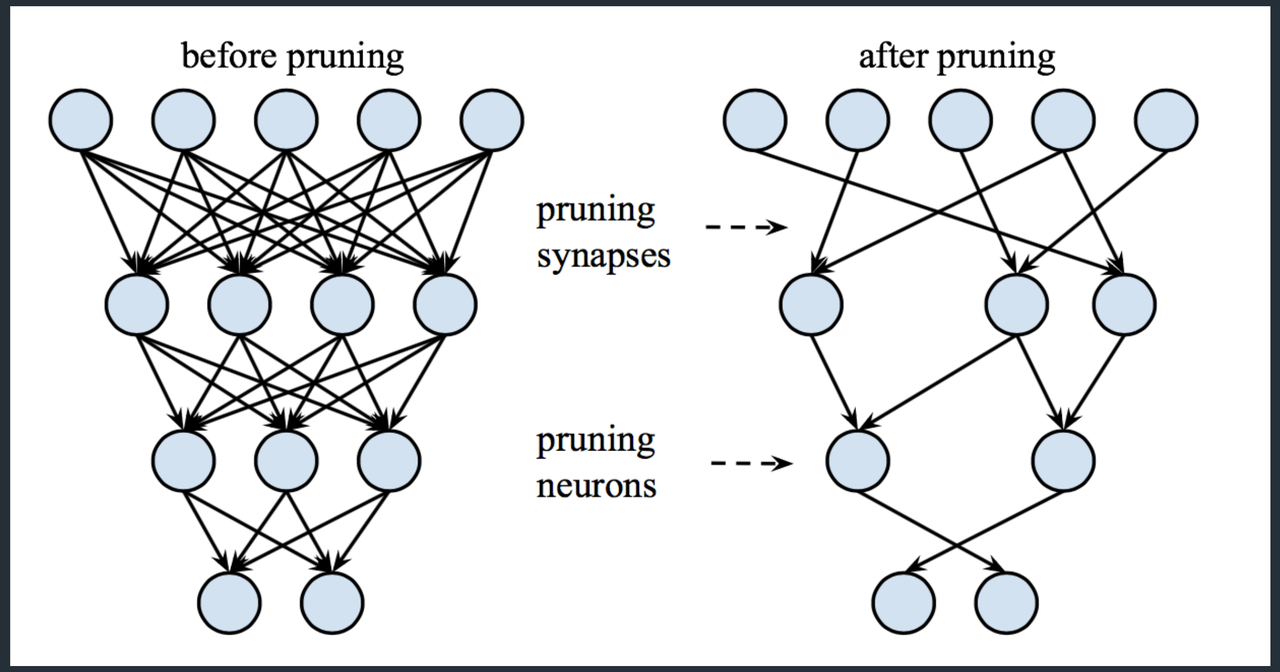
模型压缩

- 模型压缩是指在不丢失有用信息的前提下，缩减参数量以减少存储空间，提高其计算和存储效率，或按照一定的算法对数据进行重新组织，减少数据的冗余和存储的空间的一种技术方法。

- 4类方法：
  - 设计浅层网络。通过设计一个更浅的网络结构来实现和复杂模型相当的效果。但是因为浅层网络的表达能力往往很难和深层网络匹敌，因此一般用在解决简单问题上。
  - **压缩训练好的复杂模型。**采用的主要方法有 参数稀疏化（剪枝）、 参数量化表示（量化），从而达到参数量减少、计算量减少、存储减少的目的。
  - 多值网络。最为典型就是二值网络、XNOR网络。其主要原理就是采用0和1两个值对网络的输入和权重进行编码，原始网络的卷积操作可以被位运算代替。在减少模型大小的同时，极大提升了模型的计算速度。但是由于二值网络会很大程度降低模型的表达能力。因此也在研究n-bit编码方式。
  - 知识蒸馏（Knowledge Distilling）。采用迁移学习， 将复杂模型的输出做为 soft target 来训练一个简单网络。

