

王思文的知识库

自然语言处理

Transformer

从word2vec(数学上)到Glove到ELMo

Embedding的发展历程

Style Transformer

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention

DA-Transformer (将加权距离map到re-scale的系数中)

Self-Attention with Relative Position Representations (利用edge信息)

Transformer的位置编码(小结)

Transformer改进

REFORMER (三个技巧减少内存占用)

Star-Transformer

Transformer-XL(当前最优自回归模型)->XLNET

BPT: BP-Transformer

Multi-Scale Self-Attention

Low-Rank and Locality Constrained Self-Attention

External Attention(Attention(x)=Linear(Norm(Linear(x))))&Do You Even Need Attention?

BPE解决OOV (Out-Of-Vocabulary) 问题

BPE的改进算法BPEmb

mosesdecoder数据预处理

NLP机器翻译评价度量 (BLEU,ROUGE,METEOR,ROUGE)

LSTM结合知识蒸馏

BERT的发展

ELECTRA (MLM改为replaced token detection)

ALBERT (NSP换成SOP)

Distillation

BERT的模型蒸馏 (一)

BERT的模型蒸馏 (二)

BERT-TNF(对Rare word提出notes想法)

预训练模型

预训练模型---- ELMO&GPT&Bert

预训练模型---- Transformer-XL&GPT2&XLNet

预训练模型---- RoBERTa

预训练模型---- XLNet (用自回归本身的特点克服 BERT 的缺点)

预训练模型---- ALBert (减少参数量减少内存)

预训练模型---- BART: (融合BERT和GPT)

预训练模型---- SpanBERT&ERNIE2

预训练模型---- MASS

预训练模型---- ELECTRA

预训练模型---- T5: Text-To-Text Transfer Transformer

预训练模型---- 小结

预训练模型---- 华为2021newest

预训练模型---- NEZHA

预训练模型---- PanGu

预训练模型---- TinyBERT

深度学习

机器学习

机器学习的可行性

MLE 与 MAP

ROC 曲线与 PR 曲线

梯度裁剪及其作用

归纳偏差

梯度下降、牛顿法凸优化、L1、L2正则化、softmax、Batchnorm、dropout、Targeted Dropout

优化器

梯度弥散 (BN和shattered gradient)

Loss Function总结

深度学习中的Normalization模型

Batch Normalization

BN和Dropout一起使用并不会 $1+1>2$

CNN的10个思考

Distributional Representation和Distributed Representation

MLP-Mixer(LeCun说MLP-Mixer是一个挂羊头卖狗肉的算法)&RepMLP(CNN和MLP一起训)

无监督模型

生成模型的理论分析

流模型

自监督学习

去噪自编码器

对比学习

Domain Adaptation

Moment Matching for Multi-Source Domain Adaptation

Multimodel

Multimodel Neural Language Models

Deep Visual-Semantic Alignments for Generating Image Descriptions

UniT: Multimodal Multitask Learning with a Unified Transformer

算法

RE

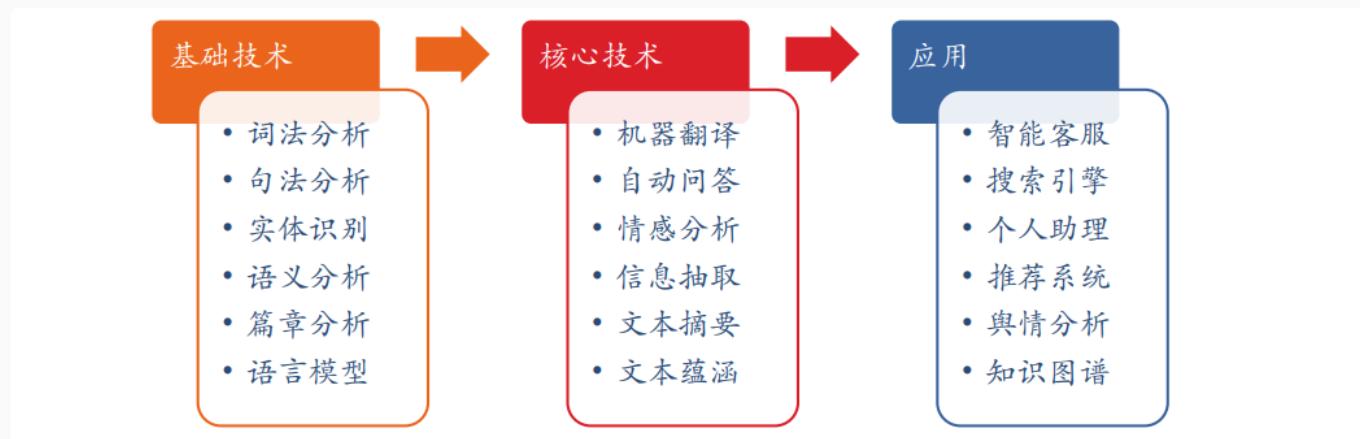
自然语言处理

参考

发展历程



WHAT



Difference



文本与图像信息的差异

	输入量	信息量	关系	底层特征
图像	二维像素集 200X200	黑白: 128-256 彩色: 3 (128-256)	欧氏空间	纹理, 形状 彩色
文本	一维离散词符号序列 几千-几万个词	共250K (英文词类) 一般用几千个词	语法关系 句法关系 语义关系	句子长度, 句子在段落中的位置, 段落在文章中的位置, ..

CCL 2016 中科院张钹院士《后深度学习时代的计算语言学》

Transformer

原文: <https://arxiv.org/pdf/1706.03762v5.pdf>

代码: <https://paperswithcode.com/paper/attention-is-all-you-need>

从word2vec(数学上)到Glove到ELMo

概述

首要问题就是如何使用向量表示词？词向量的获取方式可以大体分为两类：一类是基于统计方法（例如：基于共现矩阵、SVD），另一种是基于语言模型的。

时间线

2013年，Tomas Mikolov等人提出word2vec；

2014年，Jeffrey Pennington, Richard Socher, Christopher D. Manning三人提出了Glove算法

2018年，3月份ELMo出世

基于统计方法

参考：https://blog.csdn.net/weixin_37947156/article/details/83146141?spm=1001.2014.3001.5501

矩阵定义的词向量在一定程度上缓解了one-hot向量相似度为0的问题，但没有解决数据稀疏性和维度灾难的问题。

SVD得到了word的稠密（dense）矩阵，该矩阵具有很多良好的性质：语义相近的词在向量空间相近，甚至可以一定程度反映word间的线性关系，但这样的传统做法有很多问题：

1. 由于很多词没有出现，导致矩阵极其稀疏，因此需要对词频做额外处理来达到好的矩阵分解效果；
2. 矩阵非常大，维度太高
3. 需要手动去掉停用词（如although, a,...），不然这些频繁出现的词也会影响矩阵分解的效果。

基于语言模型

语言模型旨在为语句的联合概率函数 $P(w_1, \dots, w_T)$ 建模，其中 w_i 表示句子中的第*i*个词。语言模型的目标是，希望模型对有意义的句子赋予大概率，对没意义的句子赋予小概率。这样的模型可以应用于很多领域，如机器翻译、语音识别、信息检索、词性标注、手写识别等，它们都希望能得到一个连续序列的概率。

2013年，Google团队发表了word2vec工具。word2vec工具主要包含两个模型：跳字模型（skip-gram）和连续词袋模型（continuous bag of words，简称CBOW），当词典较大时，例如几十万到上百万，这种训练方法的计算开销会较大。因此，我们将使用近似的方法来计算这些梯度，从而减小计算开销。常用的近似训练法包括负采样（negative sampling）和层序

softmax (hierarchical softmax)。word2vec的词向量可以较好地表达不同词之间的相似和类比关系。

§5 基于 Negative Sampling 的模型

本节将介绍基于 Negative Sampling 的 CBOW 和 Skip-gram 模型. Negative Sampling (简称为 NEG) 是 Tomas Mikolov 等人在文 [4] 中提出的, 它是 NCE (Noise Contrastive Estimation) 的一个简化版本, 目的是用来提高训练速度并改善所得词向量的质量. 与 Hierarchical Softmax 相比, NEG 不再使用使用 (复杂的) Huffman 树, 而是利用 (相对简单的) 随机负采样, 能大幅度提高性能, 因而可作为 Hierarchical Softmax 的一种替代.

ELMO

但是word2vec无法处理多义性问题, ELMo是双向语言模型biLM的多层表示的组合, 基于大量文本, ELMo模型是从深层的双向语言模型 (deep bidirectional language model) 中的内部状态 (internal state)学习而来的, 而这些词向量很容易加入到QA、文本对齐、文本分类等模型中。

数学上word2vec分析

📎 [word2vec_中的数学原理详解.pdf](#)

什么是Glove?

参考: <https://www.biaodianfu.com/glove.html>

GloVe的全称叫Global Vectors for Word Representation, 它是一个基于全局词频统计 (count-based & overall statistics) 的词表征 (word representation) 工具。

Glove与LSA的区别

LSA (Latent Semantic Analysis) 是一种比较早的count-based的词向量表征工具, 它也是基于co-occurrence matrix的, 只不过采用了基于奇异值分解 (SVD) 的矩阵分解技术对大矩阵进行降维, 而我们知道SVD的复杂度是很高的, 所以它的计算代价比较大。还有一点是它对所有单词的统计权重都是一致的。而这些缺点在GloVe中被一一克服了。

GloVe与word2vec的区别

GloVe与word2vec, 两个模型都可以根据词汇的“共现co-occurrence”信息, 将词汇编码成一个向量 (所谓共现, 即语料中词汇一块出现的频率)。两者最直观的区别在于, word2vec

是“predictive”的模型，而GloVe是“count-based”的模型。

- **Predictive的模型**，如Word2vec，根据context预测中间的词汇，要么根据中间的词汇预测context，分别对应了word2vec的两种训练方式cbow和skip-gram。对于word2vec，采用三层神经网络就能训练，最后一层的输出要用一个Huffman树进行词的预测。
- **Count-based模型**，如GloVe，本质上是对共现矩阵进行降维。首先，构建一个词汇的共现矩阵，每一行是一个word，每一列是context。共现矩阵就是计算每个word在每个context出现的频率。由于context是多种词汇的组合，其维度非常大，我们希望像network embedding一样，在context的维度上降维，学习word的低维表示。这一过程可以视为共现矩阵的重构问题，即reconstruction loss。

相比Word2Vec，GloVe更容易并行化，所以对于较大的训练数据，GloVe更快。

Embedding的发展历程

参考：<https://mp.weixin.qq.com/s/BEPtBbL9qt5qk1ZvJX4luQ>

textRank

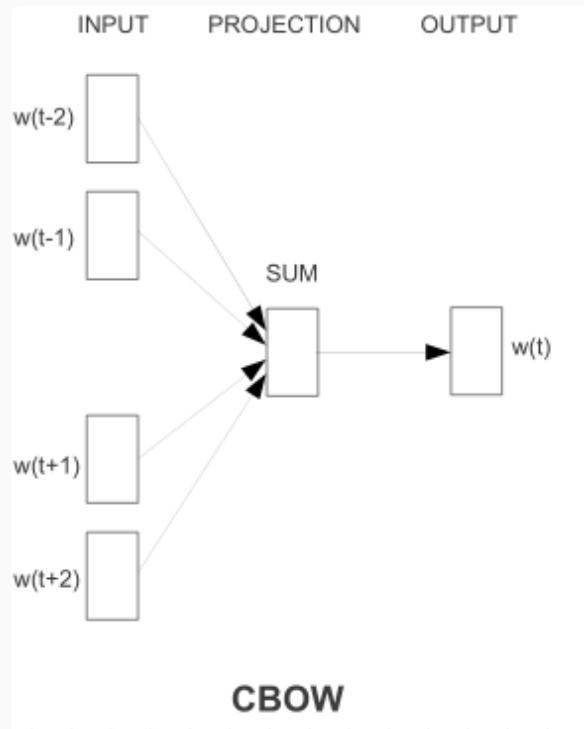
textRank 脱胎于 Google 网页搜索技术 PageRank，主要思想是将文档视作图，文本单元、单元间关系或者整个句子作为图中节点。

word2vec

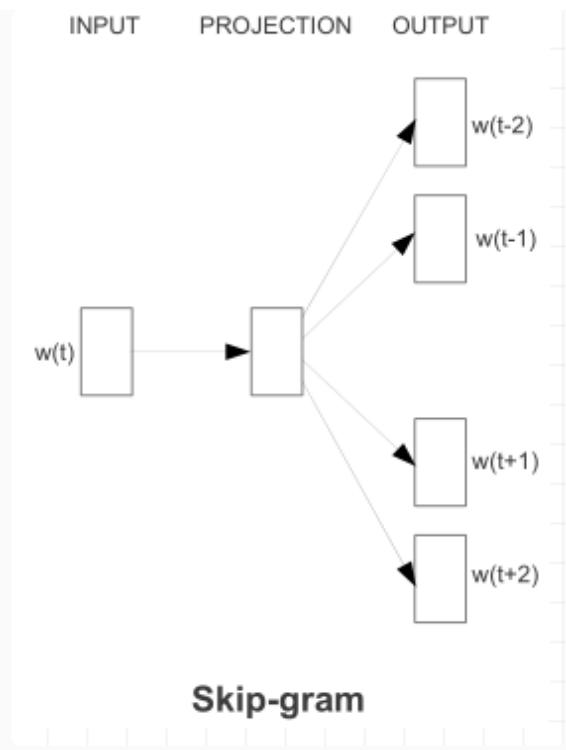
2013 年 Google 研究团队在前人模型基础上提出了两个新模型架构 CBOW (Continuous bag-of-word) 和 skip-gram，在词相似度任务上大幅提高了准确率，大幅降低了计算成本，同时虽是针对特殊任务提出，但对其他任务也很有效，自此掀起了一股「2Vec」风潮。

相较早期的词袋、n-gram等模型，word2vec 优势在于不仅在词空间内相似词离得更近，而且每个词可以从多维度计算相似性，语义运算则是“意外之喜”。预训练、迁移学习思想自此开始流行。

CBOW



skip-gram



Skip-gram

Glove

此前的两种主流方法：

1. 全局矩阵分解，如 LSA，
2. 基于局部窗口的方法，如 skip-gram

FastText

受分布式词表示学习方法启发，脸书团队尝试针对性改进文本分类任务基准性能。过往基于神经网络的模型通常效果好，效率低，训练时间、测试时间都很长，限制了模型使用数据集规模。

为此提出的 FastText 依旧是一个线性分类器，一大亮点在于，该模型在学习词表示的过程中，文本表示作为隐变量顺带被求出。

整体而言，FastText 结构很像 word2vec，只是预测的不再是中心词而是标签。使用了 Hierarchical Softmax 技巧，额外添加了 n-gram 特征。同时，输入不再是近邻圈定的固定窗口，而是标识序列

Style Transformer

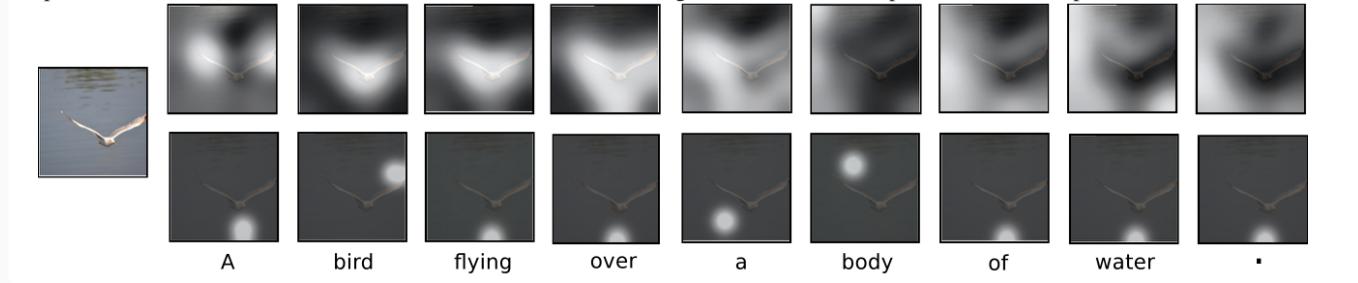
原文: <https://arxiv.org/pdf/1905.05621.pdf>

笔记: <https://drive.google.com/drive/u/0/my-drive>

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention

[Show, Attend and Tell Neural Image Caption.pdf](#)

Figure 2. Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image. “soft” (top row) vs “hard” (bottom row) attention. (Note that both models generated the same captions in this example.)



如图，“一只鸟正在一片水域上空飞翔。”每一个单词都准确对应图片中的相应位置。机器看一张图片，就能输出对应的文字语言描述，这样的效果怎么做到的？

Introduction

challenge: Automatically generating captions of an image is a task very close to the heart of scene understanding — one of the primary goals of computer vision. They must also be capable of capturing and expressing their relationships in a natural language.

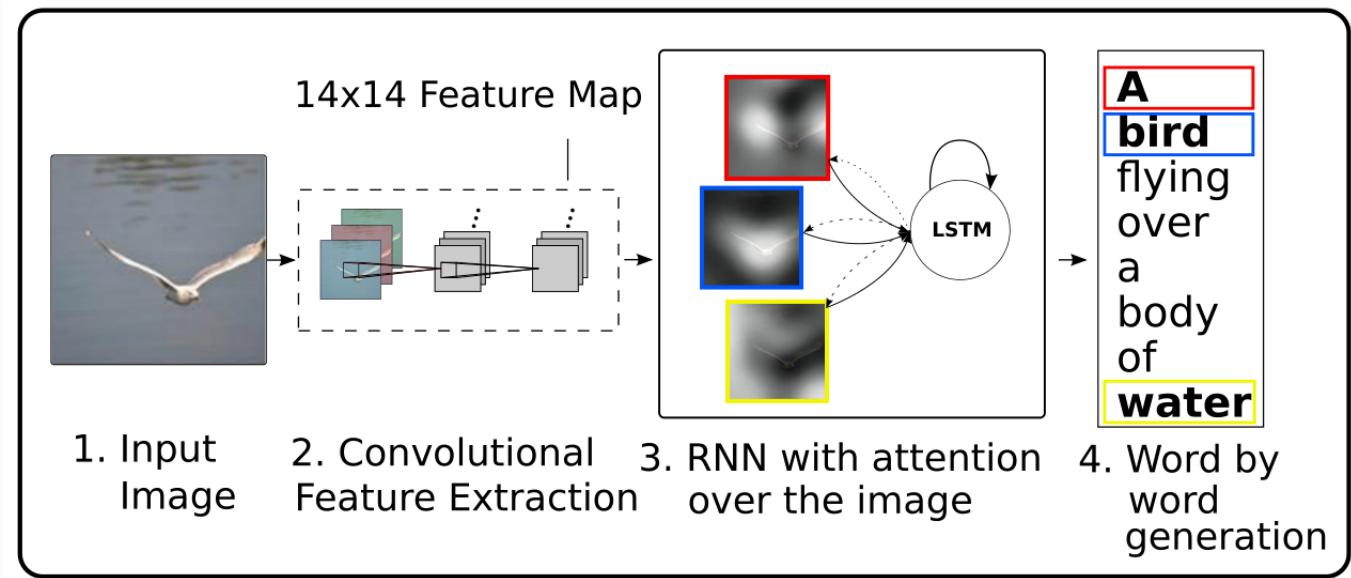
Recent work: 用CNN获得图像的矢量表示，用RNN将这些表示解码成自然语言。

“One of the most curious facets of the human visual system is the presence of attention (Rensink, 2000; Corbetta & Shulman, 2002). ”图像杂乱时，attention特别有用。 Rather than compress an entire image into a static representation, attention allows for salient features to dynamically come to the forefront as needed.

This paper introduced an attention based model that automatically learns to describe the content of images in a deterministic manner using standard backpropagation techniques and stochastically by maximizing a variational lower bound. Through visualization, we know how the model is able to automatically learn to fix its gaze on salient objects while generating the corresponding words in the output sequence.

This paper described approaches to caption generation that attempt to incorporate a form of attention with two variants: a “hard” attention mechanism and a “soft” attention mechanism. We also show how one advantage of including attention is the ability to visualize what the model “sees”.

Image Caption Generation with Attention Mechanism



Input: raw image \rightarrow Output: caption $y = \{y_1, \dots, y_C\}$, $y_i \in \mathbb{R}^K$

ENCODER: convolutional features

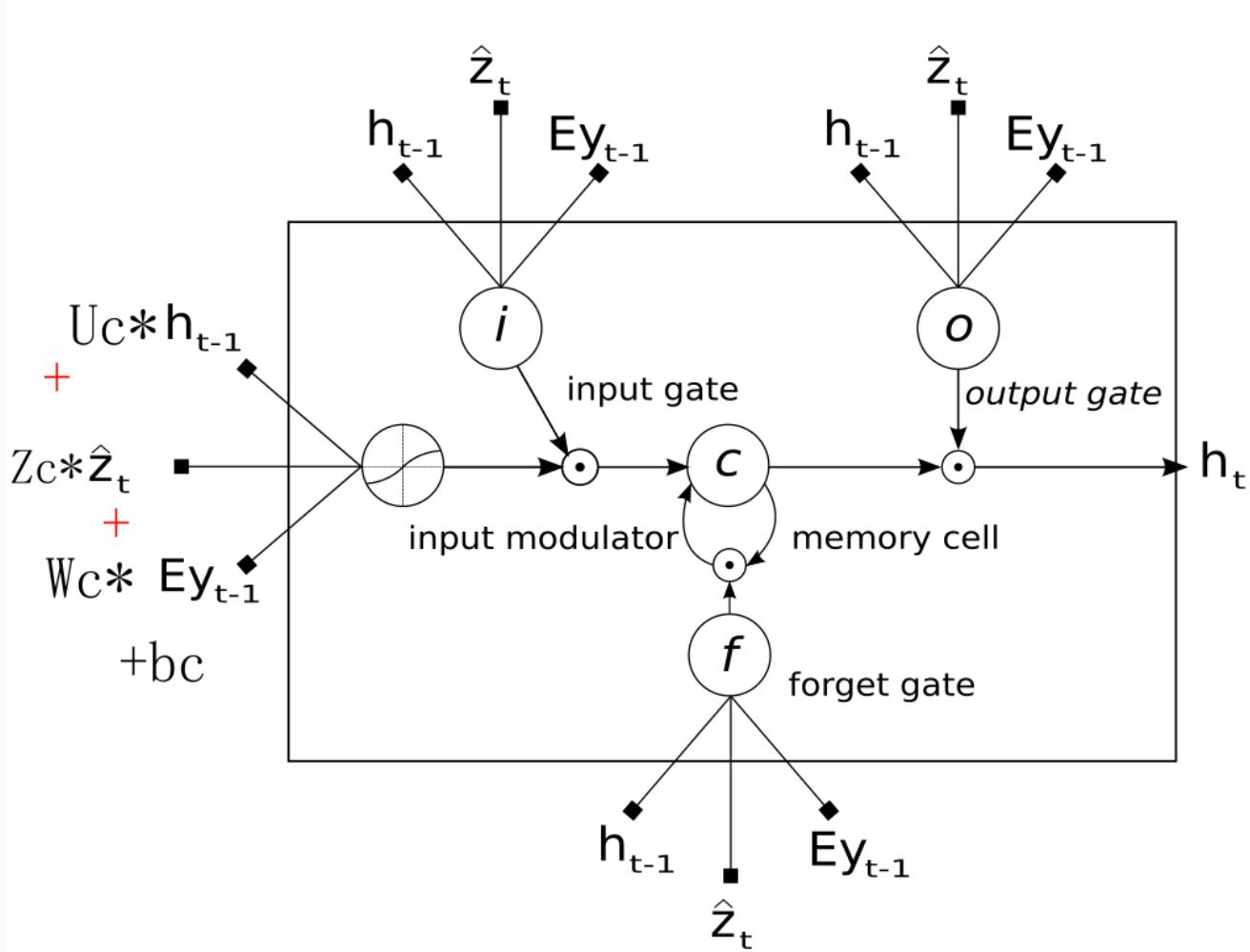
We use a convolutional neural network in order to extract a set of feature vectors which we refer to as **annotation vectors**. The extractor

produces L vectors, each of which is a D-dimensional representation corresponding to a part of the image.

$$a = a1, \dots, aL, ai \in RD$$

In order to obtain a correspondence between the feature vectors and portions of the 2-D image, we **extract features from a lower convolutional layer** unlike previous work which instead used a fully connected layer. This allows the decoder to selectively **focus on certain parts** of an image by selecting a subset of all the feature vectors.

DECODER: LSTM



$$C_t = f_t * C_{t-1} + i_t * \tanh(w_c * Ey_{t-1} + Z_c * \hat{z}_t + U_c * h_{t-1} + b_c)$$

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \mathbf{g}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} T_{D+m+n,n} \begin{pmatrix} \mathbf{E}\mathbf{y}_{t-1} \\ \mathbf{h}_{t-1} \\ \hat{\mathbf{z}}_t \end{pmatrix} \quad (1)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \quad (2)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t). \quad (3)$$

$\mathbf{z} \in \mathbb{R}^D$ is **上下文向量**, capturing the visual information associated with a particular input location, as explained below. $\mathbf{E} \in \mathbb{R}^{m \times K}$ is an **embedding matrix**. Let m and n denote the embedding and LSTM dimensionality respectively and σ and \odot be the logistic sigmoid activation and element-wise multiplication respectively.

In simple terms, the context vector $\hat{\mathbf{z}}_t$ (equations (1)–(3)) is a dynamic representation of the relevant part of the image input at time t . We define a mechanism ϕ that computes $\hat{\mathbf{z}}_t$ from the annotation vectors \mathbf{a}_i , $i = 1, \dots, L$ corresponding to the **features extracted at different image locations**. For each location i , the mechanism generates a positive weight a_i , which can be interpreted either as the probability that location i is the right place to focus for producing the next word (the “hard” but stochastic attention mechanism), or as the relative importance to give to location i in blending the a_i ’s together. The weight a_i of each annotation vector \mathbf{a}_i is computed by an attention model f_{att} for which we use a multilayer perceptron conditioned on the previous hidden state \mathbf{h}_{t-1} .

The soft version of this attention mechanism was introduced by Bahdanau et al. (2014). For emphasis, we note that the hidden state varies as the output RNN advances in its output sequence: “where” the network looks next depends on the sequence of words that has already been generated.

$$e_{ti} = f_{\text{att}}(\mathbf{a}_i, \mathbf{h}_{t-1}) \quad (4)$$

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^L \exp(e_{tk})}. \quad (5)$$

$$\hat{\mathbf{z}}_t = \phi(\{\mathbf{a}_i\}, \{\alpha_i\}), \quad (6)$$

The **initial** memory state and hidden state of the LSTM are predicted by an average of the annotation vectors fed through two separate MLPs (init,c and init,h):

$$\mathbf{c}_0 = f_{\text{init},c}\left(\frac{1}{L} \sum_i^L \mathbf{a}_i\right)$$

$$\mathbf{h}_0 = f_{\text{init},h}\left(\frac{1}{L} \sum_i^L \mathbf{a}_i\right)$$

Last, we use a deep output layer (Pascanu et al., 2014) to compute the **output word probability** given the LSTM state, the context vector and the previous word:

$$p(\mathbf{y}_t | \mathbf{a}, \mathbf{y}_1^{t-1}) \propto \exp(\mathbf{L}_o(\mathbf{E}\mathbf{y}_{t-1} + \mathbf{L}_h \mathbf{h}_t + \mathbf{L}_z \hat{\mathbf{z}}_t)) \quad (7)$$

Where $\mathbf{L}_o \in \mathbb{R}^{K \times m}$, $\mathbf{L}_h \in \mathbb{R}^{m \times n}$, $\mathbf{L}_z \in \mathbb{R}^{m \times D}$, and $\mathbf{E} \in \mathbb{R}^{m \times k}$ are learned parameters initialized randomly.

那么，怎么学习这个 ϕ 呢？文中给出Hard和Soft两种方法：

Learning Stochastic “Hard” vs Deterministic “Soft” Attention

Stochastic “Hard” Attention

the location variable s_t as where the model decides to focus attention when generating the t^{th} word.

$s_{t,i}$ is an indicator one-hot variable which is set to 1 if the i -th location (out of L) is the one used to extract visual features (=1表示选中该位置).

By treating the attention locations as intermediate latent variables, we can assign a multinoulli distribution parametrized by $\{a_i\}$, a_{ti} 是 t 时刻 location i 的关注度.

$$p(s_{t,i} = 1 \mid s_{j < t}, \mathbf{a}) = \alpha_{t,i} \quad (8)$$

$$\hat{\mathbf{z}}_t = \sum_i s_{t,i} \mathbf{a}_i. \quad (9)$$

定义一个新的目标函数 L_s , 它是观察给定图像特征 \mathbf{a} 的单词序列 \mathbf{y} 的边际对数似然 $\log p(\mathbf{y} \mid \mathbf{a})$ 的变分下界, 通过直接优化 L_s 可以得到模型参数 \mathbf{W} 的学习算法:

$$\begin{aligned} L_s &= \sum_s p(s \mid \mathbf{a}) \log p(\mathbf{y} \mid s, \mathbf{a}) \\ &\leq \log \sum_s p(s \mid \mathbf{a}) p(\mathbf{y} \mid s, \mathbf{a}) \\ &= \log p(\mathbf{y} \mid \mathbf{a}) \end{aligned} \quad (10)$$

$$\frac{\partial L_s}{\partial W} = \sum_s p(s \mid \mathbf{a}) \left[\frac{\partial \log p(\mathbf{y} \mid s, \mathbf{a})}{\partial W} + \log p(\mathbf{y} \mid s, \mathbf{a}) \frac{\partial \log p(s \mid \mathbf{a})}{\partial W} \right]. \quad (11)$$

对公式 11 进行关于模型参数的梯度的基于蒙特卡罗的采样近似。这可以通过从由公式 8 定义的 multinoulli 分布中采样位置 s_t 来实现。

$$\tilde{s}_t \sim \text{Multinoulli}_L(\{\alpha_i\})$$

$$\begin{aligned} \frac{\partial L_s}{\partial W} \approx \frac{1}{N} \sum_{n=1}^N & \left[\frac{\partial \log p(\mathbf{y} | \tilde{s}^n, \mathbf{a})}{\partial W} + \right. \\ & \left. \log p(\mathbf{y} | \tilde{s}^n, \mathbf{a}) \frac{\partial \log p(\tilde{s}^n | \mathbf{a})}{\partial W} \right] \quad (12) \end{aligned}$$

A **moving average baseline** is used to reduce the variance in the Monte Carlo estimator of the gradient. Upon seeing the k^{th} mini-batch, the moving average baseline is estimated as an accumulated sum of the previous log likelihoods with exponential decay(指数衰减):

$$b_k = 0.9 \times b_{k-1} + 0.1 \times \log p(\mathbf{y} | \tilde{s}_k, \mathbf{a})$$

为了进一步降低估计量方差，在多项式分布 $H[s]$ 上增加了熵项。此外，对于给定的图像，概率为0.5时，我们将采样的注意位置 \tilde{s} 设置为其期望值 a 。这两种技术都提高了随机注意学习算法的鲁棒性。

$$\begin{aligned} \frac{\partial L_s}{\partial W} \approx \frac{1}{N} \sum_{n=1}^N & \left[\frac{\partial \log p(\mathbf{y} | \tilde{s}^n, \mathbf{a})}{\partial W} + \right. \\ & \left. \lambda_r (\log p(\mathbf{y} | \tilde{s}^n, \mathbf{a}) - b) \frac{\partial \log p(\tilde{s}^n | \mathbf{a})}{\partial W} + \lambda_e \frac{\partial H[\tilde{s}^n]}{\partial W} \right] \end{aligned}$$

λ_r 和 λ_e 是通过交叉验证设置的两个超参数，这相当于强化学习规则(Williams, 1992)，其中注意力选择动作序列的奖励是一个实值，与采样注意轨迹下目标句子的对数可能性成正比。

Deterministic “Soft” Attention

Soft对每个区域都关注，只是关注的重要程度不同 stochastic attention requires sampling the attention location s_t each time, instead we can take the expectation of the context vector $\hat{\mathbf{z}}_t$ directly:

$$\mathbb{E}_{p(s_t|a)}[\hat{\mathbf{z}}_t] = \sum_{i=1}^L \alpha_{t,i} \mathbf{a}_i \quad (13)$$

The whole model is smooth and differentiable under the deterministic attention, so learning end-to-end is trivial by using standard backpropagation.

Learning the deterministic attention can also be understood as approximately optimizing the marginal likelihood in Equation 10 under the attention location random variable s_t . 对于一节泰勒近似, $\mathbb{E}_{p(s_t|a)[ht]}$ is equal to computing ht using a single forward prop with the expected context vector $\mathbb{E}_{p(s_t|a)[\hat{z}_t]}$.

$\mathbf{n}_t = \mathbf{L}_o(\mathbf{E}\mathbf{y}_{t-1} + \mathbf{L}_h \mathbf{h}_t + \mathbf{L}_z \hat{\mathbf{z}}_t)_{n_{ti}}$ 表示通过将随机变量 \hat{z} 值设置为 a_i 而计算的 n_t 。

the NWGM of a softmax unit is obtained by applying softmax to the expectations of the underlying linear projections.

$$\begin{aligned} NWGM[p(y_t = k | \mathbf{a})] &= \frac{\prod_i \exp(n_{t,k,i})^{p(s_{t,i}=1|a)}}{\sum_j \prod_i \exp(n_{t,j,i})^{p(s_{t,i}=1|a)}} \\ &= \frac{\exp(\mathbb{E}_{p(s_t|a)}[n_{t,k}])}{\sum_j \exp(\mathbb{E}_{p(s_t|a)}[n_{t,j}])} \end{aligned}$$

公式表明，使用期望上下文向量

可以很好地逼近字幕预测的归一化加权几何平均值,其中:

$$\mathbb{E}[\mathbf{n}_t] = \mathbf{L}_o(\mathbf{E}\mathbf{y}_{t-1} + \mathbf{L}_h \mathbb{E}[\mathbf{h}_t] + \mathbf{L}_z \mathbb{E}[\hat{\mathbf{z}}_t])$$

the deterministic attention model is an **approximation to the marginal likelihood over the attention locations.**

Doubly Stochastic Attention

双随机正则化目的是：让Attention平等的对待图片的每一个区域；

设置阈值 $\beta t = \sigma(f_\beta(ht-1))$ ：让decoder决定把重点放在语言建模还是在每个时间步骤的上下文中

Soft attention通过最小化以下惩罚的负对数似然率来端到端地训练该模型：

$$L_d = -\log(P(\mathbf{y}|\mathbf{x})) + \lambda \sum_i^L (1 - \sum_t^C \alpha_{ti})^2 \quad (14)$$

Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)

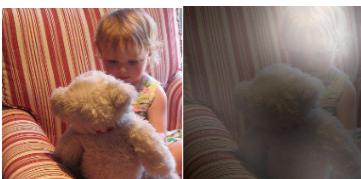


A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.



A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.



A group of people sitting on a boat in the water.



A giraffe standing in a forest with trees in the background.

Experiments

Pretrain: VCGnet, without finetune

Optimizer: Flickr8k dataset – RMSProp; Flickr30k/MS COCO dataset – Adam

In our experiments we use the **14×14×512 feature map** of the fourth convolutional layer before max pooling. This means our decoder operates on the flattened 196×512 (i.e $L \times D$) encoding.
fixed vocabulary size of 10,000

卷积特征提取器，single model versus ensemble，数据集拆分影响实验效果

NLP机器翻译评价度量：

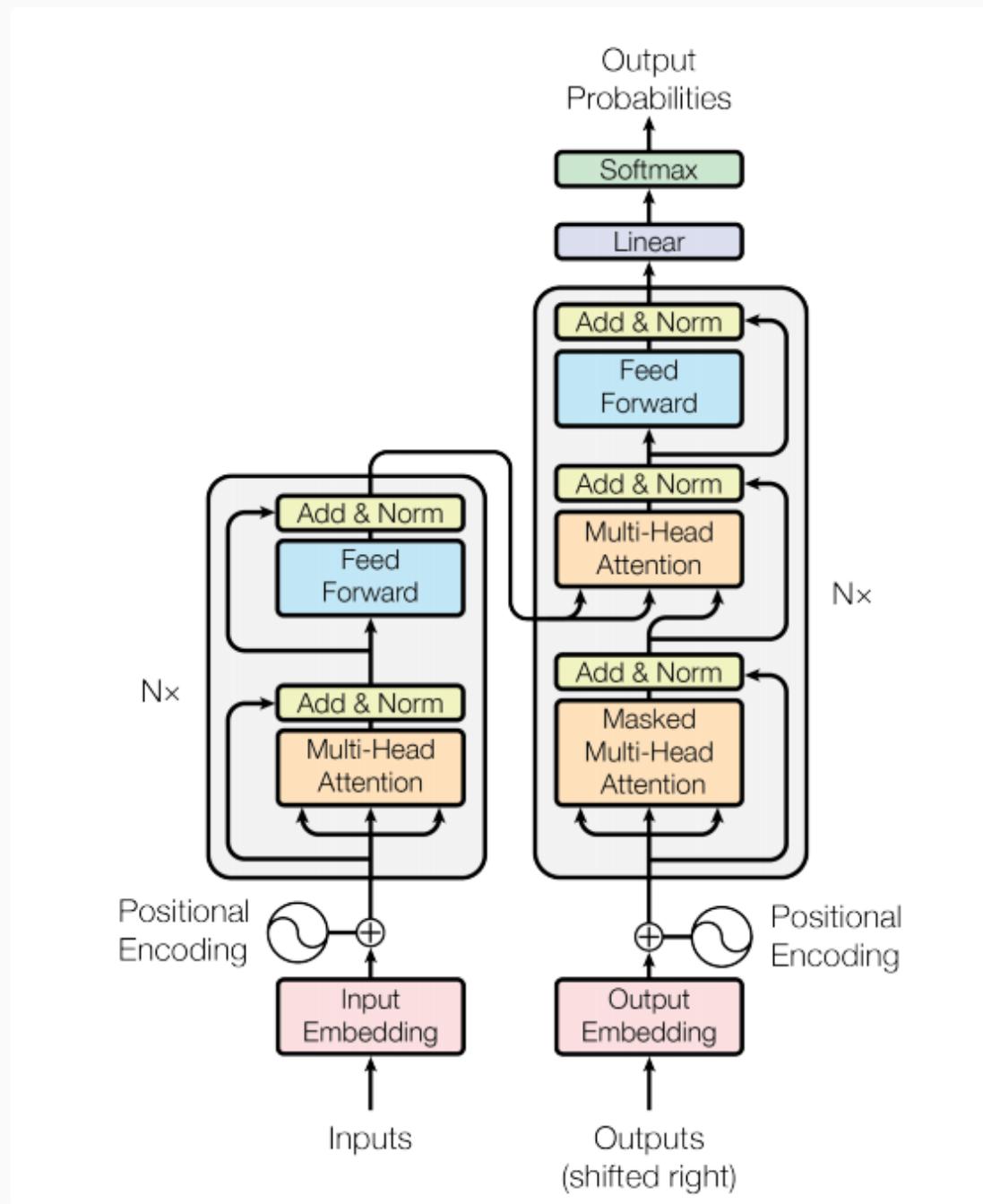
BLEU, ROUGE, METEOR, ROUGE (<https://www.jianshu.com/p/1517a75af993>)

Conclusion

本文主要介绍了两种方法：1) 可用标准反向传播方法训练的“软”确定性注意机制；
2) 可通过最大化近似变分下限或等效地通过加强训练的“硬”随机注意机制。
Encoder–Decoder&Attention可以用到其他领域

DA-Transformer (将加权距离map到re-scale的系数中)

Transformer



$$\mathbf{Q}^{(i)}, \mathbf{K}^{(i)}, \mathbf{V}^{(i)} = \mathbf{HW}_Q^{(i)}, \mathbf{HW}_K^{(i)}, \mathbf{HW}_V^{(i)}. \quad (1)$$

$$\text{Attention}(\mathbf{Q}^{(i)}, \mathbf{K}^{(i)}, \mathbf{V}^{(i)}) = \text{softmax}\left(\frac{\mathbf{Q}^{(i)} \mathbf{K}^{(i)\top}}{\sqrt{d}}\right) \mathbf{V}^{(i)}, \quad (2)$$

$$\begin{aligned} \text{Multihead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \text{Concat}(\text{head}_1, \dots, \text{head}_h) \mathbf{W}_O, \\ \text{where } \text{head}_i &= \text{Attention}(\mathbf{Q}^{(i)}, \mathbf{K}^{(i)}, \mathbf{V}^{(i)}), \end{aligned} \quad (3)$$

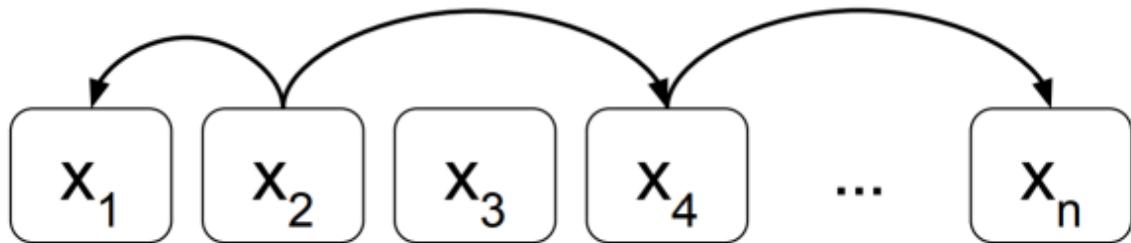
$$FFN(\mathbf{x}) = \max(0, \mathbf{x} \mathbf{W}_1 + \mathbf{b}_1) \mathbf{W}_2 + \mathbf{b}_2, \quad (4)$$

$$\begin{aligned} PE_{(pos, 2i)} &= \sin(pos / 10000^{2i/d_{\text{model}}}) \\ PE_{(pos, 2i+1)} &= \cos(pos / 10000^{2i/d_{\text{model}}}) \end{aligned}$$

Since self–attention network does not distinguish the order and position of input tokens, Transformer adds the **sin** embeddings of positions to the input embeddings to capture position information. However, position embeddings may not be optimal for distance modeling in Transformer because **distances cannot be precisely recovered** from the dot–product between two position embeddings.

Self–Attention with Relative Position Representations

$$\begin{array}{lll}
 a^V_{2,1} = w^V_{-1} & a^V_{2,4} = w^V_2 & a^V_{4,n} = w^V_k \\
 a^K_{2,1} = w^K_{-1} & a^K_{2,4} = w^K_2 & a^K_{4,n} = w^K_k
 \end{array}$$



DA-Transformer

Head-wise Distance Weighting

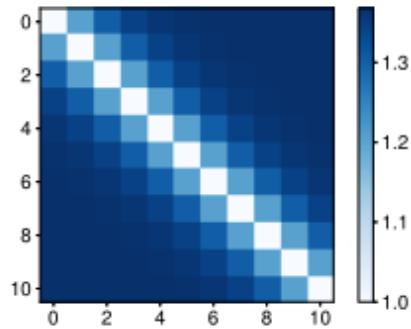
Token : $H = [h_1, h_2, \dots, h_N]$

real relative distance : $R_{i,j} = |i - j|$. 给distance加weight, re-scaled

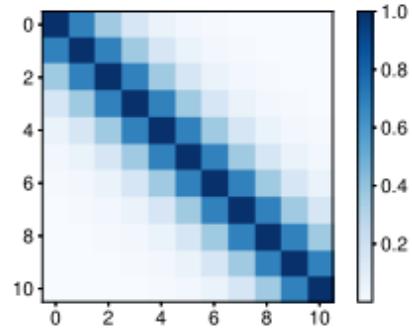
coefficients: $R^{(i)} = w_i R$

更正的 $R^{(i)}$ 会更强烈地放大注意力权重，观察远距离信息；而更负的 $R^{(i)}$ 会更强烈地降低注意力权重，观察局部信息。

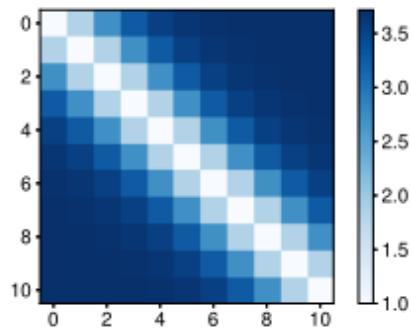
w_i 控制Attention head的长短期偏好：



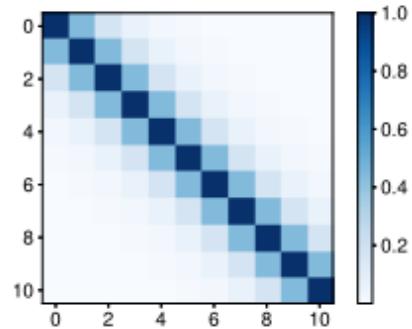
(a) $w_i = 1, v_i = -1.$



(b) $w_i = -1, v_i = -1.$



(c) $w_i = 1, v_i = 1.$



(d) $w_i = -1, v_i = 1.$

$w_i > 0$, long distance is prefer;

a,c图, c图系数range更大, capture long distance

b,d图, d图系数sharper, capture short distance

Weighted Distance Mapping

we propose to directly re-scale the attention weights based on the **mapped** relative distances instead of using sinusoidal position embeddings, which can explicitly encode real distance information to achieve more accurate distance modeling.

不同映射函数 $f(\cdot)$ 的影响:

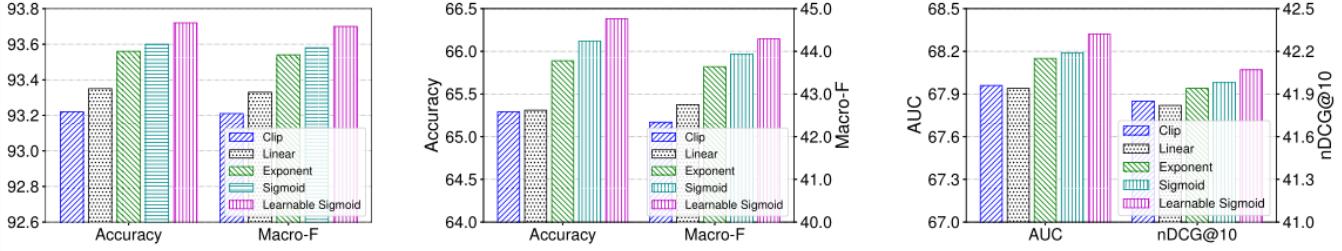


Figure 4: Influence of using different mapping functions.

clip: cannot keep the precise distance information beyond a certain range;

linear: cannot guarantee $f(\cdot)$ to be positive;

exponet: long sequences may lead to the problem of exponent explosion;

Sigmoid: 将加权距离map到re-scale的系数中；

ours: adjusting the activated function in a learnable way can better map the raw distance into re-scaled coefficients.

