**模型压缩的方式**

1. 网络剪枝（Network Pruning）
2. 知识蒸馏（Knowledge Distillation）
3. 参数量化（Parameter Quantization）
4. 架构设计（Architecture Design）
5. 动态计算（Dynamic Computation）

**蒸馏方法**（distillation）也叫teacher-student learning, 是一种压缩大模型的方法，核心思想就是训练小模型（small model）也就是学生（student）复现大模型（large model）也就是老师（teacher）学到的暗知识（dark knowledge）。

在监督学习领域，对于一个分类问题，定义soft label为模型的输出(即不同label的概率)， hard label为最终正确的label(也就是ground truth)，通常是通过最大化正确label的概率来进行学习的，通常采用 cross-entropy作为损失函数，即让正确label的概率尽可能预测为1，其余label的概率趋近于0，但是这些不正确趋近于0的label也是有大有小的(比把图片数字2识别成3的概率还是要比识别成9大，尽管他们都趋近于0)，这被称为"暗知识(Dark Knowledge)", 这也反应了模型的泛化能力。但因为过于趋紧0不利于student模型学习，为了让student也容易学习tearcher的输出，引入了带温度T的softmax概率为

文本

描述已自动生成

训练的时候T>1, 方便学到类间信息；预测的时候T=1，恢复到标准的softmax进行计算。T越大，输出的概率约平滑。

蒸馏损失（distillation loss）：

手机屏幕截图

描述已自动生成

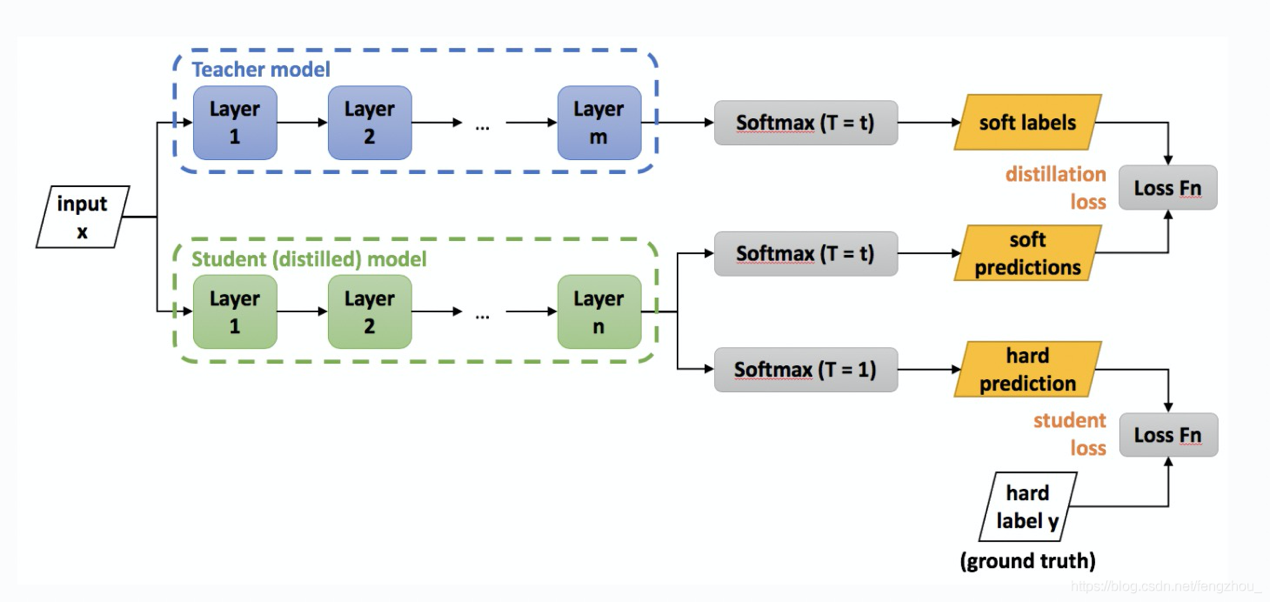
最终的损失函数是以下三种损失函数的线性组合：软标签的交叉熵的加权系数越大，表明学习到老师模型的知识越多，知识蒸馏的效果越好，蒸馏模型的泛化能力越强。

distillation loss（）：teacher网络softmax层输出的概率分布和student网络softmax层输出的概率分布的交叉熵；

masked language modeling（）以及：student网络softmax层输出的概率分布和真实的one-hot标签的交叉熵；

cosine embedding loss（）：student网络隐层输出和teacher网络隐层输出的余弦相似度值。

具体模型的训练方式如图：



Loss Fn为cross entropy，最终的损失函数为图中两个loss的线性组合。

DistilBERT: a distilled version of BERT

Student architecture：和BERT类似，只是layer的数量减半

Student initialization：因为Student模型和Teacher模型每层的layer一样，因此每两层保留一层，利用相关的参数

Distillation：采用了RoBERTa的优化策略，动态mask，增大batch size，取消NSP任务的损失函数。

另外作者在这里移除了句子向量和pooler层，在这里也没有看到NSP任务的损失函数，因此个人认为作者也去除了NSP任务（主要是很多人证明该任务并没有什么效果）。

DistilBERT利用知识蒸馏的技术，达到不过分降低性能的情况下，减少模型大小和inference速度。DistilBERT减少了BERT40%的参数，保留了BERT 97%的计算能力，但是提高了60%的速度。