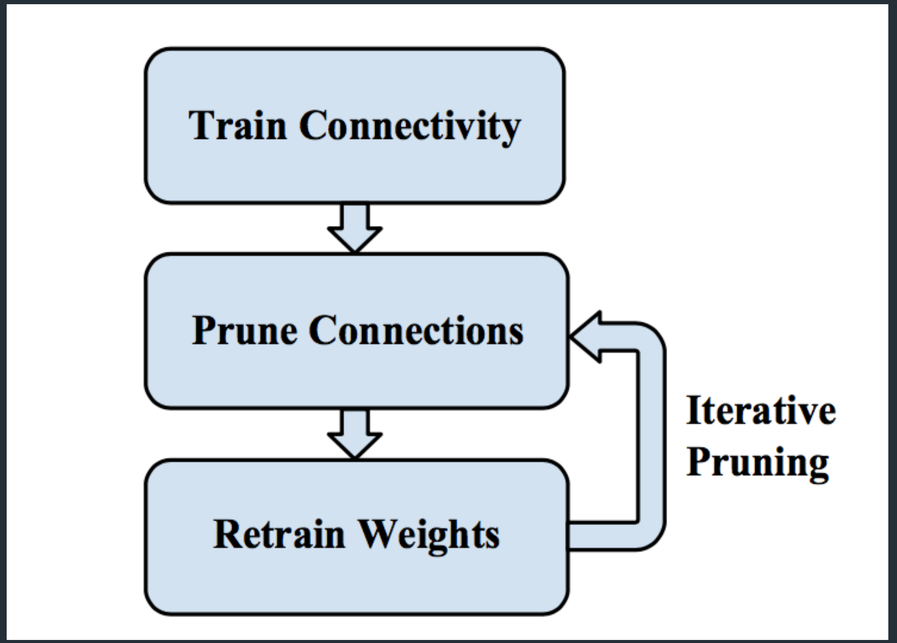
剪枝（Prunes the network）

- 剪枝就是将网络转化为稀疏网络，即大部分权值都为0，只保留一些重要的连接。



剪枝（Prunes the network）

- 事实上，我们一般是逐层对神经网络进行敏感度分析（sensitive analysis），看哪一部分权重置为0后，对精度的影响较小。然后将权重排序，设置一个置零阈值，将阈值以下的权重置零，保持这些权重不变，继续训练至模型精度恢复；反复进行上述过程，通过增大置零的阈值提高模型中被置零的比例。



剪枝（Prunes the network）

- 剪枝的特点：
  - 通用于各种网络结构与各种任务， 且实现简单， 性能稳定；
  - 稀疏网络具有更低的功耗，在CPU上具有更快的计算速度;
  - 剪枝后的稀疏矩阵通常采取特殊的存储方式，例如常用MKL中的CSR格式。

剪枝（Prunes the network）

- 剪枝的结果：
- 通过在现有的经典神经网络上做实验，发现压缩倍数在9-12倍之间。如下图所示：

Network	Top-1 Error	Top-5 Error	Parameters	Pruning Rate
LeNet-300-100	1.64%	-	267K	
LeNet-300-100 Pruned	1.59%	-	**22K**	**12**×
LeNet-5	0.80%	-	431K	
LeNet-5 Pruned	0.77%	-	**36K**	**12**×
AlexNet	42.78%	19.73%	61M	
AlexNet Pruned	42.77%	19.67%	**6.7M**	**9**×
VGG-16	31.50%	11.32%	138M	
VGG-16 Pruned	31.34%	10.88%	**10.3M**	**13**×
GoogleNet	31.14%	10.96%	7.0M	
GoogleNet Pruned	31.04%	10.88%	**2.0M**	**3.5**×
SqueezeNet	42.56%	19.52%	1.2M	
SqueezeNet Pruned	42.26%	19.34%	**0.38M**	**3.2**×
ResNet-50	23.85%	7.13%	25.5M	
ResNet-50 Pruned	23.65%	6.85%	**7.47M**	**3.4**×

- 压缩的多是全连接层，CNN层参数少，因此能压缩的倍数也较少
-
- 根据经验，压缩率到60%以上，模型大小才会下降比较多

量化（Quantize the weights）

- 量化（Quantization）又称定点，是用更少的数据位宽进行神经网络存储和计算。它的优势在于能够节省存储，并进行更快地访存和计算。

- 神经网络训练时要求速度和准确率，训练通常在GPU上进行，所以使用浮点数影响不大。但是在预测阶段，使用浮点数会影响速度。量化可以在加快速度的同时，保持较高的精度。