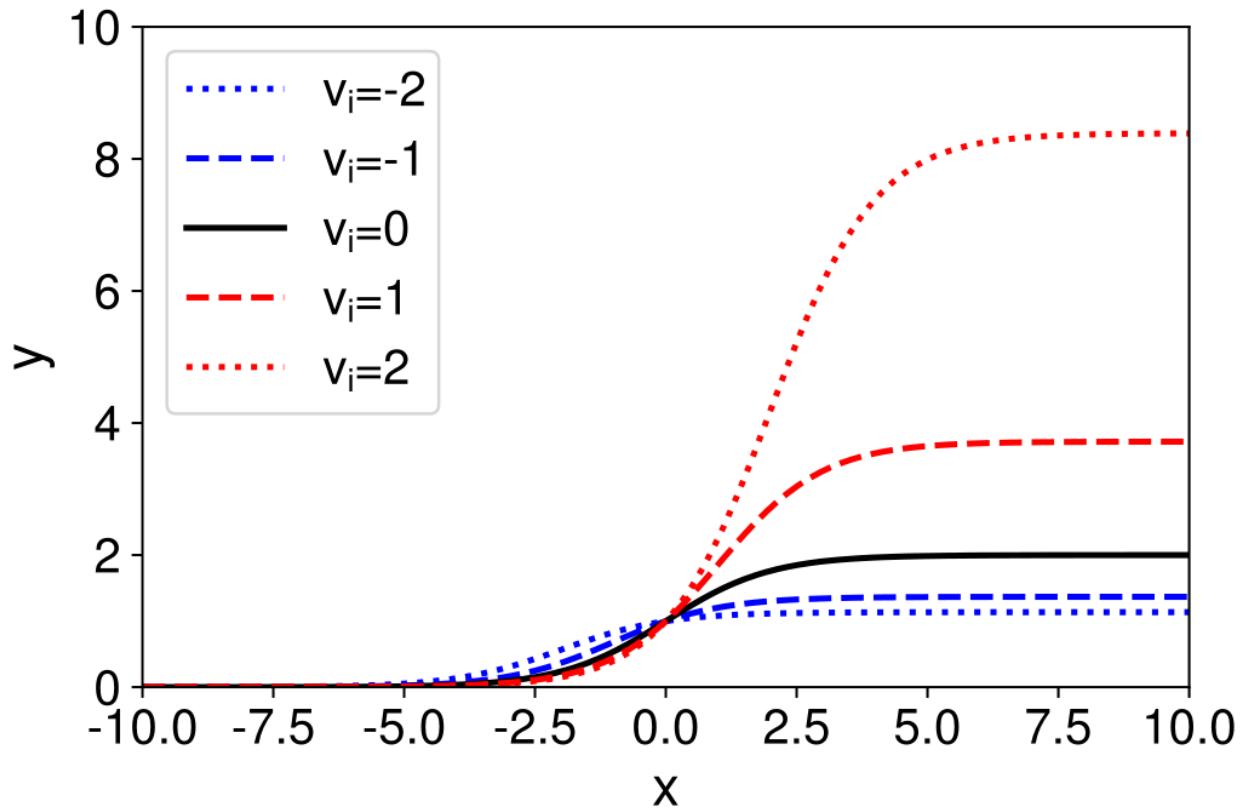
本文采用的：

$$f(\mathbf{R}^{(i)}; v_i) = \frac{1 + \exp(v_i)}{1 + \exp(v_i - \mathbf{R}^{(i)})}, \quad (5)$$

满足： (1) $f(0) = 1$. (2) The value of $f(\mathbf{R}^{(i)})=0$ when $\mathbf{R}^{(i)} \rightarrow -\infty$. This requirement is to guarantee that if an attention head prefers to capture local information ($v_i < 0$), the long-distance information should be surpassed.

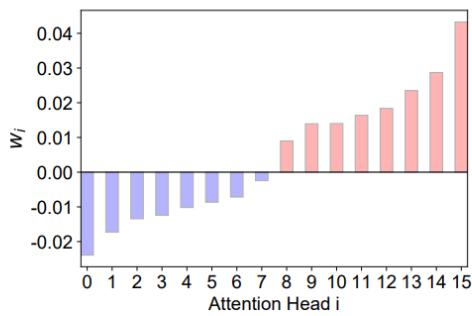
(3) The value of $f(\mathbf{R}^{(i)})$ should be limited when $\mathbf{R}^{(i)} \rightarrow +\infty$. This requirement is to ensure that the model is able to process long sequences without over-emphasize distant contexts. (4) The scale of $f(\cdot)$ needs to be tunable. This aims to help the model better adjust the intensity of distance information. (5) The function $f(\cdot)$ needs to be monotone.

v_i 控制 Sigmoid 的形状：

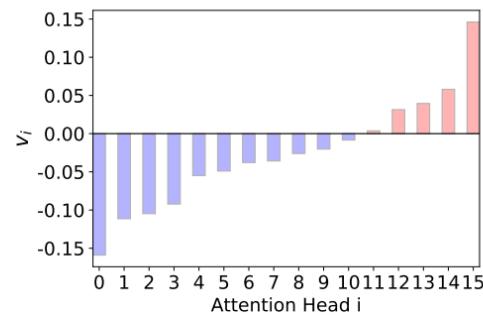


上界越大，距离信息越密集； v_i 控制上界和陡度； $v_i=0$ 时，是标准的Sigmoid函数

DA-Transformer can flexibly adjust its preference on short-term or long-term information and the intensity of attention adjustment by learning different values of w_i and v_i according to the task characteristics.



(a) w_i .



(b) v_i .

Attention Adjustment

Then, we use the re-scaled coefficients to adjust the raw attention weights that are computed by the dot product between the query and key.

$$\text{Attention}(\mathbf{Q}^{(i)}, \mathbf{K}^{(i)}, \mathbf{V}^{(i)}) = \text{softmax}\left(\frac{\mathbf{Q}^{(i)} \mathbf{K}^{(i)\top}}{\sqrt{d}}\right) \mathbf{V}^{(i)}, \quad (2)$$

$$\mathbf{O}^{(i)} = \text{softmax}\left(\frac{\text{ReLU}(\mathbf{Q}^{(i)} \mathbf{K}^{(i)\top}) * \hat{\mathbf{R}}^{(i)}}{\sqrt{d}}\right) \mathbf{V}^{(i)}, \quad (6)$$

RELU:

(1) the sign of attention weights $\mathbf{Q}(i)\mathbf{K}(i)/\sqrt{d}$ is indefinite and the multiplied results cannot accurately reflect the influence of distance information.

Thus, we propose to add a ReLU (Glorot et al., 2011) activation function to the raw attention weights **to keep non-negativity**.

(2) 还可以将稀疏性引入self-attention, 因为只有正的关注权重可以被重新缩放的系数放大, 这使得关注权重更加敏锐"focused"。

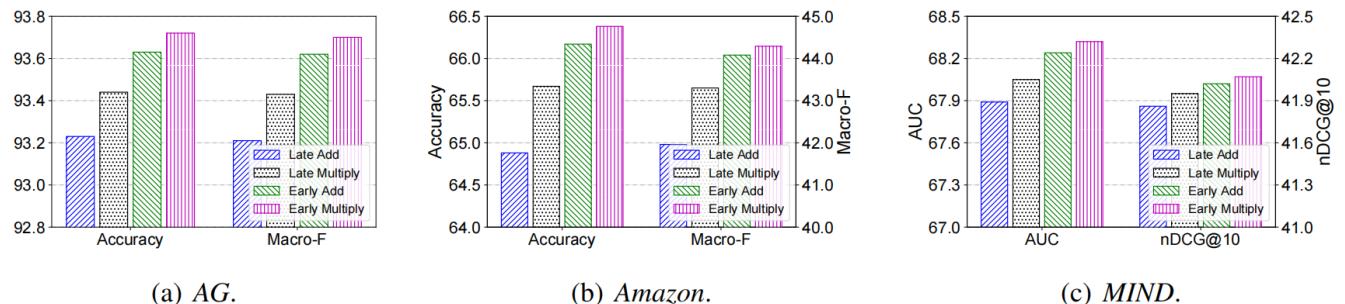


Figure 5: Influence of using different attention adjusting methods.

(1): early adjustment is better than late adjustment.

(2): multiplying is better than adding for both early and late adjustment

将来自 h 个独立注意头的输出连接起来, 并将其投影到一个统一的输出中。此外, 保持与标准Transformer相同的层归一化和剩余连接策略。

Experiments

Methods	AG		Amazon	
	Accuracy	Macro-F	Accuracy	Macro-F
Transformer	93.01±0.13	93.00±0.13	65.15±0.40	42.14±0.41
Transformer-RPR	93.14±0.12	93.13±0.13	65.29±0.38	42.40±0.40
Transformer-XL	93.35±0.10	93.34±0.11	65.50±0.40	42.88±0.43
Adapted Transformer	93.28±0.13	93.27±0.14	65.47±0.39	42.69±0.42
*DA-Transformer	93.72±0.11	93.70±0.12	66.38±0.39	44.29±0.40

Table 3: Results on AG and Amazon. *Improvement over the underlined second best results is significant at $p < 0.05$.

Methods	SST		SNLI	
	Accuracy	Macro-F	Accuracy	Macro-F
Transformer	89.67±0.22	89.59±0.24	81.45±0.30	81.42±0.31
Transformer-RPR	89.94±0.19	89.90±0.20	82.20±0.31	82.18±0.31
Transformer-XL	90.06±0.20	90.02±0.21	83.19±0.29	83.15±0.30
Adapted Transformer	90.15±0.19	90.10±0.1	82.35±0.28	82.31±0.30
*DA-Transformer	90.49±0.17	90.43±0.19	84.18±0.27	84.16±0.29

Table 4: Results on SST and SNLI. *Improvement over the underlined second best results is significant at $p < 0.05$.

Methods	AUC	MRR	nDCG@5	nDCG@10
Transformer	67.76±0.18	33.05±0.16	35.94±0.19	41.63±0.20
Transformer-RPR	67.81±0.16	33.10±0.17	35.98±0.20	41.65±0.21
Transformer-XL	67.92±0.16	33.15±0.16	36.04±0.20	41.70±0.19
Adapted Transformer	67.70±0.22	33.01±0.20	35.89±0.17	41.58±0.23
*DA-Transformer	68.32±0.15	33.36±0.16	36.34±0.14	42.07±0.17

Benchmark: AG,Amazon, SST, SNLI, MIND

Accuracy: 分类正确的样本数 与 样本总数之比。即: $(TP + TN) / (\text{ALL})$.

Precision (精确率、查准率) 被正确检索的样本数 与 被检索到样本总数之比。即: $TP / (TP + FP)$.

Recall (召回率、查全率) 被正确检索的样本数 与 应当被检索到的样本数之比。即: $TP / (TP + FN)$.

Macro-F: (适用环境: 多分类问题, 不受数据不平衡影响, 容易受到识别性高 (高recall、高precision) 的类别影响)

a) 计算每个类别的 $F1-score_i = 2 \frac{Recall_i \times Precision_i}{Recall_i + Precision_i}$.

b) 计算 macro F1score = $\frac{F1-score_1 + F1-score_2 + F1-score_3}{3}$

AUC: ROC曲线下方的面积

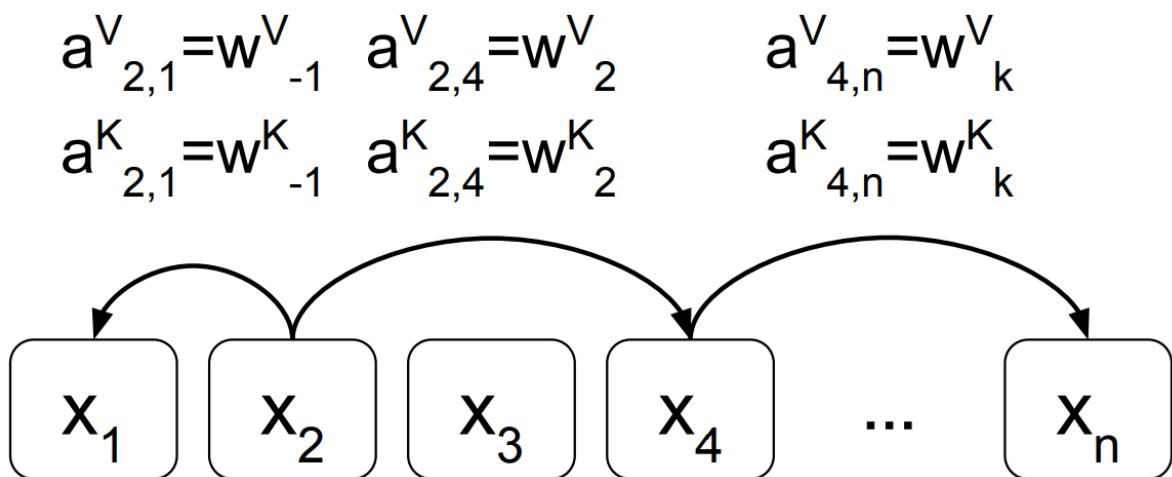
MRR:返回的结果集的优劣，跟第一个正确答案的位置有关，第一个正确答案越靠前，结果越好。

1. 考虑距离信息的方法取得了更好的性能，距离信息在上下文建模中很重要
2. Transformer-RPR没有保存精确的远距离信息
3. Adapted Transformer more suitable for modeling local contexts;Transformer-XL ,long sequences
4. ours:显式的编码真实的距离信息，而不是使用位置编码

Self-Attention with Relative Position Representations (利用edge信息)

<https://arxiv.org/pdf/1803.02155.pdf>

Question1:



边的关系怎么表示? 2,1 ;2,4 ;4,n?

Question2:

$$a_{ij}^K = w_{\text{clip}(j-i, k)}^K$$

$$a_{ij}^V = w_{\text{clip}(j-i, k)}^V$$

$$\text{clip}(x, k) = \max(-k, \min(k, x))$$

why clip?

Question3:

reduce the space complexity of **storing relative position representations** from $O(hn^2d_a)$ to $O(n^2d_a)$ by sharing them across each heads.

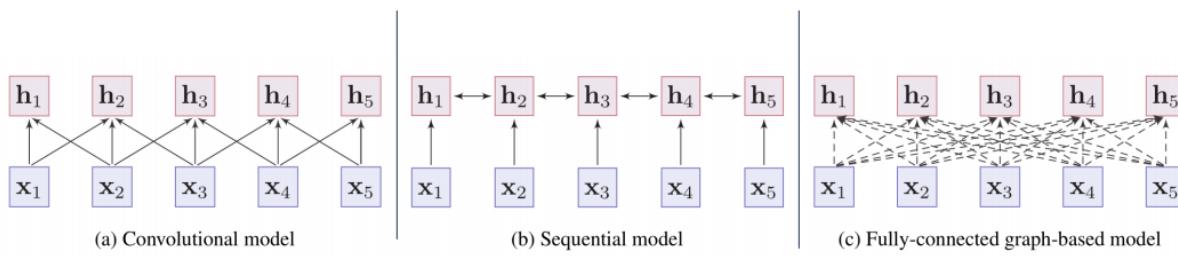
Additionally, relative position representations can be shared across sequences. Therefore, the **overall self–attention space complexity increases from $O(bhnd_z)$ to $O(bhnd_z + n^2d_a)$.** 这说明了什么?

Transformer的位置编码(小结)

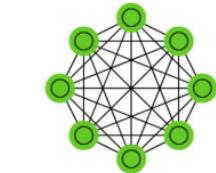
<https://mp.weixin.qq.com/s/oeXim7Z1hWV57BfH75bOsw>

Transformer改进

NLP中三大模型



隐含先验：局部组合
只建模了输入信息的局部依赖关系



连接权重 α_{ij} 由注意力
机制动态生成

Transformer至今为止是做强大的

自注意力模型分析



- ▶ 模型特点
 - ▶ 全连接结构
 - ▶ 没有任何先验假设
 - ▶ 通过位置编码来建模序列信息
- ▶ 复杂度： $O(L^2D)$
 - ▶ 无法处理长文档
 - ▶ 容易过拟合

解决思路



Transformer改进



Transformer改进汇总

2019

2020



Star-Transformer
Transformer-XL

BP-Transformer
Set Transformer
Multi-Scale Self-Attention

Reformer
Routing Transformer
ETC
Lformer
Big Bird
...

REFORMER（三个技巧减少内存占用）

论文：<https://arxiv.org/abs/2001.04451>

Transformer 的「Self–Attention & Feed–Forward Networks」的小组件多达6层，带来了计算复杂度高和内存占用大的问题。

为什么说计算复杂度高呢？

因为 Transformer 中 Self–Attention 的计算是「Scaled Dot–Product Attention」（点积自注意力），输入的序列要计算两两之间的 Attention Score，如果序列长度为 L ，那么时间复杂度是。Bert 的最大输入长度限定为512，和这个不无关系。

为什么说内存占用大呢？

一个原因是使用 Scaled Dot–Product Attention 计算 Attention Score 的矩阵，空间复杂度为，和序列长度 L 呈平方倍数增长。

第二个原因是 Transformer 中每一层的激活值（Activations）需要存储起来，用于进行反向传播。存储激活值的内存占用随层数增加而线性增长。

第三个原因是 Self–Attention & Feed–Forward Networks 的小组件中，FFN 是由两个全连接层组成，第一个全连接层的输出和第二个全连接层的输入的维度是2048，而 Self–Attention 的输出维度是512[1]，远大于，因此 FFN 这个 sub–layer 会消耗大量内存。

针对这些问题，作者提出了Reformer，希望在单块 GPU 上处理上万长度的输入序列。

Reformer 使用了以下三个技巧进行优化：

- 用 LSH Attention 来近似计算 Attention Score，将 Self–Attention 的复杂度优化到，从而加快计算速度和减少内存占用
- 用可逆残差网络（Reversible residual networks, RevNet）来减少 Activations 的存储，对 N 层的 Transformer 只存储最后一层的Activations，中间层的通过反推得到，从而减少内存占用
- 把 FFN 进行分块（Chunk），每次只对单独的一块进行计算，以时间换空间，减少内存占用

一：三种 Attention 回顾

我们先来回顾一下 Transformer 中的 Attention。

Transformer 中 Attention 的计算方式是 Scaled Dot–Product Attention，也就是缩放的点积注意力，由 Q、K、V 三个矩阵计算而得。

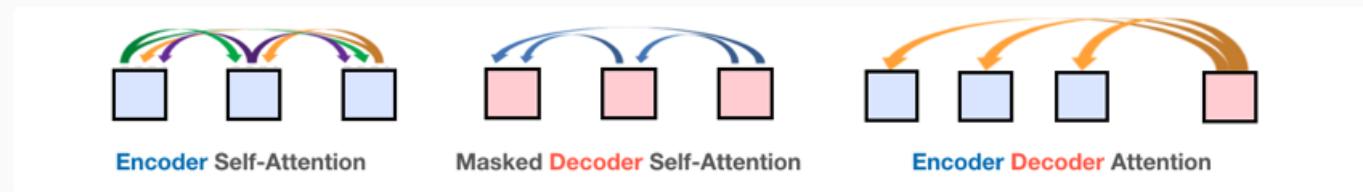
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

如果维度是512，那把512维的单个 Attention 分为了8个64维的，分别进行计算再拼接，也就是 Multi-Head Attention。

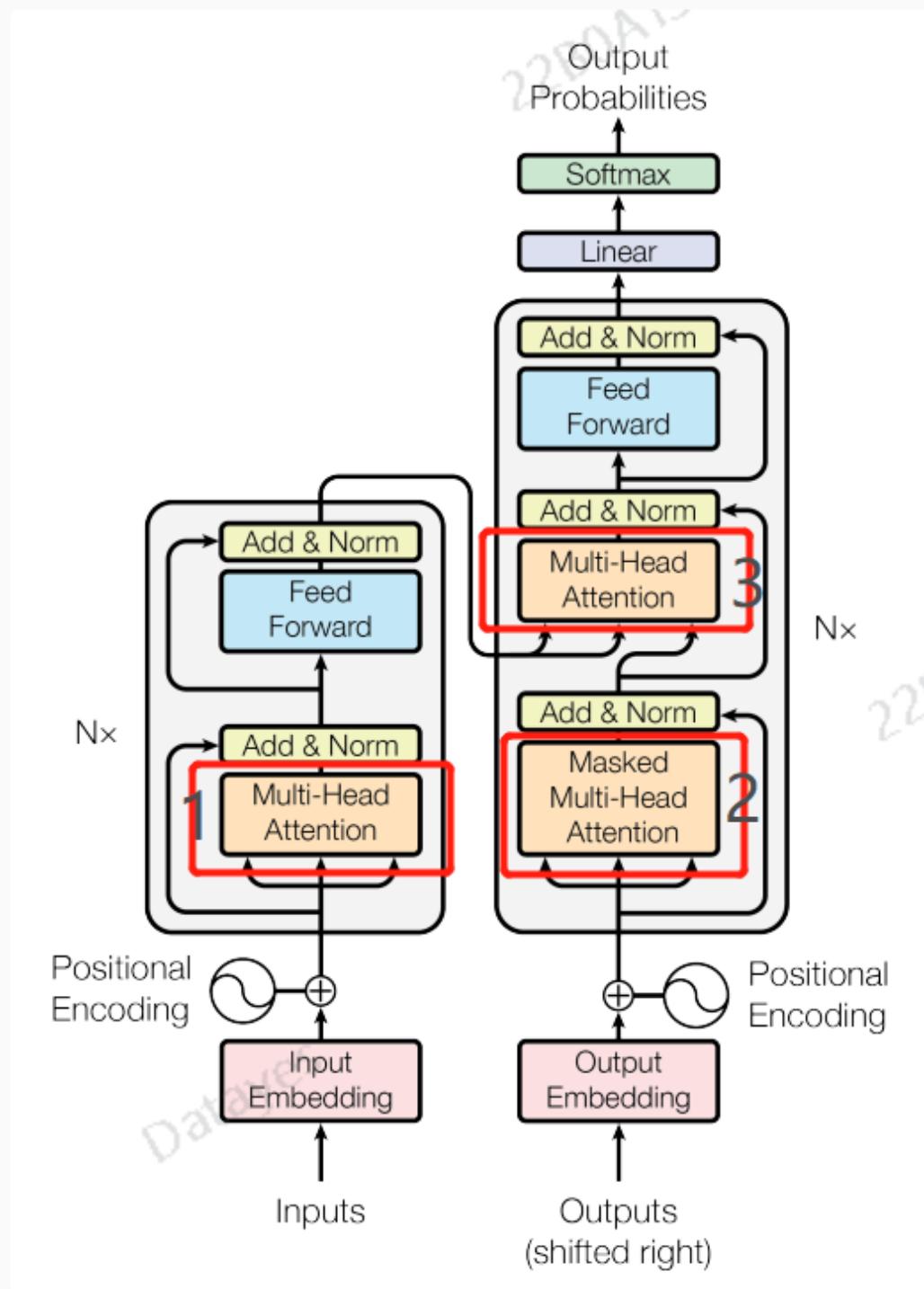
而 Multi-Head Attention 又有三种，分别是：

- Encoder Self-Attention：BERT 使用的结构，即双向 Transformer
- Decoder Self-Attention：GPT 使用的结构，带遮罩的 Self-Attention，即单向 Transformer
- Encoder-Decoder Attention：Seq2Seq 使用的结构，也叫 Cross-Attention

如下图[3]：



分别对应到 transformer 结构图中的1、2、3部位：



注意，上面说的 Attention 复杂度为，指的是 Encoder Self–Attention 和 Decoder Self–Attention。Decoder Self–Attention 尽管对右边位置的 Token 进行了 Mask，但并非不进行计算，而是通过把和右边 Token 计算的值设为从而 softmax 的值为0来实现，还是一种 Scaled Dot–Product Attention。、

二：改造 Attention

Self-Attention 要减少内存占用，「一个方法叫做 Memory-Efficient Attention，一次只计算一个 Query 的 Attention Score。」

标准的 Scaled Dot-Product Attention 是一个矩阵乘法，Q、K、V 的维度都是：

那么做矩阵乘法之后维度是：

如果 Length 特别大，比如输入序列长达 64000，那么一个样本就会得到一个 64000×64000 的矩阵，每个值是32位浮点数的话会消耗16G的内存。

矩阵乘法一次性就得到了所有 Query 的 Attention Score，速度快但是内存吃不消。换一种方式，每次只计算一个 Query 的 Attention Score，在内存中就存储一个 Query 的计算结果，以时间换空间。

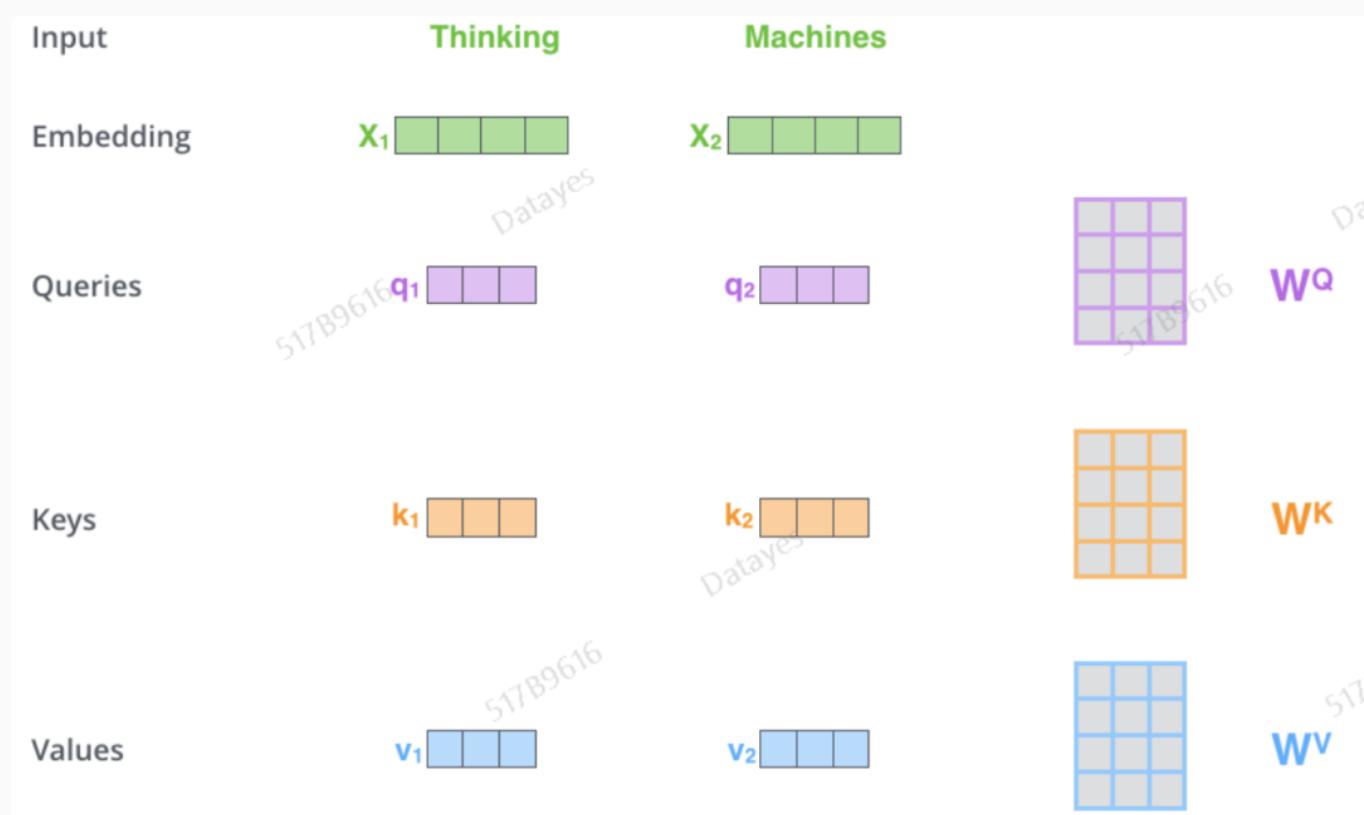
很不幸的是，在反向传播计算梯度的时候，还要再计算一次，双倍快乐。

所以这种方法非常鸡肋，顾此失彼，没啥应用价值。

而 Reformer 对 Self-Attention 做了两个改造。

「第一个改造叫做 Shared-QK Transformer，也就是让 Query 和 Key 共享参数。」

Q、K、V 这三个向量怎么来的呢？将输入序列进行 Embedding 得到一个维度是的向量A，再用三个有着不同参数的线性层对向量A进行投影，就得到了三个不同的向量 Q、K、V。



这么做显而易见的原因是通过Q和K共享参数，减少内存占用，但更重要的原因是「为了解决 LSH Attention 分桶后，桶内 query 和 key 数量不平衡的问题」。

实验证明 Shared-QK 并不会降低模型的精度。

第二个改造就是 LSH Attention 了，也就是把局部敏感哈希引入自注意力的计算中。

(1) LSH Attention 的动机

前面说了 Self-Attention 复杂度高，是因为的矩阵乘法运算。

考虑到最终是要计算，而对一个向量做 softmax，最大的概率值和对应的 index 是取决于前K个最大的元素。

```
● ● ●

import numpy as np

arr = np.array([10, 9, 8, 1.2, 0.5, 0.2])
print(np.exp(arr)/sum(np.exp(arr)))

print("=====")

arr = np.array([10, 9, 8, -100000, -100000, -100000])
print(np.exp(arr) / sum(np.exp(arr)))

输出：

[6.65116608e-01 2.44682726e-01 9.00137445e-02 1.00255072e-04
 4.97851952e-05 3.68817797e-05]
=====
[0.66524096 0.24472847 0.09003057 0.          0.          ]
```

对于每个 Query 的向量，和它最相近的几个 Key 的向量相乘可以得到最大的K个元素。

思考1：相近的元素相乘的结果就比较大吗？1*1和1*10相比，明显后面的结果大？

如果输入序列有64000个 Token，那么对于其中一个 Token，我们不再需要计算它和64000个 Token 的 score，只和最相近的32个 Token 做计算即可，效率就大大提高了。

那怎么找到最相近的 Token 呢？

这就用到了 LSH (Locality sensitive hashing，局部敏感哈希)。

(2) LSH 的实现方式

LSH 是近似最近邻查找的常用方法，可以把在高维空间中位置相近的点以较大的概率映射到同样的哈希值，也就是选择一个函数 $\text{hash}(x)$ ，「如果两个点p和q相近，那么大概率有 $\text{hash}(p)=\text{hash}(q)$ 。」

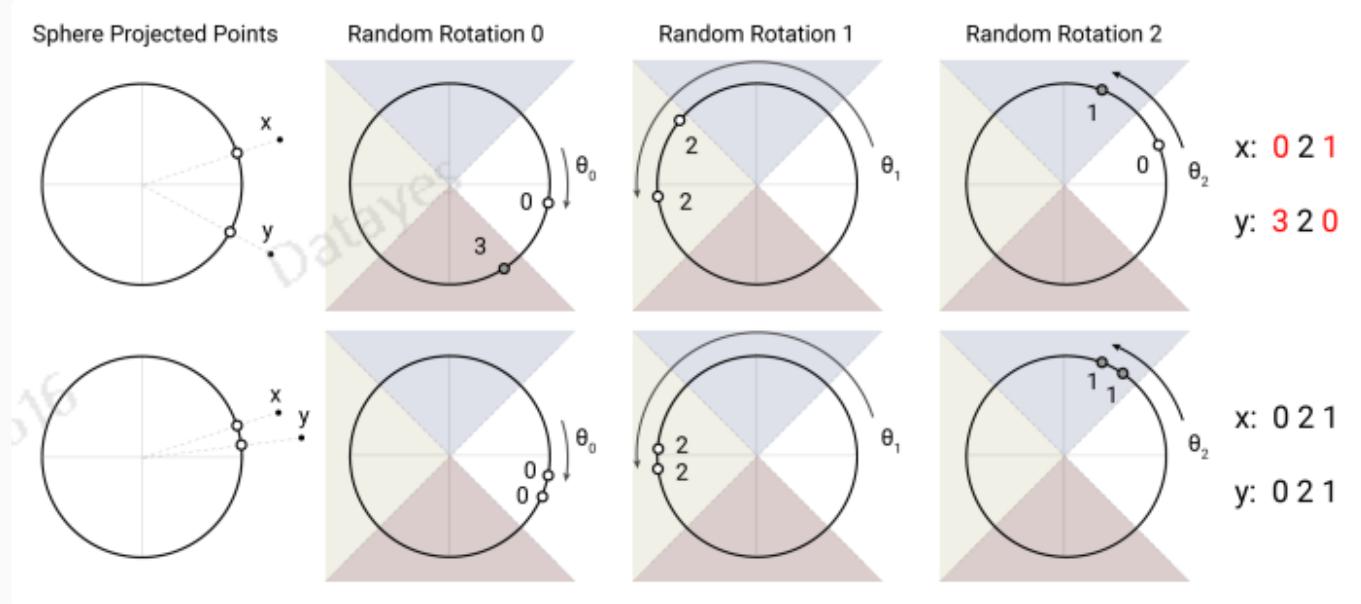
这样就把所有的点进行了分桶 (Bucket)，把相似的点以较大的概率分到同一个桶里。

作者希望把和最相似的 key 分到一个桶里，这通过 LSH 可以做到。

还希望每个桶里，key 的数量差不多，以方便做并行计算，这是后边分桶以后再分块来做到的。

LSH 具体的实现方式是 Angular LSH。

从图形上来看，是把点投影到一个二维球面上，那么在高维空间中相近的点，在二维球面上的位置也相近。



上面的两点x和y位置比较远，下面的两点位置就相对较近了。

把球面划分为4个区域，每个区域的值分别为0、1、2、3，点落在哪个区域，就得到相应的哈希值。

第一次旋转，上面的x落在了0的区域，即 $\text{hash}(x)=0$ ，上面的y落在了3的区域，即 $\text{hash}(y)=3$ ，而下面的x和y都落在了0的区域。

如此旋转3次，得到上面x的三个哈希值为 (0,2,1)，上面y的哈希值为 (3,2,0)，可见哈希值不同，对应到x和y的距离较远。

而下面 x和y 的哈希值都是 (0,2,1)，可见相近的点得到了相同的哈希值，分到了同样的哈希桶里。

从哈希函数的角度来看，如果哈希值有b位（上面的例子b为4，即0、1、2、3），那么定义一个随机旋转的参数矩阵为R，维度是，每次旋转得到的哈希值由以下公式获得（;表示拼接）：

所以旋转三次，是随机生成了三个参数矩阵，得到了三个哈希值，分桶更严格。

(3) LSH Attention

LSH Attention 是在 Decoder Self–Attention 的基础上改造过来的，因为它们都需要做 Mask。

首先看 Decoder Self–Attention。是可以计算 Attention Score 的所有 token 集合，也就是左边位置的 Token，是本来作为分母的归一化项：

$$o_i = \sum_{j \in \mathcal{P}_i} \exp(q_i \cdot k_j - z(i, \mathcal{P}_i)) v_j \quad \text{where } \mathcal{P}_i = \{j : i \geq j\}$$

对右边位置的 Token 做 Mask，可以通过减去正无穷来实现，因为负无穷经过 softmax 就得到了0：

$$o_i = \sum_{j \in \tilde{\mathcal{P}}_i} \exp(q_i \cdot k_j - m(j, \mathcal{P}_i) - z(i, \mathcal{P}_i)) v_j \quad \text{where } m(j, \mathcal{P}_i) = \begin{cases} \infty & \text{if } j \notin \mathcal{P}_i \\ 0 & \text{otherwise} \end{cases}$$

LSH Attention 把可以计算 Attention Score 的集合, 由左边位置的 Tokens: $\mathcal{P}_i = \{j : i \geq j\}$ 变为了和在同一个哈希桶中 key: $\mathcal{P}_i = \{j : h(q_i) = h(k_j)\}$

思考2: 对不在一个哈希桶内的做 Mask, 并不意味着不做计算, 而是把 $q_i * k_j$ 这一个向量乘法直接置为负无穷, 从而减少计算量。

但是做 LSH 之后, 可能出现两个问题:

- 每个哈希桶中向量的数量可能不平衡, 那么没法做 batch 并行计算
- 同一个桶中的 query 和 key 的数量可能不相等, 甚至出现一个桶中只有 query 没有 key 的情况, 没法做 Attention 计算

先来解决第2个问题, 如果 Q和K 相等, 那么同一个桶中 query 和 key 的数量一定相等。

于是作者采用了 「Shared-QK Attention」, 让 $h(k_j) = h(q_j)$ 。

再来解决第1个问题, 把哈希桶按照包含的 query 的数量进行排序, 每个桶内部再按照 query 的原始位置进行排序, 位置下标也由换成了。

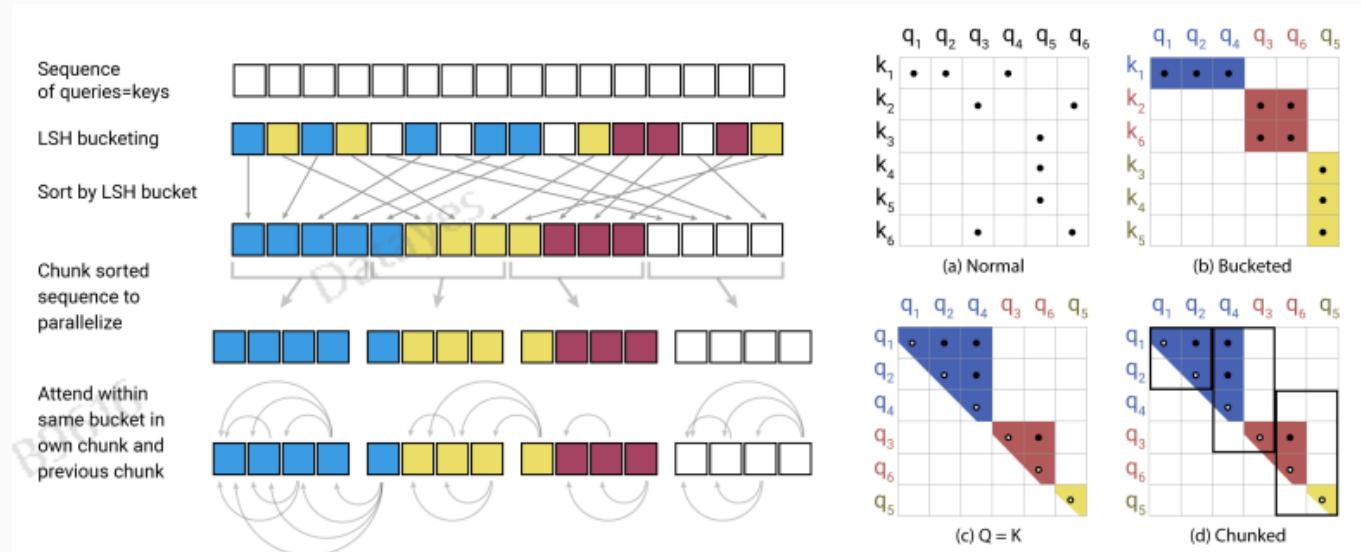
接着对所有 query 进行分块 (Chunk), 每个块包含相同数量的 query, 但是一个块可能会跨多个桶。由于同一个位置的 query 和 key 相同, 所以 query 和 query 之间进行计算 Score 就行。

在 Decoder Self–Attention 上做 LSH Attention, 和去计算 Score 的, 要满足2个条件:

- 的位置在的左边
- 和在同一个桶内

如果同一个桶的 query 被分到了两个块, 那只能是后一个块中的 query 和前一个块的 query 去计算, 即 look–back, 而不能 look–ahead。

下图就是在 Decoder Self–Attention 上做 LSH Attention 的示意图:



思考3: 分块的目的是为了做 batch 并行计算, 这意味着块与块之间必须是独立的。但是由于两个块可能包含同一个桶的 query, 块与块之间并非是完全独立的。

因此我认为并行计算也不是完全并行，而是分两步：先对每个块内部属于同一个桶的 query 计算 Attention，也就是并行计算，然后对跨块但是属于同一个桶的 query 做 look-back 的计算。

(4) Multi-round LSH attention

LSH 不是精准查找，而是近似查找，所以相似的 query 也可能被分到不同的哈希桶中。

于是对于 query 并行做多轮 LSH，每一轮都使用不同的哈希函数，然后把落入相同哈希桶的 query 求并集，提高召回率：

$$\mathcal{P}_i = \bigcup_{r=1}^{n_{\text{rounds}}} \mathcal{P}_i^{(r)} \quad \text{where } \mathcal{P}_i^{(r)} = \left\{ j : h^{(r)}(q_i) = h^{(r)}(q_j) \right\}$$

至此，Attention 改造完毕，复杂度由降低到。

思考4：为什么是 $O(L \log L)$ ？

三：改造残差连接

Transformer 中每一层的 Activations 需要存储起来，用于进行反向传播。

而存储 Activations 的内存占用随层数呈线性增长，考虑标准的 Transformer 有6层，每一层又有 Self-Attention 和 FFN 两个 sub-layer，也就是有12层的 Activations。

这是导致内存占用过大的一个重要原因。

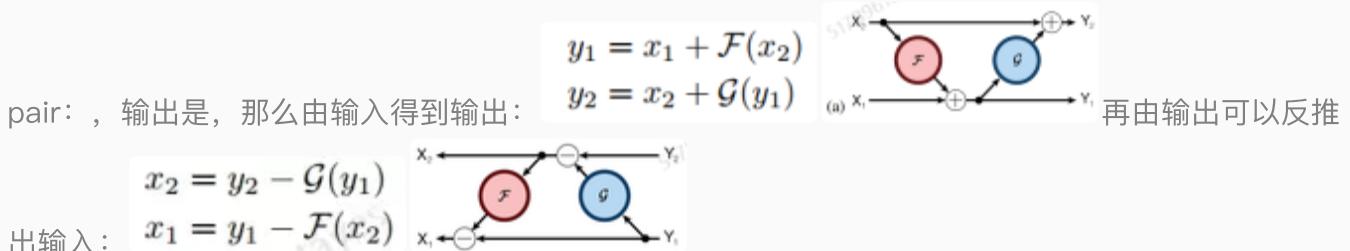
于是作者把 「RevNet (Reversible residual networks, 可逆残差网络)」 应用于 Transformer 中，来代替残差连接。

(1) RevNet

RevNet 通过下一层的 Activations 和模型参数来还原上一层的 Activations，因此只需要存储最外层的 Activations，就可以不断反推出前面每一层的 Activations。

这样一来 Activations 的内存占用就和层数无关了。

残差连接的公式是 $y = x + F(x)$ ，而和残差连接不同的是，RevNet 中一个可逆残差块将输入划分为一对



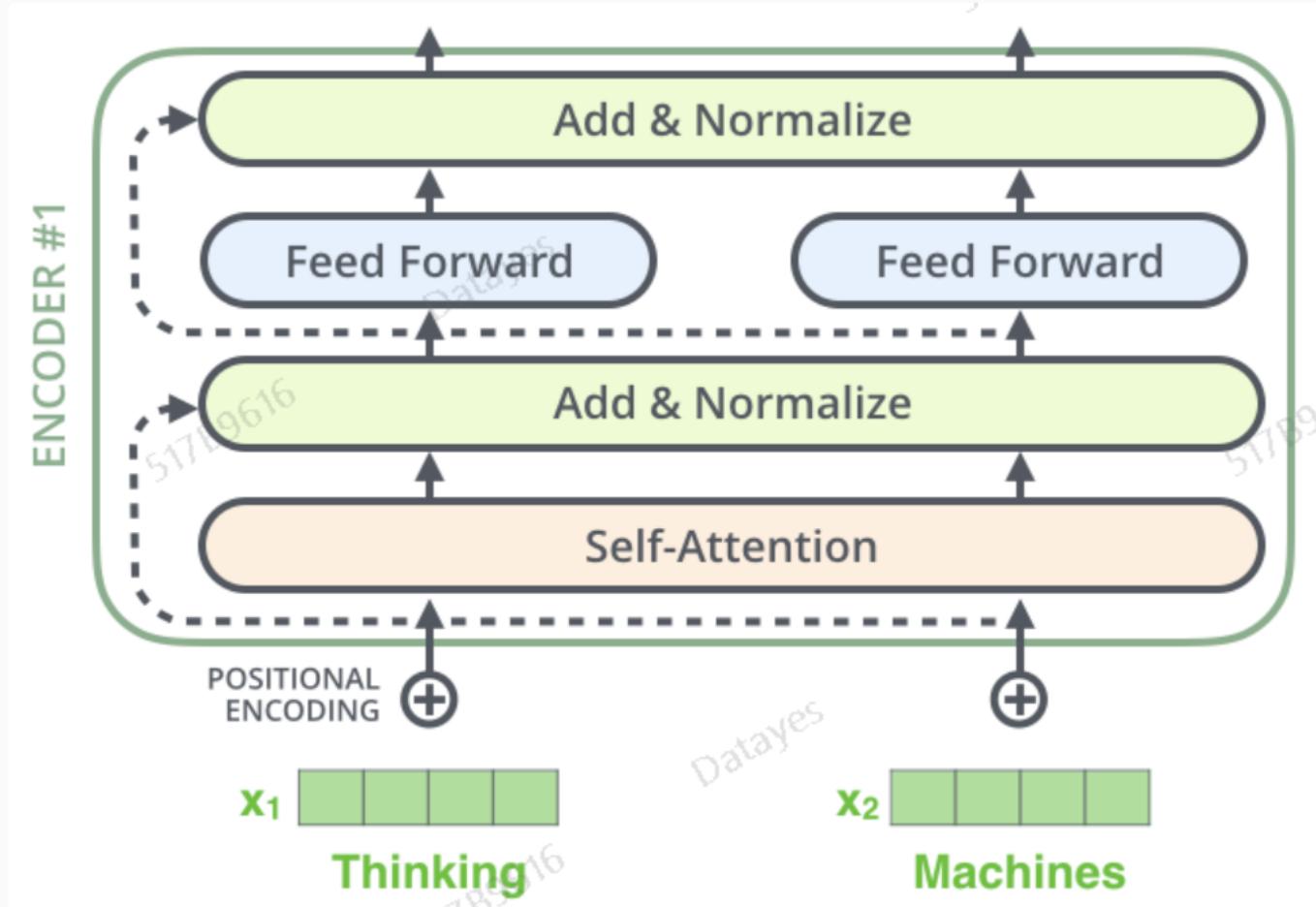
另外，RevNet 中的可逆残差连接不可以跳步，也就是 Stride=1，不然会存在不可逆层，需要额外存储 Activations。而 ResNet 则往往 Stride 非常大，Activations 都需要存储在内存中。[4]

思考5: RevNet 巧妙地把输入 x 一分为二: , 就可以由输出反推输入了, 而 ResNet 中的残差连接不行。

(2) Reversible Transformer

原本的残差连接非常简单, 用公式表示为: $y = x + F(x)$, 这个 $F(x)$ 就是 Self-Attention 或者 FFN 的操作。

也就是说 Self-Attention 和 FFN 是两个残差连接块。



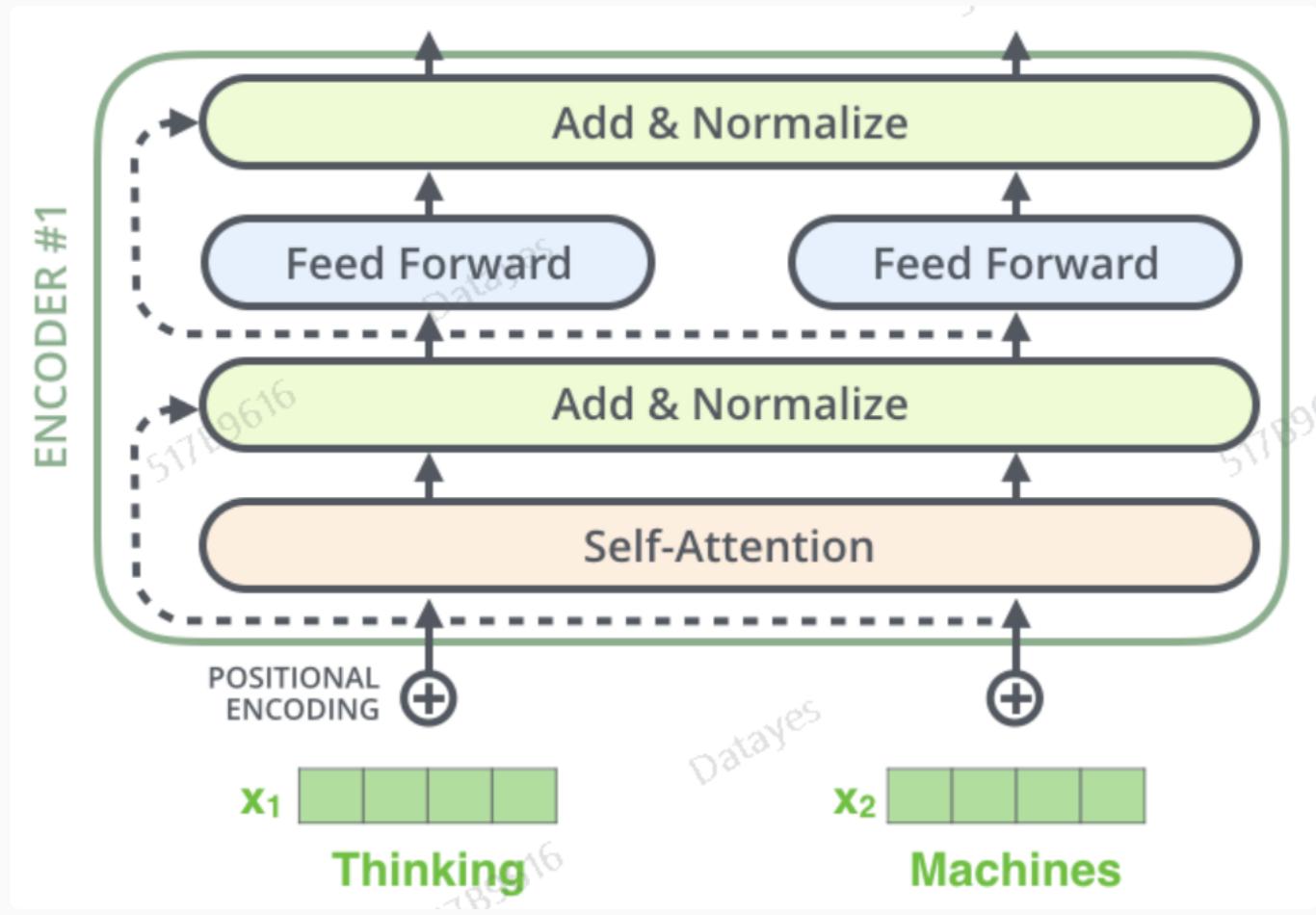
把可逆残差网络应用于 Transformer 则不一样, 一个可逆残差块就包含了 Self-Attention 和 FFN 两个 sub-layer, 对应到上面的F和G两个函数: $Y_1 = X_1 + \text{Attention}(X_1)$ $Y_2 = X_2 + \text{FeedForward}(Y_1)$ 而且 Layer Norm 由残差连接的外部操作变成了可逆残差连接的内部操作:

Note that Layer Normalization is moved inside the residual blocks.