**Albert**：

ALBERT主要是从模型架构上做的改变，能极大的减小模型大小，但是没有提升推断速度。ALBERT主要做了三点改变：

Factorized embedding parameterization ：在这里V为vocab size（通常比较大，bert中就是20000多），因此当E很大时，这里的参数就非常多，又基于上面的分析，E可以不用设这么大，因此作者在这里做了一个矩阵分解，将矩阵V x H(E)分解为两个小的矩阵V x E，E x H，E << H。在这里不再将E=H，而是将E设置为一个远小于H的值，然后再经过一个矩阵E x H将词向量维度映射到H。

Cross-layer parameter sharing ：在这里作者共享了encoder中所有的参数（包括self attention和FFN）

Inter-sentence coherence loss 。作者认为在bert中的NSP任务实际上是有问题的，当初设计这个任务时，是希望能学到句子一致性的，但事实上由于当初在构造句子对负例时，是从不同的document中选择的句子，因此网络并不需要学到句子的一致性，而只要学到句子的主题（因为不同的document，主题很可能不一样），就可以判断两个句子是否是负例，因此作者认为NSP任务只学到了句子的主题，而主题分类是一个浅层语义的NLP任务。

　　因此在这里作者提出了一种新的任务SOP（句子顺序预测），在这里构造正例和bert中一样，而负例就是将正例中两个句子的顺序颠倒，这里应为负例中的句子都来自于同一文档，所以通过句子的主题是无法区分正例和负例的，需要理解句子的深层语义才能区分。学习SOP（86.5）任务，也能在NSP（78.9）任务上取得不错的效果，但是学习NSP（90.5）任务，是无法在SOP（52.0）任务上取得效果的。

**TinyBERT：**

TinyBERT主要做了以下两点创新：

1. 提供一种新的针对 transformer-based 模型进行蒸馏的方法，使得BERT中具有的语言知识可以迁移到TinyBERT中去。

TinyBERT的distillation分为三部分：

* Transformer-layer Distillation.

MHA（multi-head attention）层:

文本

描述已自动生成

文本

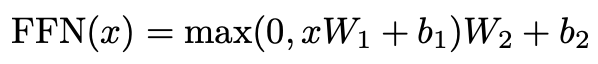
中度可信度描述已自动生成

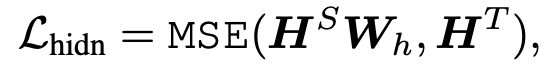
图示

低可信度描述已自动生成

h为head，attention matrix为标准化的，而不是soft Max后的，因为这样的收敛速度更快，性能更好。

FFN（fully connected feed-forward network）:





文本

描述已自动生成

* Embedding-layer Distillation.

文本

中度可信度描述已自动生成

* Prediction-layer Distillation.

卡通人物

描述已自动生成

整体可以表示为：

文本, 信件

描述已自动生成

文本, 信件

描述已自动生成

文本, 信件

描述已自动生成

图示

描述已自动生成

1. 提出一个两阶段学习框架，在预训练阶段和fine-tuning阶段都进行蒸馏，确保TinyBERT可以充分的从BERT中学习到一般领域和特定任务两部分的知识。

图示

描述已自动生成

知识蒸馏的过程也被分为两个阶段， General distillation (Pre-training阶段)，使用大规模无监督的数据， 帮助student网络TinyBERT学习到尚未微调的teacher BERT中的知识，有利于提高TinyBERT的泛化能力。此时，由于词向量维度的减小，隐层神经元的减少，以及网络层数的减少，tinybert的表现远不如teacher bert。（注意：在Pre-training阶段，蒸馏的特征表示未使用预测层输出）

task-specific distillation(Fine-tuning阶段)，使用具体任务的数据，帮助TinyBERT学习到更多任务相关的具体知识。值得注意的是，在Fine-tuning阶段，论文采用了数据增强的策略，从后面的实验中可以看出，数据增强起到了很重要的作用。 数据扩充的过程如下：对于特定任务的数据中每一条文本，首先使用bert自带的方式进行bpe分词，bpe分词之后是完整单词（single-piece word），用[MASK]符号代替，然后使用bert进行预测并选择其对应的候选词N个；如果bpe分词之后不是完整单词，则使用Glove词向量以及余弦相似度来选择对应的N个候选词，最后以概率pt选择是否替换这个单词，从而产生更多的文本数据。

文本

描述已自动生成

实验设置

TinyBERT(参数个数14.5M):

网络层数M = 4，隐层维度，FFN层维度，h=12；

Techer BERT(参数个数109M):

网络层数N= 12, 隐层维度d = 768, FFN层维度，h=12；

映射函数： ；

每层的重要性参数设置为1。

表格

描述已自动生成