量化（Quantize the weights）

- 量化网络的动机主要有两个：
  - 减小模型文件的大小。
    - 模型文件往往占据很大的磁盘空间。很多模型都接近200 MB，模型中存储的是分布在大量层中的权值。在存储模型的时候用8位整数，模型大小可以缩小为原来32位的25%左右。
  - 降低预测过程需要的计算资源。
    - 这在嵌入式和移动端非常有意义，能够更快地运行模型，功耗更低。
      - 从体系架构的角度来说，8位的访问次数要比32位多，在读取8位整数时只需要32位浮点数的1/4的内存带宽，例如，在32位内存带宽的情况下，8位整数可以一次访问4个，32位浮点数只能1次访问1个。而且使用SIMD指令，可以在一个时钟周期里实现更多的计算。
      - 另一方面，8位对嵌入式设备的利用更充分，因为很多嵌入式芯片都是8位、16位的，如单片机、数字信号处理器（DSP芯片），8位可以充分利用这些。

量化（Quantize the weights）

- 思考：那能否用低精度格式来直接训练呢？

量化（Quantize the weights）

- 不同精度（FP32、FP16、INT8）表示的数据范围：

	Dynamic Range	Min Positive Value
FP32	$-3.4 \times 10^{38} \sim +3.4 \times 10^{38}$	1.4×10^{-45}
FP16	-65504 ~ +65504	5.96×10^{-8}
INT8	-128 ~ +127	1

TensorFlow下的模型压缩工具

- **量化示例：**
- 我们举个将GoogleNet模型转换成8位模型的例子，看看模型的大小减小多少，以及用它预测的结果怎么样。
- 从官方网站上下载训练好的GoogleNet模型，解压后，放在/tmp目录下，然后执行：

- bazel build tensorflow/tools/quantization:quantize_graph
- bazel-bin/tensorflow/tools/quantization/quantize_graph \
- --input=/tmp/classify_image_graph_def.pb \
- --output_node_names="softmax" --output=/tmp/quantized_graph.pb \
- --mode=eightbit
-