四：改造 FFN 的计算

标准的 Transformer 中, FFN 是由两个全连接层组成, 第一个全连接层的输出和第二个全连接层的输入的维度是2048, 远大于 Self-Attention 的输出维度, Bert 中的更是高达3072, 因此 FFN 也是内存占用高的一个重要原因。

不过好在 FFN 的计算和 Self-Attention 不同，输入序列的 Token 之间是独立，这从下图可以看出：



于是可以把经过 Self-Attention 计算后得到的输入序列进行分块，牺牲并行计算，每次只计算一个分块，同样以时间换空间，减少内存占用：

$$Y_2 = \left[Y_2^{(1)}; \dots; Y_2^{(c)} \right] = \left[X_2^{(1)} + \text{FeedForward}(Y_1^{(1)}); \dots; X_2^{(c)} + \text{FeedForward}(Y_1^{(c)}) \right]$$

把FFN分块和可逆残差连接结合，由输出反推输入的时候也要进行分块处理。

五：实验结果

作者在NLP和CV上都做了实验，分别是 enwik8–64K 和 imagenet64。

首先来看 Multi-round LSH attention 的效果，当 LSH 的轮数为8时，准确率就非常接近 Full Attention了：

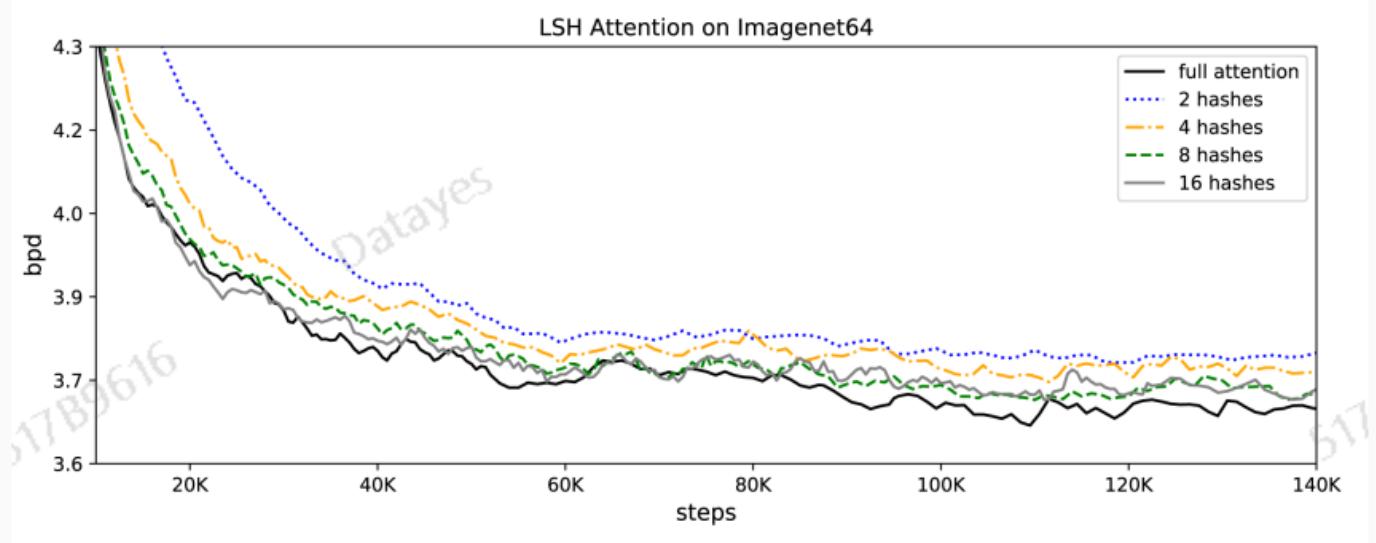
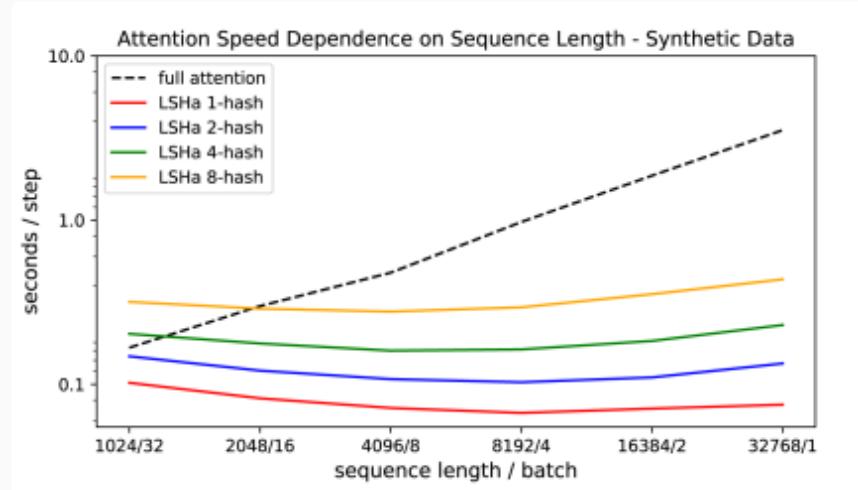


Figure 4: LSH attention performance as a function of hashing rounds on imagenet64.

从速度上来看，随着输入序列长度的增加，Full Attention 的耗时也直线上升，而 LSH Attention 却保持稳定：



而 Shared-QK 和 Reversible Transformer 在减少内存占用的同时，模型的准确率却没有明显下降：

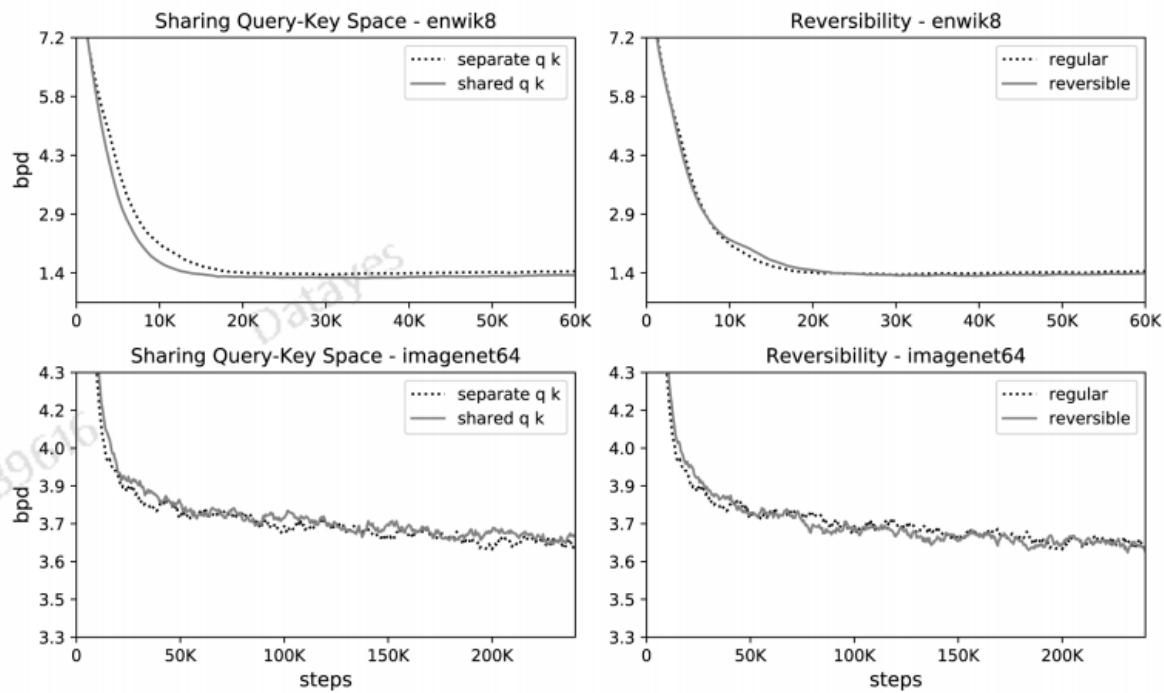


Figure 3: Effect of shared query-key space (left) and reversibility (right) on performance on enwik8 and imagenet64 training. The curves show bits per dim on held-out data.

Star-Transformer

<https://arxiv.org/pdf/1902.09113.pdf>



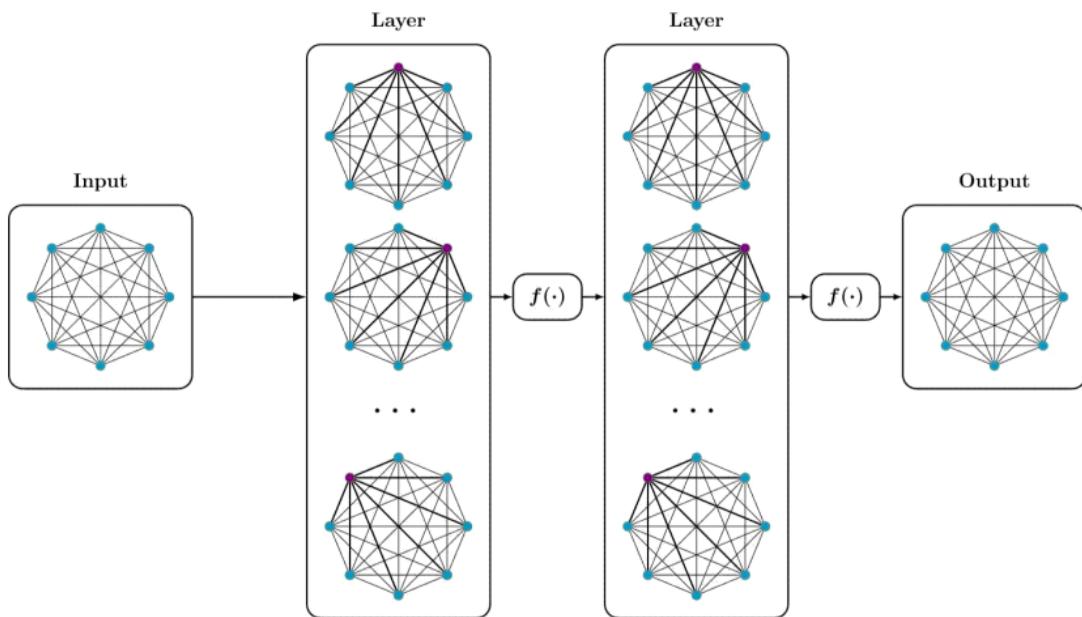
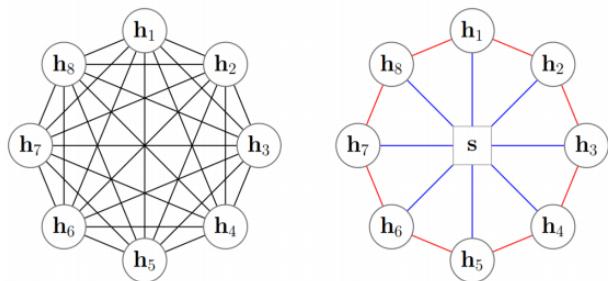
Star-Transformer

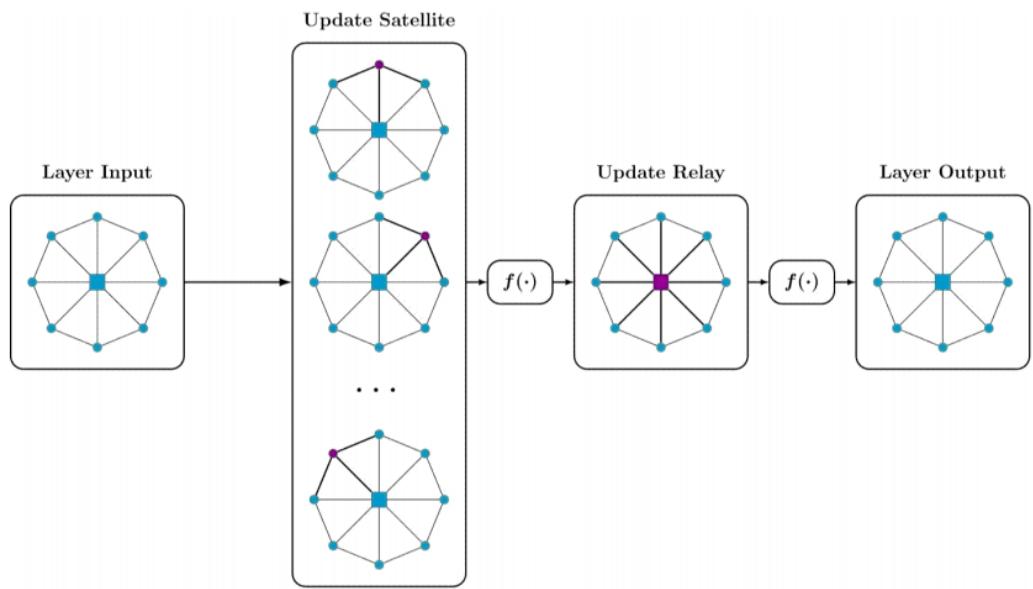
▶ Transformer

- ▶ 全连接
- ▶ 复杂度: $O(L^2d)$
 - ▶ 无法处理长文档
- ▶ 容易过拟合

▶ Star-Transformer

- ▶ 复杂度: $O(2Ld)$
- ▶ 引入局部性先验
 - ▶ 不需要Position Embedding
- ▶ 适用于小规模或中等规模数据





Transformer(上)Star–Transformer(下)

Transformer-XL(当前最优自回归模型)->XLNET

🔗 [Transformer-xl.pdf](#)

Motivation

To address the limitation of fixed-length contexts, we propose a new architecture called Transformer–XL. We introduce the notion of recurrence into our deep self–attentionnetwork.

In particular, instead of computing the hidden states from scratch for each new segment, we reuse the hidden states obtained in previous segments. The reused hidden states serve as memory for the current segment, which builds up a recurrent connection between the segments.

BPT: BP-Transformer

<https://arxiv.org/abs/1911.04070>

Transformer-XL只能在一个方向上建模，很难处理双向信息

Star-Transformer不适用于自回归模型

Introduction

the quadratic complexity of self-attention limit its application on long text.解决方式是 (1)

Hierarchical Transformers (2) Lightweight Transformers

Besides the computational cost, the fully-connected nature of Transformer does not incorporate the common sensible inductive bias of language, such as sequential or syntax structure.

BP-Transformer (BPT)



- ▶ Transformer's self-attention mechanism
 - ▶ A quadratic cost with respect to sequence length, limiting its wider application, especially for long text.
- ▶ Improvements:
 - ▶ Hierarchical Transformers
 - ▶ use two Transformers in a hierarchical architecture
 - ▶ Lightweight Transformers
 - ▶ reduce the complexity by reconstructing the connections between tokens

BPT incorporates the advantages of both the hierarchical and lightweight Transformers.

什么是BPT？

BP-Transformer (BPT for short), which partitions the input sequence into different multi-scale spans via binary partitioning (BP). BPT incorporates an inductive bias of attending the context information from fine-grain to coarse-grain as the relative distance increases. The farther the

context information is, the coarser its representation is. BPT can be regarded as graph neural network, whose nodes are the multi-scale spans.

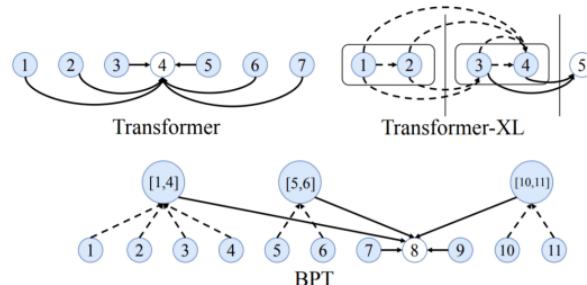
to better represent the position information of the span nodes and token nodes, we generalize the notion of **relative position from sequences to trees** and show that it better captures position bias.

Thus, BPT incorporates the **advantages of both hierarchical and lightweight Transformers**: (1) it models the long-range context in an hierarchical fashion, (2) reduces computation cost with fewer edges, and finally, (3) introduces coarse-to-fine connections to approximate the reasonable inductive bias of language, with a net effect of making BPT easier to train.

BP-Transformer (BPT)



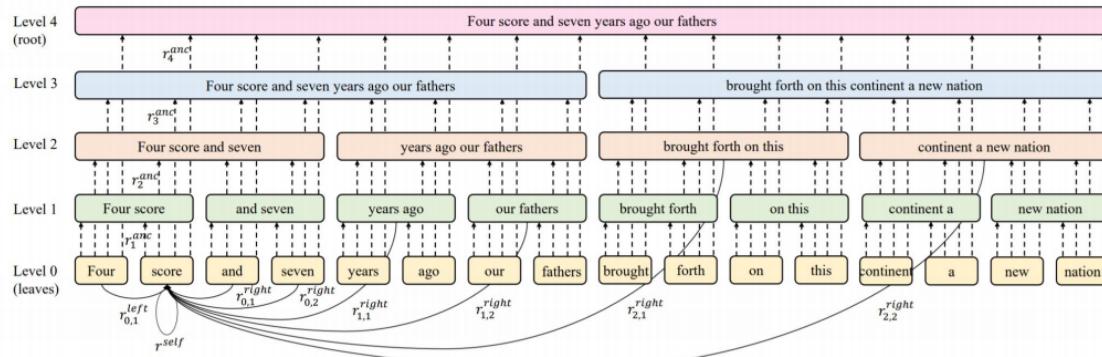
- ▶ BPT incorporates the advantages of both the hierarchical and lightweight Transformers.
 - ▶ model long-range context in an hierarchical fashion
 - ▶ reduce computation cost with fewer edges
 - ▶ introduce coarse-to-fine connections to approximate the reasonable inductive bias of language
 - ▶ make BPT easier to train



BP-Transformer (BPT)



Binary Partitioning



Multi-Scale Self-Attention

论文: <https://arxiv.org/abs/1912.00544>

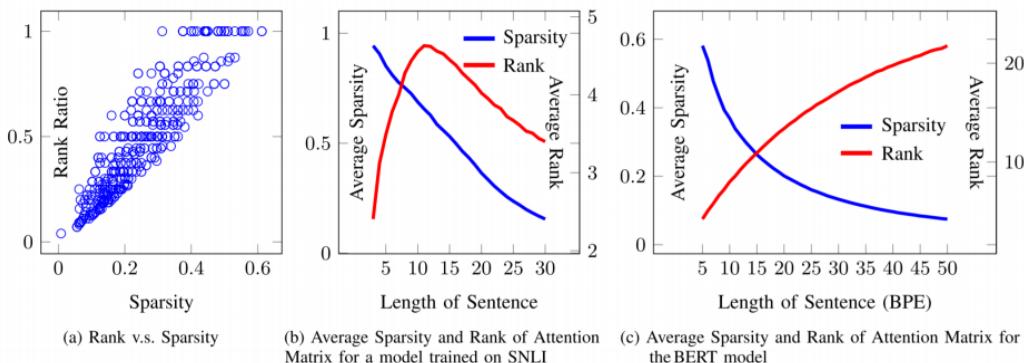
Low-Rank and Locality Constrained Self-Attention

Three Properties From Linguistic Viewpoint



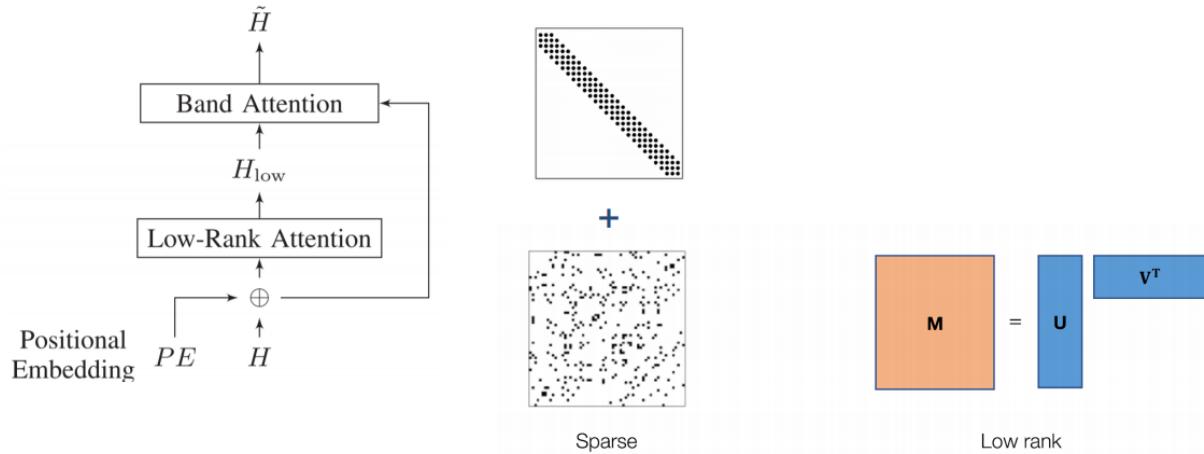
- ▶ Sparsity
 - ▶ Intuitively, the number of dependent relations between tokens should be far lower than L2.
- ▶ Locality
 - ▶ Most of the dependency relations occur in the adjacent tokens.
- ▶ Low-Rank
 - ▶ Detaching the local relations, the remaining long-range relations usually many-to-one relations.

Model Analysis



Statistics of the learned attention matrix of a Transformer which trained on SNLI and the BERT model.

Model Design



Results

Model	Acc
BiLSTM [26]	49.8
Tree-LSTM [27]	51.0
CNN-Tensor [28]	51.2
Emb + self-att [29]	48.9
BiLSTM + self-att [30]	50.4
CNN + self-att [30]	50.6
Dynamic self-att [30]	50.6
DiSAN [29]	51.7
Transformer	50.4
Ours	52.0

results on the SST dataset

Model	Acc
sentence-vector based methods:	
BiLSTM [32]	83.3
BiLSTM + self-att [32]	84.2
SPINN-PI [18]	83.2
Tree-based CNN [33]	82.1
BiLSTM-max [34]	84.5
Residual encoders [35]	86.0
DiSAN [29]	85.6
Gumbel TreeLSTM [36]	86.0
Reinforced self-att [37]	86.3
Multiple DSA [30]	87.4
Transformer	81.5
Ours	85.7
interaction methods:	
BiLSTM+intra-att [32]	85.0
LSTMN+deep att [38]	86.3
BiMPM [39]	87.5
DIIN [40]	88.0
BCN [41]	88.1
ESIM + ELMo [42]	88.7
Pretrained Transformer[3]	89.9
Transformer	82.3
Ours	88.2

results on the SNLI dataset



External

Attention(Attention(x)=Linear(Norm(Linear(x))))&Do You Even Need Attention?

论文: <https://arxiv.org/pdf/2105.02723v1.pdf>

<https://arxiv.org/pdf/2105.02358.pdf>

参考: <https://www.cnblogs.com/liuyangcode/p/14760367.html>

External Attention

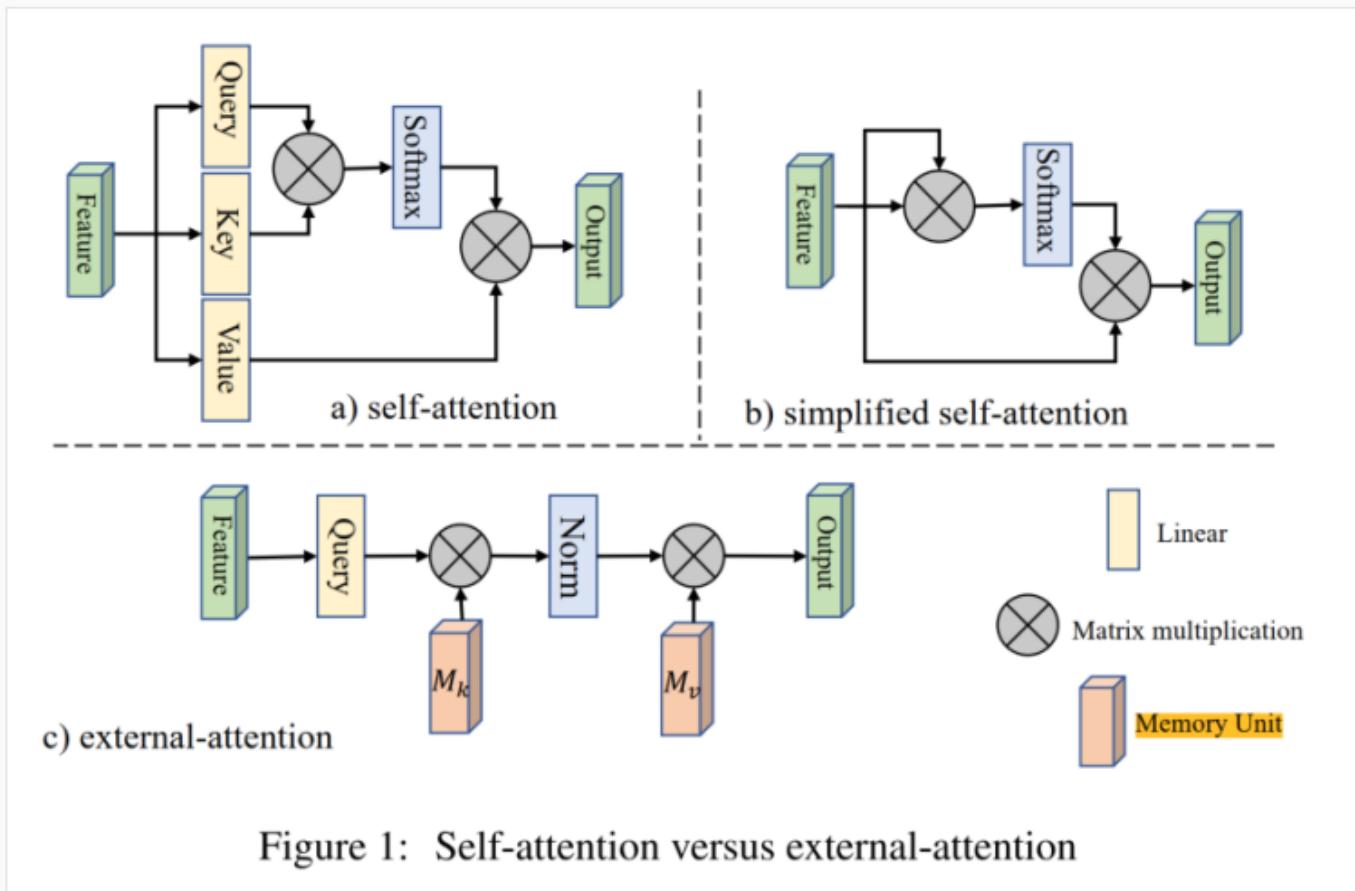


Figure 1: Self-attention versus external-attention

- 我们是在self-attention的基础上来实现external-attention的，对于一个输入特征图 $F \in \mathbb{R}^{N \times d}$, N 是像素数量, d 是特征维数, self-attention首先通过线性变化将其变成
 - query matrix $Q \in \mathbb{R}^{N \times d'}$
 - key matrix $K \in \mathbb{R}^{N \times d'}$
 - value matrix $V \in \mathbb{R}^{N \times d}$
 - self-attention可以描述成

$$\begin{aligned} A &= (\alpha)_{i,j} = \text{softmax}(QK^T) \\ F_{out} &= AV \end{aligned}$$

- 对于简化版的self-attention, 就直接利用特征图来计算attention, 可以描述成

$$\begin{aligned} A &= (\alpha)_{i,j} = \text{softmax}(FF^T) \\ F_{out} &= AF \end{aligned}$$

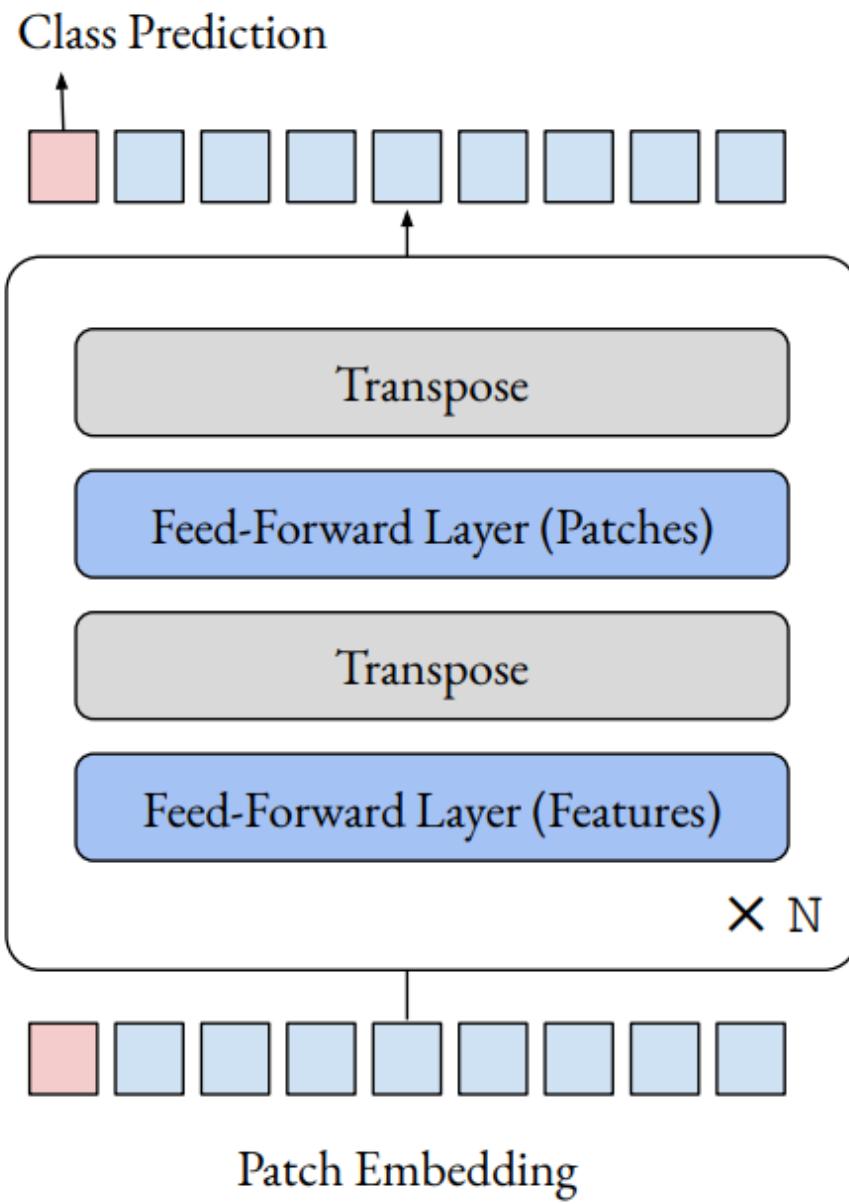
- 对attention进行可视化之后可以发现大多数像素点只和一部分像素点有关, 没有必要计算 $N \times N$ 的attention map, 所以我们提出了额外attention模块, 通过计算像素与额外的存储单元 $M \in \mathbb{R}^{S \times d}$ 得到attention, 用和self-attention相似的方法进行归一化, M 是一个独立于输入可学习的参数, 作为整个数据集的存储单元, 并且我们用不同的两个 M 分别作为key和value, 所以整个external attention可以描述成

$$\begin{aligned} A &= (\alpha)_{i,j} = \text{Norm}(FM_k^T) \\ F_{out} &= AM_v \end{aligned}$$

- d 和 S 都属于超参, 一个较小的 S 在实验中表现就很不错, 所以external attention比self-attention更有效, 应用在了多种任务上, 论文中给出的伪代码如下图

Do You Even Need Attention?

我们问：关注层有必要吗？具体地说，我们将视觉转换器中的关注层替换为应用在patch维度上的前馈层。由此产生的体系结构只是一系列以交替方式应用于patch和特征维度上的前馈层。在ImageNet上的实验中，该架构表现得出人意料地好：ViT/DeiT-Base大小的模型获得了74.9%的TOP-1准确率，而ViT和DeiT的准确率分别为77.9%和79.9%。这些结果表明，视觉转换器的注意力以外的其他方面，如patch embedding，可能比之前认为的更应该对它们的强劲表现负责。



BPE解决OOV（Out-Of-Vocabulary）问题

原文：<https://arxiv.org/abs/1508.07909>

参考：https://blog.csdn.net/weixin_38937984/article/details/101723700

OOV (Out-Of-Vocabulary) 问题：在数据集中出现次数极少的词的词向量质量较低^{^1}。如果我们使用预训练的词向量，而同时在实际数据中出现了未登录词，我们只能不得不用一个<UNK>标记去指代这个词，这样的词一旦很多，就必然会严重影响模型的结果^{^2}。

有时相对于以一个word作为基本单位整体去翻译，这种通过subword来翻译则显得更有效率和意义。如何将word切分成合适的subword呢？论文中提出了采用Byte pair encoding (BPE)压缩算法，首先以字符划分，然后再合并。也就是不断的以出现频次最高的2-gram进行合并操作，直到打到词表大小为止。这种以频次合并是比较符合常识的，例如像'er','ing,'ed'这样比较有意义的后缀，'e'和'r'同时出现的频次应该比较多。“ing”和“ed”类似道理。将稀有词划分成subword，能比较好的处理oov问题。例如训练集中大量出现“runner”和“thinking”，那按word-level词频来说，word-level词表中应该包含这两个词，此时词“running”只出现一次，则该词很有可能是oov，但是如果以subword切词，则subword词表中应该含有“run”，“er”，“think”，“ing”，那么对于“running”，其subword切词后为“run”，“ing”均在词表中。

通过对稀有词进行适当的切分，得到subword units，机器翻译模型能更好的处理透明翻译，并且能生成一些unseen words。

机器翻译模型通常都会用到注意力机制，在word-level模型中，模型每次只能计算在word级别上的注意力，我们希望模型可以在每一步学习中将注意力放在不同的subword上，显然这样更有意义和效率。

BPE的改进算法BPEmb

<https://arxiv.org/pdf/1710.02187.pdf>

mosesdecoder数据预处理

https://blog.csdn.net/weixin_38937984/article/details/103995209?spm=1001.2014.3001.5501

NLP机器翻译评价度量

(BLEU, ROUGE, METEOR, ROUGE)

参考: <https://www.jianshu.com/p/1517a75af993>

LSTM结合知识蒸馏

https://mp.weixin.qq.com/s/cPCpWz31Aad2_vS0qFmhEQ

BERT的发展

参考

前言

BERT是在Google论文《BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding》中被提出的，是一个面向NLP的无监督预训练模型，并在多达11个任务上取得了优秀的结果。这个模型的最大意义是使得NLP任务可以向CV一样使用与训练模型，这极大的方便了一个新的任务开始，因为在NLP领域，海量数据的获取还是有难度的。

模型概述：BERT是一个无监督的NLP与训练模型，结构上是Transformer的编码部分，每个block主要由多头self-Attention、标准化(Norm)、残差连接、Feed Forward组成。在具体任务中，主要分为模型预训练和模型微调两个阶段。在模型预训练阶段，因为模型参数巨大，通常是上千万乃至上亿的数量级，所以需要大量的数据训练，所幸这时候模型是无监督的，只需要爬取或使用开源数据集即可；在模型微调阶段，需要针对具体的任务来微调模型，已达到较好的效果。

1. 模型整体结构

Bert就是Transformer的编码部分，下图是Transformer的具体结构：

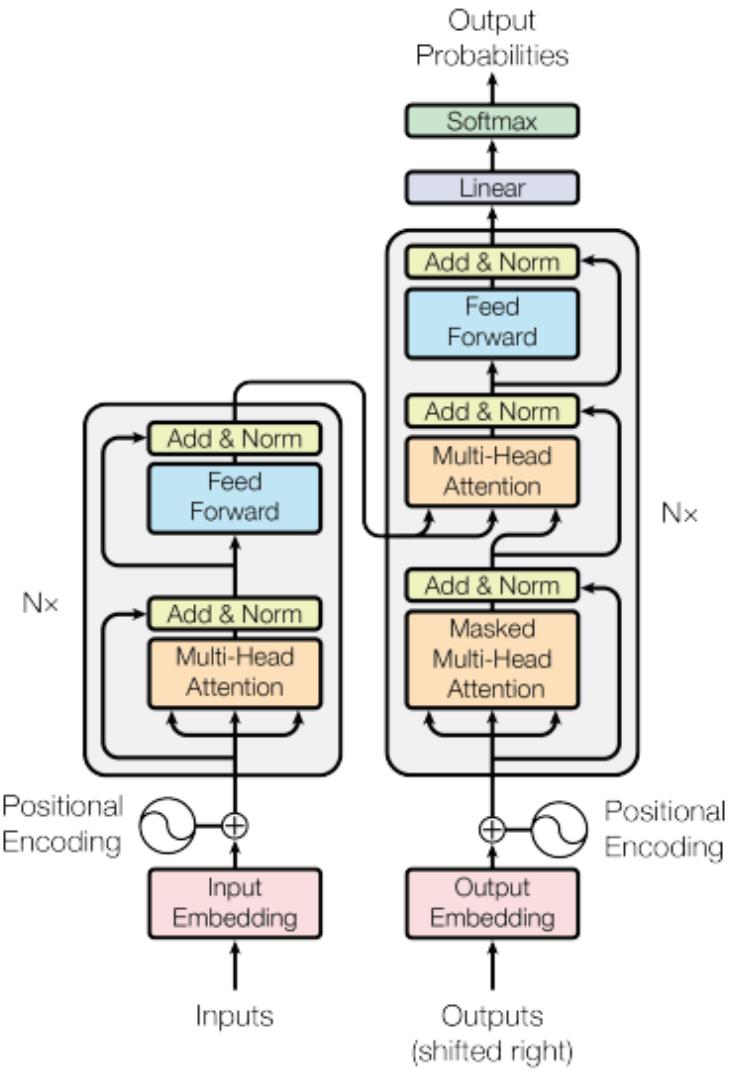


Figure 1: The Transformer - model architecture.

左侧为Transformer的编码部分，右侧为Transformer的解码部分，主要以编码部分详细讲解Bert的结构。

左侧的编码部分包括输入，添加位置编码，以self-Attention、Add&Norm、Feed Forward的block。

2. 位置编码

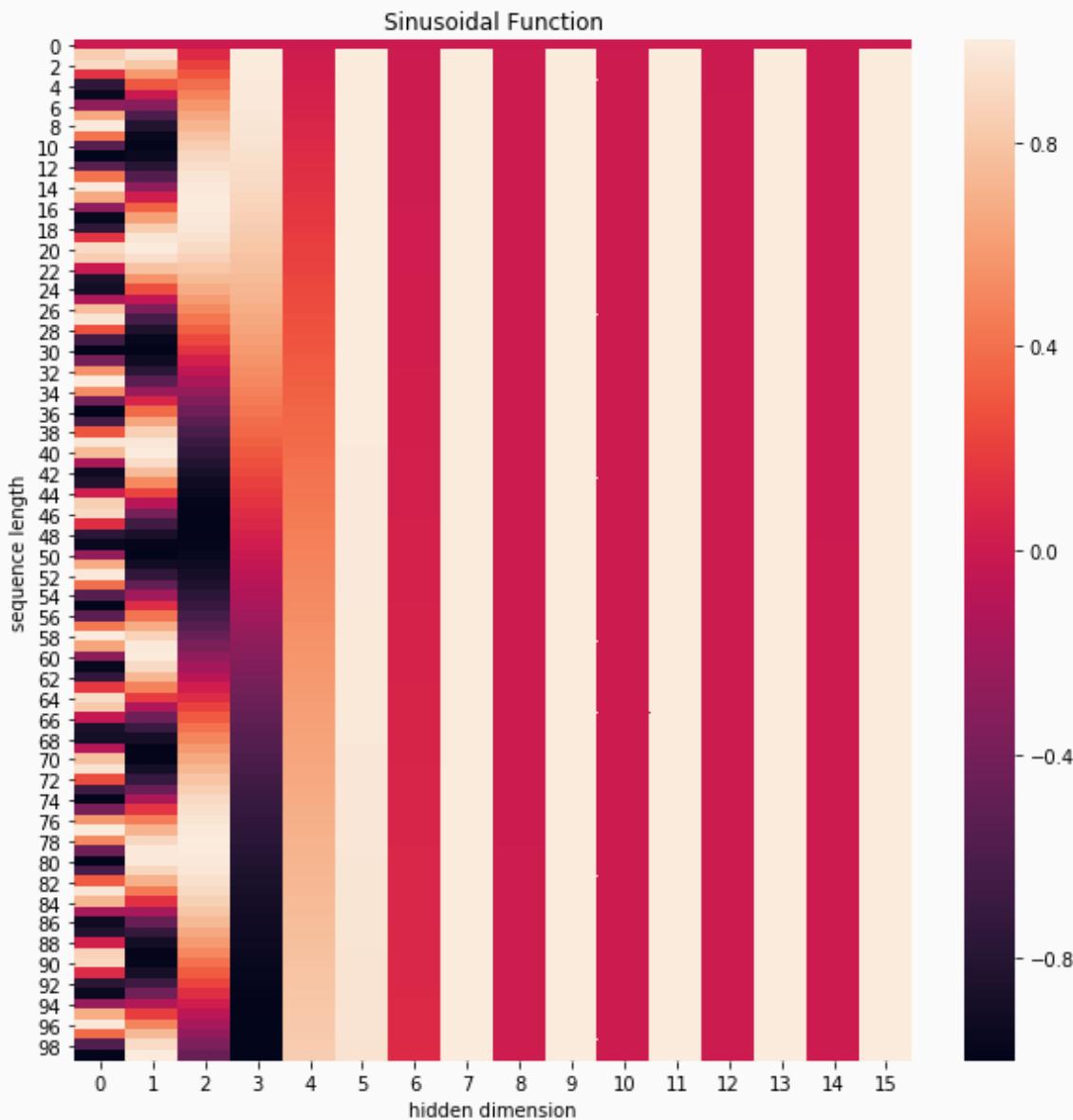
位置编码是用来捕获文本之间的时序关联性的，例如打开现在热度第一的新闻的第一句话：“重庆主城区一栋30层的居民楼发生大火，造成百余名群众被困，重庆市政府迅速调集消防、公安、卫生等数百名人员赶赴现场施救。”其中，“重庆市”与“主城区”相关度最高，位置最近。当对NLP文本处理时，位置更近的文本一般相关性更大，所以将位置编码融入到数据中是很有必要的。需要说明的是与Bert这种全部基于Attention不同的是，之前基于RNN的模型在模型结构上已经可以将这种时序信息考虑在内。

在具体处理方式上，采用的是Embedding+Positional的方法，将数据之间的关联性融入到数据中。Embedding是嵌入到相应维度的文本数据，Positional在论文中使用了sinesine和cosinecosine函数的线性变换来提供模型的位置信息，公式如下：

$$PE(pos, 2i) = \sin(pos/100002i/dmodel)$$

$$PE(pos, 2i+1) = \cos(pos/100002i/dmodel)$$

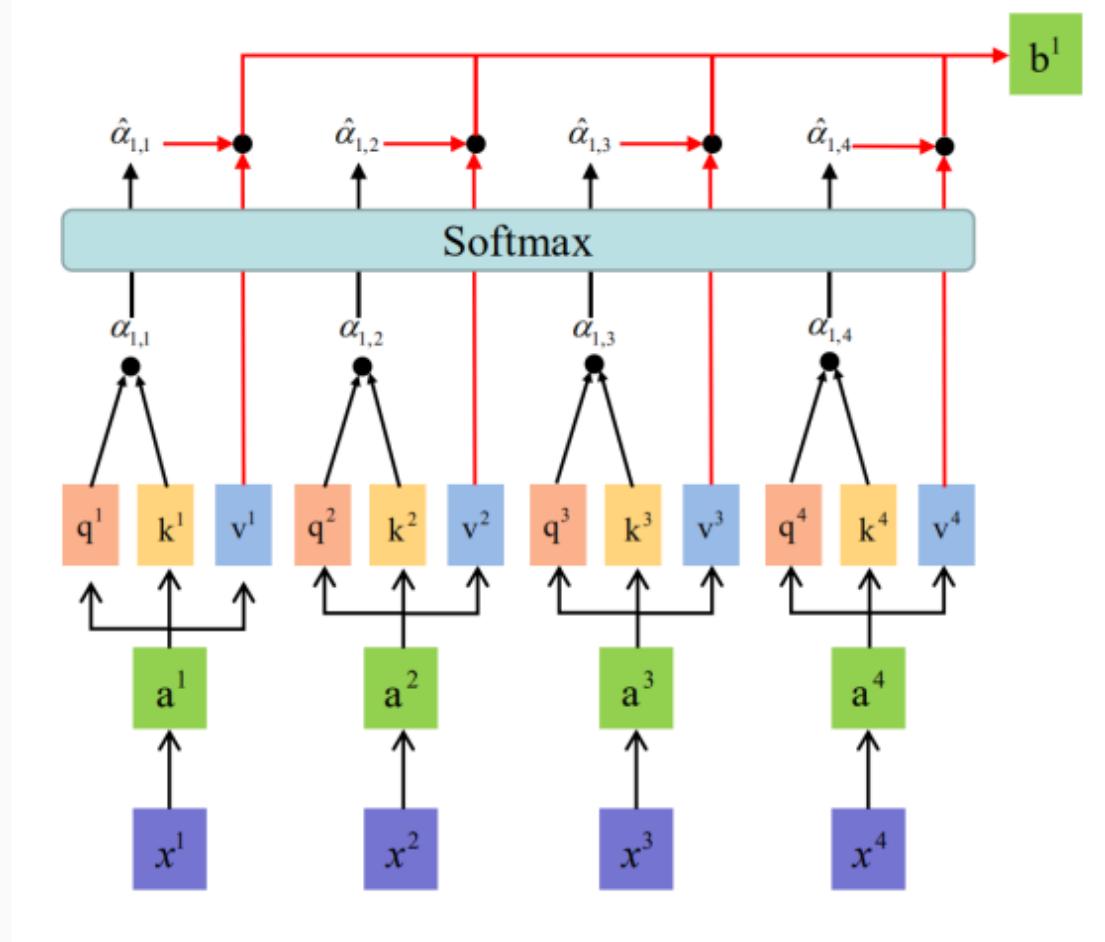
那为何加了位置编码就能获取数据间位置的特征呢？在self-attention的结构中，在对每维数据计算权重时，是采用点积的形式，本质上就是计算向量之间的相关性。而位置编码将临近的数据加上频率接近的位置编码，就是增加了相邻数据的相关性。下图是位置编码向量的热图，可以看出距离越近，频率就更加接近。