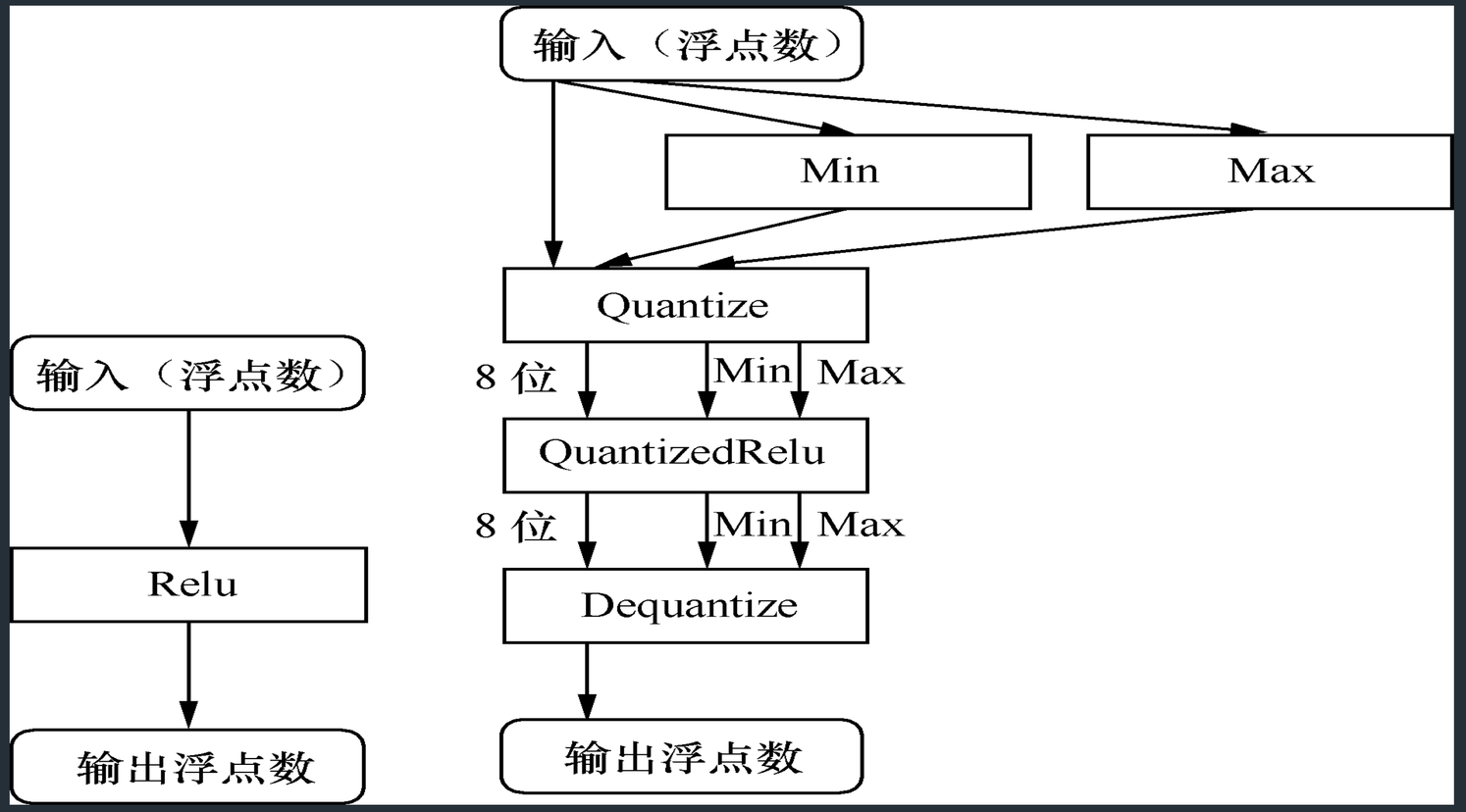
TensorFlow下的模型压缩工具

- 生成量化后的模型quantized_graph.pb大小只有23 MB，是原来模型classify_image_graph_ def.pb（91 MB）的1/4。它的预测效果怎么样呢？执行：
- bazel build tensorflow/examples/label_image:label_image
- bazel-bin/tensorflow/examples/label_image/label_image \
- --image=/tmp/cropped_panda.jpg \
- --graph=/tmp/quantized_graph.pb \
- --labels=/tmp/imagenet_synset_to_human_label_map.txt \
- --input_width=299 \
- --input_height=299 \
- --input_mean=128 \
- --input_std=128 \
- --input_layer="Mul:0" \
- --output_layer="softmax:0"
-
- 运行结果如下图所示，可以看出8位模型预测的结果也很好。

```
2017-04-16 21:34:26.337147: I tensorflow/examples/label_image/main.cc:206] n00483508    doubles (169): 0.891074
2017-04-16 21:34:26.337170: I tensorflow/examples/label_image/main.cc:206] n00449796    hydroplane racing (75): 0.00779054
2017-04-16 21:34:26.337176: I tensorflow/examples/label_image/main.cc:206] n00021265    food, nutrient (7): 0.00295912
2017-04-16 21:34:26.337181: I tensorflow/examples/label_image/main.cc:206] n01405616    bladderwrack, Ascophyllum nodosum (325): 0.00146577
2017-04-16 21:34:26.337186: I tensorflow/examples/label_image/main.cc:206] n01622779    great grey owl, great gray owl, Strix nebulosa (878): 0.00117424
```

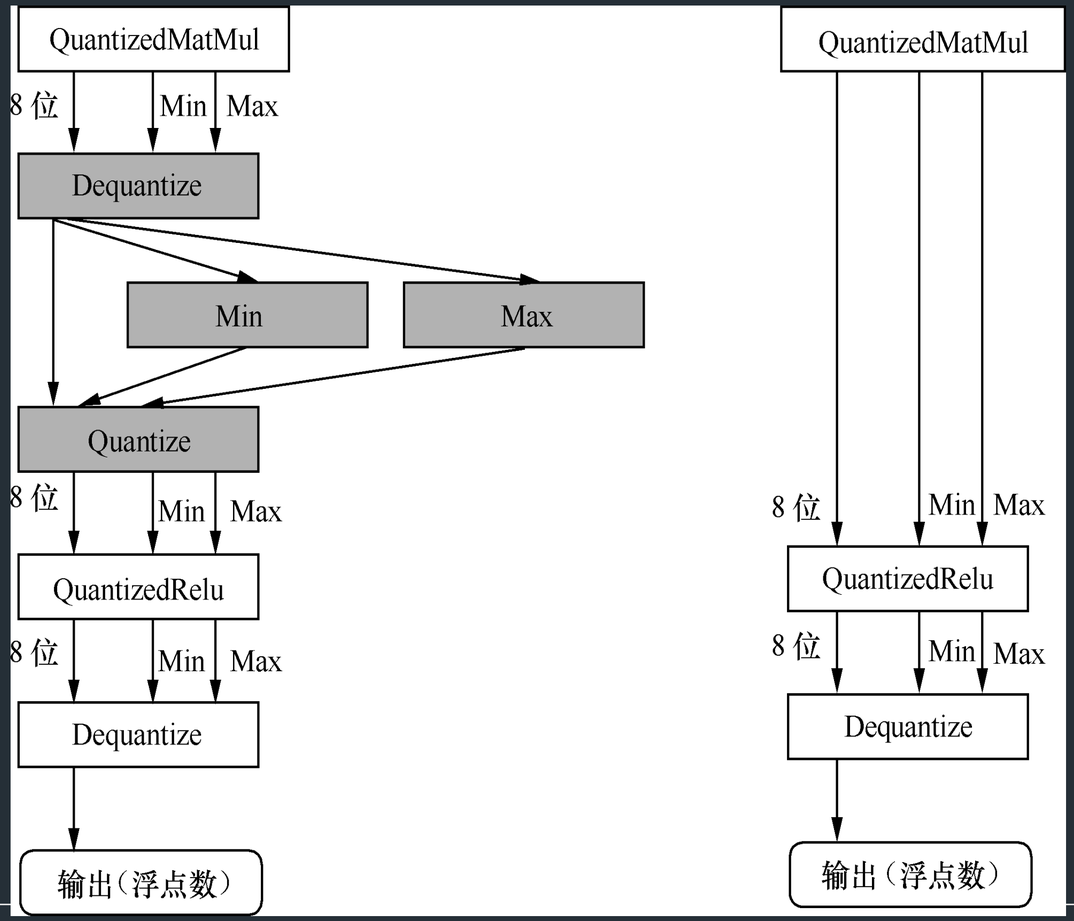
TensorFlow下量化过程的实现

- TensorFlow的量化是通过将预测的操作转换成等价的8位版本的操作来实现的。



TensorFlow下量化过程的实现

- 实际上，我们会在每个量化操作（如QuantizedMatMul、QuantizedRelu等）的后面执行反量化操作（Dequantize），如下图左侧所示在QuantizedMatMul后执行反量化和量化操作可以相互抵消。因此，如下图右侧所示，在输出层之前做一次反量化操作就可以了。



量化数据的表示

- 将浮点数转换为8位的表示实际上是一个压缩问题。实际上，权重和经过激活函数处理过的上一层的输出（也就是下一层的输入）实际上是分布在一个范围内的值。量化的过程一般是找出最大值和最小值后，将分布在其中的浮点数认为是线性分布，做线性扩展。因此，假设最小值是-10.0f，最大值是30.0f，那量化后的结果如下图所示。