3. self-Attention

self-attention是BERT的重要思想，其与位置编码结合，解决了文本数据的时序相关性的问题，从而一举结束了依靠RNN、LSTM、GRU等之前一直用来解决时序问题的网络模型。**self-attention**通俗的说就是信息向前传播时动态的计算权重的一种方式，与CNN常见的MaxPooling、MeanPooling不同的

是，attention模型是经过训练，当不同信息传入时，自动的调整权重的一种结构。self-attention的具体结构如下图所示：



具体的，将上图的过程进行详细的解释，主要是拆分成4个步骤：

1) x^1, x^2, x^3, x^4 代表的是经过embedding的4条时序文本信息，首先将4条信息加上位置向量，得到 a^1, a^2, a^3, a^4 ，这样做的目的上文已经说过，是为了获取文本的时序相关性。

2) 对每条信息分配三个权重 WQ, WK, WV ($\text{embed.dim} * \text{embed.dim}$)，分别与 a^1, a^2, a^3, a^4 相乘后形成3个矩阵 Q, K, V 也就是上图的 q^i, k^i, v^i 。

$$Q = \text{Linear}(a^1) = a^1 WQ \quad K = \text{Linear}(a^1) = a^1 WK \quad V = \text{Linear}(a^1) = a^1 WV$$

3) 将 q^1 分别与 k^1, k^2, \dots, k^4 点乘，得到 a^1, i ，再有softmax的计算公式，计算得 a^1, i 。

$$a^1, i = q^1 * k^i \quad a^1, i = \exp(a^1, i) / \sum_j \exp(a^1, j)$$

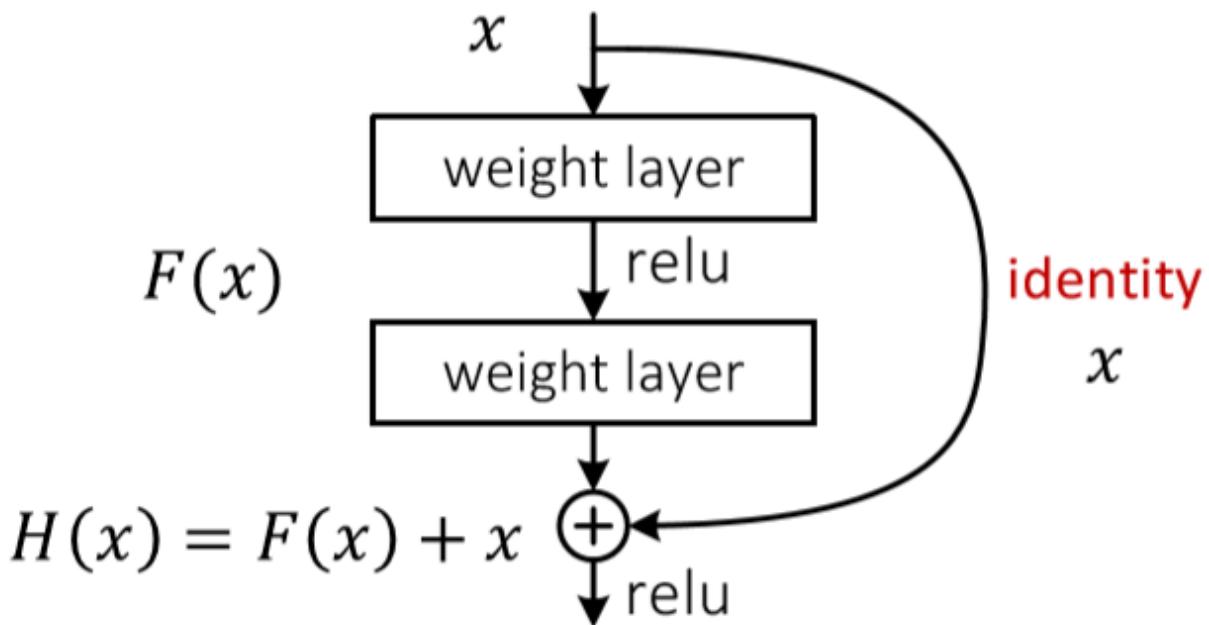
4) 最后按照softmax输出的权重对 V 进行加权，计算得 b^1 。使用同样的方法计算得 b^2, b^3, \dots, b^4 。将 $b^1, b^2, b^3, \dots, b^4$ 进行合并，完成self-attention。

$$b_1 = \sum i \alpha_{1,i} * v_i$$

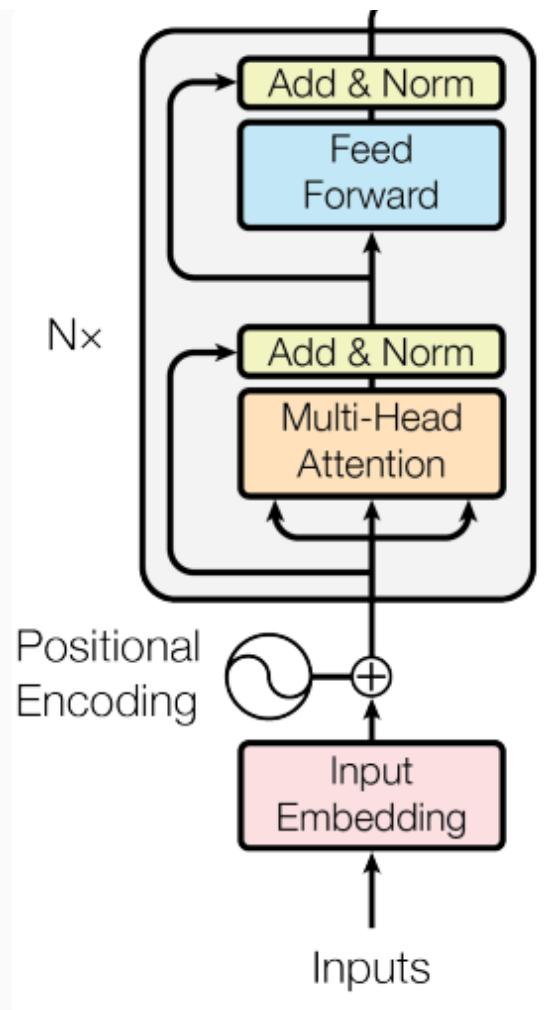
4. 残差连接

残差连接是训练深层模型时惯用的方法，主要是为了避免模型较深时，在进行反向传播时，梯度消失等问题。具体实现时，当网络进行前向传播时，不仅仅时按照网络层数进行逐层传播，还会由当前层隔一层或多层向前传播，如下图所示：

• Residual net



5. 模型实现



以上是BERT的整体结构，Input输入的是文本数据，经过Embedding加上位置向量Positional Encoding。Multi-Head Atention为多头的self-Attention，实际上就是将self-attention的Q、K、V均分成n份，分别进行计算。Add&Norm为残差计算和标准化；Feedward为全连接层，进行前向传播。其中Nx为基本单元的个数，是可以条调整的超参数。

6. Bert模型预训练策略

在预训练Bert模型时，论文提供了两种策略：

(1) Masked LM

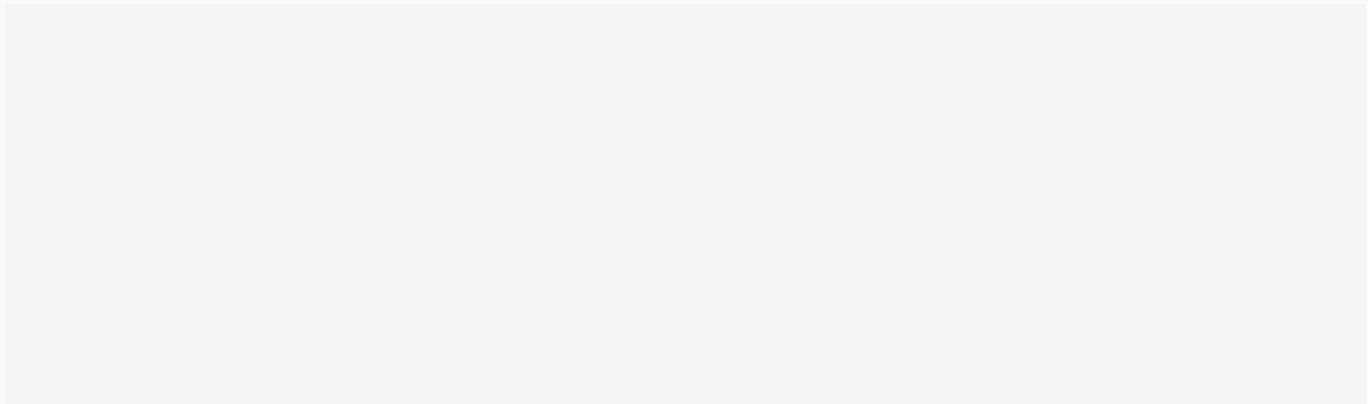
在BERT中，Masked LM(Masked language Model)构建了语言模型，这也是BERT的预训练中任务之一，简单来说，就是随机遮盖或替换一句话里面任意字或词，然后让模型通过上下文的理解预测那一个被遮盖或替换的部分，之后做Loss的时候只计算被遮盖部分的Loss，其实是一个很容易理解的任务，实际操作方式如下：

1. 随机把一句话中 $token$ 替换成以下内容：
 - 这些 $token$ 有 $[mask]$ ；
 - 有10%的几率被替换成任意一个其他的 $token$ ；

- 有10%的几率原封不动.
2. 之后让模型预测和还原被遮盖掉或替换掉的部分, 模型最终输出的隐藏层的计算结果的维度是:
 $X_{hidden} : [batch_size, seq_len, embedding_dim]$ 我们初始化一个映射层的权重 W_{vocab} :
 $W_{vocab} : [embedding_dim, vocab_size]$ 我们用 W_{vocab} 完成隐藏维度到字向量数量的映射, 只要求 X_{hidden} 和 W_{vocab} 的矩阵乘(点积): $vocab_size$ 的和为1, 我们就可以通过 $vocab_size$ 里概率最大的字来得到模型的预测结果, 就可以和我们准备好的 $Label$ 做损失($Loss$)并反传梯度了. 注意做损失的时候, 只计算在第1步里当句中随机遮盖或替换的部分, 其余部分不做损失, 对于其他部分, 模型输出什么东西, 我们不在意.

(2) Next Sentence Predict(NSP)

1. 首先我们拿到属于上下文的一对句子, 也就是两个句子, 之后我们要在这两段连续的句子里面加一些特殊 $token$: $[cls][cls]$ 上一句话, $[sep][sep]$ 下一句话 $[sep][sep]$. 也就是在句子开头加一个 $[cls]$, 在两句话之中和句末加 $[sep]$, 具体地就像下图一样:



我们看到上图中两句话是 $[cls]my dog is cute[sep]he likes playing[sep]$, $[cls]$ 我的狗很可爱 $[sep]$ 他喜欢玩耍 $[sep]$, 除此之外, 我们还要准备同样格式的两句话, 但他们不属于上下文关系的情况; $[cls]$ 我的狗很可爱 $[sep]$ 企鹅不擅长飞行 $[sep]$, 可见这属于上下句不属于上下文关系的情况; 在实际的训练中, 我们让上面两种情况出现的比例为 1:1, 也就是一半的时间输出的文本属于上下文关系, 一半时间不是.

我们进行完上述步骤之后, 还要随机初始化一个可训练的 $segment embeddings$, 见上图中, 作用就是用 $embeddings$ 的信息让模型分开上下句, 我们一把给上句全 0 的 $token$, 下句全 1 的 $token$, 让模型得以判断上下句的起止位置, 例如: $[cls]$ 我的狗很可爱 $[sep]$ 企鹅不擅长飞行 $[sep]$ 0 0 0 0 0 0 1 1 1 1 1 1 1 上面 0 和 1 就是 $segment embeddings$.

注意力机制就是, 让每句话中的每一个字对应的一条向量里, 都融入这句话所有字的信息, 那么我们在最终隐藏层的计算结果里, 只要取出 $[cls]token$ 所对应的一条向量, 里面就含有整个句子的信息, 因为我们期望这个句子里面所有信息都会往 $[cls]token$ 所对应的一条向量里汇总: 模型最终输出的隐藏层的计算结果的维度是: 我们 $X_{hidden} : [batch_size, seq_len, embedding_dim]$ 我们要取出 $[cls]token$ 所对应的一条向量, $[cls]$ 对应着 seq_len 维度的第 0 条: $cls_vector = X_{hidden}[:, 0, :]$ $cls_vector \in \mathbb{R}^{batch_size, embedding_dim}$ 之后我们再初始化一个权重, 完成从 $embedding_dim$ 维度到 1 的

映射, 也就是逻辑回归, 之后用 *sigmoid* 函数激活, 就得到了而分类问题的推断. 我们用 \hat{y} 来表示模型的输出的推断, 他的值介于(0, 1)之间: $\hat{y} = \text{sigmoid}(\text{Linear}(\text{cls_vector}))$ $\hat{y} \in (0, 1)$

ELECTRA (MLM改为replaced token detection)

原文: <https://openreview.net/pdf?id=r1xMH1BtvB>

本文最主要是将Masked Language Model的方式改为了replaced token detection, 采用generator和discriminator的结合方式, 使用了weight sharing的方式将generator的embedding的信息共享给discriminator, 在fine-tuning 的时候, 丢弃generator, 只使用discrinator。

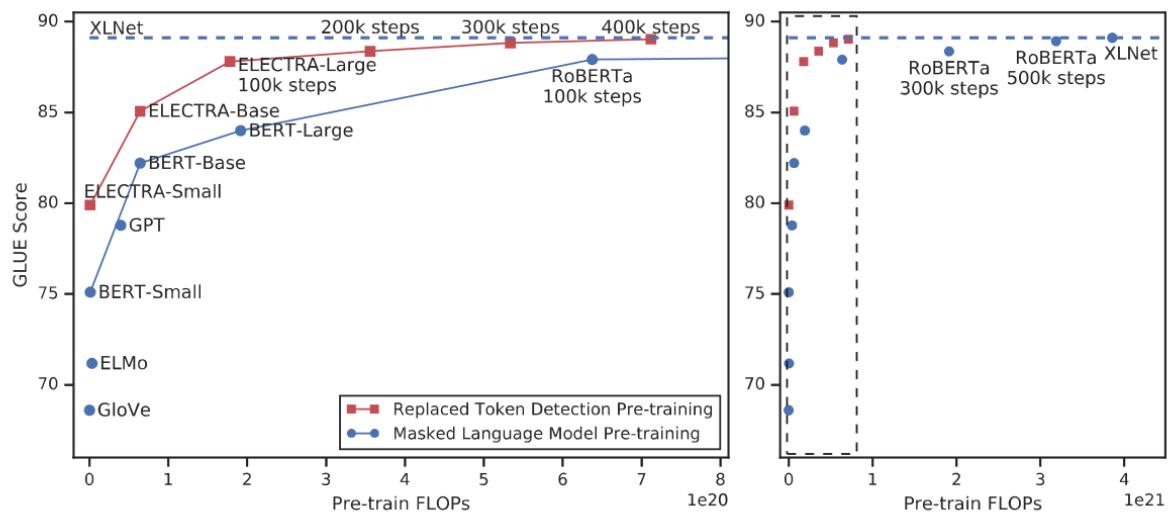


Figure 1: Replaced token detection pre-training consistently outperforms masked language model pre-training given the same compute budget. The left figure is a zoomed-in view of the dashed box.

Method

- BERT的MLM的实现, 并不是非常高效的, 只有15%的tokens对参数的更新有用, 其他的85%是不参与gradients的update的
- 并且存在了预训练和fine-tuning的mismatch, 因为在fine-tuning阶段, 并不会有[MASK]的token。

replaced token detection:

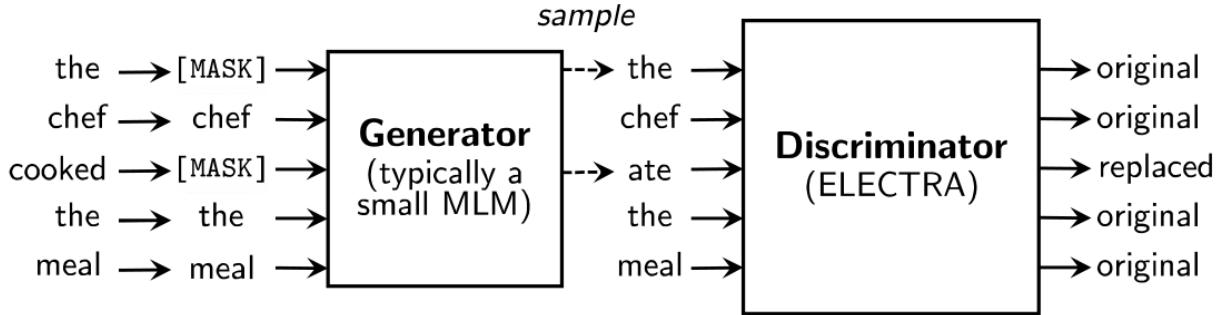


Figure 2: An overview of replaced token detection. The generator can be any model that produces an output distribution over tokens, but we usually use a small masked language model that is trained jointly with the discriminator. Although the models are structured like in a GAN, we train the generator with maximum likelihood rather than adversarially due to the difficulty of applying GANs to text. After pre-training, we throw out the generator and only fine-tune the discriminator (the ELECTRA model) on downstream tasks.

该模型由两部分组成，分别是generator以及discriminator，两个都是transformer的encoder结构，只是两者的size不同：

- generator：就是一个小的 masked language model（通常是 1/4 的discriminator的size），该模块的具体作用是他采用了经典的bert的MLM方式：
 - 首先随机选取15%的tokens，替代为[MASK]token，（取消了bert的80% [MASK], 10% unchange, 10% random replaced 的操作，具体原因也是因为没必要，因为我们 finetuning使用的discriminator）
 - 使用generator去训练模型，使得模型预测masked token，得到corrupted tokens
 - generator的目标函数和bert一样，都是希望被masked的能够被还原成原本的original tokens
如上图，token, the和cooked被随机选为被masked，然后generator预测得到corrupted tokens，变成了the和ate
- discriminator：discriminator的作用是分辨输入的每一个token是original的还是replaced，注意：如果generator生成的token和原始token一致，那么这个token仍然是original的
 - 所以，对于每个token，discriminator都会进行一个二分类，最后获得loss

以上的方式被称为replaced token detection。

- learn from all tokens (instead of 15%)
- compute efficient
- parameter efficient
- improves downstream task performance

该模型采用了minimize the combined loss的方式进行训练：

$$\min_{\theta_G, \theta_D} \sum_{\mathbf{x} \in \mathcal{X}} \mathcal{L}_{\text{MLM}}(\mathbf{x}, \theta_G) + \lambda \mathcal{L}_{\text{Disc}}(\mathbf{x}, \theta_D)$$

$$p_G(x_t | \mathbf{x}) = \exp(e(x_t)^T h_G(\mathbf{x})_t) / \sum_{x'} \exp(e(x')^T h_G(\mathbf{x})_t)$$

$$D(\mathbf{x}, t) = \text{sigmoid}(w^T h_D(\mathbf{x})_t)$$

$$\begin{aligned} m_i &\sim \text{unif}\{1, n\} \text{ for } i = 1 \text{ to } k & \mathbf{x}^{\text{masked}} &= \text{REPLACE}(\mathbf{x}, \mathbf{m}, [\text{MASK}]) \\ \hat{x}_i &\sim p_G(x_i | \mathbf{x}^{\text{masked}}) \text{ for } i \in \mathbf{m} & \mathbf{x}^{\text{corrupt}} &= \text{REPLACE}(\mathbf{x}, \mathbf{m}, \hat{x}) \end{aligned}$$

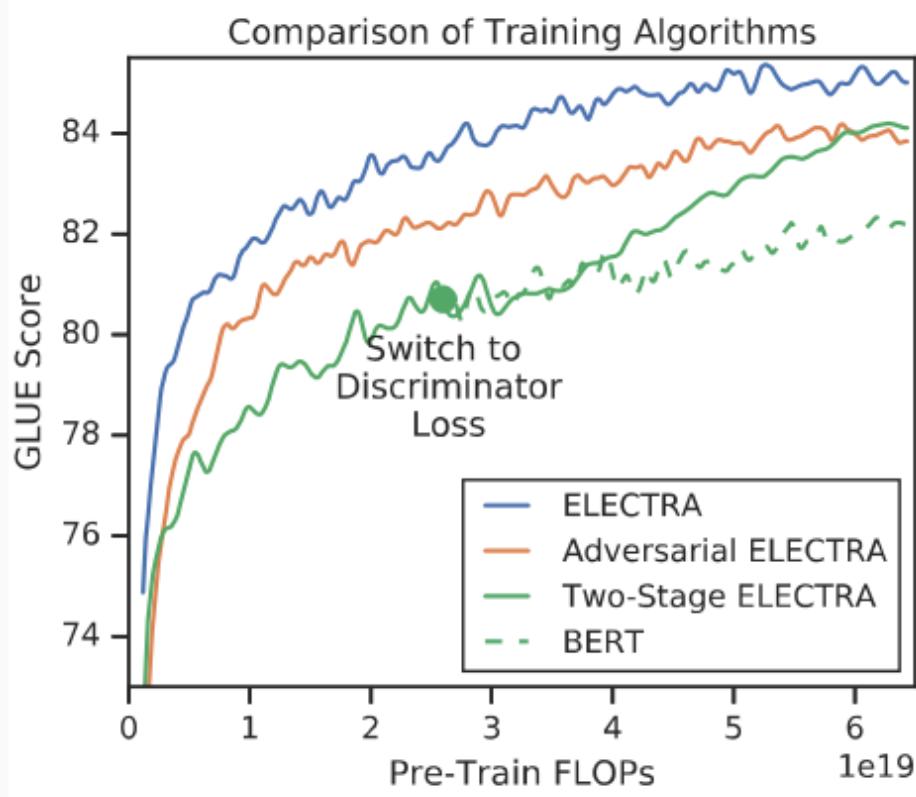
and the loss functions are

$$\begin{aligned} \mathcal{L}_{\text{MLM}}(\mathbf{x}, \theta_G) &= \mathbb{E} \left(\sum_{i \in \mathbf{m}} -\log p_G(x_i | \mathbf{x}^{\text{masked}}) \right) \\ \mathcal{L}_{\text{Disc}}(\mathbf{x}, \theta_D) &= \mathbb{E} \left(\sum_{t=1}^n -\mathbb{1}(x_t^{\text{corrupt}} = x_t) \log D(\mathbf{x}^{\text{corrupt}}, t) - \mathbb{1}(x_t^{\text{corrupt}} \neq x_t) \log(1 - D(\mathbf{x}^{\text{corrupt}}, t)) \right) \end{aligned}$$

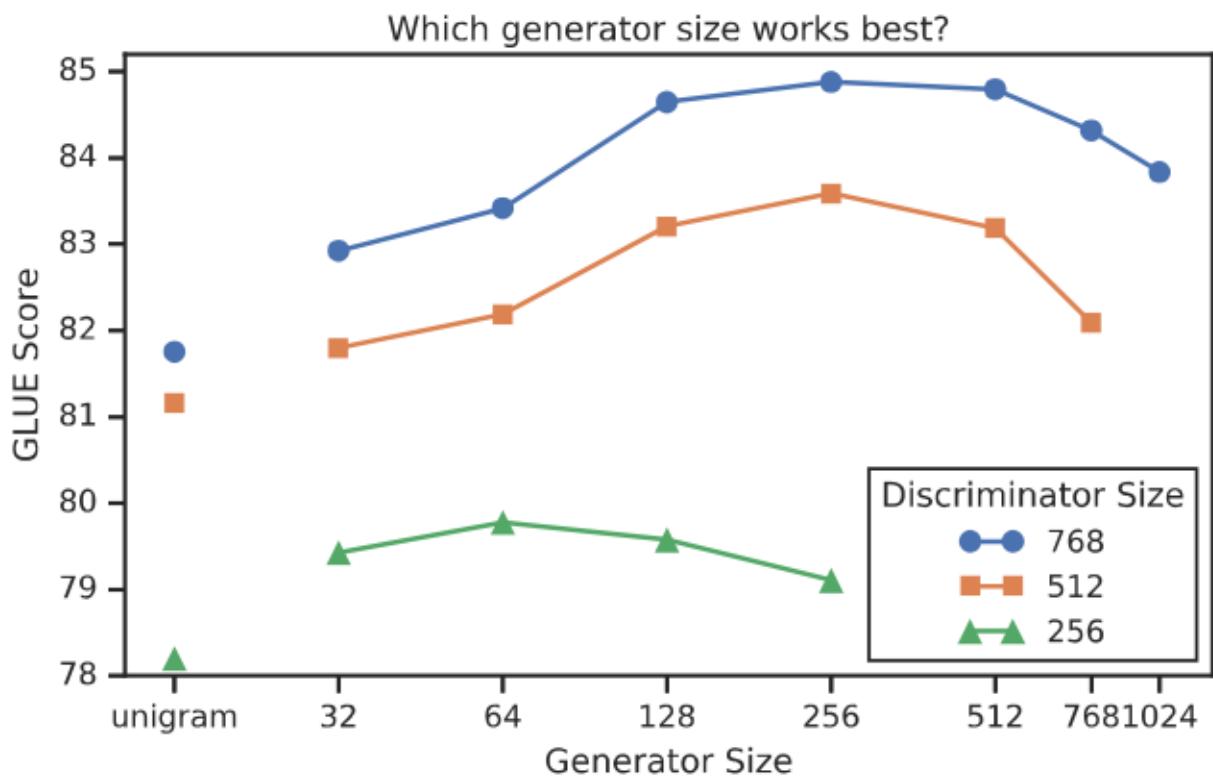
\mathcal{L}_{MLM} loss的计算只计算m个，即m个被masked tokens; \mathcal{L}_{Dis} 的t的取值1....n, 每个token都会更新参数。在训练过程中，discriminator的loss不会反向传播到generator (因为generator的sampling的步骤导致)，在pre-training之后，我们只使用discriminator进行fine-tuning.两者同时训练，但判别器的梯度不会传给生成器

论文中还提出两种训练方式

- 一种是GAN: ELECTRA 以一种对抗学习的思想来训练。作者将生成器的目标函数由最小化MLM loss换成了最大化判别器在被替换token上的RTD loss。但还有一个问题，就是新的生成器loss无法用梯度下降更新生成器，于是作者用强化学习Policy Gradient的思想，将被替换token的交叉熵作为生成器的reward，然后进行梯度下降。强化方法优化下来生成器在MLM任务上可以达到54%的准确率，而之前MLE优化下可以达到65%。
- 一种是two-stage 训练，先训练 \mathcal{L}_{MLM} n steps，然后froze住 generator，再训练 discriminator n steps但是效果都没有共同训练效果好



如果generator过强，那么discriminator就无法成功训练，这其实也很好理解。因为如果generator非常强，那么预测出来的token都非常好，即都是original tokens，那么discriminator并不需要如何学习就收敛，因为它只需要把所有二分类都认为是1就行（假设1代表real）。所以我们的generator不能够过大，过大的话，说明它太powerful了，而且如果他和discriminator一样大的话，那么我们训练一次相当于要训练两个bert的参数，也不能达到efficiency的效果。



从图中看出来，discriminator size = 256 时，best generator size = 64；discriminator size = 512 时，best generator size = 256；discriminator size = 768 时，best generator size = 256；我们可以发现，当generator的size 为discriminator size 的 1/2–1/4 时，模型效果最好

在bert中，mask是随机的，很容易会出现mask的token是非常简单的，而在electra中，corrupted tokens 是有一定难度的，而不是简单的mask，所以就使得discriminator能更好的学习。比如说，输入是：一个聪明的模型，如果随机mask之后是：一[MASK]聪明[MASK]模型，那么就很对模型来说很简单。而一个[MASK][MASK]的模型，则对于模型来说就更复杂了。根据高质量的mask，那么我们的模型能学得更好discriminator 的二分类模型，将MLM连接在一起，而且它不需要考虑到每个position的数据分布，能够达到更高效训练的成果。比如小时候学习语文，老师为了加深你对汉语的理解，总是给出一段话，把一些词去掉(当然老师会有目的性的选词，bert是随机的)，让你根据上下文来填写空缺词。我们可能会很快的根据上下文或者常识填好空缺词。这时，语文老师加大了难度，会给你一段话，让你挑出这段话中哪里用词不当。这就是electra 判别器的预训练任务。

weight sharing

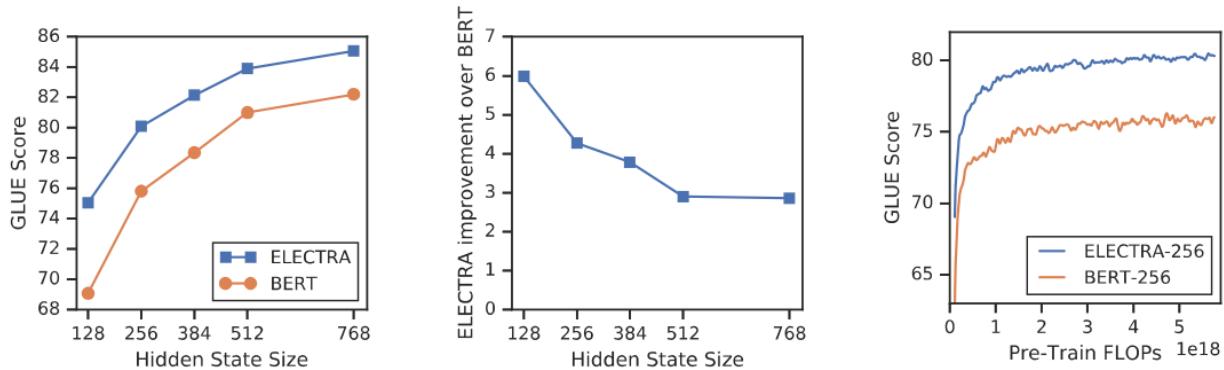
generator和discriminator的weight 存在共享，但是并不是所有的参数都共享，型采用了共享generator的embedding 权重。为什么会选择共享embedding 权重呢，主要的原因是generator采用的MLM的方式训练，MLM根据token周围的context预测该token，可以很好地学习到embedding的表示。discriminator 只更新input或者被 generator sample 的tokens（二分类），而generator的softmax over vocabulary 会更新所有的token embeddings。然而在此基础上，replaced token detection 从所有的input tokens都做更新，所以参数更新的更有效率。

difference from GAN

GAN 的训练是训练一个generator 产生结果，去骗过discriminator。所以generator的产生结果，在 discriminator的都会认为是假的。

- 在我们的训练过程中，如果generator 产生的token和original token一样，discriminator应该认为这个token是real的
- generator是按照最大似然训练的，和discriminator 并没有交互
- generator 不是用来fool discriminator。
- generator的输入是真实文本，而GAN的输入是随机噪声
- discriminator的梯度不会传到generator，而GAN的梯度是会从discriminator传到generator的

由于ELECTRA是判别式任务，不用对每个位置的整个数据分布建模，所以更parameter-efficient。ELECTRA 模型在小模型的时候，效果提升显著，而随着模型大小的增加，效果降低。



ALBERT (NSP换成SOP)

Distillation

distillation 可简单分为 model distillation 和 feature distillation。蒸馏是对原来的模型/特征进行了压缩，其原因可能是为了减少模型的大小（model distillation）或某些特征只能在 training 时获取，serving 无法获取(feature distillation)；在实际业务中可根据具体场景灵活地应用这两类技术。

基本原理

Distillation 可分为 Model Distillation 和 Feature Distillation，其思想都是在训练时同时训练两个模型：teacher 模型和 student 模型，而在 serving 时只用 student 模型。这里的假设是：teacher 模型比起 student 模型，在模型结构上更复杂(Model Distillation)，或在特征集上更为丰富(Feature Distillation)；因此其准确率也会比 student 模型要好。

如下图所示是 Model Distillation 和 Feature Distillation 示例（下面的图和公式基本摘自 [Privileged Features Distillation for E-Commerce Recommendations](#)）

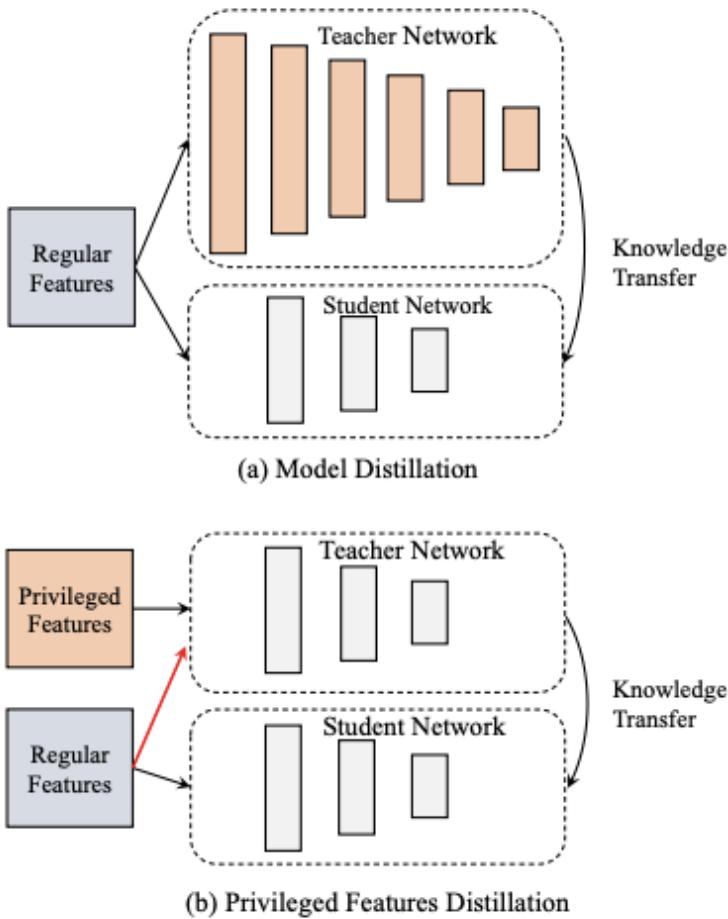


Figure 1: Illustration of model distillation (MD) [13] and privileged features distillation (PFD) proposed in this work. In MD, the knowledge is distilled from the more complex model. While in PFD, the knowledge is distilled from *both* the privileged *and* the regular features. PFD also differs from the original learning using privileged information (LUPI) [24], where the teacher *only* processes the privileged features.

那如何利用 teacher 模型指导 student 模型学得更好？基本的做法是将 teacher 模型的输出作为 soft label(相对于作为 ground truth 的 hard label)，为 student 模型添加额外的 loss 项；如下公式 (1) 所示

$$\min_{W_s} (1-\lambda)L_s(y, f_s(X; W_s)) + \lambda * L_d(f_t(X; W_t), f_s(X; W_s)) \quad (1)$$

上式中各项符号含义如下

- $f_s(X; W_s)$: student 模型的预估值
- $f_t(X; W_t)$: teacher 模型的预估值
- L_s : student 模型原始的 loss
- L_d : 利用 teacher 模型预估值输出作为 soft label 计算的 distillation loss;
- λ : 平衡 L_s 和 L_d 的超参

上面公式 (1) 是 Model Distillation 的典型做法，可以看到输入 teacher 模型和 student 模型的特征都是相同的即 X ；而公式(2)描述的 Feature Distillation 则认为 teacher 模型的特征 (X_*) 比 student 模型的特征(X) 更为丰富，

$$\min_{W_s} (1-\lambda)L_s(y, f_s(X; W_s)) + \lambda * L_d(f_t(X_*; W_t), f_s(X; W_s)) \quad (2)$$

上面两条公式是 Distillation 的核心思想了，且在使用理论上应该首先训练好 teacher 网络，再训练 student 网络；但是在实际训练的时候，为了加快训练速度，会令 teacher 模型和 student 模型同时进行训练；因此最终的损失函数变为了如下公式(3) 形式，其中 L_s L_s 和 L_t L_t 是 logloss，而 L_d L_d 是 cross entropy loss

$$\min_{W_s, W_t} (1-\lambda)L_s(y, f_s(X; W_s)) + \lambda * L_d(f_t(X_*; W_t), f_s(X; W_s)) + L_t(y, f_t(X_*; W_t)) \quad (3)$$

综上，在 training 和 serving 时的模型结构分别如下所示

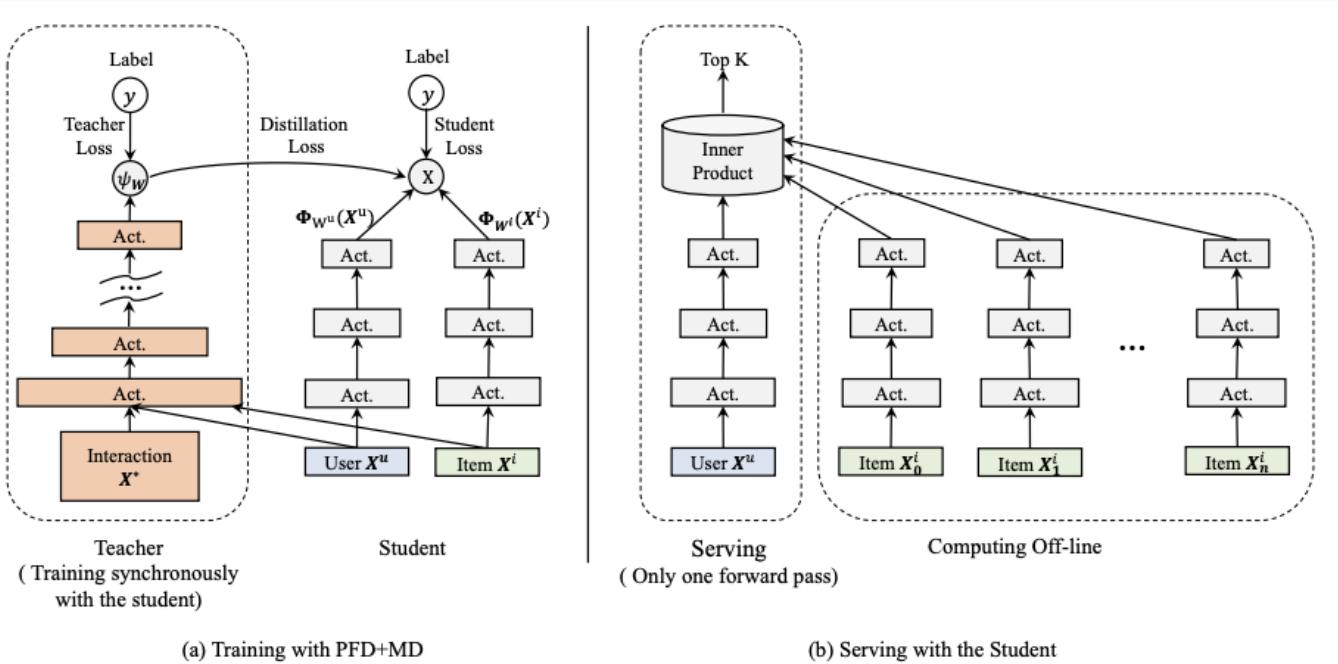


Figure 4: Illustration of training the inner-product model with PFD+MD and (b) its deployment during serving. At the training time, the privileged features, i.e., interaction X^* between X^u and X^i , and the more complex DNN model together form a strong teacher to instruct the student. During serving, we compute the mappings $\Phi_{W^u}(\cdot)$ of all items off-line in advance. When a request comes, we only need to execute one forward pass to derive the user mapping $\Phi_{W^u}(X^u)$.

训练注意事项

上面提到，distillation 需要训练 teacher 和 student 两个网络，因此也有两种训练模式：

- (1) 先训练 teacher 网络，再训练 student 网络，也被称为 asynchronous training
- (2) 同时训练 teacher 网络和 student 网络，也被称为 synchronous training

理论上应该采用方式(1)，但是由于需要串行训练两个模型，会导致训练的时间过长，因此才提出了方式 (2) 的方法；而方式 (2) 会带来训练效果不稳定的问题，其原因是在 teacher 在训练初期，其效果往往还不好，而将其输出结果作为 label 很容易导致 student 网络学飞了

因此更常用的做法在这两个之间做个权衡，基本做法就是在训练的初期，将公式(3) 中的 λ 设为 0，然后后面逐渐增大入这个值

上面提到的paper在这点上提出了一个更简单策略，就是在kk个 step 后才让 teacher 网络的输出作为 loss 影响 student 网络，kk是一个拍定的超参，因此其详细训练方式如下

Algorithm 1 Minimizing the student, distillation, and teacher loss in Eq.(5) synchronously with SGD.

Input: Hyper-parameter λ , swapping step k , and learning rate η

```
1: Initialize  $(W_s, W_t)$  and let  $i = 0$ .
2: while not converged do
3:   Get training data  $(y, X, X^*)$ .
4:   if  $i < k$  then
5:      $W_s = W_s - \eta \nabla_{W_s} L_s$ .
6:   else
7:      $W_s = W_s - \eta \nabla_{W_s} \{(1 - \lambda) * L_s + \lambda * L_d\}$ .
8:   end if
9:    $W_t = W_t - \eta \nabla_{W_t} L_t$ . //No distillation loss  $L_d$ 
9:   Update  $i = i + 1$ 
10: end while
```

Output: (W_s, W_t)

实现

tensorflow 提供的一个distillation 的实现[distillation.py](#)，使用见 stack-overflow 上的[这个回答](#)核心代码如下所示，注释写得已经非常清晰了，下面默认的模式是先训练好了 Teacher 网络，再训练 Student 网络，也就是上面提到的 asynchronous training 模式；但是也可以比较容易将下面的逻辑改成 synchronous training 的。

```

1  ### Teacher Network
2  with tf.variable_scope("teacher"):
3      teacher_outputs = self.teacher_model.body(features)
4      tf.logging.info("teacher output shape: %s" % teacher_outputs.get_shape())
5      teacher_outputs = tf.reduce_mean(teacher_outputs, axis=[1, 2])
6      teacher_logits = tf.layers.dense(teacher_outputs, hp.num_classes)
7
8      teacher_task_xent = tf.nn.softmax_cross_entropy_with_logits_v2(
9          labels=one_hot_targets, logits=teacher_logits)
10     outputs = teacher_logits
11
12 if is_distill:
13     # Load teacher weights
14     tf.train.init_from_checkpoint(hp.teacher_dir, {"teacher/": "teacher/"})
15     # Do not train the teacher
16     trainable_vars = tf.get_collection_ref(tf.GraphKeys.TRAINABLE_VARIABLES)
17     del trainable_vars[:]
18
19
20  ### Student Network
21 if is_distill:
22     with tf.variable_scope("student"):
23         student_outputs = self.student_model.body(features)
24         tf.logging.info(
25             "student output shape: %s" % student_outputs.get_shape())
26         student_outputs = tf.reduce_mean(student_outputs, axis=[1, 2])
27         student_logits = tf.layers.dense(student_outputs, hp.num_classes)
28
29         student_task_xent = tf.nn.softmax_cross_entropy_with_logits_v2(
30             labels=one_hot_targets, logits=student_logits)
31         teacher_targets = tf.nn.softmax(teacher_logits / hp.distill_temperature)
32         student_distill_xent = tf.nn.softmax_cross_entropy_with_logits_v2(
33             labels=tf.stop_gradient(teacher_targets),
34             logits=student_logits / hp.distill_temperature)
35         # scale soft target obj. to match hard target obj. scale
36         student_distill_xent *= hp.distill_temperature**2
37
38     outputs = student_logits
39
40     # Summaries
41     tf.summary.scalar("distill_xent", student_distill_xent)
42
43 if not is_distill:
44     phase_loss = teacher_task_xent
45 else:
46     phase_loss = hp.task_balance * student_task_xent
47     phase_loss += (1 - hp.task_balance) * student_distill_xent
48
49 losses = {"training": phase_loss}

```

```
50  outputs = tf.reshape(outputs, [-1, 1, 1, 1, outputs.shape[1]])
51
52  return outputs, losses
```

BERT的模型蒸馏（一）

参考

BERT的模型蒸馏（二）

参考

BERT-TNF(对Rare word提出notes想法)

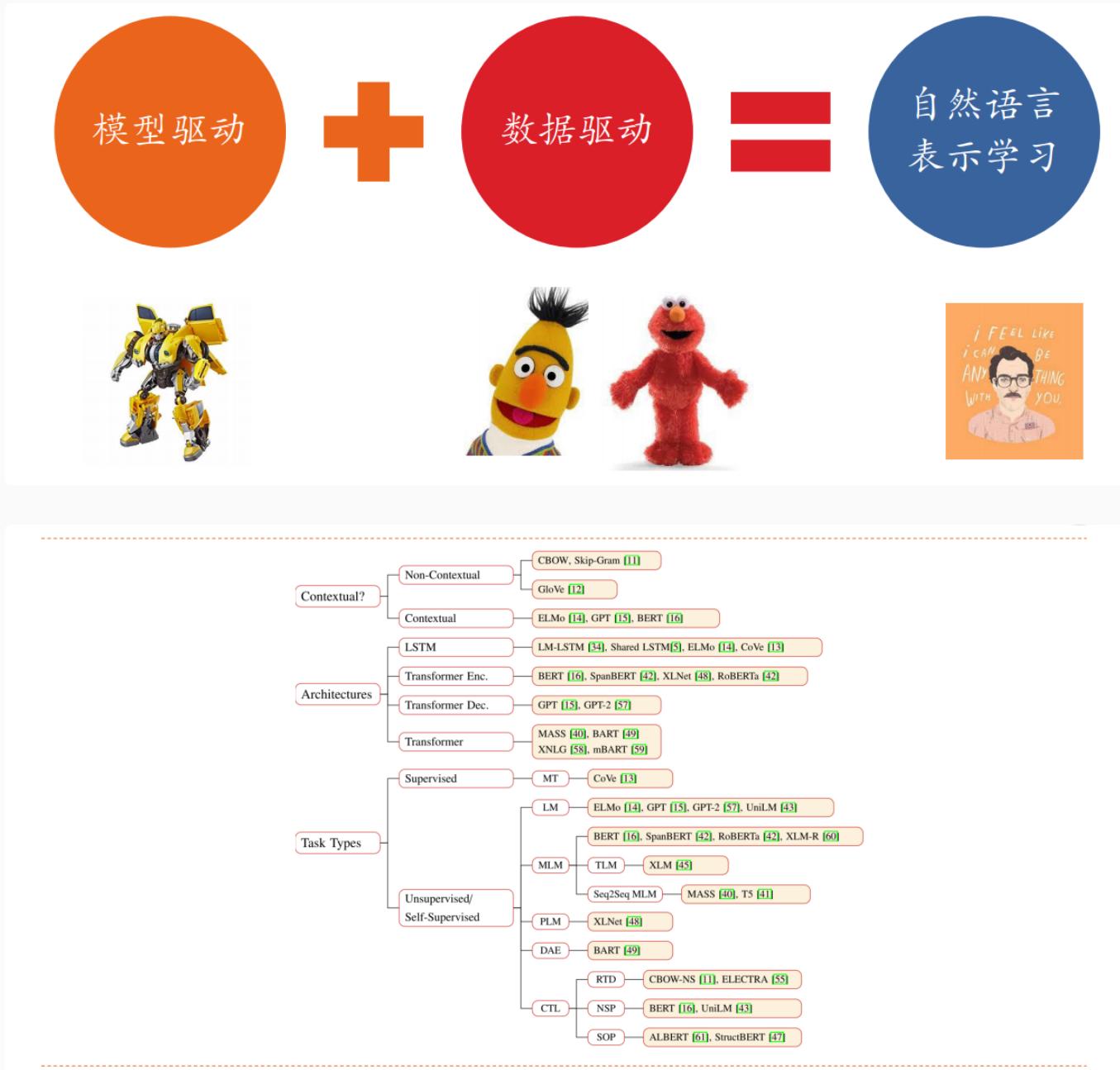
原文: https://openreview.net/forum?id=IU5Rs_wCweN