量化后的值	原始的浮点数
0	-10.0
255	30.0
128	10.0

经典神经网络ResNet50上的模型压缩实验

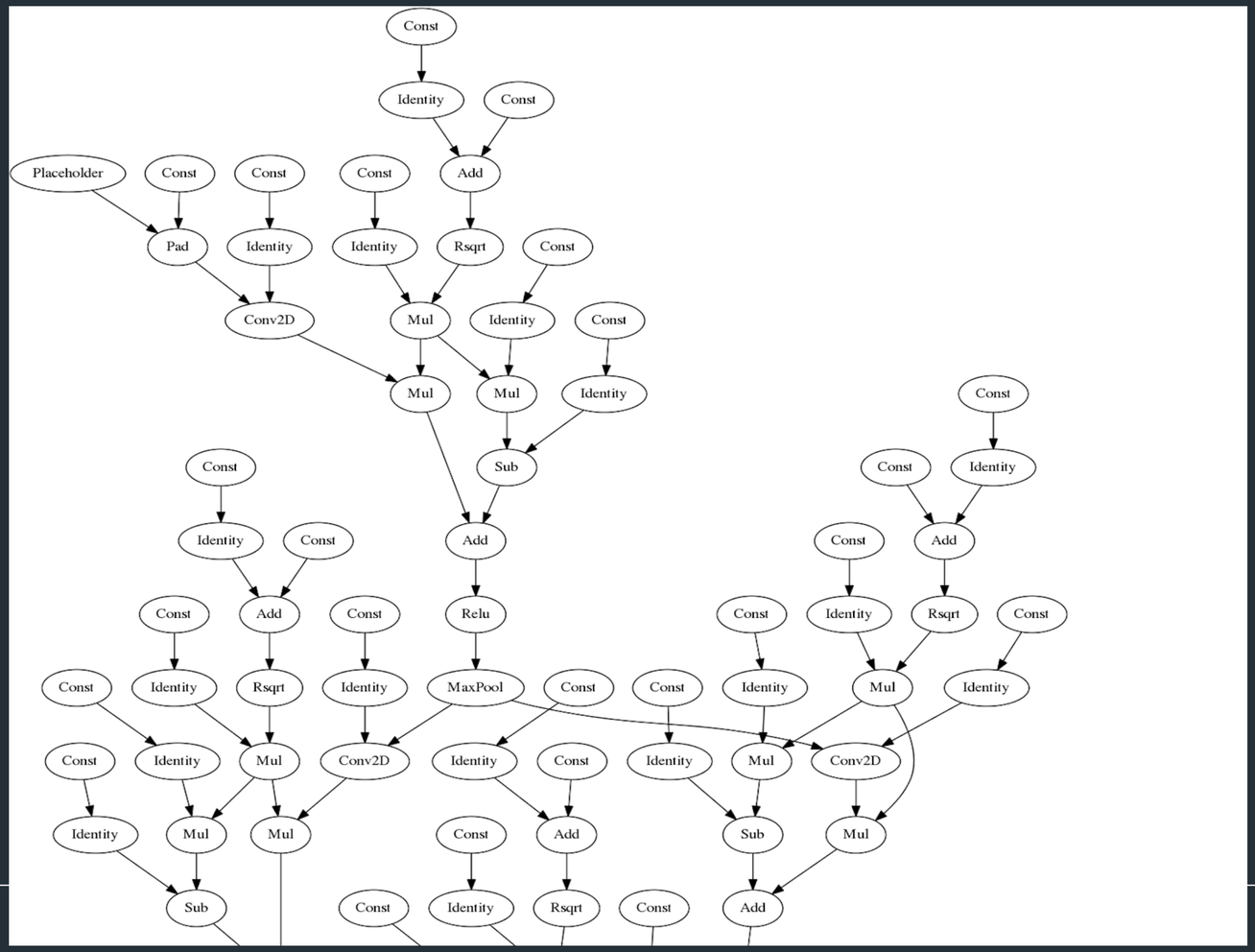
- 在ResNet50-v1上，采用官方GitHub上提供的模型作为Baseline，在ImageNet测试集5万张图片上进行测试。

	Baseline			均匀量化					
网络模型	Baseline			8bit		8bit（存储Range-10张图片）		8bit（存储Range-1000张图片）	
	Top1	Recall_5	文件大小	Top1	文件大小	Top1	文件大小	Top1	文件大小
ResNet-50-v1	74.768	92.074	98M	72.8	25M	72.9	25M	73.1	25M

- 在均匀量化的过程中，首先是仅仅对权重进行量化，得到精度为72.8%。随后，分别用模型对测试集的10张、1000张图片的范围进行提前计算最值（Max和Min），并进行存储，得到的精度分别为72.9%和73.1%。

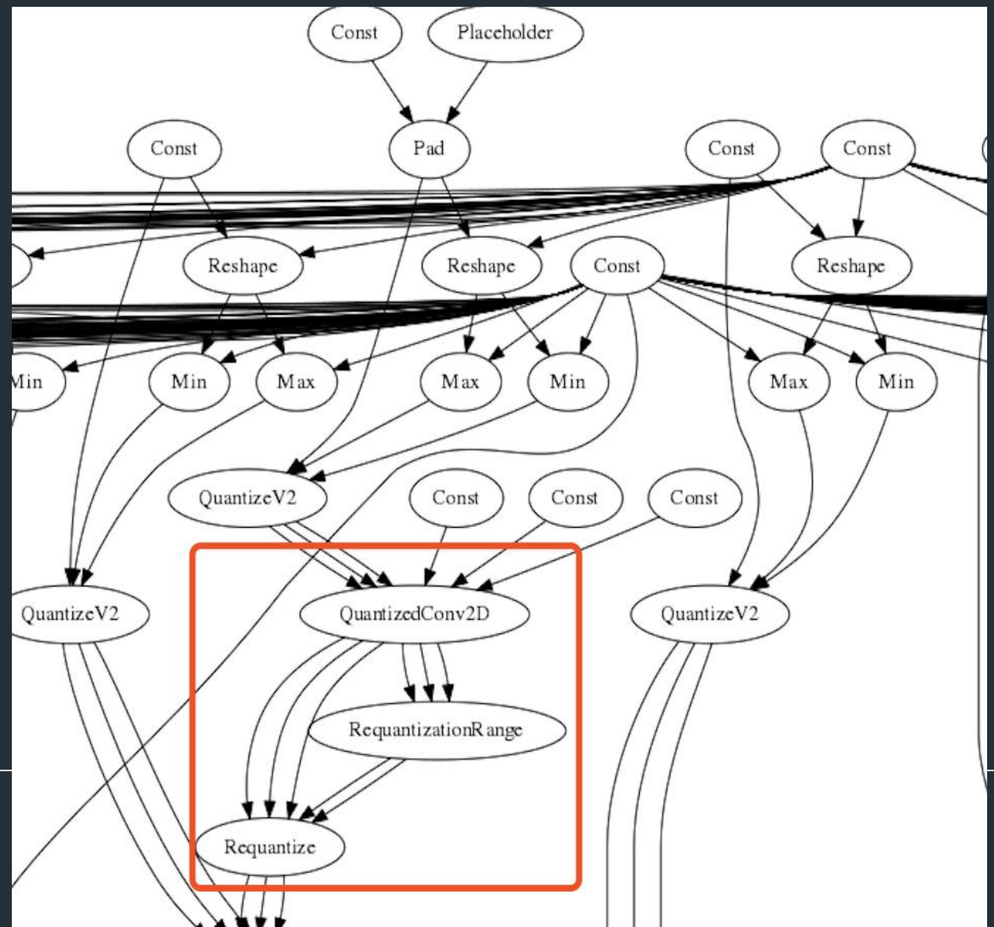
经典神经网络ResNet50上的模型压缩实验

- 从量化前后的可视化的模型对比，也可以看成量化对模型做了哪些操作。下图是未经量化的原始模型：



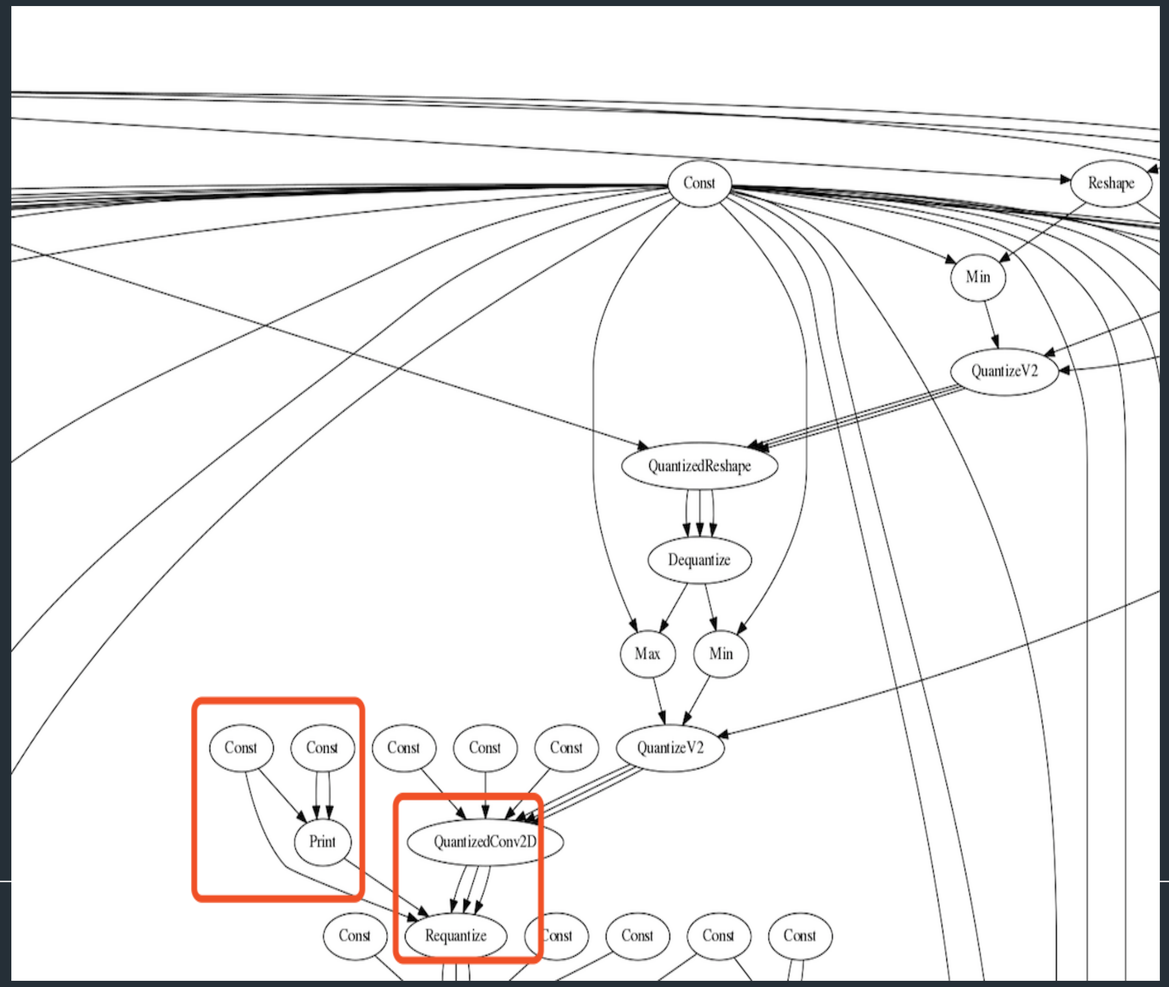
经典神经网络ResNet50上的模型压缩实验

- 下图是仅仅对权重进行量化，没有计算输入图片的最值范围的可视化模型。可以看出原本的Conv2D等节点都转换为QuantizedConv2D的对应节点。并且在进行QuantizedConv2D操作后，得到INT32类型的结果，需要对操作的结果转换为8位（ReQuantize操作），而转换的过程需要知道INT32结果的最值范围，因此也加入了ReQuantizationRange节点。



经典神经网络ResNet50上的模型压缩实验

- 如果已经预先使用10张或者1000张图片计算了每一个Conv2D等操作之后需要计算的范围，则ReQuantizationRange的计算过程就可以省去，直接从存储的计算好最值文件中读取。



优化神经网络模型的其他建议

- 1. 设计小模型
-
- 可以将模型大小做为约束，在模型结构设计和选择时便加以考虑。例如，对于全连接，使用 bottleneck 是一个有效的手段。
- 例如，我们使用TensorFlow官方网站提供的预训练好的Inception V3模型在此花卉数据集上进行训练。在项目根目录下执行：
- python tensorflow/examples/image_retraining/retrain.py \
- --bottleneck_dir=/tmp/bottlenecks/ \
- --how_many_training_steps 10 \