笔记: <https://drive.google.com/drive/u/0/my-drive>

代码: https://openreview.net/forum?id=IU5Rs_wCweN

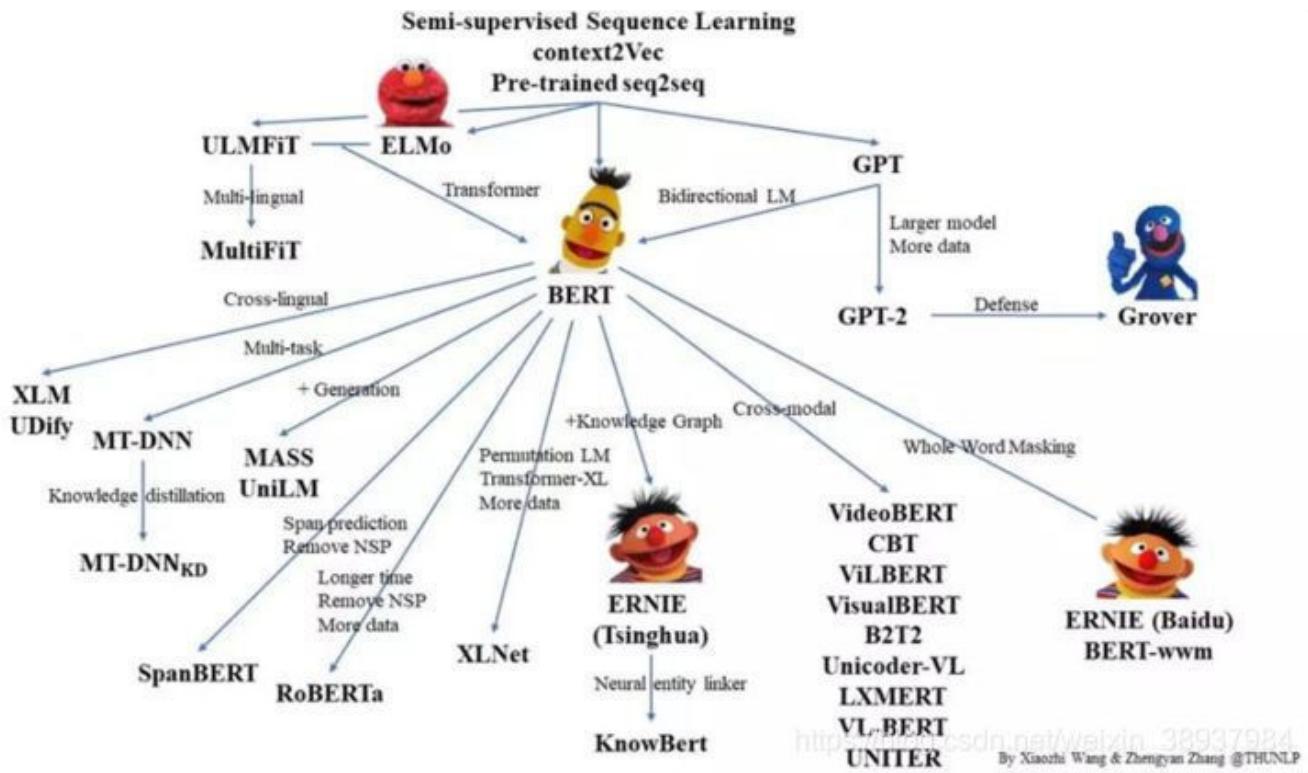
预训练模型

预训练论文Summary: <https://github.com/thunlp/PLMpapers>



PTMs	Architecture [†]	Input	Pre-Training Task	Corpus	Params	GLUE [‡]	FT [§]
ELMo [14]	LSTM	Text	BiLM	WikiText-103		No	
GPT [15]	Transformer Dec.	Text	LM	BookCorpus	117M	72.8	Yes
GPT-2 [57]	Transformer Dec.	Text	LM	WebText	117M ~ 1542M	No	
BERT [16]	Transformer Enc.	Text	MLM & NSP	WikiEn+BookCorpus	110M ~ 340M	81.9*	Yes
InfoWord [54]	Transformer Enc.	Text	DIM+MLM	WikiEn+BookCorpus	=BERT	81.1*	Yes
RoBERTa [42]	Transformer Enc.	Text	MLM	BookCorpus+CC-News+OpenWebText+ STORIES	355M	88.5	Yes
XLNet [48]	Two-Stream	Text	PLM	WikiEn+ BookCorpus+Giga5 +ClueWeb+Common Crawl	≈BERT	90.5 [§]	Yes
ELECTRA [55]	Transformer Enc.	Text	RTD+MLM	same to XLNet	335M	88.6	Yes
UniLM [43]	Transformer Enc.	Text	MLM [‡] NSP	WikiEn+BookCorpus	340M	80.8	Yes
MASS [40]	Transformer	Text	Seq2Seq MLM	*Task-dependent			Yes
BART [49]	Transformer	Text	DAE	same to RoBERTa	110% of BERT	88.4*	Yes
T5 [41]	Transformer	Text	Seq2Seq MLM	Colossal Clean Crawled Corpus (C4)	220M ~ 11B	89.7*	Yes

模型	语言模型	特征抽取	上下文表征	最大亮点	https://zhuanlan.zhihu.com/p/76912493
ELMO	BiLM	BiLSTM	单向	2个单向语言模型拼接；	
ULMFiT	LM	AWD-LSTM	单向	引入逐层解冻解决finetune中的灾难性问题；	
SiATL	LM	LSTM	单向	引入逐层解冻+辅助LM解决finetune中的灾难性问题；	
GPT1.0	LM	Transformer	单向	统一下游任务框架，验证Transformer在LM中的强大；	
GPT2.0	LM	Transformer	单向	没有特定模型的精调流程，生成任务取得很好效果；	
BERT	MLM	Transformer	双向	MLM获取上下文相关的双向特征表示；	
MASS	LM+MLM	Transformer	单向/双向	改进BERT生成任务：统一为类似Seq2Seq的预训练框架；	
UNILM	LM+MLM+S2SLM	Transformer	单向/双向	改进BERT生成任务：直接从mask矩阵的角度出发；	
ENRIE1.0	MLM(BPE)	Transformer	双向	引入知识：3种[MASK]策略(BPE)预测短语和实体；	
ENRIE	MLM+DEA	Transformer	双向	引入知识：将实体向量与文本表示融合；	
MTDNN	MLM	Transformer	双向	引入多任务学习：在下游阶段；	
ENRIE2.0	MLM+Multi-Task	Transformer	双向	引入多任务学习：在预训练阶段，连续增量学习；	
SpanBERT	MLM+SPO	Transformer	双向	不需要按照边界信息进行mask；	
RoBERTa	MLM	Transformer	双向	精细调参，舍弃NSP；	知乎 @JayLou
XLNet	PLM	Transformer-XL	双向	排列语言模型+双注意力流+Transformer	



预训练的方法最初是在图像领域提出的，达到了良好的效果，后来被应用到自然语言处理。预训练一般分为两步，首先用某个较大的数据集训练好模型(这种模型往往比较大，训练需要大量的内存资源)，使模型训练到一个良好的状态，然后下一步根据不同的任务，改造预训练模型，用这个任务的数据集在预训练模型上进行微调。

这种做法的好处是训练代价很小，预训练的模型参数可以让新的模型达到更快的收敛速度，并且能够有效地提高模型性能，尤其是对一些训练数据比较稀缺的任务，在神经网络参数十分庞大的情况下，仅仅依靠任务自身的训练数据可能无法训练充分，预训练方法可以认为是让模型基于一个更好的初始状态进行学习，从而能够达到更好的性能。

预训练模型---- ELMO&GPT&Bert

参考: https://blog.csdn.net/weixin_38937984/article/details/101759107?spm=1001.2014.3001.5501

ppt:  从ELMO到GPT到BERT.pptx

回答: https://mp.weixin.qq.com/s/ao9.bn_2p0CrFUa_urtmOg

预训练模型---- Transformer-XL&GPT2&XLNet

https://blog.csdn.net/weixin_38937984/article/details/101759331?spm=1001.2014.3001.5501

预训练模型---- RoBERTa

<https://arxiv.org/pdf/1907.11692v1.pdf>

RoBERTa主要在三方面对之前提出的BERT做了改进，其一是模型的具体细节层面，改进了优化函数；其二是训练策略层面，改用了动态掩码的方式训练模型，证明了NSP（Next Sentence Prediction）训练策略的不足，采用了更大的batch size；其三是数据层面，一方面使用了更大的数据集，另一方面是使用BPE（Byte-Pair Encoding）来处理文本数据。

1. RoBERTa对一般BERT的模型细节进行了优化

Optimization

原始BERT优化函数采用的是Adam默认的参数，其中 $\beta_1=0.9, \beta_2=0.999$ ，在RoBERTa模型中考虑采用了更大的batches，所以将 β_2 改为了0.98。

2. RoBERTa对一般BERT的训练策略进行了优化

(1) 动态掩码与静态掩码

原始静态mask：BERT中是准备训练数据时，每个样本只会进行一次随机mask（因此每个epoch都是重复），后续的每个训练步都采用相同的mask，这是原始静态mask，即单个静态mask，这是原始BERT的做法。

修改版静态mask：在预处理的时候将数据集拷贝10次，每次拷贝采用不同的mask（总共40 epochs，所以每一个mask对应的数据被训练4个epoch）。这等价于原始的数据集采用10种静态mask来训练40个epoch。

动态mask：并没有在预处理的时候执行mask，而是在每次向模型提供输入时动态生成mask，所以是时刻变化的。不同模式的实验效果如下表所示。其中reference为BERT用到的原始静态mask，

static 为修改版的静态mask。

Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
<i>Our reimplementation:</i>			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

(2) 对NSP训练策略的探索

为了探索NSP训练策略对模型结果的影响，将以下4种训练方式及进行对比：

SEGMENT-PAIR + NSP：这是原始 BERT 的做法。输入包含两部分，每个部分是来自同一文档或者不同文档的 segment（segment 是连续的多个句子），这两个segment 的token总数少于 512。预训练包含 MLM 任务和 NSP 任务。

SENTENCE-PAIR + NSP：输入也是包含两部分，每个部分是来自同一个文档或者不同文档的单个句子，这两个句子的token 总数少于 512。由于这些输入明显少于512 个tokens，因此增加batch size的大小，以使 tokens 总数保持与 SEGMENT-PAIR + NSP 相似。预训练包含 MLM 任务和 NSP 任务。

FULL-SENTENCES：输入只有一部分（而不是两部分），来自同一个文档或者不同文档的连续多个句子，token 总数不超过 512。输入可能跨越文档边界，如果跨文档，则在上一个文档末尾添加文档边界 token。预训练不包含 NSP 任务。

DOC-SENTENCES：输入只有一部分（而不是两部分），输入的构造类似于 FULL-SENTENCES，只是不需要跨越文档边界，其输入来自同一个文档的连续句子，token 总数不超过 512。在文档末尾附近采样的输入可以短于 512个tokens，因此在这些情况下动态增加batch size大小以达到与 FULL-SENTENCES 相同的tokens总数。预训练不包含 NSP 任务。

以下是论文中4种方法的实验结果：

Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
<i>Our reimplementation (with NSP loss):</i>				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
<i>Our reimplementation (without NSP loss):</i>				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6
BERT _{BASE}	88.5/76.3	84.3	92.8	64.3
XLNet _{BASE} (K = 7)	-/81.3	85.8	92.7	66.1
XLNet _{BASE} (K = 6)	-/81.0	85.6	93.4	66.7

Table 2: Development set results for base models pretrained over BOOKCORPUS and WIKIPEDIA. All models are trained for 1M steps with a batch size of 256 sequences. We report F1 for SQuAD and accuracy for MNLI-m, SST-2 and RACE. Reported results are medians over five random initializations (seeds). Results for BERT_{BASE} and XLNet_{BASE} are from Yang et al. (2019).

从实验结果来看，如果在采用NSP loss的情况下，将SEGMENT-PAIR与SENTENCE-PAIR进行对比，结果显示前者优于后者。发现单个句子会损害下游任务的性能，可能是如此模型无法学习远程依赖。接下来把重点放在没有NSP loss的FULL-SENTENCES上，发现其在四种方法中结果最好。可能的原因：原始 BERT 实现采用仅仅是去掉NSP的损失项，但是仍然保持SEGMENT-PARI的输入形式。最后，实验还发现将序列限制为来自单个文档(doc-sentence)的性能略好于序列来自多个文档(FULL-SENTENCES)。但是DOC-SENTENCES策略中，位于文档末尾的样本可能小于 512 个 token。为了保证每个 batch 的 token 总数维持在一个较高水平，需要动态调整 batch-size。出于处理方便，后面采用DOC-SENTENCES输入格式。

(3) Training with large batches

虽然在以往的经验中，当学习速率适当提高时，采用非常大mini-batches的训练既可以提高优化速度，又可以提高最终任务性能。但是论文中通过实验，证明了更大的batches可以得到更好的结果，实验结果下表所示。

bsz	steps	lr	ppl	MNLI-m	SST-2
256	1M	1e-4	3.99	84.7	92.7
2K	125K	7e-4	3.68	85.2	92.9
8K	31K	1e-3	3.77	84.6	92.8

Table 3: Perplexity on held-out training data (*ppl*) and development set accuracy for base models trained over BOOKCORPUS and WIKIPEDIA with varying batch sizes (*bsz*). We tune the learning rate (*lr*) for each setting. Models make the same number of passes over the data (epochs) and have the same computational cost.

论文考虑了并行计算等因素，在后续的实验中使用batch size=8k进行训练。

3. RoBARTa在数据层面对模型进行了优化

(1) 使用了更大的训练数据集

将16G的数据集提升到160G数据集，并改变多个steps，寻找最佳的超参数。

Model		data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa							
with BOOKS + WIKI		16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)		160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer		160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer		160GB	8K	500K	94.6/89.4	90.2	96.4
BERT_{LARGE}							
with BOOKS + WIKI		13GB	256	1M	90.9/81.8	86.6	93.7
XLNet_{LARGE}							
with BOOKS + WIKI		13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data		126GB	2K	500K	94.5/88.8	89.8	95.6

(2) Text Encoding

字节对编码(BPE)(Sennrich et al.,2016)是字符级和单词级表示的混合，该编码方案可以处理自然语言语料库中常见的大量词汇。BPE不依赖于完整的单词，而是依赖于子词(sub-word)单元，这些子词单元是通过对训练语料库进行统计分析而提取的，其词表大小通常在1万到10万之间。当对海量多样语料建模时，unicode characters占据了该词表的大部分。Radford et al.(2019)的工作中介绍了一个简单但高效的BPE，该BPE使用字节对而非unicode characters作为子词单元。

总结下两种BPE实现方式：

基于 char-level：原始 BERT 的方式，它通过对输入文本进行启发式的词干化之后处理得到。

基于 bytes-level：与 char-level 的区别在于bytes-level 使用 bytes 而不是 unicode 字符作为 sub-word 的基本单位，因此可以编码任何输入文本而不会引入 UNKNOWN 标记。

当采用 bytes-level 的 BPE 之后，词表大小从3万（原始 BERT 的 char-level ）增加到5万。这分别为 BERT-base 和 BERT-large 增加了1500万和2000万额外的参数。之前有研究表明，这样的做法在有些下游任务上会导致轻微的性能下降。但是本文作者相信：这种统一编码的优势会超过性能的轻微下降。且作者在未来工作中将进一步对比不同的encoding方案。

4. 简单来说

(数据) 更多 (训练) 更久 (批次) 更大：

总共训练了多少个token (字) 呢？

50万步*batch size为 8K,长度为512=20480亿个token。

估计8卡的TPU上训练，需要3200个小时即2个月时间。

1.让数据训练不重复。dynamic masking:each train has different training data

2.取消下一个句子预测，数据连续的从一个或多个文档中获得，直到长度为512。full sentence without NSP

- 3.优化器参数调整。adam epsilon
- 4.更多更对样性的数据。数据more data:130g维基百科、书、新闻（6300万新闻）、社区讨论reddit、故事类数据
- 5.更大batch size。large batch size: $2k * (7e-4)$ to 8k
- 6.训练更长时间：train longer:100k to 300k steps

预训练模型---- XLNet（用自回归本身的特点克服 BERT 的缺点）

参考1: https://blog.csdn.net/weixin_38937984/article/details/101929869?spm=1001.2014.3001.5501

参考2: <https://zhuanlan.zhihu.com/p/70257427>

参考3: https://blog.csdn.net/weixin_37947156/article/details/93035607

题目：XLNet: Generalized Autoregressive Pretraining for Language Understanding

机构：1Carnegie Mellon University, 2Google AI Brain Team

作者：Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, Quoc V. Le

收录会议：ACL 2020

论文地址：<https://arxiv.org/pdf/1906.08237.pdf>

预训练模型及代码地址：<https://github.com/zihangdai/xlnet>

BERT 这样基于去噪自编码器的预训练模型可以很好地建模双向语境信息，性能优于基于自回归语言模型的预训练方法。然而，由于需要 mask 一部分输入，BERT 忽略了被 mask 位置之间的依赖关系，因此出现预训练和微调效果的差异 (*pretrain–finetune discrepancy*)。

基于这些优缺点，该研究提出了一种泛化的自回归预训练模型 XLNet。XLNet 可以：1) 通过最大化所有可能的因式分解顺序的对数似然，学习双向语境信息；2) 用自回归本身的特点克服 BERT 的缺点。此外，XLNet 还融合了当前最优自回归模型 Transformer-XL 的思路。

最终，XLNet 在 20 个任务上超过了 BERT 的表现，并在 18 个任务上取得了当前最佳效果 (state-of-the-art)，包括机器问答、自然语言推断、情感分析和文档排序。

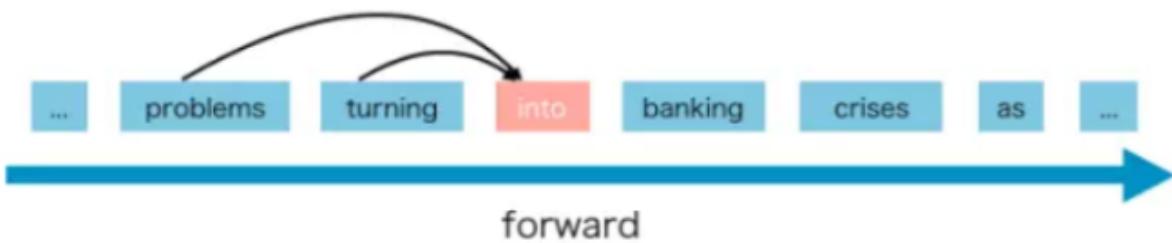
以前超越 BERT 的模型很多都在它的基础上做一些修改，本质上模型架构和任务都没有太大变化。但是在这篇新论文中，作者从自回归 (autoregressive) 和自编码 (autoencoding) 两大范式分析了当前的预训练语言模型，并发现它们虽然各自都有优势，但也都有难以解决的困难。为此，研究者提出 XLNet，并希望结合大阵营的优秀属性。

1、AR 与 AE 两大阵营

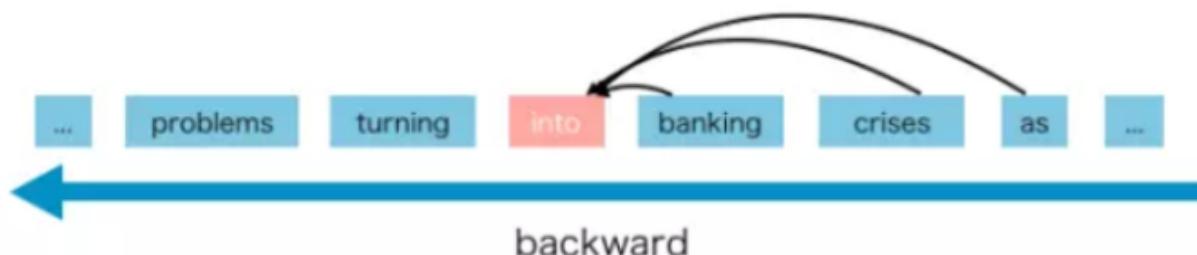
什么是 XLNet

首先，XLNet 是一个类似 BERT 的模型，而不是完全不同的模型。总之，XLNet 是一种通用的自回归预训练方法。

那么什么是自回归 (AR) 语言模型？



前向



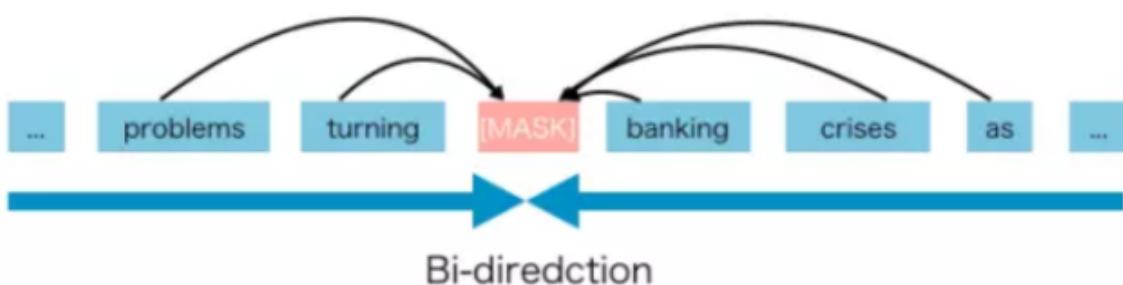
后向

https://blog.csdn.net/weixin_37947156

AR语言模型是一种使用上下文词来预测下一个词的模型。但是在这里，上下文单词被限制在两个方向，前向或后向，GPT 和 GPT-2 都 AR 语言模型。AR 语言模型的优势是擅长生成式自然语言处理任务。因为在生成上下文时，通常是前向的。AR 语言模型很自然地适用于此类 NLP 任务。但AR语言模型有一些缺点，它只能使用前向上下文或后向上下文，这意味着它不能同时使用前向和后向上下文。

XLNet和BERT有什么区别

与 AR 语言模型不同，BERT 被归类为自动编码器（AE）语言模型。AE 语言模型旨在从损坏的输入重建原始数据。



双向

https://blog.csdn.net/weixin_37947156

损坏的输入意味着我们在预训练阶段用 [MASK] 替换原始词 into。目标是预测 into 得到原始句子。

AE 语言模型的优势是，它可以从向前和向后的方向看到上下文。

但 AE 语言模型也有其缺点。它在预训练中使用 [MASK]，但这种人为的符号在调优时在真实数据中并不存在，会导致预训练–调优的差异。[MASK] 的另一个缺点是它假设预测（掩蔽的）词 在给定未屏蔽的词的情况下彼此独立。例如，我们有一句话“它表明住房危机已经变成银行危机”。我们掩蔽“银行业”和“危机”。在这里注意，我们知道掩蔽的“银行业”和“危机”包含彼此的隐含关系。但 AE 模型试图预测“银行业”给予未掩蔽的词，并预测“危机”分别给出未掩蔽的词。它忽略了“银行业”与“危机”之间的关系。换句话说，它假设预测（掩蔽）的标记彼此独立。但是我们知道模型应该学习预测（掩蔽）词之间的这种相关性来预测其中一个词。

作者想要强调的是，XLNet 提出了一种让 AR 语言模型从双向上下文中学习的新方法，以避免 MASK 方法在 AE 语言模型中带来的缺点。

1.2 两大阵营间需要新的 XLNet

现有的语言预训练目标各有优劣，这篇新研究提出了一种泛化自回归方法 XLNet，既集合了 AR 和 AE 方法的优势，又避免了二者的缺陷。

首先，XLNet 不使用传统 AR 模型中固定的前向或后向因式分解顺序，而是**最大化所有可能因式分解顺序的期望对数似然**。由于对因式分解顺序的排列操作，每个位置的语境都包含来自左侧和右侧的 token。因此，每个位置都能学习来自所有位置的语境信息，即捕捉双向语境。

其次，作为一个泛化 AR 语言模型，XLNet 不依赖残缺数据。因此，XLNet 不会有 BERT 的预训练–微调差异。同时，自回归目标提供一种自然的方式，来利用乘法法则对预测 token 的联合概率执行因式分解 (factorize)，这消除了 BERT 中的独立性假设。

除了提出一个新的预训练目标，XLNet 还改进了预训练的架构设计。

受到 AR 语言建模领域最新进展的启发，XLNet 将 Transformer–XL 的分割循环机制 (segment recurrence mechanism) 和相对编码范式 (relative encoding) 整合到预训练中，实验表明，这种做法提高了性能，尤其是在那些包含较长文本序列的任务中。

简单地使用 Transformer(–XL) 架构进行基于排列的 (permutation-based) 语言建模是不成功的，因为因式分解顺序是任意的、训练目标是模糊的。因此，研究人员提出，对 Transformer(–XL) 网络的参数化方式进行修改，移除模糊性。

2、背景

在论文里作者使用了一些术语，比如自回归(Autoregressive, AR)语言模型和自编码(autoencoding)模型等，这可能让不熟悉的读者感到困惑，因此我们先简单的解释一下。自回归是时间序列分析或者信号处理领域喜欢用的一个术语，我们这里理解成语言模型就好了：一个句子的生成过程如下：首先根据概率分布生成第一个词，然后根据第一个词生成第二个词，然后根据前两个词生成第三个词，……，直到生成整个句子。而所谓的自编码器是一种无监督学习输入的特征的方法：我们用一个神经网络把输入(输入通常还会增加一些噪声)变成一个低维的特征，这就是编码部分，然后再用一个Decoder尝试把特征恢复成原始的信号。我们可以把BERT看成一种AutoEncoder，它通过Mask改变了部分Token，然后试图通过其上下文的其它Token来恢复这些被Mask的Token。

给定文本序列 $x=[x_1, \dots, x_T]$, 语言模型的目标是调整参数使得训练数据上的似然函数最大:

$$\max_{\theta} \log p_{\theta}(x) = \sum_{t=1}^T \log p_{\theta}(x_t | \mathbf{x}_{<t}) = \sum_{t=1}^T \log \frac{\exp(h_{\theta}(\mathbf{x}_{1:t-1})^\top e(x_t))}{\sum_{x'} \exp(h_{\theta}(\mathbf{x}_{1:t-1})^\top e(x'))}, \quad (1)$$

记号 $\mathbf{x}_{<t}$ 表示 t 时刻之前的所有 x , 也就是 $x_{1:t-1}$ 。 $h_{\theta}(x_{1:t-1})$ 是RNN或者Transformer(注: Transformer也可以用于语言模型, 比如在OpenAI GPT)编码的 t 时刻之前的隐状态。 $e(x)$ 是词 x 的embedding。

而BERT是去噪(denoising)自编码的方法。对于序列 x , BERT会随机挑选15%的Token变成[MASK]得到带噪声版本的 x^{\wedge} 。假设被Mask的原始值为 x^- , 那么BERT希望尽量根据上下文恢复(猜测)出原始值了, 也就是:

$$\max_{\theta} \log p_{\theta}(\bar{\mathbf{x}} | \hat{\mathbf{x}}) \approx \sum_{t=1}^T m_t \log p_{\theta}(x_t | \hat{\mathbf{x}}) = \sum_{t=1}^T m_t \log \frac{\exp(H_{\theta}(\hat{\mathbf{x}})_t^\top e(x_t))}{\sum_{x'} \exp(H_{\theta}(\hat{\mathbf{x}})_t^\top e(x'))}, \quad (2)$$

上式中 $m_t=1$ 表示 t 时刻是一个Mask, 需要恢复。 H_{θ} 是一个Transformer, 它把长度为 T 的序列 xx 映射为隐状态的序列 $H_{\theta}(x)=[H_{\theta}(x)_1, H_{\theta}(x)_2, \dots, H_{\theta}(x)_T]$ 。注意: 前面的语言模型的RNN在 t 时刻只能看到之前的时刻, 因此记号是 $h_{\theta}(x_{1:t-1})$; 而BERT的Transformer(不同与用于语言模型的Transformer)可以同时看到整个句子的所有Token, 因此记号是 $H_{\theta}(x)$ 。

这两个模型的优缺点分别为:

- 独立假设

注意等式(2)的约等号 \approx , 它的意思是假设在给定 $x^{\wedge}x^{\wedge}$ 的条件下被Mask的词是独立的(没有关系的), 这个显然并不成立, 比如”New York is a city”, 假设我们Mask住”New”和”York”两个词, 那么给定”is a city”的条件下”New”和”York”并不独立, 因为”New York”是一个实体, 看到”New”则后面出现”York”的概率要比看到”Old”后面出现”York”概率要大得多。而公式(1)没有这样的独立性假设, 它是严格的等号。

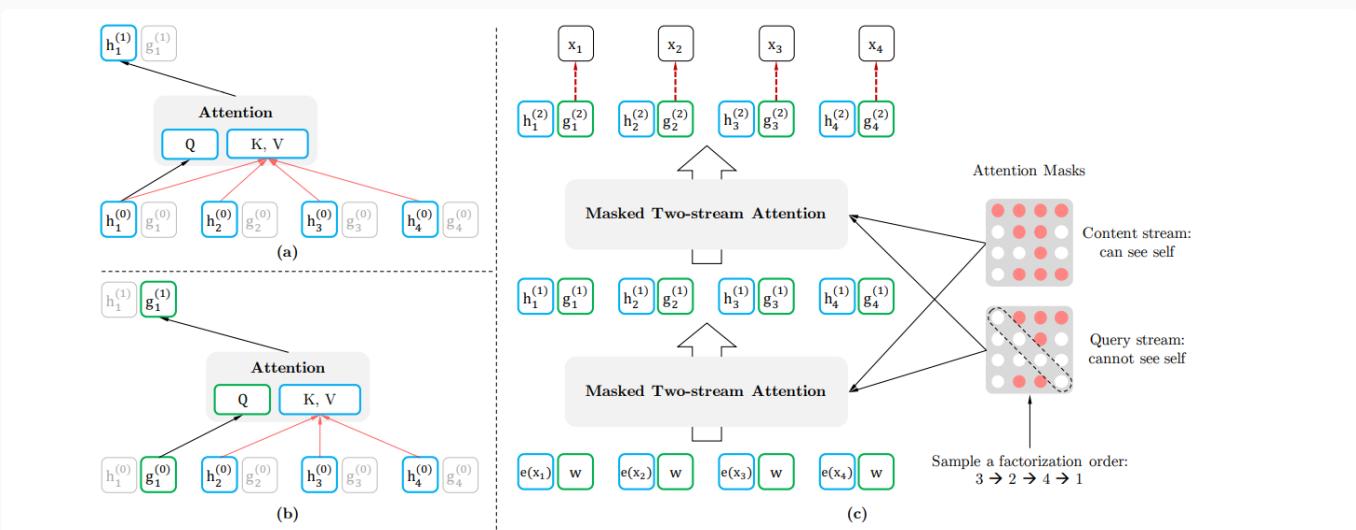
- 输入噪声

BERT的在预训练时会出现特殊的[MASK], 但是它在下游的fine-tuning中不会出现, 这就是出现了不匹配。而语言模型不会有这个问题。

- 双向上下文

语言模型只能参考一个方向的上下文, 而BERT可以参考双向整个句子的上下文, 因此这一点BERT更好一些。

ELMo和GPT最大的问题就是传统的语言模型是单向的——我们是根据之前的历史来预测当前词。但是我们不能利用后面的信息。比如句子”The animal didn’t cross the street because it was too tired”。我们在编码it的语义的时候需要同时利用前后的信息，因为在这个句子中，it可能指代animal也可能指代street。根据tired，我们推断它指的是animal，因为street是不能tired。但是如果把tired改成wide，那么it就是指代street了。传统的语言模型，不管是RNN还是Transformer，它都只能利用单方向的信息。比如前向的RNN，在编码it的时候它看到了animal和street，但是它还没有看到tired，因此它不能确定it到底指代什么。如果是后向的RNN，在编码的时候它看到了tired，但是它还根本没看到animal，因此它也不能知道指代的是animal。Transformer的Self–Attention理论上是可以同时attend to到这两个词的，但是根据前面的介绍，由于我们需要用Transformer来学习语言模型，因此必须用Mask来让它看不到未来的信息，所以它也不能解决这个问题的。注意：即使ELMo训练了双向的两个RNN，但是一个RNN只能看一个方向，因此也是无法”同时”利用前后两个方向的信息的。如果RNN有很多层，比如第一层的正向RNN在编码it的时候编码了animal和street的语义，反向RNN编码了tired的语义，然后第二层的RNN就能同时看到这两个语义，然后判断出it指代animal。理论上是有这种可能，但是实际上很难。举个反例，理论上一个三层(一个隐层)的全连接网络能够拟合任何函数，那我们还需要更多层词的全连接网络或者CNN、RNN干什么呢？如果数据不是足够足够多，如果不对网络结构做任何约束，那么它有很多中拟合的方法，其中很多是过拟合的。但是通过对网络结构的约束，比如CNN的局部特效，RNN的时序特效，多层次的层次结构，对它进行了很多约束，从而使得它能够更好的收敛到最佳的参数。我们研究不同的网络结构(包括resnet、dropout、batchnorm等等)都是为了对网络增加额外的(先验的)约束。



图的左上是Content流Attention的计算，假设排列为 $3 \rightarrow 2 \rightarrow 4 \rightarrow 1$ ，并且我们现在预测第1个位置的词的概率。根据排列，我们可以参考所有4个词的信息，因此 $KV = [h_1^{(0)}, h_2^{(0)}, h_3^{(0)}, h_4^{(0)}]$ ，而 $Q = h_1^{(0)}$ 。

左下是Query流的计算，因为不能参考自己的内容，因此 $KV = [h_2^{(0)}, h_3^{(0)}, h_4^{(0)}]$ ，而 $Q = g_1^{(0)}$ 。

而图的右边是完整的计算过程，我们从下往上看，首先 h 和 g 分别被初始化为 $e(x_i)$ 和 W ，然后Content Mask和Query Mask计算第一层的输出 $h^{(1)}$ 和 $g^{(1)}$ ，然后计算第二层……。注意最右边的Content Mask和Query Mask，我们先看Content Mask。它的第一行全是红点，表示第一个词可以attend to所有的词(根据 $3 \rightarrow 2 \rightarrow 4 \rightarrow 1$)，第二个词可以attend to它自己和第三个词，……。而Query Mask和Content Mask的区别就是不能attend to自己，因此对角线都是白点。
https://kexue.baidu.com/jet/weixin_37947156

预训练模型---- ALBert (减少参数量减少内存)

https://blog.csdn.net/weixin_38937984/article/details/101759506?spm=1001.2014.3001.5501

Introduction

ALBERT 架构的骨干网络与 BERT 是相似的，即使用 Transformer 编码器和 GELU 非线性激活函数。现在先约定一下 BERT 的表示方式，即指定词嵌入大小为 E 、编码器层数为 L 、隐藏层大小为 H 。与 Devlin 等人的研究一样，这篇论文将前馈网络/滤波器大小设置为 $4H$ ，将注意力 Head 的数量设置为 $H/64$ 。

ALBert

Factorized embedding parameterization (嵌入向量参数化的因式分解)

在 BERT 以及后续的 XLNet 和 RoBERTa 中，WordPiece 词嵌入大小 E 和隐藏层大小 H 是相等的，即 $E \equiv H$ 。由于建模和实际使用的原因，这个决策看起来可能并不是最优的。这个选择有两方面缺点：

1. 从建模的角度来说，WordPiece 词嵌入的目标是学习上下文无关的表示，而隐藏层嵌入的目标是学习上下文相关的表示。通过上下文相关的实验，BERT 的表征能力很大一部分来自于使用上下文为学习过程提供上下文相关的表征信号。因此，将 WordPiece 词嵌入大小 E 从隐藏层大小 H 分离出来，可以更高效地利用总体的模型参数，其中 H 要远远大于 E 。
2. 从实践的角度，自然语言处理使用的词典大小 V 非常庞大，如果 E 恒等于 H ，那么增加 H 将直接加大嵌入矩阵的大小，这种增加还会通过 V 进行放大。

因此，对于 ALBERT 而言，研究者对词嵌入参数进行了因式分解，将它们分解为两个小矩阵。研究者不再将 one-hot 向量直接映射到大小为 H 的隐藏空间，而是先将它们映射到一个低维词嵌入空间 E ，然后再映射到隐藏空间。通过这种分解，研究者可以将词嵌入参数从 $O(V \times H)$ 降低到 $O(V \times E + E \times H)$ ，这在 H 远远大于 E 的时候，参数量减少得非常明显。

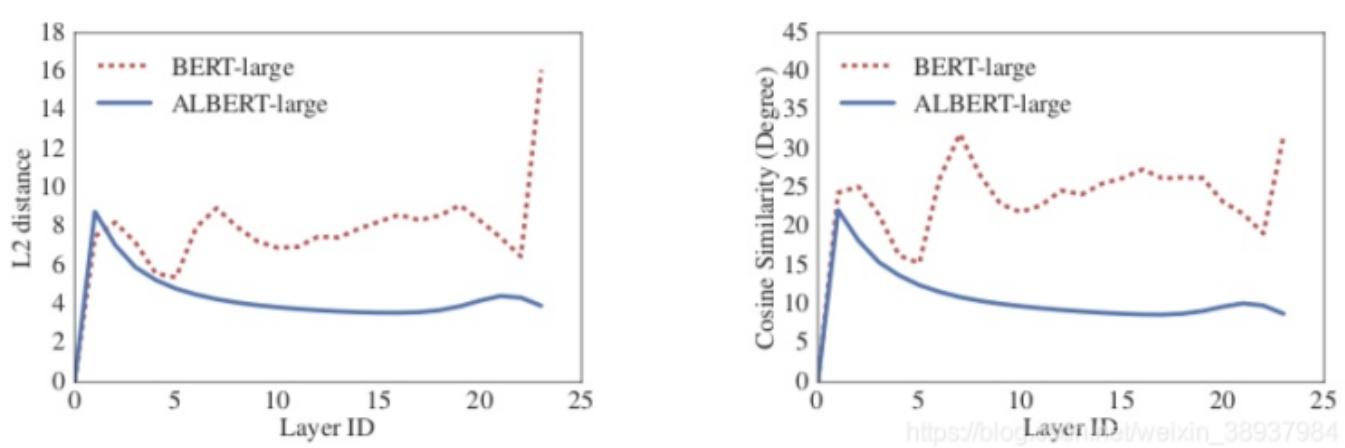
Model	E	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base not-shared	64	87M	89.9/82.9	80.1/77.8	82.9	91.5	66.7	81.3
	128	89M	89.9/82.8	80.3/77.3	83.7	91.5	67.9	81.7
	256	93M	90.2/83.2	80.3/77.4	84.1	91.9	67.3	81.8
	768	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base all-shared	64	10M	88.7/81.4	77.5/74.8	80.8	89.4	63.5	79.0
	128	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1
	256	16M	88.8/81.5	79.1/76.3	81.5	90.3	63.4	79.6
	768	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8

从后续的实验中来看，E的大小与实验效果也不是完全正相关，因此其他实验中E都取的128。

Cross-layer parameter sharing (跨层参数共享)

对于 ALBERT，研究者提出了另一种跨层参数共享机制来进一步提升参数效率。其实目前有很多方式来共享参数，例如只贡献前馈网络不同层之间的参数，或者只贡献注意力机制的参数，而 ALBERT 采用的是贡献所有层的所有参数。

这种机制之前也是有的，但研究者的度量发现词嵌入的 L2 距离和余弦相似性是震荡而不是收敛。如下图 2 展示了每一层输入与输出嵌入矩阵间的 L2 距离与余弦相似性。



研究者发现 ALBERT 从一层到另一层的转换要比 BERT 平滑得多，结果表明，权重共享有效地提升了神经网络参数的鲁棒性。即使相比于 BERT 这两个指标都有所下降，但在 24 层以后，它们也不会收敛到 0。

Inter-sentence coherence loss (句间连贯性损失)

除了自编码语言建模损失外，BERT 还是用了额外的下一句预测损失。下一句预测损失本来是为了提升下游任务的性能，但是后来很多研究者发现这种机制并不是很高效，因此决定去除它。

研究者猜测，下一句预测任务低效的原因，主要是它的难度太小。因为下一句预测将主题预测和连贯性预测结合到单个任务中，然而主题预测比连贯性预测简单得多，因此它与语言建模损失函数学到的内容是有重合的。

研究者表示，句间建模在语言理解中是非常重要的，因此他们提出了一种基于语言连贯性的损失函数。对于 ALBERT，研究者使用了一个句子顺序预测（SOP）损失函数，它会避免预测主题，而只关注建模句子之间的连贯性。

具体的损失函数表达式读者可以查阅原论文，但研究者表示，在使用了该损失函数后，ALBERT 能显著提升下游多句子编码任务的性能。

预训练模型---- BART: (融合BERT和GPT)

背景题目：

BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension

机构：Facebook AI

作者：Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, Luke Zettlemoyer

论文地址：<https://arxiv.org/abs/1910.13461>

收录会议：ACL 2020

代码：<https://github.com/pytorch/fairseq/tree/master/examples/bart>

摘要

文章提出一个预训练sequence-to-sequence的去噪自编码器：BART。BART的训练主要由2个步骤组成：(1)使用任意噪声函数破坏文本(2) 模型学习重建原始文本。BART 使用基于 Transformer 的标准神经机器翻译架构，可视为BERT(双向编码器)、GPT(从左至右的解码器)等近期出现的预训练模型的泛化。文中评估了多种噪声方法，最终发现通过随机打乱原始句子的顺序，再使用首创的新型文本填充方法(即用单个 mask token 替换文本片段，换句话说不管是被mask掉多少个token，都只用一个特定的 mask token表示该位置有token被遮蔽了)能够获取最优性能。BART 尤其擅长处理文本生成任务，不过在自然语言理解任务中也颇有可圈可点之处。在同等训练资源下，BART 在 GLUE 和 SQuAD 数据集上的效果与 RoBERTa 不相伯仲，并在对话摘要、问答和文本摘要等任务中斩获新的记录，在 XSum 数据集上的性能比之前的最佳结果高出了6个ROUGE。在机器翻译任务中，BART 在仅使用目标语言预训练的情况下，获得了比回译系统高出 1.1 个 BLEU 值的结果。此外，文章还使用控制变量在BART 框架内使用其他预训练机制，从而更好地评估影响下游任务性能的因素。

模型

BART结合双向(比如BERT)和自回归(比如GPT) Transformer对模型进行预训练。此外，BART参考了GPT中的激活函数，将ReLU也改为GeLU。BART、BERT和GPT之间的对比如 Figure 1所示。



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.

(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.

(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitrary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

Figure 1: A schematic comparison of BART with BERT (Devlin et al., 2019) and GPT (Radford et al., 2018).

(a)BERT: 用掩码替换随机 token，双向编码文档。由于缺失 token 被单独预测，因此 BERT 较难用于生成任务。(b)GPT: 使用自回归方式预测 token，这意味着GPT可用于生成任务。但是，该模型仅基于左侧上下文预测单词，无法学习双向交互。

©BART: 编码器输入与解码器输出无需对齐，即允许任意噪声变换。使用掩码符号替换文本段，从而破坏文本。使用双向模型编码被破坏的文本（左），然后使用自回归解码器计算原始文档的似然（右）。至于微调，未被破坏的文档是编码器和解码器的输入，研究者使用来自解码器最终隐藏状态的表征。

BART的base版中的encoder和decoder都是6层网络，large则分别12层。BART与BERT还有2点不同
(1)decoder中的每一层都与encoder最后隐藏层执行交叉关注(cross–attention，就像在transformer序列到序列模型中一样)。(2)BERT在预测token之前接一个前馈网络，而BART没有。总的来说，BART比同等大小的BERT模型多了大约10%的参数。

BART的一个关键优势是噪声的随意性，可以动用任何方式(包括改变长度)对原始文本进行破坏。这种方式让模型学习更多地考虑句子的整体长度，并对输入进行更大范围的转换，从而将BERT中MLM和NSP目标统一起来。此外，BART也为微调开创了一个新思路。BART做机器翻译的时候，将BART堆叠在一些额外的Transformer层之上，这些附加的Transformer层实质上是把其他语种翻译成带噪的英语，再通过BART模型，从而将BART作为一个预训练好的目标端语言模型。这种方法在WMT Romanian–English数据集上高出回译系统1.1个BLEU。

预训练

BART的损失函数是decoder的输出与原始文本之间的交叉熵。与其他去噪自编码器(一般需要定制特定的噪声方案)不同的是BART可以使用任何的加噪方式。在极端情况下，源信息都可以全部缺失，此时的BART就蜕化成了一个语言模型。文章中用到的加噪方案(即原始文本如何被破坏)如Figure 2所示。

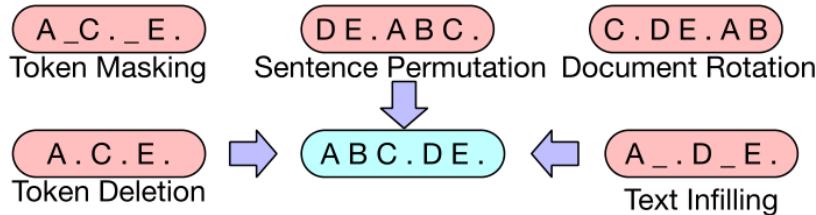


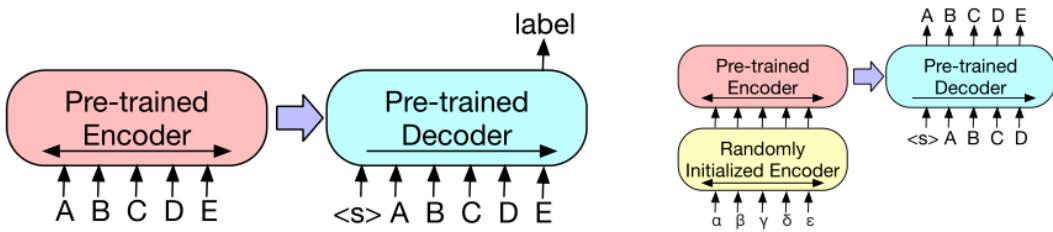
Figure 2: Transformations for noising the input that we experiment with. These transformations can be composed.

主要有：

- (1)Token Masking：与 BERT 一样，BART 随机采样 token，并用 [MASK] 这一预定义的特殊token进行替换。
- (2)Token Deletion：从输入中随机删除 token。与 Token Masking不同，模型必须同时确定输入中缺失的位置。
- (3)Text Infilling：采样多个文本片段，每个文本片段长度服从 $\lambda = 3$ 的泊松分布。每个文本片段用单个 [MASK] token替换。从泊松分布中采样出长度为 0 的文本片段对应 插入 [MASK] token。这种文本填充方法的思想源于SpanBERT，但SpanBERT采样的文本片段长度服从的是几何分布，且用等长的 [MASK] token 序列替换掉文本片段。因此，BART能够迫使模型学习到一个片段中所缺失的token数量。
- (4)Sentence Permutation：这里的句子排列变换是指按句号将文档分割成多个句子，然后随机打乱这些句子。
- (5)Document Rotation：随机均匀地选择一个token，再旋转文档使文档以该 token 作为起始。该任务的目的是训练模型识别文档开头。

Fine-tuning

BART在文本分类和翻译任务中的微调如Figure 3所示。以下具体介绍 BART 在各个下游任务的微调。



(a) To use BART for classification problems, the same input is fed into the encoder and decoder, and the representation from the final output is used.

(b) For machine translation, we learn a small additional encoder that replaces the word embeddings in BART. The new encoder can use a disjoint vocabulary.

Figure 3: Fine tuning BART for classification and translation.

Figure 3: a: 当使用 BART 解决分类问题，用相同的输入文本输入到encoder和decoder，使用最终输出的表征。b: 对于机器翻译任务，训练一个额外的小型encoder来替换 BART 中的词嵌入。新encoder 可使用不同的词汇。

序列分类任务：

同一个输入同时输入到encoder 和decoder，将最后decoder的token的最终隐层状态被输入到一个新的多类别线性分类器中。该方法与 BERT 中的 CLS token 类似，不过 BART 在decoder最后额外添加了一个 token，如此该 token 在decoder中的表征可以关注到完整输入的decoder状态（见Figure 3a）。

token 分类任务：

token的分类任务，比如SQuAD中答案端点的分类。将完整文档输入到encoder和decoder中，使用decoder最上方的隐状态作为每个token的表征以判断该 token 的类别，比如是否为答案端部。

序列生成任务：

由于 BART 具备自回归解码器，因此可以直接应用到序列生成任务(如生成式问答和文本摘要)进行微调。在这两项任务中，从输入复制经过处理的信息，这与去噪预训练目标紧密相关。encoder的输入是输入序列，decoder以自回归的方式生成输出。

机器翻译：

BART用以机器翻译的时候，将整个BART(包括encoder和decoder)作为一个单独的预训练decoder，并增加一系列的从双语语料学习而得的encoder，如 Figure 3b所示。具体是用一个新的随机初始化encoder替换 BART encoder的嵌入层。该模型以端到端的方式训练，即训练一个新的encoder将外来词映射到输入(BART可将其去噪为英文)。这个新的encoder可以使用不同于原始 BART 模型的词汇。

源encoder的训练分两步，均需要将BART模型输出的交叉熵损失进行反向传播。

- (1)冻结 BART 的大部分参数，仅更新随机初始化的源encoder、BART 位置嵌入和 BART encoder第一层的自注意力输入投影矩阵。
- (2)将所有模型参数进行少量迭代训练。

实验

预训练目标对比

文章中还充分对比了不同预训练目标的影响，包括：

- (1)语言模型：与GPT类似，训练一个从左到右的Transformer语言模型。该模型相当于BART的decoder，只是没有交叉注意(cross-attention)。
- (2)排列语言模型：该模型基于XLNet，采样1/6的token，并以自回归的随机顺序生成。为了与其他模型保持一致，这里没有引入相对位置编码和XLNet中的片段级的循环注意力机制。
- (3)带遮蔽的语言模型：与BERT相同，15%的token用 [MASK] token替换，训练模型重建出这些被遮蔽掉的token。
- (4)多任务遮蔽的语言模型：与 UniLM 一样，使用额外self-attention mask训练带遮蔽的语言模型。自注意力遮蔽按如下比例随机选择:1/6从左到右；1/6从右到左；1/3未遮蔽；剩余的1/3中前50%的未遮蔽，其余的从左到右遮蔽。
- (5)带遮蔽的seq-to-seq：与MASS模型类似，遮蔽一个片段中50%的token，并训练一个序列到序列模型预测被遮蔽的tokens。

实验过程对比了两种方案：

- (1)将所有任务视为sequence-to-sequence问题，source端输入到encoder，decoder端的输出即为target结果。
- (2)在decoder端将source作为target的一个前缀，且只在序列的target部分有损失函数。

实验发现前者对BART模型更有效，后者对其他模型更有效。更加详细的实验结果如 Table 1所示。

Model	SQuAD 1.1	MNLI	ELI5	XSum	ConvAI2	CNN/DM
	F1	Acc	PPL	PPL	PPL	PPL
BERT Base (Devlin et al., 2019)	88.5	84.3	-	-	-	-
Masked Language Model	90.0	83.5	24.77	7.87	12.59	7.06
Masked Seq2seq	87.0	82.1	23.40	6.80	11.43	6.19
Language Model	76.7	80.1	21.40	7.00	11.51	6.56
Permuted Language Model	89.1	83.7	24.03	7.69	12.23	6.96
Multitask Masked Language Model	89.2	82.4	23.73	7.50	12.39	6.74
<hr/>						
BART Base						
w/ Token Masking	90.4	84.1	25.05	7.08	11.73	6.10
w/ Token Deletion	90.4	84.1	24.61	6.90	11.46	5.87
w/ Text Infilling	90.8	84.0	24.26	6.61	11.05	5.83
w/ Document Rotation	77.2	75.3	53.69	17.14	19.87	10.59
w/ Sentence Shuffling	85.4	81.5	41.87	10.93	16.67	7.89
w/ Text Infilling + Sentence Shuffling	90.8	83.8	24.17	6.62	11.12	5.41

Table 1: Comparison of pre-training objectives. All models are of comparable size and are trained for 1M steps on a combination of books and Wikipedia data. Entries in the bottom two blocks are trained on identical data using the same code-base, and fine-tuned with the same procedures. Entries in the second block are inspired by pre-training objectives proposed in previous work, but have been simplified to focus on evaluation objectives (see §4.1). Performance varies considerably across tasks, but the BART models with text infilling demonstrate the most consistently strong performance.

Table 1: 预训练目标对比。各个预训练目标源于BERT, MASS, GPT, XLNet和UniLM。对比的模型都是尺寸近似，训练步数都是1M，预训练使用的数据也相同。

从中可以看出使用文本填充方案(Text Infilling)的BART战绩斐然。从中可以得出以下结论：

- (1)在不同的任务中，预训练方法的表现有显著差异。换句话说，**预训练方法的有效性高度依赖于任务本身**。比如，一个简单的语言模型在ELI5数据集上可以夺冠，但是在SQuAD上的结果却是最差的。
- (2)遮蔽Token至关重要。只使用旋转文档或句子组合的预训练目标则效果较差，效果较好的都是使用了token的删除或遮蔽作为预训练目标。此外，在生成任务上，**删除token似乎比遮蔽token更胜一筹**。
- (3)从左到右的预训练目标有助于文本生成任务。遮蔽语言模型和排列语言模型在文本生成任务上不如其他模型。而这两种模型在预训练阶段都没有用到从左到右的自回归语言模型。
- (4)对于SQuAD而言双向的encoder至关重要。因为上下文在分类决策中至关重要，**BART仅用双向层数的一半就能达到BERT类似的性能**。
- (5)预训练目标并不是唯一重要的因素。这里的排列语言模型略逊于XLNet，其中一些差异可能是由于没有使用XLNet架构中的其他的改进，如相对位置编码和片段级的循环机制。
- (6)纯语言模型在ELI5数据集上技压群雄，其困惑度远优于其他模型。这表明当输出仅受到输入的松散约束时，BART较为低效。

总而言之，使用文本填充预训练目标的BAR在多项任务上(除了ELI5之外)效果都很好。

Large版模型对比

自然语言理解任务

由于更大模型和更大batch size有助于下游任务性能的提升，所以文章还进一步对比各模型的large版。Large版的BART，encoder和decoder分别有12层，隐层大小为1024，batch size与RoBERTa一样都是8000，模型预训练了500000个step。tokenized方法借用 GPT-2 中的字节对编码(BPE)。各个模型在GLUE上的实验对比结果如 Table 2所示。

	SQuAD 1.1 EM/F1	SQuAD 2.0 EM/F1	MNLI m/mm	SST Acc	QQP Acc	QNLI Acc	STS-B Acc	RTE Acc	MRPC Acc	CoLA Mcc
BERT	84.1/90.9	79.0/81.8	86.6/-	93.2	91.3	92.3	90.0	70.4	88.0	60.6
UniLM	-/-	80.5/83.4	87.0/85.9	94.5	-	92.7	-	70.9	-	61.1
XLNet	89.0/94.5	86.1/88.8	89.8/-	95.6	91.8	93.9	91.8	83.8	89.2	63.6
RoBERTa	88.9/ 94.6	86.5/89.4	90.2/90.2	96.4	92.2	94.7	92.4	86.6	90.9	68.0
BART	88.8/ 94.6	86.1/89.2	89.9/90.1	96.6	92.5	94.9	91.2	87.0	90.4	62.8

Table 2: Results for large models on SQuAD and GLUE tasks. BART performs comparably to RoBERTa and XLNet, suggesting that BART’s uni-directional decoder layers do not reduce performance on discriminative tasks.

Table 2: Large版模型在 SQuAD 和 GLUE 上的实验结果。BART 的效果可比肩 RoBERTa 和 XLNet，这表明 BART 的单向decoder层并不会降低模型在判别任务上的性能。

各个模型在SQuAD上的对比结果如Table 3所示。

	CNN/DailyMail			XSum		
	R1	R2	RL	R1	R2	RL
Lead-3	40.42	17.62	36.67	16.30	1.60	11.95
PTGEN (See et al., 2017)	36.44	15.66	33.42	29.70	9.21	23.24
PTGEN+COV (See et al., 2017)	39.53	17.28	36.38	28.10	8.02	21.72
UniLM	43.33	20.21	40.51	-	-	-
BERTSUMABS (Liu & Lapata, 2019)	41.72	19.39	38.76	38.76	16.33	31.15
BERTSUMEXTABS (Liu & Lapata, 2019)	42.13	19.60	39.18	38.81	16.50	31.27
BART	44.16	21.28	40.90	45.14	22.27	37.25

Table 3: Results on two standard summarization datasets. BART outperforms previous work on summarization on two tasks and all metrics, with gains of roughly 6 points on the more abstractive dataset.

Table 3: BART的结果与XLNet和RoBERTa不相伯仲。

总体而言，BART在自然语言理解任务上与其他先进模型不相上下。这表明BART在生成任务上的进一步突破并不是以牺牲自然语言理解性能为代价。

自然语言生成任务

在文本生成任务中选用了摘要生成(CNN/DailyMail 和XSum)、对话(CONVAl2)和生成式问答(ELI5，是一个长篇问答数据集)中对应的数据集进行评测，结果如 Table 4所示。

ConvAI2		
	Valid F1	Valid PPL
Seq2Seq + Attention	16.02	35.07
Best System	19.09	17.51
BART	20.72	11.85

Table 4: BART outperforms previous work on conversational response generation. Perplexities are renormalized based on official tokenizer for ConvAI2.

Table 4: 在两个标准摘要数据集上的结果。

从结果可以看出，在这两个摘要任务上，BART 在所有度量指标上均优于之前的模型。BART在更抽象的 XSum 数据集上的比之前最优的RoBERTa高出3.5个点(所有的ROUGE指标)。此外，从人工评测的角度来看，BART也大幅优于之前的模型。但是，与人类的摘要结果仍然有差距。

各模型在CONVAI2上的实验结果如Table 5所示。

ELI5			
	R1	R2	RL
Best Extractive	23.5	3.1	17.5
Language Model	27.8	4.7	23.1
Seq2Seq	28.3	5.1	22.8
Seq2Seq Multitask	28.9	5.4	23.1
BART	30.6	6.2	24.3

Table 5: BART achieves state-of-the-art results on the challenging ELI5 abstractive question answering dataset. Comparison models are from [Fan et al. \(2019\)](#).

Table 5: BART 在对话生成任务上的性能优于之前的研究。其中困惑度基于 ConvAI2 官方 tokenizer 进行了重新归一化。

在ELI5数据集上的评测结果如Table 6所示。

RO-EN	
Baseline	36.80
Fixed BART	36.29
Tuned BART	37.96

Table 6: The performance (BLEU) of baseline and BART on WMT'16 RO-EN augmented with back-translation data. BART improves over a strong back-translation (BT) baseline by using monolingual English pre-training.

Table 6: BART在具有挑战性的ELI5生产式问答数据集上达到了最先进的结果。

发现BART的性能比之前最好的工作(指Seq2Seq Multi-task)高出1.2个 ROUGE-L。因为该数据集中的问题只对答案进行了微弱的指定，从而提升了该数据集的难度。

leader board结果如下：

机器翻译任务

BART在WMT16 Romanian–English上与其他模型的对比结果如Table 7所示。

Source Document (abbreviated)	BART Summary
The researchers examined three types of coral in reefs off the coast of Fiji ... The researchers found when fish were plentiful, they would eat algae and seaweed off the corals, which appeared to leave them more resistant to the bacterium <i>Vibrio coralliilyticus</i> , a bacterium associated with bleaching. The researchers suggested the algae, like warming temperatures, might render the corals' chemical defenses less effective, and the fish were protecting the coral by removing the algae.	Fisheries off the coast of Fiji are protecting coral reefs from the effects of global warming, according to a study in the journal Science.
Sacoolas, who has immunity as a diplomat's wife, was involved in a traffic collision ... Prime Minister Johnson was questioned about the case while speaking to the press at a hospital in Watford. He said, "I hope that Anne Sacoolas will come back ... if we can't resolve it then of course I will be raising it myself personally with the White House."	Boris Johnson has said he will raise the issue of US diplomat Anne Sacoolas' diplomatic immunity with the White House.
According to Syrian state media, government forces began deploying into previously SDF controlled territory yesterday. ... On October 6, US President Donald Trump and Turkish President Recep Tayyip Erdogan spoke on the phone. Then both nations issued statements speaking of an imminent incursion into northeast Syria On Wednesday, Turkey began a military offensive with airstrikes followed by a ground invasion.	Syrian government forces have entered territory held by the US-backed Syrian Democratic Forces (SDF) in response to Turkey's incursion into the region.
This is the first time anyone has been recorded to run a full marathon of 42.195 kilometers (approximately 26 miles) under this pursued landmark time. It was not, however, an officially sanctioned world record, as it was not an "open race" of the IAAF. His time was 1 hour 59 minutes 40.2 seconds. Kipchoge ran in Vienna, Austria. It was an event specifically designed to help Kipchoge break the two hour barrier.	Kenyan runner Eliud Kipchoge has run a marathon in less than two hours.
PG&E stated it scheduled the blackouts in response to forecasts for high winds amid dry conditions. The aim is to reduce the risk of wildfires. Nearly 800 thousand customers were scheduled to be affected by the shutoffs which were expected to last through at least midday tomorrow.	Power has been turned off to millions of customers in California as part of a power shutoff plan.

Table 7: Example summaries from the XSum-tuned BART model on WikiNews articles. For clarity, only relevant excerpts of the source are shown. Summaries combine information from across the article and prior knowledge.

Table 7: BART 和基线模型(Transformer)在机器翻译任务上的性能对比情况。

参与对比的模型使用数据集包括 WMT16 RO-EN 和用回译系统做的扩增数据。可以看出BART使用单语英文预训练，性能结果优于基线模型。

总结

文本介绍了一种预训练模型：BART。该模型是一个预训练sequence-to-sequence的去噪自编码器。BART不同于一些只能针对特定的噪声的去噪自编码器，可以使用任意方式破坏原始文本，最极端的情况下，源文本信息全部丧失，此时BART蜕变为一个语言模型。BART模型中的文本填充方法让模型学习更多地考虑句子的整体长度，并对输入进行更大范围的转换，从而将BERT中MLM和NSP目标统一起来，加大了模型学习难度。BART不仅能在自然语言理解任务上与先进模型不相上下，而且在文本生成任务上可以碾压其他模型。

预训练模型---- SpanBERT&ERNIE2

https://blog.csdn.net/weixin_38937984/article/details/101997286?spm=1001.2014.3001.5501

预训练模型---- MASS

https://blog.csdn.net/weixin_38937984/article/details/102479003?spm=1001.2014.3001.5501

预训练模型---- ELECTRA

https://blog.csdn.net/weixin_38937984/article/details/102893601?spm=1001.2014.3001.5501

预训练模型---- T5: Text-To-Text Transfer Transformer

https://blog.csdn.net/weixin_38937984/article/details/102894767?spm=1001.2014.3001.5501

预训练模型---- 小结

https://blog.csdn.net/weixin_38937984/article/details/102894927?spm=1001.2014.3001.5501

预训练模型---- 华为2021newest

https://mp.weixin.qq.com/s/AzWfSgSfEYUFZxqM7Ws_BA

预训练模型---- NEZHA

论文: <https://arxiv.org/pdf/1909.00204.pdf>

代码: <https://github.com/huawei-noah/Pretrained-Language-Model/tree/master/NEZHA-PyTorch>

Functional Relative Positional Encoding

$$\begin{aligned} a_{ij}[2k] &= \sin((j - i)/(10000^{\frac{2 \cdot k}{d_z}})), \\ a_{ij}[2k + 1] &= \cos((j - i)/(10000^{\frac{2 \cdot k}{d_z}})). \end{aligned}$$

Whole Word Masking

[Original Sentence]

使用语言模型来预测下一个词的probability。

[Original Sentence with CWS]

使用语言模型来预测下一个词的 probability。

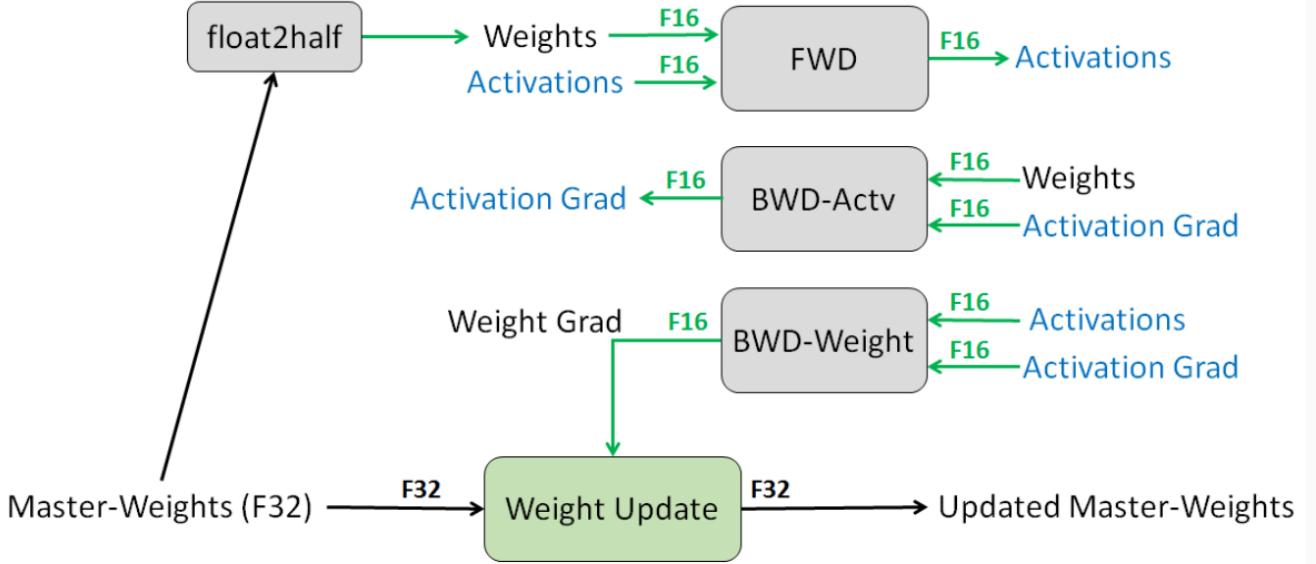
[Original BERT Input]

使用语言 [MASK] 型来 [MASK] 测下一个词的 pro [MASK] ##lity。

[Whold Word Masking Input]

使用语言 [MASK] [MASK] 来 [MASK] [MASK] 下一个词的 [MASK] [MASK] [MASK]。

Mixed Precision Training



LAMB Optimizer

Algorithm 1 The LAMB optimizer

Data: the number of iterations T ; the number of layers L ; the learning rate η ; (the users only need to input η , default setting: weight decay rate $\lambda=0.01$, $\beta_1=0.9$, $\beta_2=0.999$, $\epsilon=1e-6$)

Result: the trained model w

initialization

$t = 0$

while $t < T$ **do**

***** within each iteration *****

Get a batch of data x_t

$l = 0$

while $l < L$ **do**

***** within each layer *****

$$g_t^l = \nabla L(w_{t-1}^l, x_t)$$

$$m_t^l = \beta_1 m_{t-1}^l + (1 - \beta_1) g_t^l$$

$$v_t^l = \beta_2 v_{t-1}^l + (1 - \beta_2) g_t^l \odot g_t^l$$

$$\hat{m}_t^l = m_t^l / (1 - \beta_1^t)$$

$$\hat{v}_t^l = v_t^l / (1 - \beta_2^t)$$

$$r_1 = \|w_{t-1}^l\|_2$$

$$r_2 = \left\| \frac{\hat{m}_t^l}{\sqrt{\hat{v}_t^l + \epsilon}} + \lambda w_{t-1}^l \right\|_2$$

$$r = r_1 / r_2$$

$$\eta^l = r \times \eta$$

$$w_t^l = w_{t-1}^l - \eta^l \times \left(\frac{\hat{m}_t^l}{\sqrt{\hat{v}_t^l + \epsilon}} + \lambda w_{t-1}^l \right)$$

end

end

预训练模型---- PanGu

论文: <https://git.openi.org.cn/PCL-Platform.Intelligence/PanGu-Alpha/src/branch/master/PANGU-%CE%B1.pdf>

预训练模型----TinyBERT

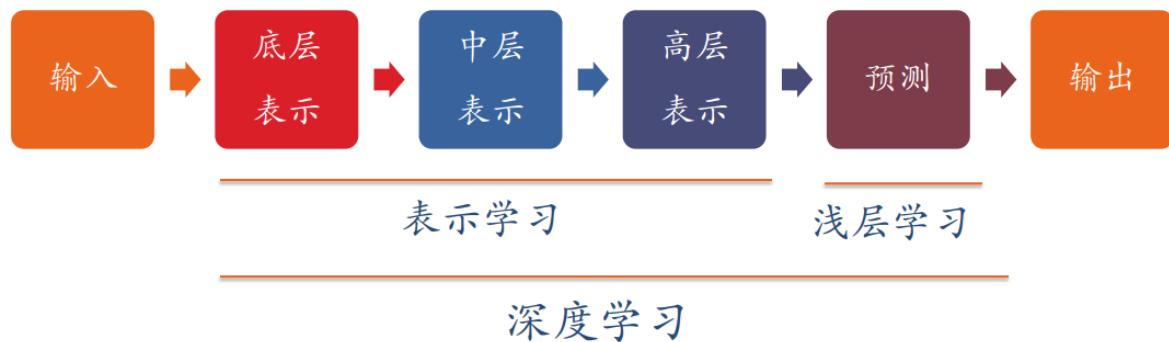
论文: <https://arxiv.org/abs/1909.10351>

深度学习



深度学习

- ▶ 深度学习=表示学习+浅层学习
- ▶ 难点：贡献度分配问题



机器学习

机器学习的可行性

通过Hoeffding 不等式，引出了机器学习中学习的可行性。

机器学习中的“学习”二字到底指的是什么，或者说机器为什么能够学习，到底学到了什么东西？要回答这个问题，首先要确认一个大前提：数据集（包括训练集和测试集）是从相同分布中产生的，也就是说产生数据的环境应该是一致的，否则假如训练集与数据集的产生方式不一样，那么从训练集中是不可能学到训练集中相关的知识的

有了上面这个大前提，从概率论的角度来讲，机器学习学的就是这个概率分布，；或者说是模式识别中的模式（pattern）；然后用学习到的概率分布去预测未见过但是也是在这个概率分布下产生的样本，这样一来，便称机器能够“学习”了。

下面的内容便是将这一过程通过数学来严谨化

Hoeffding 不等式

为了引入 Hoeffding 不等式，首先来看一下概率论中一个简单的例子：假如一个罐子中有绿色和橙色两种弹珠，现在想知道罐子中橙色弹珠的比例，该怎么做？

最直观的方法就是将罐子中所有的弹珠分类并计数，然后计算橙色弹珠的比例。但是当罐子中的弹珠数目变得很大的时候，在实际中显然是无法将所有弹珠都数一遍。

这时便需要进行抽样并从抽出的样本（sample）中估计橙色弹珠的比例，但是抽样一定会带来一定的误差的，而且直观上来看，抽样的样本数目越多，误差越小。而 Hoeffding 不等式就是描述这个误差跟抽样数目之间的关系，假如橙色弹珠的真实比例为 μ , 而从样本中估计出的比例为 ν , 样本大小为 N , 则对应的 Hoeffding 不等式如下

$$p(|\nu - \mu| > \epsilon) \leq 2 \exp(-2\epsilon^2 N)$$

上式中的 ϵ 表示允许的误差范围

从 Hoeffding 不等式到机器学习

假如将上面的罐子中的一个弹珠抽象为机器学习中的一个样本，考虑一个二分类问题，绿色弹珠表示样本标签与我们的模型 h 预测出的标签一致，而橙色弹珠则表示样本标签与预测标签不一致。则橙色弹珠的比例就是模型 h 的错误率。同时将模型 h 在全部弹珠中的错误率记为 $E_{out}(h)$, 而在样本中的错误率记为 $E_{in}(h)$, 则根据 Hoeffding 不等式有

$$p(|E_{in}(h) - E_{out}(h)| > \epsilon) \leq 2 \exp(-2\epsilon^2 N)$$

也就是说，训练样本的数目越大， $E_{in}(h)$ 和 $E_{out}(h)$ ，也就是训练误差和泛化误差越接近。

这一等式实际上代表了PAC(probably approximately correct) 学习理论中的 probably 部分，PAC 理论简单描述如下(摘自 Wikipedia)

In this framework, the learner receives samples and must select a generalization function (called the hypothesis) from a certain class of possible functions. The goal is that, with high probability (the “probably” part), the selected function will have low generalization error (the “approximately correct” part).

上面的 Hoeffding 不等式只是说明了训练误差和泛化误差可以很接近，但是这一接近必须要在训练误差也就是 $E_{in}(h)$ 很小的情况下才有意义，否则大的训练误差就有大的泛化误差，而大的泛化误差的模型实际上是没有任何意义的。而小的训练误差对应到 PAC 中的 AC(approximately correct) 部分。

从一个 hypothesis 到多个 hypothesis

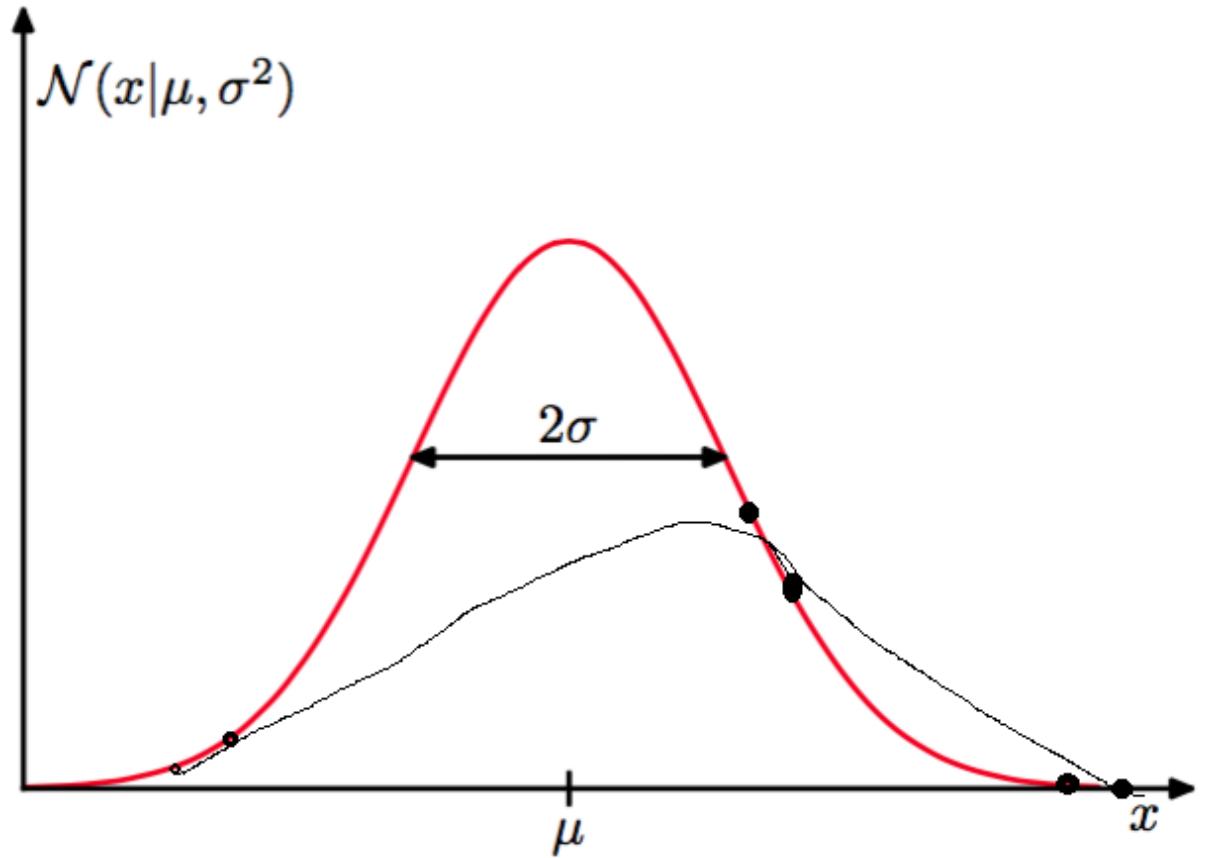
上面的 Hoeffding 不等式描述的是一个 hypothesis 也就是一个 h 的情况，但是实际中往往有多个 hypothesis 可选，这个时候的 Hoeffding 不等式又会变成怎样？

首先回顾一下单个 h 的 Hoeffding 不等式，它告诉我们下面这个事情发生的概率很大： h 在抽取的样本上得到的样本误差（也就是训练误差）跟 h 的总体误差（也就是泛化误差）很接近。

从另一个角度来讲，也就是说还是有很小的概率抽出一些样本，使得 h 在样本上得到的误差与其在总体上得到的误差相差很大（实际上就是抽出的样本不能很好反映总体），讲义中将这部分的 sample 称为 **bad data**，就是使得 h 的 $E_{in}(h)$ 很小， $E_{out}(h)$ 很大的样本。如下图所示，如果进行多次抽样，那么肯定有一些样本会导致 $E_{in}(h)$ 和 $E_{out}(h)$ 的差距较大。

	\mathcal{D}_1	\mathcal{D}_2	...	\mathcal{D}_{1126}	...	\mathcal{D}_{5678}	...	Hoeffding
h	BAD					BAD		$\mathbb{P}_{\mathcal{D}} [\text{BAD } \mathcal{D} \text{ for } h] \leq \dots$

注：这里 $E_{in}(h)$ 很小， $E_{out}(h)$ 很大其实已经是我们常听到的过拟合现象，影响过拟合的因素有很多，而抽样的数据的分布是否能够代表整体数据的分布则是其中一个因素。下面是一个简单的例子：对于一个高斯分布产生的数据，如果抽样数据是图中的黑色点，那么拟合出来的曲线可能是图中的黑线，也就是说假如抽样数据的分布如果跟原始数据分布不一致，我们的模型拟合了抽样的数据，对于原始数据而言，自然没有预测能力，也就是 $E_{in}(h)$ 很小， $E_{out}(h)$ 很大，可以说是过拟合了抽样的数据。



回到讨论的话题，如果对于MM个hypothesis 呢？上图可以改为如下形式

	\mathcal{D}_1	\mathcal{D}_2	...	\mathcal{D}_{1126}	...	\mathcal{D}_{5678}	Hoeffding
h_1	BAD					BAD	$\mathbb{P}_{\mathcal{D}} [\text{BAD } \mathcal{D} \text{ for } h_1] \leq \dots$
h_2		BAD					$\mathbb{P}_{\mathcal{D}} [\text{BAD } \mathcal{D} \text{ for } h_2] \leq \dots$
h_3	BAD	BAD				BAD	$\mathbb{P}_{\mathcal{D}} [\text{BAD } \mathcal{D} \text{ for } h_3] \leq \dots$
...							
h_M	BAD					BAD	$\mathbb{P}_{\mathcal{D}} [\text{BAD } \mathcal{D} \text{ for } h_M] \leq \dots$
all	BAD	BAD				BAD	?

在上图中，由于每个 hypothesis 都不同，因此对各个 hypothesis 而言其 bad data 也不同，只有当样本对各个 hypothesis 而言都不是 bad 的时候，才不会泛化误差和训练误差差距很大的情况。在有M个 hypothesis 的时候，用 Hoeffding 不等式表示选择了 bad data 的概率为

$$\begin{aligned}
& \mathbb{P}_{\mathcal{D}}[\text{BAD } \mathcal{D}] \\
&= \mathbb{P}_{\mathcal{D}}[\text{BAD } \mathcal{D} \text{ for } h_1 \text{ or } \text{BAD } \mathcal{D} \text{ for } h_2 \text{ or } \dots \text{ or } \text{BAD } \mathcal{D} \text{ for } h_M] \\
&\leq \mathbb{P}_{\mathcal{D}}[\text{BAD } \mathcal{D} \text{ for } h_1] + \mathbb{P}_{\mathcal{D}}[\text{BAD } \mathcal{D} \text{ for } h_2] + \dots + \mathbb{P}_{\mathcal{D}}[\text{BAD } \mathcal{D} \text{ for } h_M] \\
&\quad (\text{union bound}) \\
&\leq 2 \exp(-2\epsilon^2 N) + 2 \exp(-2\epsilon^2 N) + \dots + 2 \exp(-2\epsilon^2 N) \\
&= 2M \exp(-2\epsilon^2 N)
\end{aligned}$$

上面的不等式表明，对于有限多个 hypothesis 而言， $E_{in}(h) \approx E_{out}(h)$ 还是 PAC 的，只是两者误差的 upper bound 变大了，但是数据量 N 的增大能够抵消这一影响。

在上面的前提下，在有多个 hypothesis 的情况下，只需要选择 $E_{in}(h)$ 小的，就能保证 $E_{out}(h)$ 也是小的，也就是学习是可行的，样本上小，全部就小。

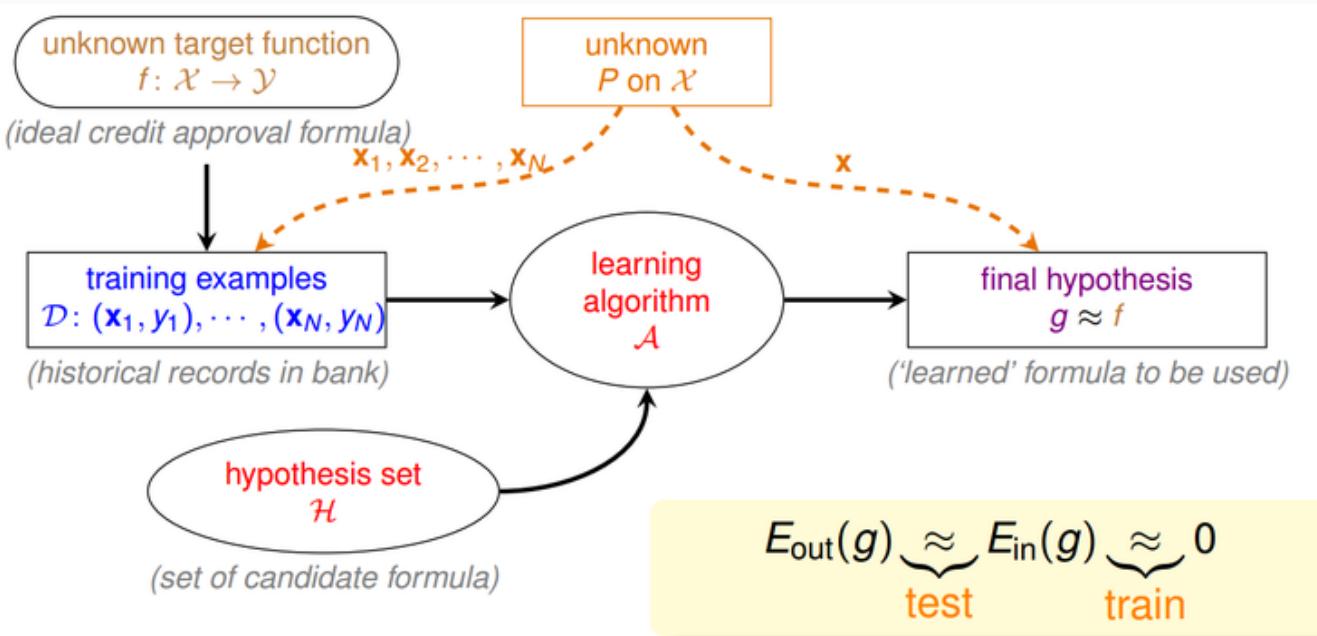
小结

本文主要是通过 Hoeffding 不等式证明了当模型的所有 hypothesis 的个数 M 为有限个时，样本数目 N 足够大时，就能够保证泛化误差 $E_{out}(h)$ 和训练误差 $E_{in}(h)$ 很接近。

这时候只要找到一个 hypothesis 使得 $E_{in}(h)$ 很小，那么 $E_{out}(h)$ 也会很小，从而达到学习的目的。

当然有一个大前提就是训练样本和测试样本必须要在同一分布下产生，否则学习无从谈起。

上面的内容可通过下图进行描述



MLE 与 MAP

参考: http://www.cs.cmu.edu/~aarti/Class/10701_Spring14/slides/MLE_MAP_Part1.pdf

参考: https://personal.utdallas.edu/~nrr150130/cs7301/2016fa/lects/Lecture_14_Bayes.pdf

两大学派的争论

频率学派(Frequentist)和贝叶斯学派(Bayesian)对世界的认知有本质不同：频率学派认为世界是确定的，有一个本体，这个本体的真值是不变的，我们的目标就是要找到这个真值或真值所在的范围；而贝叶斯学派认为世界是不确定的，人们对世界先有一个预判，而后通过观测数据对这个预判做调整，我们的目标是要找到最优的描述这个世界的概率分布。

在对事物建模时，用 θ 表示模型的参数，解决问题的本质就是求 θ 。那么频率学派和贝叶斯学派的区别在于：

- **频率学派：存在唯一真值 θ 。**举一个简单直观的例子—抛硬币，我们用 $P(\text{head})$ 来表示硬币的 bias。抛一枚硬币 100 次，有 20 次正面朝上，要估计抛硬币正面朝上的 bias $P(\text{head})=\theta P(\text{head})=\theta$ 。在频率学派来看， $\theta = 20 / 100 = 0.2$ ，很直观。当数据量趋于无穷时，这种方法能给出精准的估计；然而缺乏数据时则可能产生严重的偏差。例如，对于一枚均匀硬币，即 $\theta = 0.5$ ，抛掷 5 次，出现 5 次正面（这种情况出现的概率是 $1/2^5=3.125\%$ ），频率学派会直接估计这枚硬币 $\theta = 1$ ，出现严重错误。
- **贝叶斯学派： θ 是一个随机变量，符合一定的概率分布。**在贝叶斯学派里有两大输入和一大输出，输入是先验 (prior) 和似然 (likelihood)，输出是后验 (posterior)。先验，即 $P(\theta)$ ，指的是在没有观测到任何数据时对 θ 的预先判断，例如给我一个硬币，一种可行的先验是认为这个硬币有很大的概率是均匀的，有较小的概率是不均匀的；似然，即 $P(X|\theta)$ ，是假设 θ 已知后我们观察到的数据应该是什么样子的；后验，即 $P(\theta|X)$ ，是最终的参数分布。贝叶斯估计的基础是贝叶斯公式，如下：

$$P(\theta|X)=P(X|\theta)\times P(\theta)P(X)$$

同样是抛硬币的例子，对一枚均匀硬币抛 5 次得到 5 次正面，如果先验认为大概率下这个硬币是均匀的（例如最大值取在 0.5 处的 Beta 分布），那么 $P(\text{head})$ ，即 $P(\theta|X)$ ，是一个 distribution，最大值会介于 0.5~1 之间，而不是武断的 $\theta = 1$ 。

这里有两点值得注意的地方：

- (1) 随着数据量的增加，参数分布会越来越向数据靠拢，先验的影响力会越来越小(2)如果先验是 uniform distribution，则贝叶斯方法等价于频率方法。因为直观上来讲，先验是 uniform distribution 本质上表示对事物没有任何预判

MLE – 最大似然估计

Maximum Likelihood Estimation, MLE是频率学派常用的估计方法!

假设数据 x_1, x_2, \dots, x_n 是 i.i.d.的一组抽样, $X=(x_1, x_2, \dots, x_n)$ 。其中i.i.d.表示Independent and identical distribution, 独立同分布。那么MLE对 θ 的估计方法可以如下推导:

$$\begin{aligned}\hat{\theta}_{\text{MLE}} &= \arg \max P(X; \theta) \\&= \arg \max P(x_1; \theta)P(x_2; \theta) \cdots P(x_n; \theta) \\&= \arg \max \log \prod_{i=1}^n P(x_i; \theta) \\&= \arg \max \sum_{i=1}^n \log P(x_i; \theta) \\&= \arg \min - \sum_{i=1}^n \log P(x_i; \theta)\end{aligned}$$

最后这一行所优化的函数被称为Negative Log Likelihood (NLL), 这个概念和上面的推导是非常重要的!

我们经常在不经意间使用MLE, 例如

上文中关于频率学派求硬币概率的例子, 其方法其实本质是由优化NLL得出。给定一些数据, 求对应的高斯分布时, 我们经常会算这些数据点的均值和方差然后带入到高斯分布的公式, 其理论依据是优化NLL深度学习做分类任务时所用的cross entropy loss, 其本质也是MLE

MAP – 最大后验估计

Maximum A Posteriori, MAP是贝叶斯学派常用的估计方法!

同样的, 假设数据 x_1, x_2, \dots, x_n 是i.i.d.的一组抽样, $X=(x_1, x_2, \dots, x_n)$ 。那么MAP对 θ 的估计方法可以如下推导:

$$\begin{aligned}\hat{\theta}_{\text{MAP}} &= \arg \max P(\theta|X) \\&= \arg \min - \log P(\theta|X) \\&= \arg \min - \log P(X|\theta) - \log P(\theta) + \log P(X) \\&= \arg \min - \log P(X|\theta) - \log P(\theta)\end{aligned}$$

其中, 第二行到第三行使用了贝叶斯定理, 第三行到第四行 $P(X)$ 可以丢掉因为与 θ 无关。注意 $-\log P(X|\theta)$ 其实就是NLL, 所以MLE和MAP在优化时的不同就是在于先验项 $-\log P(\theta)$ 。假定先验

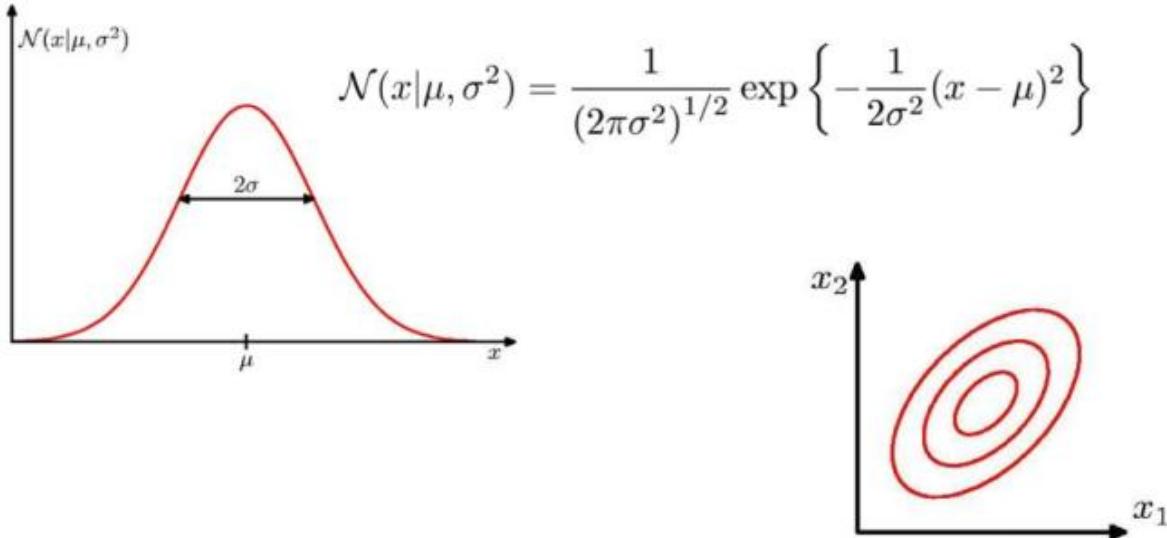
是一个高斯分布，即

$$P(\theta) = \text{constant} \times e^{-\theta^2/2\sigma^2}$$

那么， $-\log P(\theta) = \text{constant} + \theta^2/2\sigma^2$ 。至此，一件神奇的事情发生了：

在MAP中使用一个高斯分布的先验等价于在MLE中采用L2的regularization！

The Gaussian Distribution



更一般地，假如参数分布不是以0为中心的

$$P(\theta) = N(\theta|\mu, \sigma^2) \log P(\theta) = \text{const} + (\theta - \mu)^2$$

其本质是优化一个 $(\theta - \mu)^2$ MSE

最小化交叉熵 = (广义) 伯努利分布极大似然估计

Summary

- 频率学派 – Frequentist – Maximum Likelihood Estimation (MLE, 最大似然估计)
 - 世界是确定的，概率客观存在，需要做的就是找到置信区间
- 贝叶斯学派 – Bayesian – Maximum A Posteriori (MAP, 最大后验估计)
 - 世界是变化的，需要根据观察调整预判

$MAE \approx MLE + \text{Prior}$

- 交叉熵 $\approx MLE$
- 正态先验 $MAE \approx L2$ 正则化

ROC 曲线与 PR 曲线

ROC 曲线和 PR 曲线是评估机器学习算法性能的两条重要曲线，两者概念比较容易混淆，但是两者的使用场景是不同的。

定义

ROC 曲线和 PR 曲线都是用在二分类中，且涉及到下图的几个概念(摘自 [The Relationship Between Precision–Recall and ROC Curves](#))

	actual positive	actual negative
predicted positive	TP	FP
predicted negative	FN	TN

(a) Confusion Matrix

$$\begin{aligned} \text{Recall} &= \frac{TP}{TP+FN} \\ \text{Precision} &= \frac{TP}{TP+FP} \\ \text{True Positive Rate} &= \frac{TP}{TP+FN} \\ \text{False Positive Rate} &= \frac{FP}{FP+TN} \end{aligned}$$

(b) Definitions of metrics

Recall: 查全率，正样本中被预测出来是正的比例(越大越好)

Precision: 查准率，预测的正样本中被正确预测的比例(越大越好)

True Positive Rate: 跟 Recall 定义一样 (越大越好)

FPR : 负样本中被预测为正的比例(越小越好)

对于一个二分类问题，往往要设定一个 threshold，当预测值大于这个 threshold 时预测为正样本，小于这个 threshold 时预测为负样本。如果以 Recall 为横轴，Precision 为纵轴，那么设定一个 threshold 时，便可在坐标轴上画出一个点，设定多个 threshold 则可以画出一条曲线，这条曲线便是 PR 曲线。

PR 曲线是以 Recall 为横轴，Precision 为纵轴；而 ROC 曲线则是以 FPR 为横轴，TPR 为纵轴。

那么两者的关系是怎样的？

对比

The Relationship Between Precision–Recall and ROC Curves中证明了以下两条定理

定理1：对于一个给定的数据集，ROC空间和PR空间存在一一对应的关系，因为二者包含完全一致的混淆矩阵。我们可以将ROC曲线转化为PR曲线，反之亦然。

定理2：对于一个给定数目的正负样本数据集，曲线 A 在 ROC 空间优于曲线 B，当且仅当在 PR 空间中曲线 A 也优于曲线 B。

定理 2 中“曲线A优于曲线B”是指曲线 B 的所有部分与曲线 A 重合或在曲线 A 之下。而在ROC空间，**ROC曲线越凸向左上方向效果越好**。与ROC曲线左上凸不同的是，**PR曲线是右上凸效果越好**。

从定理 2 来看，ROC 空间和 PR 空间两个指标似乎具有冗余性，那么为什么还需要这两个指标呢？答案是在两者在样本不均衡的情况下表现有较大差异。

下图是ROC曲线和Precision–Recall曲线的对比，摘自An introduction to ROC analysis

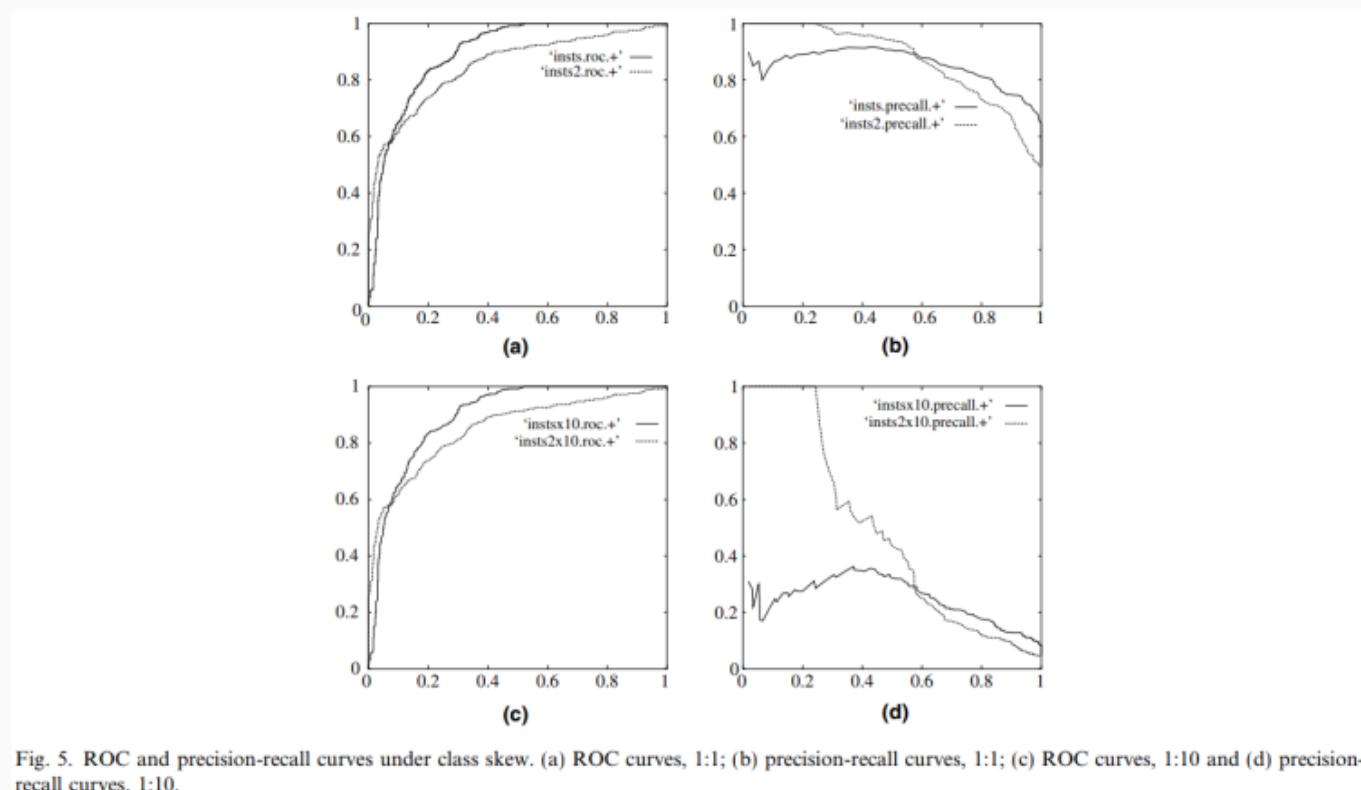


Fig. 5. ROC and precision-recall curves under class skew. (a) ROC curves, 1:1; (b) precision-recall curves, 1:1; (c) ROC curves, 1:10 and (d) precision-recall curves, 1:10.