- --model_dir=/tmp/inception \
- --output_graph=/tmp/retrained_graph.pb \
- --output_labels=/tmp/retrained_labels.txt \
- --image_dir /tmp/flower_photos

优化神经网络模型的其他建议

- 再如，Highway，ResNet，DenseNet 这些带有 skip connection 结构的模型，也可以用来作为设计窄而深网络的参考，从而减少模型整体参数量和计算量。

- 还如，SqueezeNet 网络结构中通过引入1 x 1的小卷积核、减少 feature map 数量等方法，最终将模型大小压缩在 1M 以内，分类精度与 AlexNet 相当，而模型大小仅是 AlexNet的1/50。

优化神经网络模型的其他建议

- 2.模型小型化
-
- 一般采用知识蒸馏。

- 蒸馏模型是采用是迁移学习，通过采用预先训练好的复杂模型（Teacher model）的输出作为监督信号去训练另外一个简单的网络，得到的简单的网络称之为Student model。

- 实验表明，蒸馏模型的方法在 MNIST 及声学建模等任务上都有着很好的表现。

总结

- 随着深度学习模型在嵌入式端的应用越来越丰富，例如安防、工业物联网、智能机器人等设备，需要解决图像、语音场景下深度学习的加速问题，减小模型大小及计算量，构建高性能神经网络模型。

- 重点讲解模型压缩和剪枝方法带来的模型大小和计算量的下降，并且能使精度维持在较高水平。

- 除此之外，剪枝的敏感度分析和重新训练（Retrain）也有很多不同的手段；量化也可以在更低精度（5bit、6bit、甚至二值网络）上尝试，期待和大家一起探讨。

谢谢