图 (a) 和 (b) 是在样本正负比例为 1:1 下的 ROC 曲线和PR 曲线，图(c) 和 (d) 是在样本正负比例为 1:100 下的 ROC 曲线和PR 曲线。

从结果来看：当测试集中的正负样本的分布变化的时候，ROC曲线能够保持不变。

文章[ROC和AUC介绍以及如何计算AUC](#)以及[An introduction to ROC analysis](#)中都认为这是个优点，原因是在实际的数据集中经常会出现类不平衡 (class imbalance) 现象，即负样本比正样本多很多 (或者相反)，而ROC 这种对不平衡样本的鲁棒性使得其曲线下的面积 AUC 不会发生突变。

那么，AUC 意味这什么？首先AUC 值是一个概率值，表示随机挑选一个正样本以及一个负样本，当前的分类算法根据计算得到的Score值将这个正样本排在负样本前面的概率。

因此，AUC值实际上反映了模型的 rank 能力，AUC值越大，当前的分类算法越有可能将正样本排在负样本前面。这个指标尤其适用在某些场景下(如 CTR 预估)，每次要返回的是最有可能点击的若干个广告(根据CTR排序，选择排在前面的若干个)，实际上便是在考验模型的排序能力。除此之外，CTR 中存在着样本不均衡的问题，正负样本比例通常会大于 1:100，如果采用 PR 曲线，则会导致 AUC 发生剧变，无法较好反映模型效果。

然而，ROC 曲线不会随着类别分布的改变而改变的优点在一定程度上也是其缺点。因为 ROC 曲线这种不变性其实影响着的是 AUC 值，或者说是评估分类器的整体性能。但是在某些场景下，我们会更关注正样本，这时候就要用到 PR 曲线了。

比如说信用卡欺诈检测，我们会更关注 precision 和 recall，比如说如果要求预测出为欺诈的人尽可能准确，那么就是要提高 precision；而如果要尽可能多地预测出潜在的欺诈人群，那么就是要提高 recall。一般来说，提高二分类的 threshold 就能提高 precision，降低 threshold 就能提高 recall，这时便可观察 PR 曲线，得到最优的 threshold。

除此之外，Quora 上的问题[What is the difference between a ROC curve and a precision-recall curve? When should I use each?](#)中也举了一下的例子说明了在欺诈检测的问题中，PR 曲线更能反映结果的变化。

Let's take an example of fraud detection problem where there are 100 frauds out of 2 million samples.

Algorithm 1: 90 relevant out of 100 identified

Evidently, algorithm 1 is more preferable because it identified less number of false positive.

In the context of ROC curve, Algorithm 1: $TPR=90/100=0.9$, $FPR=10/1,999,900=0.00000500025$

Algorithm 2: $TPR=90/100=0.9$, $FPR=910/1,999,900=0.00045502275$

The FPR difference is 0.0004500225
For PR Curve
Algorithm 1: $precision=0.9$, $recall=0.9$
Algorithm 2: $Precision=90/1000=0.09$, $recall=0.9$
Precision difference= 0.81

The difference is more apparent in PR curve

总结

综上，有以下几条结论（参考[机器学习之类别不平衡问题 \(2\) —— ROC和PR曲线](#)）

1. ROC曲线由于兼顾正例与负例，所以适用于评估分类器的整体性能(通常是会计算AUC，表示模型的rank性能)，相比而言PR曲线完全聚焦于正例。
2. 如果有多份数据且存在不同的类别分布。比如信用卡欺诈问题中每个月正例和负例的比例可能都不相同，这时候如果只想单纯地比较分类器的性能且剔除类别分布改变的影响，则ROC

曲线比较适合，因为类别分布改变可能使得PR曲线发生变化时好时坏，这种时候难以进行模型比较；反之，如果想测试不同类别分布下对分类器的性能的影响，则PR曲线比较适合。

3. 如果想要评估在**相同的类别分布下正例的预测情况**，则宜选PR曲线。类别不平衡问题中，ROC曲线通常会给出一个乐观的效果估计，所以大部分时候还是PR曲线更好。（参考上面Quora的例子）
4. 最后可以根据具体的应用，在曲线上找到最优的点，得到相对应的precision, recall, f1 score等指标，去调整模型的阈值，从而得到一个符合具体应用的模型。

梯度裁剪及其作用

梯度裁剪(gradient clipping)的方法及其作用，在训练 RNN 过程中这个机制对结果影响非常大。

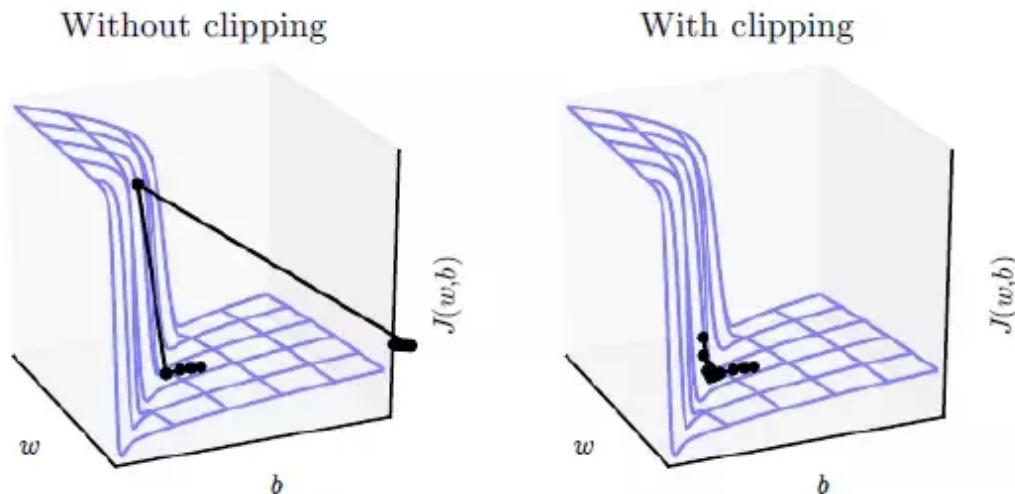
梯度裁剪一般用于解决 梯度爆炸(gradient explosion) 问题，而梯度爆炸问题在训练 RNN 过程中出现得尤为频繁，所以训练 RNN 基本都需要带上这个参数。常见的 gradient clipping 有两种做法

1. 根据参数的 gradient 的值直接进行裁剪
2. 根据若干参数的 gradient 组成的 vector 的 L2 norm 进行裁剪

第一种做法很容易理解，就是先设定一个 gradient 的范围如 $(-1, 1)$ ，小于 -1 的 gradient 设为 -1 ，大于这个 1 的 gradient 设为 1 。

第二种方法则更为常见，先设定一个 `clip_norm`，然后在某一次反向传播后，通过各个参数的 gradient 构成一个 vector，计算这个 vector 的 L2 norm (平方和后开根号) 记为 `LNorm`，然后比较 `LNorm` 和 `clip_norm` 的值，若 `LNorm <= clip_norm` 不做处理，否则计算缩放因子 `scale_factor = clip_norm / LNorm`，然后令原来的梯度乘上这个缩放因子。这样做是为了让 gradient vector 的 L2 norm 小于预设的 `clip_norm`。

关于 gradient clipping 的作用可更直观地参考下面的图，没有 gradient clipping 时，若梯度过大优化算法会越过最优点。



而在一些的框架中，设置 gradient clipping 往往也是在 Optimizer 中设置，如 tensorflow 中设置如下

Keras 中设置则更为简单

除此之外，调试 RNN 是个比较 tricky 的活，可参考知乎上这个问题：[trick 你在训练RNN的时候有哪些特殊的trick？](#)

另外，与 gradient explosion 相反的问题 gradient vanishing 的做法跟 gradient explosion 不同，不能简单地采用 scaling 的方法，具体可参考这个问题[梯度消失问题为什么不能通过 gradient scaling 来解决？](#)，实际的处理方法一般是采用 LSTM 或 GRU 这类有记忆的 RNN 单元。

参考：

1. [Gradient Clipping](#)
2. [caffe里的clip gradient是什么意思？](#)
3. [What is gradient clipping and why is it necessary?](#)

归纳偏差

归纳偏置在机器学习中是一种很微妙的概念：在机器学习中，很多学习算法经常会对学习的问题做一些假设，这些假设就称为归纳偏置(Inductive Bias)。归纳偏置这个译名可能不能很好地帮助理解，不妨拆解开来：**归纳(Induction)**是自然科学中常用的两大方法之一(归纳与演绎, induction and deduction)，指的是从一些例子中寻找共性、泛化，形成一个比较通用的规则的过程；**偏置(Bias)**是指我们对模型的偏好。

因此，归纳偏置可以理解为，从现实生活中观察到的现象中归纳出一定的规则(heuristics)，然后对模型做一定的约束，从而可以起到“模型选择”的作用，即从假设空间中选择出更符合现实规则的模型。其实，贝叶斯学习中的“先验(Prior)”这个叫法，可能比“归纳偏置”更直观一些。

归纳偏置在机器学习中几乎无处不可见。老生常谈的“奥卡姆剃刀”原理，即希望学到的模型复杂度更低，就是一种归纳偏置。另外，还可以看见一些更强的一些假设：KNN中假设特征空间中相邻的样本倾向于属于同一类；SVM中假设好的分类器应该最大化类别边界距离；等等。

在深度学习方面也是一样。以神经网络为例，各式各样的网络结构/组件/机制往往就来源于归纳偏置。在卷积神经网络中，我们假设特征具有局部性(Locality)的特性，即当我们把相邻的一些特征放在一起，会更容易得到“解”；在循环神经网络中，我们假设每一时刻的计算依赖于历史计算结果；还有注意力机制，也是基于从人的直觉、生活经验归纳得到的规则。

在自然语言处理领域赫赫有名的word2vec，以及一些基于共现窗口的词嵌入方法，都是基于分布式假设：A word's meaning is given by the words that frequently appear close-by. 这当然也可以看作是一种归纳偏置；一些自然语言理解的模型中加入解析树，也可以类似地理解。都是为了选择“更好”的模型。

- 归纳偏差：
 - 归纳偏差是一个关于机器学习算法的目标函数的假设。
 - 其实这个指的就是目标函数评分的标准. 我们利用机器学习算法要做的是，利用一个学习器，通过学习样本使得学习器对于任意输入(可以不包含在训练数据中)可以产生正确的预测. 那么，这个假设就决定了在面对未知数据下如何去作出判断. 例如，在线性回归中，作出的假设(归纳偏置)是，输出与输入是线性的. 下面是stackoverflow上的一个解释.Every machine learning algorithm with any ability to generalize beyond the training data that it sees has some type of inductive bias. This is the assumptions made by the model to learn the target function and to generalize beyond training data.For example in linear regression the model assumes that the output or dependent variable is related to independent variable linearly (in the weights).This is inductive bias in the model

- 模型的指导规则, 可以同时应用于训练和预测的时候的东西, 是一种超参数(因为是模型的一部分).上面举的例子是线性回归, 还有一种是决策树, 决策树的假设就是,
 - 优先选择较短的树而不是较长的。
 - 选择那些信息增益高的属性里根节点较近的树。这里利用了两个假设. 相比之下, 神经网络的假设是相当弱的, 举个例子: 分类神经网络模型 : 将输入通过非线性函数进行映射的结果, 正确的类别具有较高的softmax值.
- 归纳偏差的种类: 最大条件独立性 (conditional independence) : 如果假说能转成贝叶斯模型架构, 则试着使用最大化条件独立性。这是用于朴素贝叶斯分类器 (Naive Bayes classifier) 的偏置。
 - 最小交叉验证误差: 当试图在假说中做选择时, 挑选那个具有最低交叉验证误差的假说, 虽然交叉验证看起来可能无关偏置, 但天下没有免费的午餐理论显示交叉验证已是偏置的。
 - 最大边界: 当要在两个类别间画一道分界线时, 试图去最大化边界的宽度。这是用于支持向量机的偏置。这个假设是不同的类别是由宽界线来区分。
 - 最小描述长度 (Minimum description length) : 当构成一个假设时, 试图去最小化其假设的描述长度。假设越简单, 越可能为真的。见奥卡姆剃刀。
 - 最少特征数 (Minimum features) : 除非有充分的证据显示一个特征是有效用的, 否则它应当被删除。这是特征选择 (feature selection) 算法背后所使用的假设。
 - 最近邻居: 假设在特征空间 (feature space) 中一小区域内大部分的样本是同属一类。给一个未知类别的样本, 猜测它与它最紧接的大部分邻居是同属一类。这是用于最近邻居法的偏置。这个假设是相近的样本应倾向同属于一类别。

梯度下降、牛顿法凸优化、L1、L2正则化、softmax、Batchnorm、dropout、Targeted Dropout

参考: https://blog.csdn.net/weixin_37947156/article/details/84900390?spm=1001.2014.3001.5501

优化器

参考: <https://www.cnblogs.com/guoyaohua/p/8780548.html>

梯度弥散 (BN和shattered gradient)

参考: <https://arxiv.org/pdf/1702.08591.pdf>

有了bn就基本可以认为解决了梯度弥散问题，二十几层之后网络性能下降并不是因为梯度弥散，而是因为shattered gradient问题。

Loss Function总结

参考: <https://www.cnblogs.com/guoyaohua/p/9217206.html>

深度学习中的Normalization模型

参考：https://blog.csdn.net/weixin_37947156/article/details/99065938?spm=1001.2014.3001.5501

参考：<https://mp.weixin.qq.com/s/ItglnX92Pv9SuTel8gpdUw>

各种Normalization的适用场景

- 对于RNN的神经网络结构来说，目前只有LayerNorm是相对有效的；
- 如果是GAN等图片生成或图片内容改写类型的任务，可以优先尝试InstanceNorm；
- 如果使用场景约束BatchSize必须设置很小，无疑此时考虑使用GroupNorm；
- 而其它任务情形应该优先考虑使用BatchNorm。

Batch Normalization

<https://mp.weixin.qq.com/s/b8H3LhEwt58L2XEpQP4vYw>

BN和Dropout一起使用并不会 $1+1>2$

参考: [https://blog.csdn.net/weixin_37947156/article/details/98763993?
spm=1001.2014.3001.5501](https://blog.csdn.net/weixin_37947156/article/details/98763993?spm=1001.2014.3001.5501)

CNN的10个思考

参考：https://blog.csdn.net/weixin_37947156/article/details/95937001?spm=1001.2014.3001.5501

- 一、卷积只能在同一组进行吗？ — Group convolution
- 二、卷积核一定越大越好？ — 3×3 卷积核
- 四、怎样才能减少卷积层参数量？ — Bottleneck
- 五、越深的网络就越难训练吗？ — Resnet 残差网络
- 六、卷积操作时所有通道都只能用同一个过滤器吗？ — DepthWise 操作
- 七、分组卷积能否对通道进行随机分组？ — ShuffleNet
- 八、通道间的特征都是平等的吗？ — SEnet
- 九、能否让固定大小的卷积核看到更大范围的区域？ — Dilated convolution(空洞卷积)
- 十、卷积核形状一定是矩形吗？ — Deformable convolution 可变形卷积核
- 十一、为什么ResNet和DenseNet可以这么深？ — Skip Connection

现在越来越多的CNN模型从巨型网络到轻量化网络一步步演变，模型准确率也越来越高。现在工业界追求的重点已经不是准确率的提升（因为都已经很高了），都聚焦于速度与准确率的trade off，都希望模型又快又准。因此从原来AlexNet、VGGnet，到体积小一点的Inception、Resnet系列，到目前能移植到移动端的mobilenet、ShuffleNet（体积能降低到0.5mb！），我们可以看到这样一些趋势：

卷积核方面：

- 大卷积核用多个小卷积核代替；
- 单一尺寸卷积核用多尺寸卷积核代替；
- 固定形状卷积核趋于使用可变形卷积核；
- 使用 1×1 卷积核（bottleneck结构）。

卷积层通道方面：

- 标准卷积用depthwise卷积代替；
- 使用分组卷积；
- 分组卷积前使用channel shuffle；
- 通道加权计算。

卷积层连接方面：

- 使用skip connection，让模型更深；
- densely connection，使每一层都融合上其它层的特征输出(DenseNet)

启发

类比到通道加权操作，卷积层跨层连接能否也进行加权处理？bottleneck + Group conv + channel shuffle + depthwise的结合会不会成为以后降低参数量的标准配置？

Distributional Representation和Distributed Representation

参考: <https://zhuanlan.zhihu.com/p/22386230>

Distributional Representation是从**分布式假设**（即如果两个词的上下文相似，那么这两个词也是相似的）的角度，利用共生矩阵来获取词的语义表示，可以看成是一类获取词表示的方法。这里的“分布”带有统计上分布的意思。我们可以构建一个大小为 $W \times C$ 的共现矩阵 F ，其中 W 是词典大小， C 是上下文数量。上下文的类型可以为相邻词、所在句子或所在的文档等。共现矩阵的每一行可以看作对应词的向量表示。基于共现矩阵，有很多方法来得到连续的词表示，比如潜在语义分析模型（Latent Semantic Analysis, LSA）、潜在狄利克雷分配模型（Latent Dirichlet Allocation, LDA）、随机索引（random indexing）等。

而Distributed Representation中的distributed 没有统计上的“分布”含义，而是“分散”、“分配”的意思。一段文本的语义分散在一个低维空间的不同维度上，相当于将不同的文本分散到空间中不用的区域。Distributed Representation是文本的一种表示形式，具体为稠密、低维、连续的向量。向量的每一维都表示文本的某种潜在的语法或语义特征。Distributed Representation翻译为**分散式表示**可能理解起来会更明确些。

MLP-Mixer(LeCun说MLP-Mixer是一个挂羊头卖狗肉的算法)&RepMLP(CNN和MLP一起训)

论文: <https://arxiv.org/pdf/2105.01883.pdf>

<https://arxiv.org/pdf/2105.01601v1.pdf>

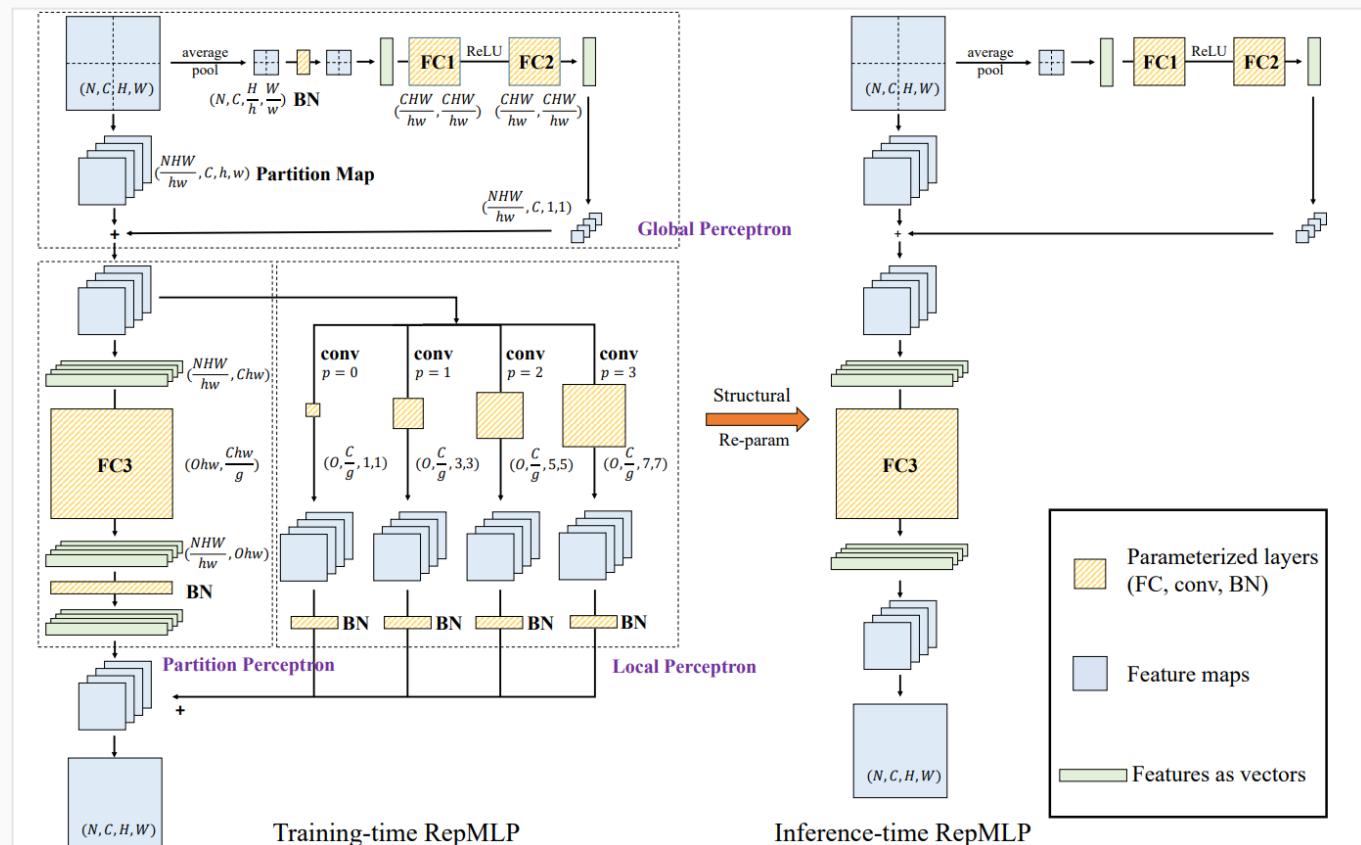
参考: <https://zhuanlan.zhihu.com/p/371738694>

<https://zhuanlan.zhihu.com/p/371738694>

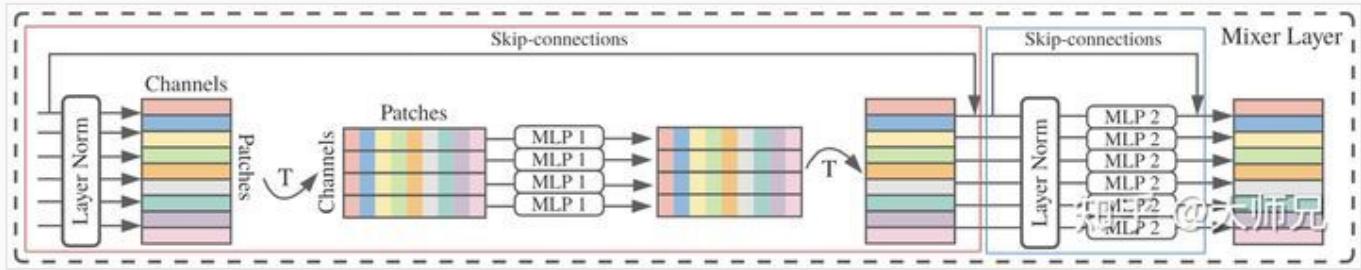
<https://mp.weixin.qq.com/s/OqXHEYZozhGnGzIyyf7OUw>

MLP-Mixer

MLP-Mixer主要是分两个步骤构建了intra和inter的联系，channel mixing可以看做是一个 1×1 的convolutions, token mixing可以看做是只有一个full receptive field卷积核的depth-wise convolutions。



MLP-Mixer的骨干网络是由 N 个Mixer Layer组成的，每个Mixer Layer的结构如图2所示。Mixer Layer的核心结构是图2中的两个MLP，其中MLP1是token-mixing全连接块（红框），MLP2是channel-mixing全连接块（蓝框）。



在Token-Mixing中，它先对输入的 \mathbf{X} 使用LN[3]进行归一化处理，然后再对其进行转置，将输入数据的格式由Channels为宽，Patchs为高的矩阵变成Patchs为宽，Channels为高的矩阵。接着在每个Channels上使用一个权值共享的MLP进行token之间的特征加工，然后再使用一个转置将矩阵还原，最后使用一个残差结构将处理前后的两个特征进行拼接。因为这一部分是实现的同一通道，不同token之间的混合，因此叫做Token-Mixing。

可以看出，图2中的MLP1其实就是MobileNet[3]中介绍的深度卷积（Depthwise Convolution），其中卷积核和步长的大小均是 P 。这里的LN则相当于对整个Feature Map做了一次白化。

在Channel-Mixing中，它也是先使用LN对特征进行了一次归一化，然后再直接使用一个共享的MLP对Feature Map的通道的特征进行混合计算。这里的MLP2其实就是 1×1 卷积。Channel Mixing中也使用了残差结构进行特征的拼接。

在Mixer Layer中每个MLP是由两个全连接和GELU激活函数组成。因此Mixer Layer其实本质上还是一个由连续两个深度卷积和连续两个点卷积组成的深度可分离卷积。

RepMLP

本文提出一种多层感知器风格的神经网络构建模块RepMLP用于图像识别，它有一系列的全连接层构成。相比卷积层，全连接层更为高效，可以进行更好的长期依赖与位置模式建模，但在局部结构提取方面较差，因此通常不太适合于图像识别。

本文提出一种结构重参数技术，它为全连接层添加了局部先验信息以使其可以进行强有力的图像识别。具体来说：在训练阶段，我们在RepMLP内部构建了卷积层，而在推理阶段，我们将这些卷积层合并到全连接层内。

在CIFAR数据集上，简简单单的MLP模型即可取得与CNN非常接近的性能。通过将RepMLP插入到现有CNN中，我们在ImageNet数据集上提升ResNets模型精度达1.8%，在人脸识别任务上提升2.9%，在Cityscapes提升2.3% mIoU精度且具有更低的FLOPs。

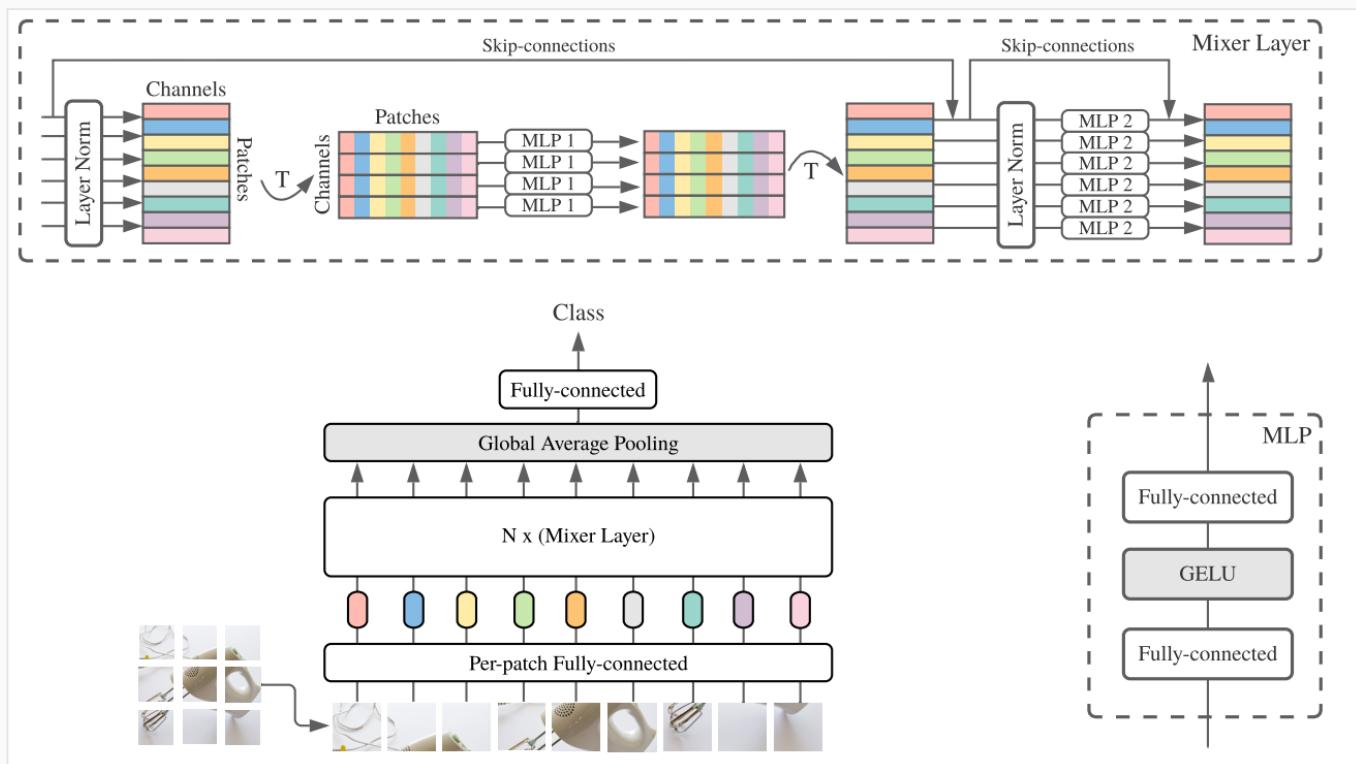
该发现说明：全连接层的全局表达、位置感知能力与卷积的局部结构提取能力的组合能够以更快的速度在平移不变任务(比如语义分割)、图像对齐+位置模式相关任务(比如人脸识别)上的性能。

本文主要贡献包含以下几点：

- 提出利用全连接层的全局建模、位置感知能力，并为其插上局部先验的翅膀，使其适用于图像识别任务；
- 提出一种简单的、平台不可知的、可微分算法用于将并行卷积、BN合并到全连接层中，使其具有局部先验且不造成任何推理耗时增加；
- 提出一种高效模块RepMLP，并在多个视觉任务上证实了其有效性。

RepMLP认为MLP具有构建long-range dependencies的能力和positional perception位置感知的能力等优点。FC对空间关系是敏感的，比如对于人脸识别，根据常识，我们知道眼睛在鼻子的上面，鼻子在嘴巴上面，FC具有构建这种位置关系的能力。

但是FC对于local信息聚合的能力显然不如CNN，因此为了弥补这一缺点，作者是CNN和MLP一起训练，最后在inference的时候再将CNN的参数整合为矩阵相乘的形式，相当于inference的时候只有MLP，但是是有local prior的MLP。



无监督模型

参考: https://mp.weixin.qq.com/s/YKUn_jDFBdYM9GyWUSHJpQ

到目前为止，学习了以下模型与概念

- 自回归模型(Autoregressive models)
 - PixelRNN, PixelCNN, PixelCNN++, PixelSNAIL
- 流模型(Flow models) (参考5.10大组会及笔记)
 - NICE, RealNVP, Autoregressive Flows, Inverse Autoregressive Flows, Glow, Flow++
- 隐变量模型(Latent Variable models)
 - 使用变分下界逼近似然函数
 - Wake Sleep
 - Variational Auto–Encoder, IWAE, IAF–VAE, PixelVAE (VLAE), VQ–VAE
- 隐含似然模型(Implicit models)
 - 生成对抗网络 (GAN)
 - 其他概念：矩匹配(moment matching), 能量模型(energy based models)
- 自监督学习(Self–supervised Learning) / 表示学习(Representation Learning)
 - 从原始数据中学习对下游任务有帮助的、有意义的特征
 - 数据+计算+核心认知原则 (常识任务、不变性先验[invariance priors]) 。

生成模型的理论分析

论文: <https://arxiv.org/pdf/1610.03483.pdf>

参考: <https://zhuanlan.zhihu.com/p/42403050>

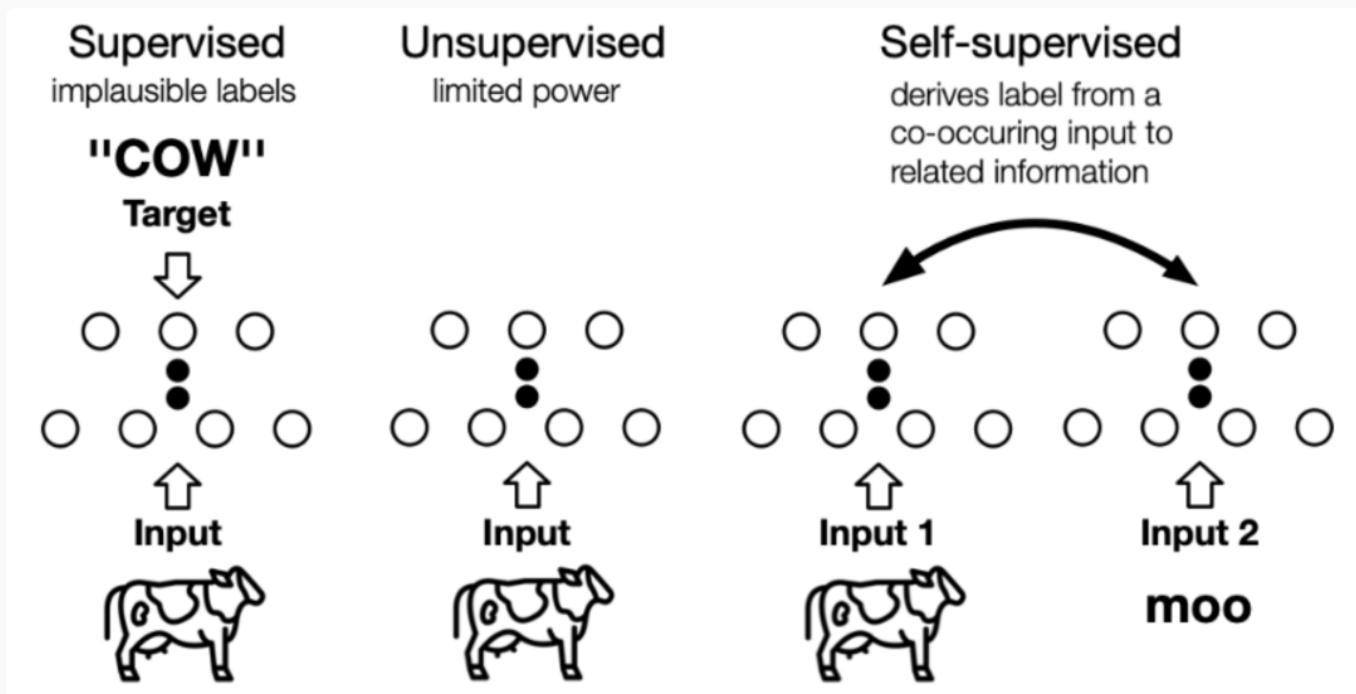
流模型

http://www.seeprettyface.com/pdf>Note_Flow.pdf

自监督学习

<https://mp.weixin.qq.com/s/sDYKh5EhC5ECsCfLNXkteQ>

监督学习在过去取得了巨大的成功，然而监督学习的研究进入了瓶颈期，因其依赖于昂贵的人工标签，却饱受泛化错误（generalization error）、伪相关（spurious correlations）和对抗攻击（adversarial attacks）的困扰。自监督学习以其良好的数据利用效率和泛化能力引起了人们的广泛关注。本文将全面研究最新的自监督学习模型的发展，并讨论其理论上的合理性，包括预训练语言模型（Pretrained Language Model, PTM）、生成对抗网络（GAN）、自动编码器及其拓展、最大化互信息（Deep Infomax, DIM）以及对比编码（Contrastive Coding）。自监督学习与无监督学习的区别主要在于，无监督学习专注于检测特定的数据模式，如聚类、社区发现或异常检测，而自监督学习的目标是恢复（recovering），仍处于监督学习的范式中。下图展示了两者之间的区别，自监督中的“related information”可以来自其他模态、输入的其他部分以及输入的不同形式。





自监督学习

- ▶ 一种监督学习和无监督学习的结合

- ▶ 监督式的学习方式

- ▶ 训练数据由无标注数据自动构建

- ▶ 典型任务

- ▶ Language Modeling (LM) ?

- ▶ Masked Language Modeling (MLM)

- ▶ Permuted Language Modeling (PLM)

- ▶ Denoising Autoencoder (DAE)

- ▶ Contrastive Learning (CTL)

- ▶ Deep InfoMax (DIM)

- ▶ Replaced Token Detection (RTD)

- ▶ Next Sentence Prediction (NSP)

- ▶ Sentence Order Prediction (SOP)

不依赖标注数据

伪监督任务

目标：学习数据中的可泛化知识

- 常与无监督学习互换使用(使用数据产生的标签);
- 通过上游人物创造出自身的标签：例如，遮挡图片的一部分作为输入，让nn预测被遮挡的部分；
- 为什么需要自监督学习：
 - 创造标签成本较高
 - 可以充分利用互联网上的无标签数据
 - 研究认知：动物与小宝宝是如何学习的
- 自监督学习目标：
 - 在无监督情况下，学习到与监督式学习一样好的特征；
 - 部署完成后，在下游任务中不需要太多标签；
 - 潜在的泛化性更好。
- 做法：假设输入的一部分未知，然后预测它。
- 认知原则：
 - 从混乱/部分数据中重构：去噪自编码器、图像修复、上色、Split–Brain 自编码器；
 - 视觉常见任务：图片部分内容的相关性、拼图、旋转；
 - 对比学习：word2vec、对比预测编码(CPC)、实例辨析、其他SOTA模型。

自监督学习举例

去噪自编码器

Split–Brain 自编码器

预测图像上部分内容的相对位置

检测图片旋转角度

词向量 word2vec

- 在自然语言处理(NLP)中，需要对语言进行数字化编码，one-hot型编码显然不合适；
- 词向量(word embeddings)就是研究字符到向量编码，有两种基本的编码学习方式：
 - 由输入的上下文推测输入内容(Continuous Bag Of Words)
 - 由输入推测其上下的内容(Skip Gram)

对比预测编码

Generative Self-supervised Learning 生成式自监督学习

Auto-regressive (AR) Model

在自回归模型中，联合分布可以被分解为条件的乘积，每个变量概率依赖于之前的变量：

$$\max_{\theta} p_{\theta}(x) = \sum_{t=1}^T \log p_{\theta}(x_t | x_{1:t-1})$$

在自然语言处理中，自回归语言模型的目标通常是最化正向自回归因子分解的似然。例如 GPT、GPT-2 使用 Transformer 解码器结构进行建模；在计算机视觉中，自回归模型用于逐像素建模图像，例如在 PixelRNN 和 PixelCNN 中，下方(右侧)像素是根据上方(左侧)像素生成的；而在图学习中，则可以通过深度自回归模型来生成图，例如 GraphRNN 的目标为最大化观察到的图生成序列的似然。自回归模型的优点是能够很好地建模上下文依赖关系。然而，其缺点是每个位置的 token 只能从一个方向访问它的上下文。

Auto-encoding (AE) Model

自动编码模型的目标是从(损坏的)输入中重构(部分)输入。标准的自动编码器，首先一个编码器对输入进行编码得到隐表示，再通过解码器重构输入，目标是使得输入和尽可能的相近。

Denoising AE, 去噪自动编码器 Denoising AE 认为表示应该对引入的噪声具有鲁棒性，MLM (masked language model) 就可以看作是去噪自动编码器模型，MLM 通过对序列中的 token 随机替换为 mask token，并根据上下文预测它们。同时为了解决下游任务中输入不存在 mask token 的问题，在实际使用中，替换 token 时可以选择以一定概率随机替换为其他单词或是保持不变。与自回归模型相比，在 MLM 模型中，所预测的 token 可以访问双向的上下文信息。然而，MLM 假设预测的 token 是相互独立的。

Variational AE, 变分自动编码器 变分自动编码模型假设数据是从潜在变量表示中生成的。给定数据，潜在变量上的后验分布通过来逼近。根据变分推断：

$$\log p(x) \geq -D_{KL}(q(z | x) \| p(z)) + \mathbb{E}_{\sim q(z|x)}[\log p(x | z)]$$

从自动编码器视角来看，第一项可以看作是正则化项保证后验分布逼近先验分布，第二项就是根据潜在变量重构输入的似然。VAE 在图像生成领域中有所应用，代表模型有向量量化 VAE 模型 VQ-VAE，而在图学习领域，则有变分图自动编码器 VGAE。

Hybrid model

结合 AR 模型和 AE 模型的各自优点，MADE 模型对 AE 做了一个简单的修改，使得 AE 模型的参数遵循 AR 模型的约束。具体来说，对于原始的 AE，相邻两层之间的神经元通过 MLPs 完全连接。然而，在 MADE 中，相邻层之间的一些连接被 mask 了，以确保输入的每个维度仅从前的维度来重构。

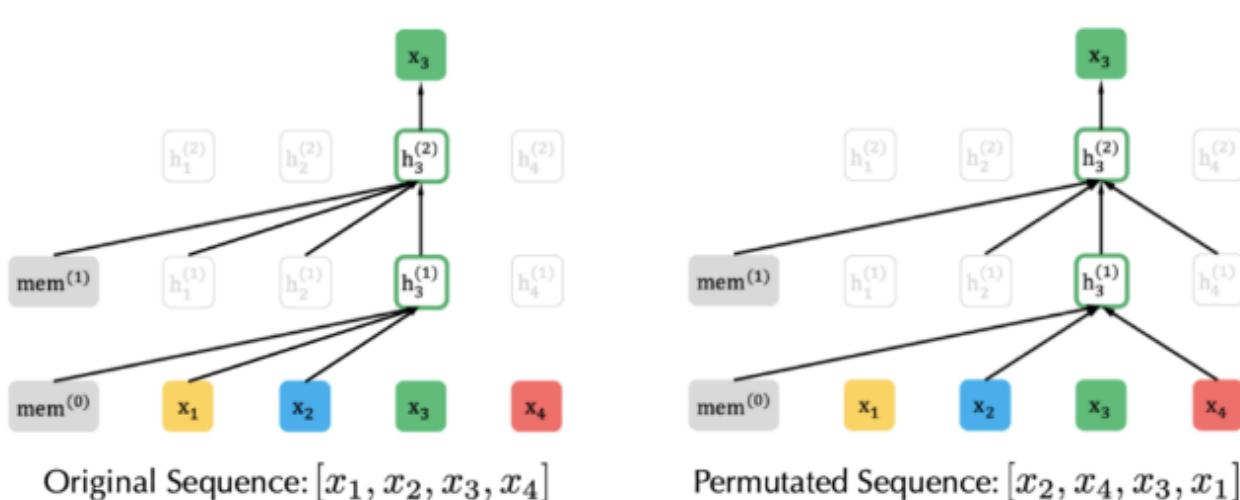


图2:PLM 实例示意图 (Image source: Z Yang et al.)

在 NLP 领域中，PLM (Permutation Language Model) 模型则是一个代表性的混合方法，XLNet 引入 PLM，是一种生成自回归预训练模型。XLNet 通过最大化因数分解顺序的所有排列的期望似然来实现学习双向上下文关系（见图）。具体的，假设是长度为的所有可能的序列排序，则 PLM 的优化目标为：

$$\max_{\theta} \mathbb{E}_{\mathbf{z} \sim \mathcal{Z}_T} \left[\sum_{t=1}^T \log p_{\theta}(x_{z_t} | \mathbf{x}_{\mathbf{z}_{<t}}) \right]$$

实际上，对于每个文本序列，都采样了不同的分解因子。因此，每个 token 都可以从双向看到它的上下文信息。在此基础上，XLNet 还对模型进行了位置重参数化，使模型知道需要预测的位置，并引入了特殊的双流自注意机制来实现目标感知预测。此外，与 BERT 不同的是，XLNet 受到 AR 模型最新改进的启发，提出了结合片段递归机制和相对位置编码机制的 Transformer-XL 集成到预训练中，比 Transformer 更好地建模了长距离依赖关系。

Contrastive Self-supervised Learning 对比式自监督学习

大多数表示学习任务都希望对样本之间的关系建模。因此长期以来，人们认为生成模型是表征学习的唯一选择。然而，随着 Deep Infomax、MoCo 和 SimCLR 等模型的提出，**对比学习 (contrastive learning)** 取得了突破性进展，如图 3 所示，在 ImageNet 上，自监督模型的 Top-1 准确率已经接近监督方法（ResNet50），揭示了判别式模型的潜力。

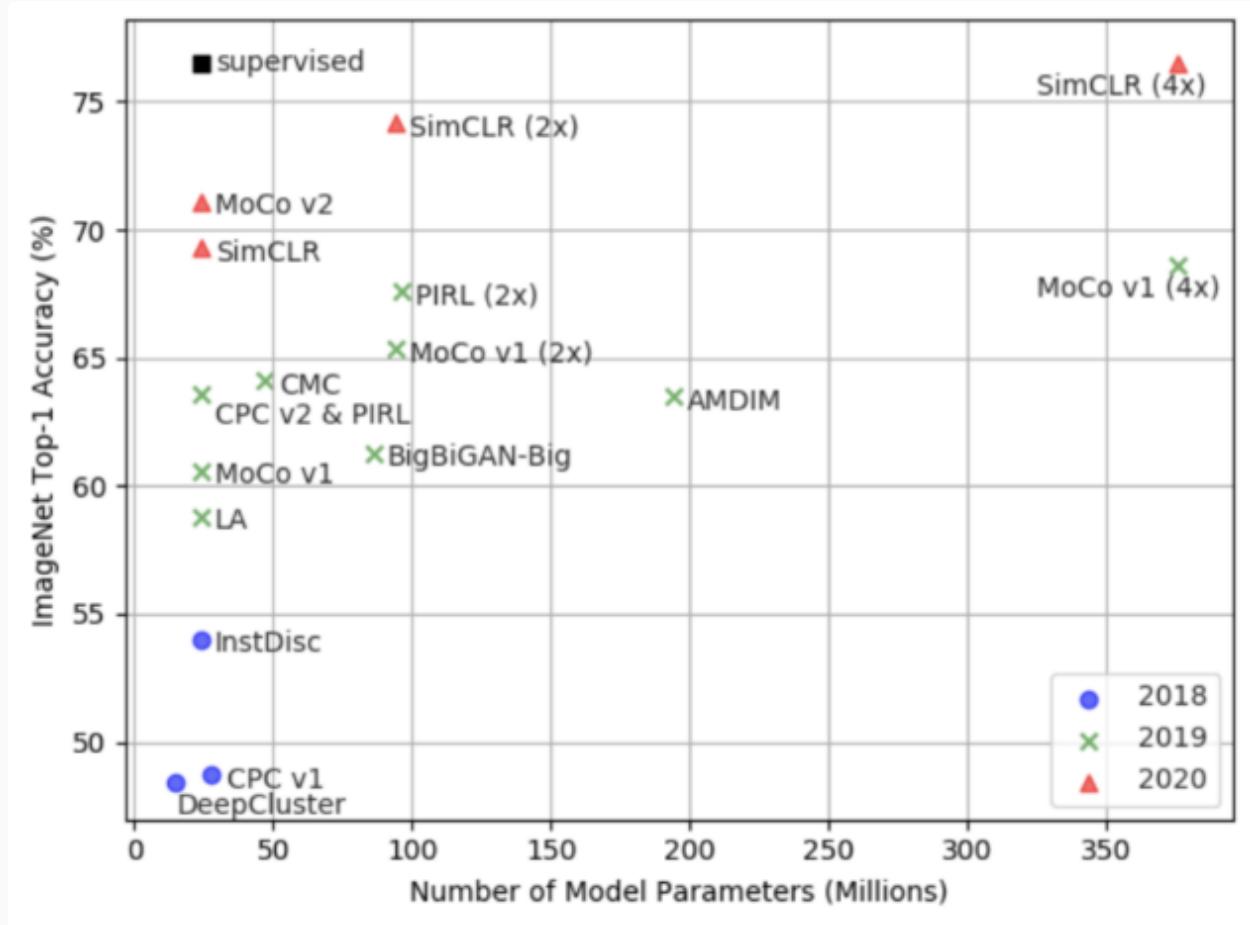


图3:自监督学习模型在 ImageNet 上 Top-1 准确率对比. (Image source: Jie Tang et al.)
对比学习的宗旨在于“learn to compare”，通过设定噪声对比估计 (Noise Contrastive Estimation, NCE) 来实现这个目标：

$$\mathcal{L} = \mathbb{E}_{x, x^+, x^-} \left[-\log \left(\frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + e^{f(x)^T f(x^-)}} \right) \right]$$

其中，与相似，而与不相似，是编码器函数（表示学习方程）。相似度量和编码函数可能会根据具体的任务有所不同，但是整体的框架仍然类似。当有更多的不相似样本对时，我们可以得到 InfoNCE：

$$\mathcal{L} = \mathbb{E}_{x, x^+, x^k} \left[-\log \left(\frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + \sum_{k=1}^K e^{f(x)^T f(x^k)}} \right) \right]$$

在这里，根据最近的对比学习模型框架，具体可细分为两大类：实例-上下文对比（context-instance contrast）和实例-实例对比（instance-instance contrast）。他们都在下游任务中有不俗的表现，尤其是分类任务。

Context–Instance Contrast

Context–Instance Contrast，又称为 Global–local Contrast，重点建模样本局部特征与全局上下文表示之间的归属关系，其初衷在于当我们学习局部特征的表示时，我们希望它与全局内容的表示相关联，比如条纹与老虎，句子与段落，节点与相邻节点。这一类方法又可以细分为两大类：预测相对位置（Predict Relative Position, PRP）和最大化互信息（Maximize Mutual Information, MI）。两者的区别在于：1) PRP 关注于学习局部组件之间的相对位置。全局上下文是预测这些关系的隐含要求（例如，了解大象的长相对于预测它头和尾巴的相对位置至关重要）；2) MI 侧重于学习局部组件和全局上下文之间的明确归属关系。局部之间的相对位置被忽略。

Predict Relative Position 许多数据蕴含丰富的时空关系，例如图 4 中，大象的头在其尾巴的右边，而在文本中，“Nice to meet you”很可能出现在“Nice to meet you, too”的前面。许多模型都把识别其各部分之间的相对位置作为辅助任务（pretext task），例如图 4 中，预测两个 patch 之间的相对位置，或是预测图像的旋转角，又或者是“解拼图”。



图4:Pretext Task示例. (Image source: Jie Tang et al.)

在 NLP 领域中, BERT 率先使用的 NSP (Next Sentence Prediction) 也属于类似的辅助任务, NSP 要求模型判断两个句子是否具有上下文关系, 然而最近的研究方法证明 NSP 可能对模型没什么帮助甚至降低了性能。为了替换 NSP, ALBERT 提出了 SOP (Sentence Order Prediction) 辅助任务, 其认为 NSP 中随机采样的句子可能来自不同的文章, 主题不同则难度自然很低, 而 SOP 中, 两个互换位置的句子被认为是负样本, 这使得模型将会集中在语义的连贯性学习。

Maximize Mutual Information MI 类任务利用统计学中的互信息, 通常来说, 模型的优化目标为:

$$\max_{g_1 \in \mathcal{G}_1, g_2 \in \mathcal{G}_1} I(g_1(x_1), g_2(x_2))$$

其中, 表示表示编码器, 则表示满足约束的一类编码器, 表示对于真实互信息的采样估计。在应用中, MI 计算困难, 一个常见的做法是通过 NCE 来最大化互信息的下界。Deep Infomax(DIM) 是第一个通过对比学习任务显式建模互信息的算法, 它最大化了局部 patch 与其全局上下文之间的 MI。在实际应用中, 以图像为例, 我们可以把一张狗的图像 x 编码为, 随后取出一个局部向量, 为了进行实例和上下文之间的对比: 首先需要通过一个总结函数 (summary function), 来生成上下文向量其次需要一个猫图像作为, 并得到其上下文向量这样就可以得到对比学习的目标函数:

$$\mathcal{L} = \mathbb{E}_{v,x} \left[-\log \left(\frac{e^{v^T \cdot s}}{e^{v^T \cdot s} + e^{v^T \cdot s^-}} \right) \right]$$

Deep Infomax 给出了自监督学习的新范式, 这方面工作较有影响力跟进, 最先有在 speech recognition 领域的 CPC 模型, CPC 将音频片段与其上下文音频之间的互信息最大化。为了提高数据利用效率, 它需要同时使用几个负的上下文向量, CPC 随后也被应用到图像分类中。AMDIM 模型提出通过随机生成一张图像的不同 views (截断、变色等) 用于生成局部特征向量和上下文向量, 其与 Deep Infomax 的对比如下图 5 所示, Deep Infomax 首先通过编码器得到图像的特征图, 随后通过 readout (summary function) 得到上下文向量, 而 AMDIM 则通过随机选择图像的另一个 view 来生成上下文向量。

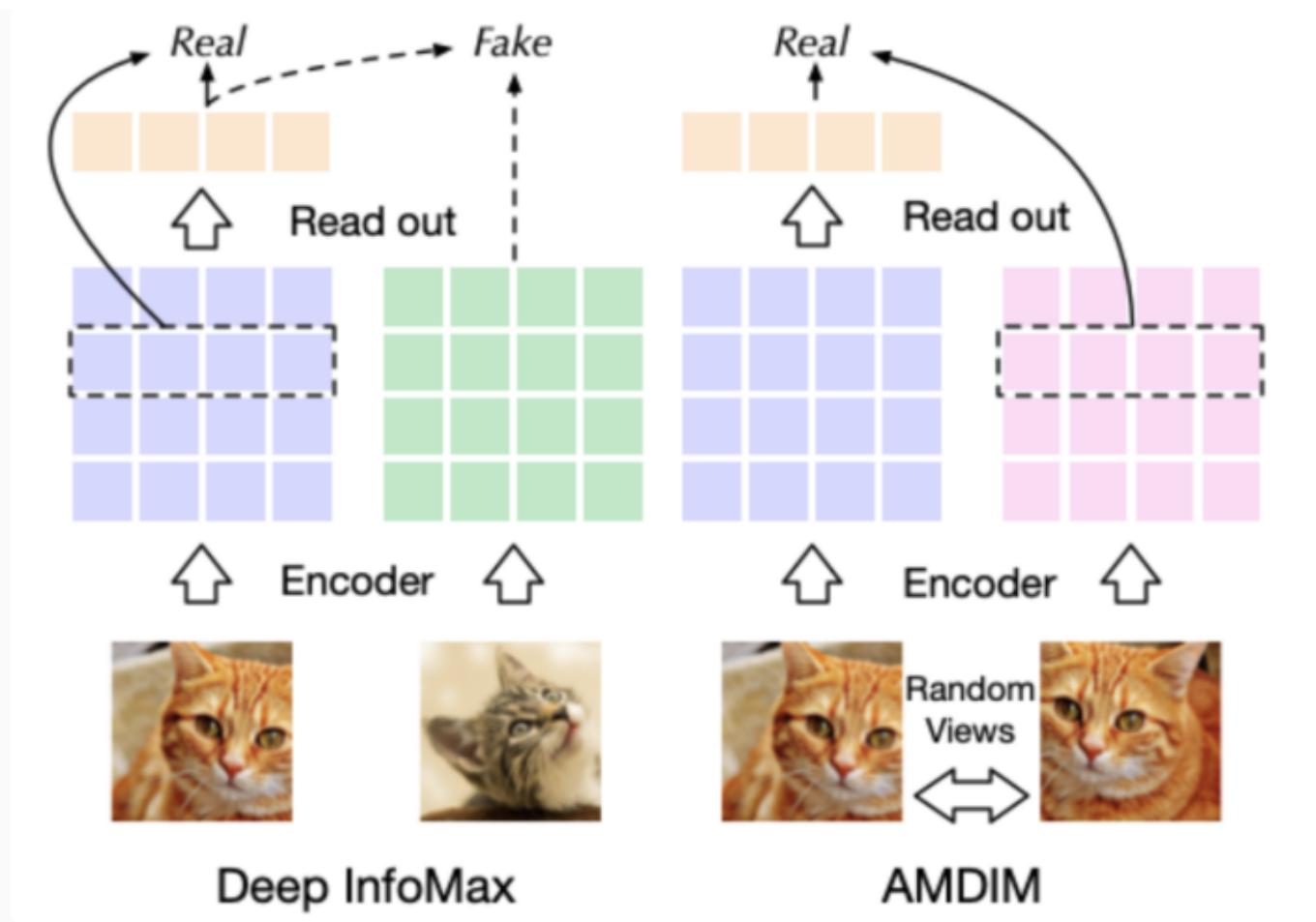


图5:Deep Infomax 与 AMDIM 示意图. (Image source: Jie Tang et al.)

在 NLP 领域, InfoWord 模型提出最大化句子的整体表示和句子的 n-grams 之间的互信息。而在图学习领域则有 DGI (Deep Graph Infomax) 模型, DGI 以节点表示作为局部特征, 以随机采样的二跳邻居节点的均值作为上下文向量, 注意图学习中的负样本难以构造, DGI 提出保留原有上下文节点的结构但打乱顺序来生成负样本。

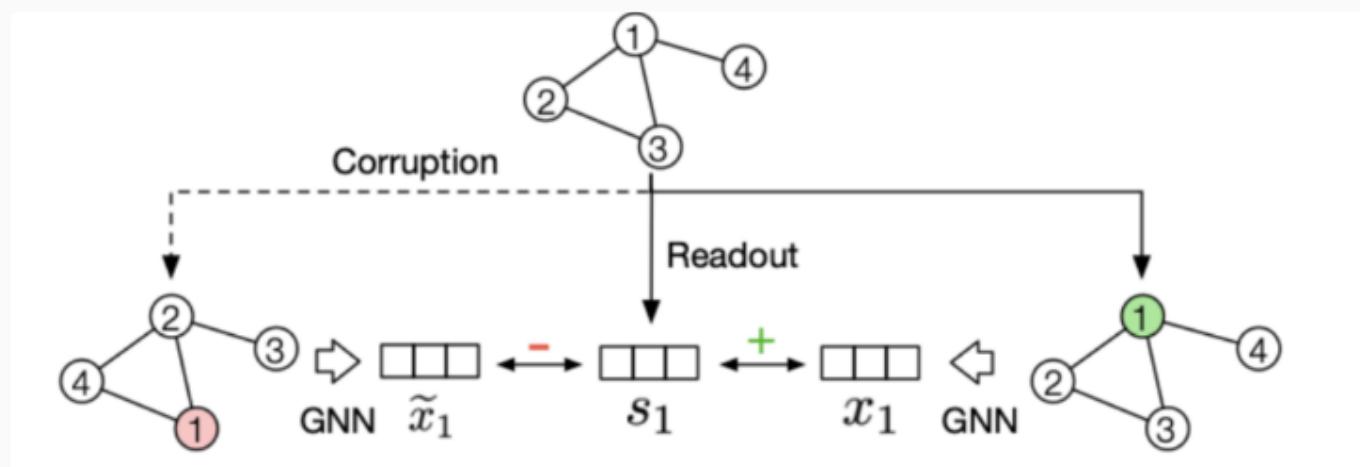


图6:Deep Graph Infomax 示意图. (Image source: Jie Tang et al.)

Instance–Instance Contrast

尽管基于最大化互信息的模型取得了成功，但是一些最近的研究对优化 MI 带来的实际增益表示怀疑。M Tschannen et al. 证明基于最大化互信息的模型的成功与 MI 本身联系并不大。相反，它们的成功应该更多地归因于编码器架构和以及与度量学习相关的负采样策略。度量学习的一个重点是在提高负采样效率的同时执行困难样本采样 (hard positive sampling)，而且它们对于基于最大化互信息的模型可能发挥了更关键的作用。而最新的相关研究 MoCo 和 SimCLR 也进一步证实了上述观点，它们的性能优于 context–instance 类方法，并通过 instance–instance 级的直接对比，相比监督方法取得了具有竞争力的结果。

Cluster-based Discrimination instance–instance contrast 类的方法最早的研究方向是基于聚类的方法，DeepCluster 取得了接近 AlexNet 的成绩。图像分类任务要求模型正确地对图像进行分类，并且希望在同一个类别中图像的表示应该是相似的。因此，目标是在嵌入空间内保持相似的图像表示相接近。在监督学习中，这一目标通过标签监督来实现；然而，在自监督学习中，由于没有标签，DeepCluster 通过聚类生成伪标签，再通过判别器预测伪标签。最近，局部聚合 (Local Aggregation, LA) 方法已经突破了基于聚类的方法的边界。LA 指出了 DeepCluster 算法的不足，并进行了相应的优化。首先，在 DeepCluster 中，样本被分配到互斥的类中，而 LA 对于每个样本邻居的识别是分开的。第二，DeepCluster 优化交叉熵的判别损失，而 LA 使用一个目标函数直接优化局部软聚类度量。这两个变化从根本上提高了 LA 表示在下游任务中的性能。

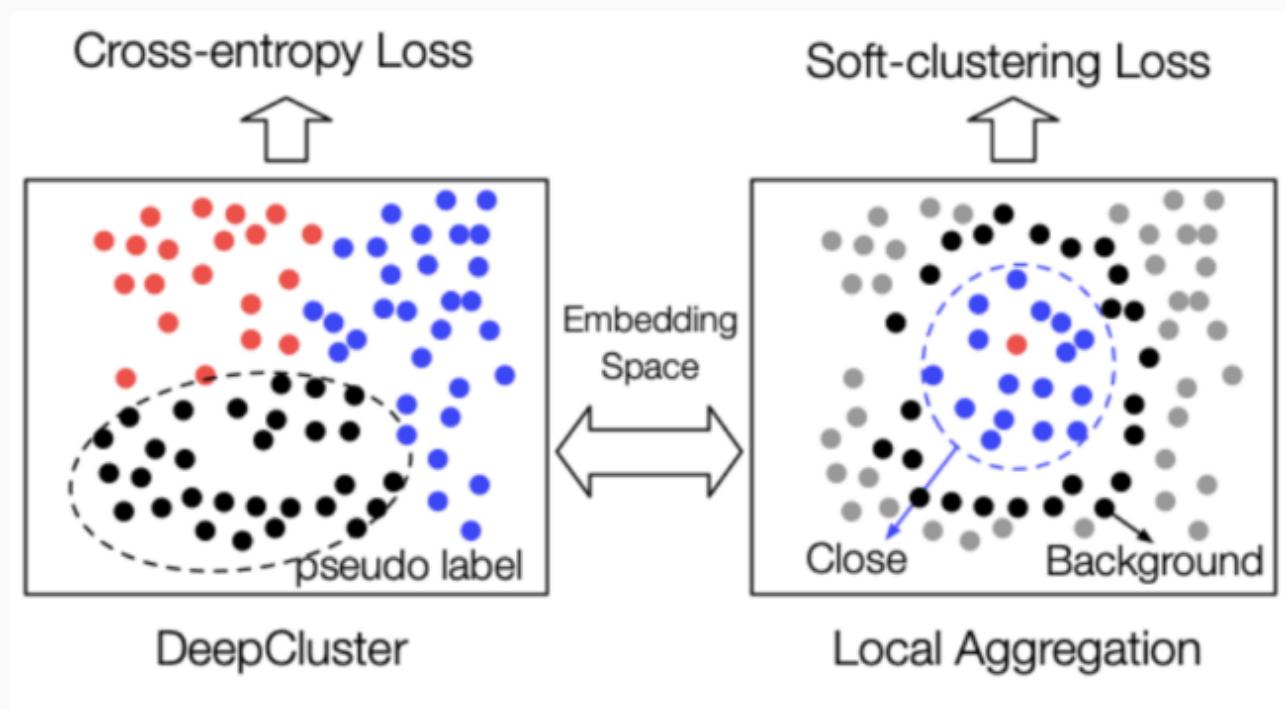


图7:DeepCluster 和 LA 方法示意图. (Image source: Jie Tang et al.)

基于聚类的判别也可以帮助预训练模型提升泛化能力，更好地将模型从辅助任务目标转移到实际任务中。传统的表示学习模型只有两个阶段：一个是预训练阶段，另一个是评估阶段。ClusterFit 在上述两个阶段之间引入了一个与 DeepCluster 相似的聚类预测微调阶段，提高了表示在下游分类评估中的性能。

Instance Discrimination 以实例判别 (instance discrimination) 作为辅助任务的模型原型是 InstDisc，在其基础之上，CMC 提出将一幅图像的多个不同 view 作为正样本，将另一幅图像作为负样本。在嵌入空间中，CMC 将使一幅图像的多个 view 尽可能靠近，并拉开与其他样本的距离。然而，它在某种程度上受到了 Deep InfoMax 思想的限制，仅仅对每个正样本采样一个负样本。在 MoCo 中，通过 momentum contrast 进一步发展了实例判别的思想，MoCo 极大的增加了负样本的数量。对于一个给定的图像，MoCo 希望通过一个 query encoder 得到一个可以区别于任何其他图像的 instinct 表示。因此，对于其他图像集中的，异步更新 key encoder 并得到以及，并优化下面的目标函数：

$$\mathcal{L} = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^K \exp(q \cdot k_i / \tau)}$$

其中，表示负样本的数量，这个式子就是 InfoNCE 的一种表达。同时，MoCo 还提出了其他两个在提升负采样效率方面至关重要的思想：首先，摒弃了传统的端到端训练，采用带有两个编码器 (query and key encoder) 的 momentum contrast 学习来避免训练初始阶段的损失的波动。其次，为了扩大负样本的容量，MoCo 采用队列 ($K=65536$) 的方式将最近编码过的批样本保存为负样本。这大大提高了负采样效率。

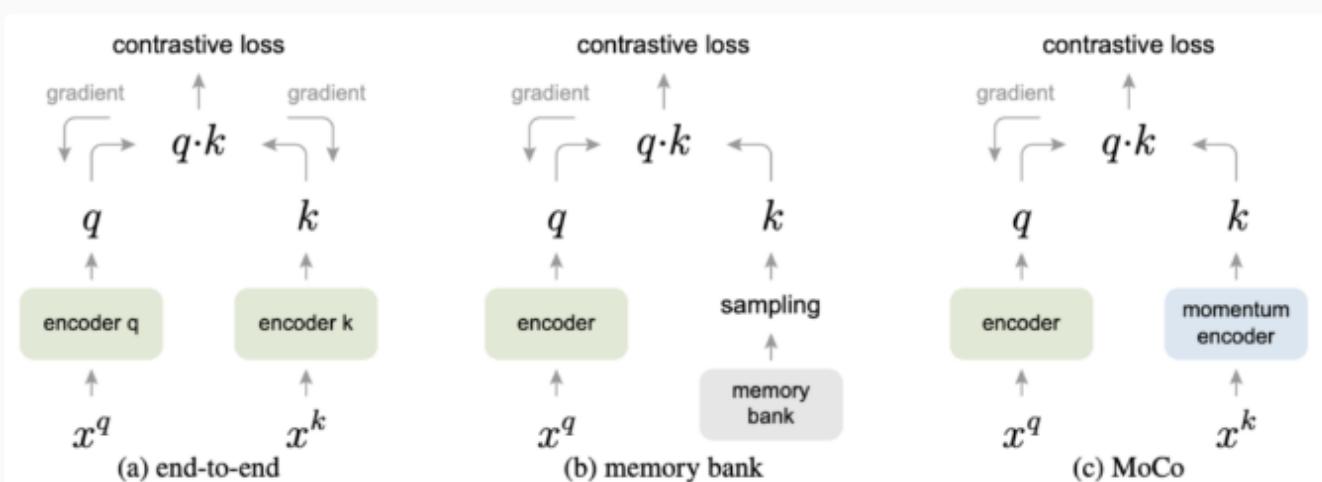


图8:MoCo 示意图. (Image source: Kaiming He et al.)

MoCo 还采用了一些辅助的技术来保证训练的收敛性，如批随机化以及 temperature 超参数的调整。然而，MoCo 采用了一种过于简单的正样本策略：正对表示来自同一样本，没有任何变换或

增强，使得正对太容易区分。PIRL 添加了上述提到的“解拼图”的困难正样本（hard positive example）增强。为了产生一个 pretext-invariant 的表示，PIRL 要求编码器把一个图像和它的拼图作为相似的对。在 SimCLR 中，作者通过引入 10 种形式的数据增强进一步说明了困难正样本（hard positive example）策略的重要性。这种数据增强类似于 CMC，它利用几个不同的 view 来增加正对。SimCLR 遵循端到端的训练框架，而不是 MoCo 的 momentum contrast，为处理大规模负样本问题，SimCLR 选择将 batch 大小 N 扩展为 8196。

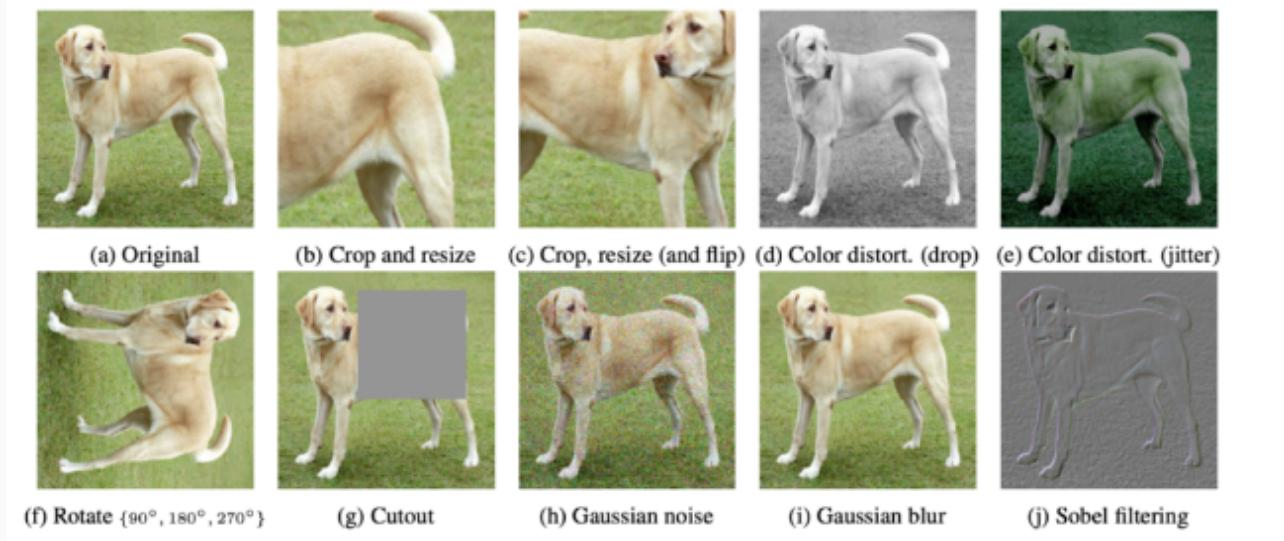


图9:SimCLR 数据增强示意图. (Image source: G Hinton et al.)

具体来看，minibatch 中的个样本将会被增强为个样本。对于一个正样本，其他的 $2(N-1)$ 个样本将会被视为负样本，我们可以得到成对对比损失 NT-Xent loss:

$$l_{i,j} = -\log \frac{\exp(\text{sim}(\hat{x}_i, \hat{x}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{I}_{[k \neq i]} \exp(\text{sim}(\hat{x}_i, \hat{x}_k)/\tau)}$$

注意，是非对称的，此处为正弦函数可以归一化表示，最终的和损失为：

$$\mathcal{L} = \frac{1}{2N} \sum_{k=1}^N [l_{2i-1,2i} + l_{2i,2i-1}]$$

SimCLR 还提供了一些其他有用的技术，包括 representation 和 contrast loss 之间的可学习的非线性转换，更多的训练步数，和更深的神经网络。Kaiming He et al. 进行消融分析研究表明，SimCLR 中的技术也可以进一步提高 MoCo 的性能。

在图学习中，图对比编码（Graph Contrastive Coding, GCC）率先将实例判别作为结构信息预训练的辅助任务。对于每个节点，通过随机游走独立采样两个不同的子图，并使用它们的归一化图 Laplacian 矩阵中的 top 特征向量作为节点的初始表示，然后使用 GNN 对它们进行编码，并像 MoCo 和 SimCLR 那样计算 InfoNCE loss，其中相同节点(在不同的子图中)的节点嵌入被视为相似正样本，其余的视为不相似的负样本。结果表明，GCC 比以前的工作，如 struc2vec、GraphWave 和 ProNE，学习了更好的可迁移的结构知识。

Generative–Contrastive (Adversarial) Self-supervised Learning 对比–生成式（对抗式）自监督学习

Why Generative–Contrastive (Adversarial)?

生成式模型往往会优化下面的目标似然方程：

$$\mathcal{L}_{MLE} = - \sum_x \log p(x | c)$$

其中， x 是我们希望建模的样本，则 c 是条件约束例如上下文信息，这个方程一般都会使用最大似然估计 MLE 来优化，然而 MLE 存在两个缺陷：

- **Sensitive and Conservative Distribution**（敏感的保守分布）：当时，会变得非常大，使得模型对于罕见样本会十分敏感，这会导致生成模型拟合的数据分布非常保守，性能受到限制。
- **Low-level Abstraction Objective**（低级抽象目标）：在使用 MLE 的过程中，表示分布建模的对象是样本级别，例如图像中的像素，文本中的单词以及图中的节点。然而，大部分分类任务需要更高层次的抽象表示，例如目标检测、长段落理解以及社区分类。

以上两个缺陷严重的限制了生成式自监督模型的发展，而使用判别式或对比式目标函数可以很好的解决上述问题。以自动编码器和 GAN 为例，自动编码器由于使用了一个 pointwise 的重构损失，因此可能建模的是样本中的 pixel-level 模式而不是样本分布，而 GAN 利用了对比式目标函数直接对比生成的样本和真实样本，建模的是 semantic-level 从而避免了这个问题。与对比学习不同，对抗学习仍然保留由编码器和解码器组成的生成器结构，而对比学习则放弃了解码器（如下图 10 所示）。这点区别非常重要，因为一方面，生成器将生成式模型所特有的强大的表达能力赋予了对抗学习；另一方面，这也使得对抗式方法的学习目标相比与对比式方法更具挑战性，也导致了收敛不稳定。在对抗式学习的设置中，解码器的存在要求表示具有“重构性”，换句话说，表示应当包含所有必要的信息来构建输入。而在对比式学习的设置中，我们只需要学习“可区分的”信息就可以区分不同的样本。

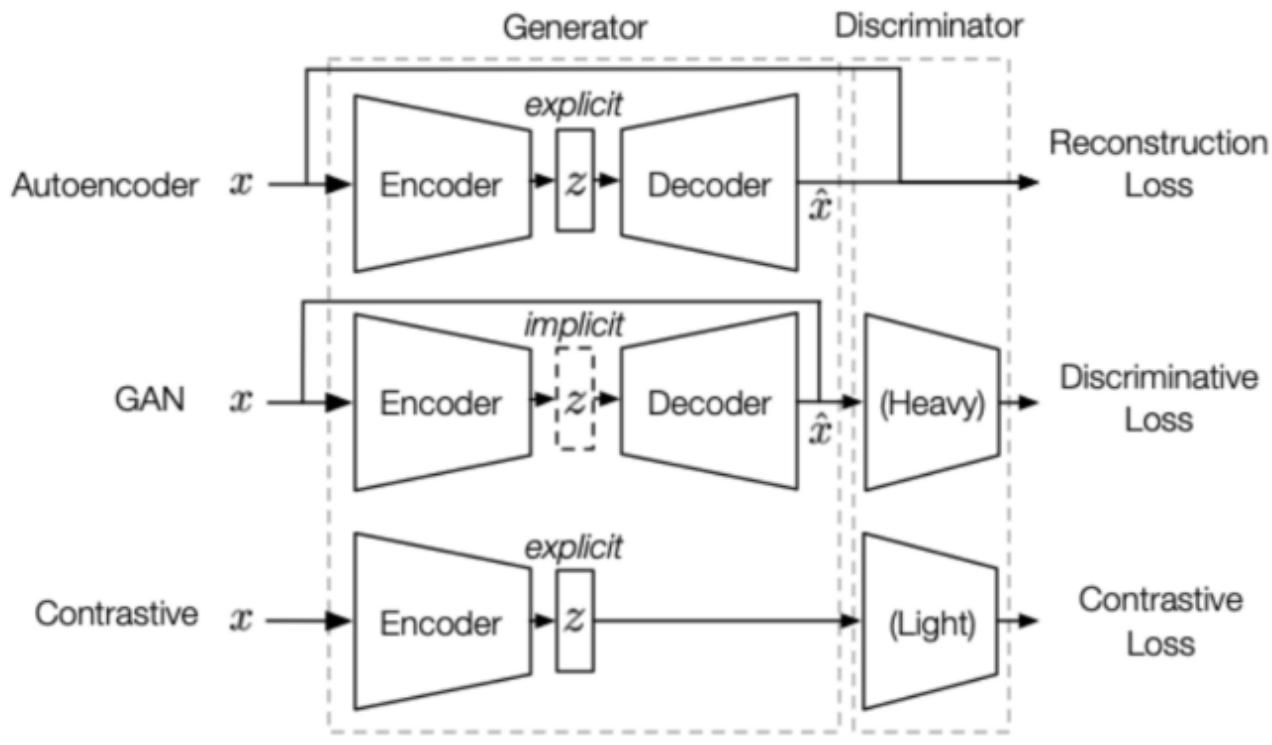


图10:AE、GAN 和 Contrastive 方法的区别，总体上它们都包含两个大的组件——生成器和判别器。生成器可以进一步分解为编码器和解码器，区别在于：1) 隐编码表示在 AE 和 Contrastive 方法中是显式的；2) AE 没有判别器是纯生成式模型，Contrastive 方法的判别器参数少于 GAN；3) AE 学习目标是生成器，而 GAN 和 Contrastive 方法的学习目标是判别器。(Image source: Jie Tang et al.)

综上所述，对抗式方法吸收了生成式方法和对比式方法各自的优点，但同时也有一些缺点。在我们需要拟合隐分布的情况下，使用对抗式方法会是一个更好的选择。下面将具体讨论它在表示学习中的各种应用。

Generate with Complete Input

本小节将介绍 GAN 及其变种如何应用到表示学习中，这些方法关注于捕捉样本的完整信息。

GAN 在图像生成中占据主导地位，然而数据的生成与表示还是存在 gap 的，这是因为在 AE 和 Contrastive 方法中，隐编码表示是显式建模的，而 GAN 是隐式的，因此我们需要提取出 GAN 的隐分布。AAE 率先尝试弥补这个 gap，AAE 参考 AE 的思想，为了提取隐分布，可以将 GAN 中的生成器替换为显式建模的 VAE，回忆一下前文提到的 VAE 损失函数：

$$\mathcal{L}_{\text{VAE}} = -\mathbb{E}_{q(z|x)}(-\log(p(x|z)) + \text{KL}(q(z|x)\|p(z)))$$

D

我们说过，AE 由于使用了损失导致建模对象可能是 pixel-level，而 GAN 使用的对比式损失可以更好地建模 high-level 表示，因此为了缓解这个问题，AAE 将上面的损失函数中的 KL 项替换为了一个判别损失：

$$\mathcal{L}_{\text{Disc}} = \text{CrossEntropy}(q(z), p(z))$$

它要求判别器区分来自编码器的表示和先验分布。尽管如此，AAE 仍然保留了重构损失与 GAN 的核心思想还是存在矛盾。基于 AAE，BiGAN 和 ALI 模型主张保留对抗性学习，并提出新的框架如下图 11 所示。

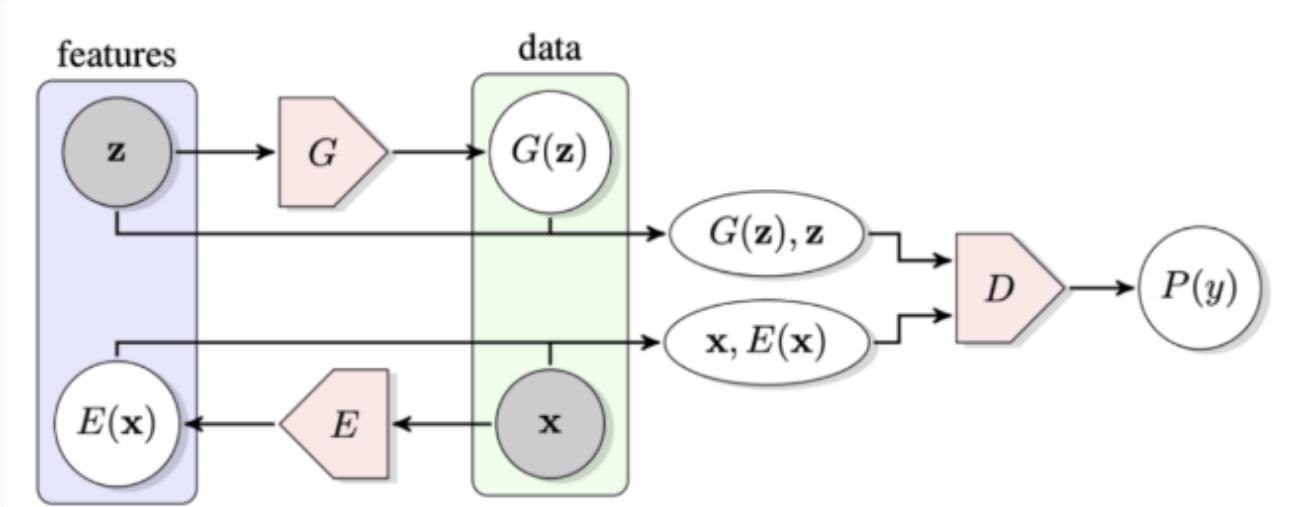


图11:BiGAN 和 ALI 的框架图。(Image source: J Donahue et al.)

给定一个真实样本:

- 生成器：这里生成器实际上扮演解码器的角色，其会根据先验隐分布生成 fake 样本
- 编码器：新引入的单元，其将真实样本 x 映射为表示，而这正是我们想要的表示
- 判别器：给定输入和，判断哪一个来自真实的样本分布。容易看出，训练的目标是，即编码器应当学会“转换“生成器。看起来似乎和 AE 很像，但是区别在于编码表示的分布不作任何关于数据的假设，完全由判别器来决定，可以捕捉 semantic-level 的差异。

Pre-trained Language Model

在很长一段时间内，预训练语言模型 PTM 都关注于根据辅助任务来最大化似然估计，因为判别式的目标函数由于语言的生动丰富性被认为是无助的。然而，最近的一些研究显示了 PTM 在对比学习方面的良好性能和潜力。这方面的先驱工作有 ELECTRA，在相同的算力下取得了超越 BERT 的表现。如下图 12 所示，ELECTRA 提出了 RTD (Replaced Token Detection)，利用 GAN 的结构构建了一个预训练语言模型，ELECTRA 的生成器可以看作一个小型的 Masked Language Model，将句子中的 masked token 替换为正常的单词，而判别器预测哪些单词被替换掉，这里的替换的意思是与原始单词不一致。

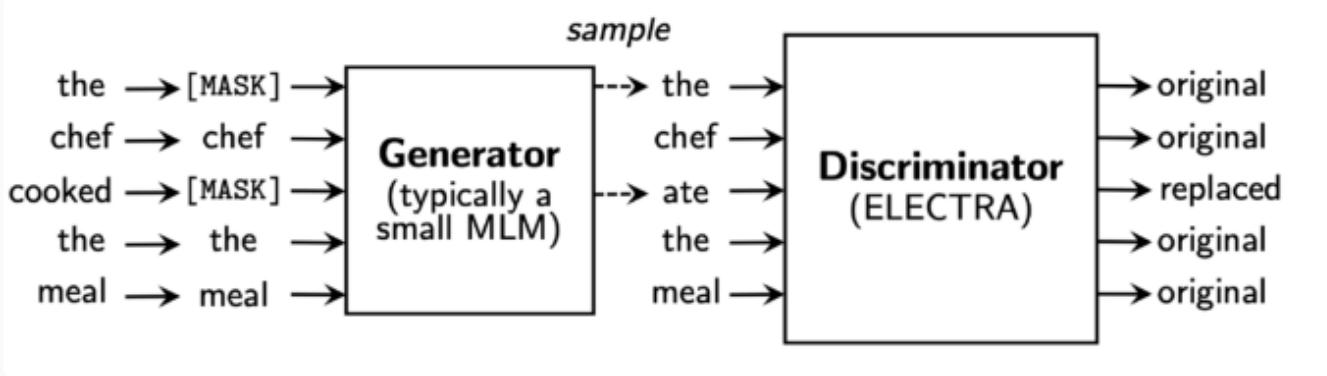


图12:ELECTRA 框架图。 (Image source: K Clark et al.)

其训练过程包含两个阶段：

- “生成器”热启动“：按照 MLM 的辅助任务训练一些步骤以对的参数进行”预热“。
- 训练判别器：判别器的参数将会初始化为的参数，并按照交叉熵判别损失进行训练，此时生成器的参数将会被冻结。最终的目标函数为：、

$$\min_{\theta_G, \theta_D} \sum_{x \in \mathcal{X}} \mathcal{L}_{MLM}(x, \theta_G) + \lambda \mathcal{L}_{Disc}(x, \theta_D)$$

尽管 ELECTRA 的设计是仿照 GAN 的，但是其训练方式不符合 GAN，这是由于图像数据连续的，而文本数据是离散的阻止了梯度的传播。另一方面，ELECTRA 实际上将多分类转化为了二分类，使得其计算量降低，但也可能因此降低性能。

Graph Learning

在图学习领域中，也有利用对抗学习的实例，但是不同模型之间的区别很大。大部分方法都跟随 BiGAN 和 ALI 选择让判别器区分生成的表示和先验分布，例如 ANE 的训练分为两步：1) 生成器将采样的图编码为嵌入表示，并计算 NCE 损失；2) 判别器区分生成的嵌入表示和先验分布。最终的损失是对抗损失加上 NCE 损失，最终可以返回一个更好的表示。GraphGAN 建模链路预测任务，并且遵循 GAN 的原始形似，直接判别节点而非表示。如下图 13 所示，GraphGAN 表示分类任务中很多的错误都是由于边缘节点造成的，通常情况下同一聚簇内的节点会在嵌入空间内聚集，聚簇之间会存在包含少量节点的 density gap，作者证明如果能够在 density gap 中生成足够多的 fake 节点，就能够改善分类的性能。在训练过程中，GraphGAN 利用生成器在 density gap 中生成节点，并利用判别器判断节点的原始类别和生成的 fake 节点的类别，在测试阶段，fake 节点将会被移除，此时分类的性能应当得到提升。

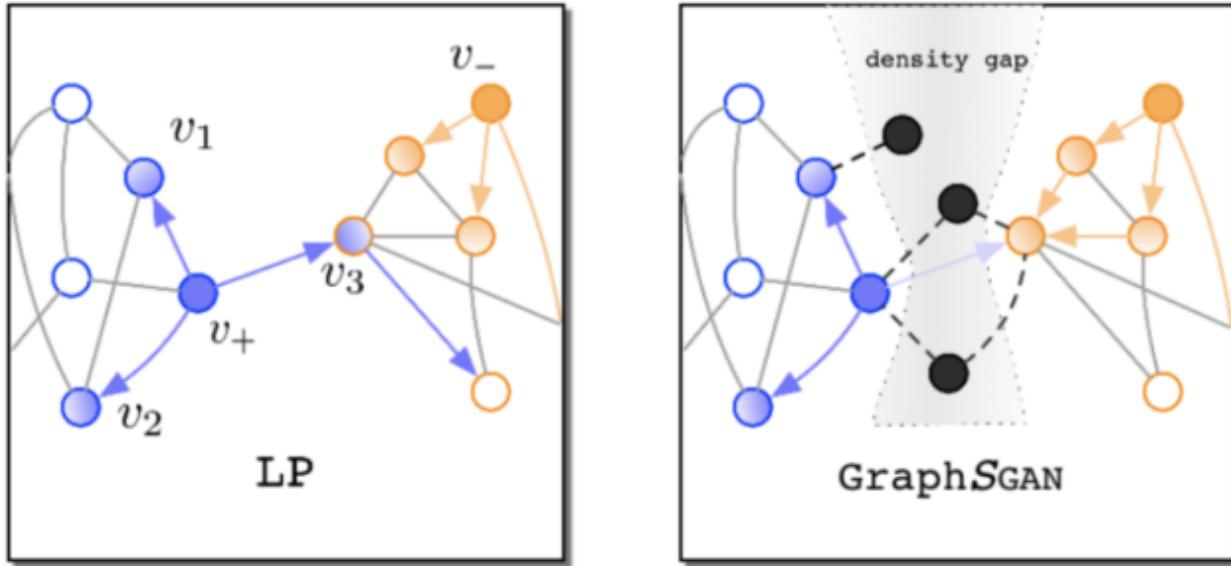


图13:GraphGAN 示意图。(Image source: H Wang et al.)

Discussions and Future Directions

Theoretical Foundation: 尽管自监督学习的方法众多，但是对于自监督学习背后的理论基础的探讨很少，理论分析十分重要，它可以避免让我们误入歧途。S Arora et al. 探讨了对比学习目标函数的泛化能力，而 M Tschannen et al. 则指出互信息与几种基于互信息的方法的成功关系不大，反而是其中采样策略和架构设计可能更重要。这类工作对于自监督学习形成坚实的基础至关重要，我们亟需更多的理论分析工作。

Transferring to downstream tasks: 在预训练和下游任务之间有一个较大的 gap。研究人员精心设计各种辅助任务 (pretext task)，以帮助模型学习数据集的一些重要特征，使得这些特征可以转移到其他下游任务中，但有时可能也很难实现。此外，选择辅助任务的过程似乎过于启发式，没有模式可循。对于预训练的任务选择问题，一个可能令人兴奋的方向是像神经架构搜索 (Neural Architecture Search) 那样为特定的下游任务自动设计预训练任务。

Transferring across datasets: 这个问题也被称为如何学习归纳偏差或归纳学习。传统上，我们将数据集分为用于学习模型参数的训练集和用于评估的测试集。这种学习范例的前提是现实世界中的数据符合我们训练数据集中的分布。然而，这种假设往往与实践中不符。自监督表示学习解决了部分问题，特别是在自然语言处理领域。大量的语料库用于语言模型的预训练，有助于涵盖大多数语言模式，因此有助于 PTM 在各种语言任务中的成功。然而，这是基于同一语言文本共享相同的嵌入空间这一事实。对于像机器翻译这样的任务和图学习这样的领域，不同数据集的嵌入空间不同，如何有效地学习可迁移的归纳偏差仍然是一个问题。

Exploring potential of sampling strategies: M Tschannen et al. 指出采样策略对于基于互信息的模型贡献较大，MoCo 和 SimCLR 以及一系列对比学习方法也支持了这一观点，他们都提出

要尽可能的增大负样本的数量并增强正样本，这一点在深度度量学习（metric learning）中有所探讨。如何进一步提升采样带来的性能，仍然是一个有待解决而又引人注目的问题。

Early Degeneration for Contrastive Learning: 尽管 MoCo 和 SimCLR 逼近了监督学习的性能，然而，它们通常仅限于分类问题。同时，语言模型预训练的对比-生成式方法 ELECTRA 在几个标准的 NLP 基准测试中也以更少的模型参数优于其他的生成式方法。然而，一些评论表明，ELECTRA 在语言生成和神经实体提取方面的表现并没有达到预期。这一问题可能是由于对比式目标函数经常陷入嵌入空间的提前退化（early degeneration）问题，即模型过早地适应了辅助任务（pretext task），从而失去了泛化能力。我们期望在保持对比学习优势的同时，会有新的技术或范式来解决提前退化问题。

去噪自编码器

参考: <https://zhuanlan.zhihu.com/p/94350902>

对比学习

<https://mp.weixin.qq.com/s/h8loG3enT5U-5F2a2UflJg>

Moment Matching for Multi-Source Domain Adaptation

论文: <https://arxiv.org/pdf/1812.01754.pdf> overview

Multimodel



自然语言中的多任务学习

多领域 (Multi-Doman) 任务

- Multi-Domain Text Classification
- Multi-Domain Sentiment Analysis

多级 (Multi-Level) 任务

- part-of-speech (POS)
- tagging, named entity recognition (NER)
- semantic role labeling (SRL)

多语言 (Multi-Linguistic) 任务

- Machine translation
- Multi-lingual parsing

多模态 (Multi-Modality) 任务

- Visual QA
- Image Caption



深度学习在自然语言处理中的“困境”

What

- Not Truly Deep
- One layer LSTM + Attention is enough for most NLP tasks.



Why

- 缺少大规模的标注数据
- 标注代价太高

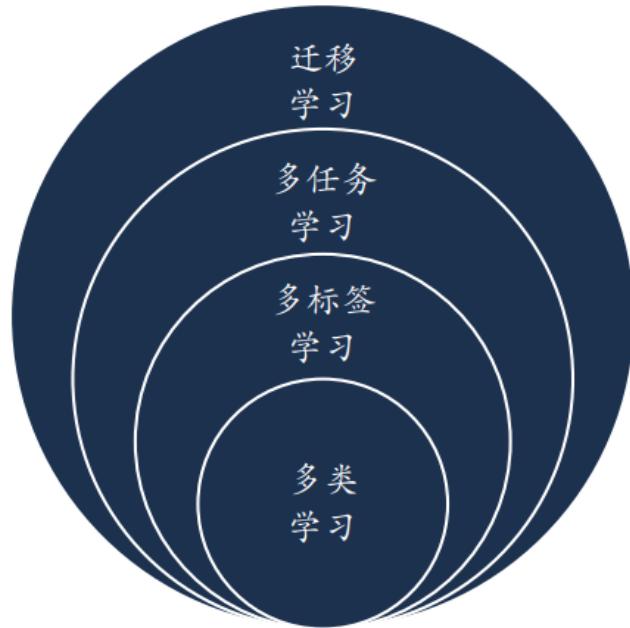


How

- 无监督预训练
- 多任务学习



不同学习范式之间的关系



- ▶ 迁移学习 Transfer Learning
 - ▶ 在源领域上学习模型
 - ▶ 泛化到目标领域上
- ▶ 多任务学习 Multi-Task Learning
 - ▶ 同时建模多个相关任务
 - ▶ 不同任务有不同的数据和标签
- ▶ 多标签学习 Multi-Label Learning
 - ▶ 一个样本可以有多个标签
 - ▶ 建模标签之间的关系
- ▶ 多类学习 Multi-Class Learning
 - ▶ 一个样本只能属于一个标签

Multimodel Neural Language Models

"The first approach to use neural networks for caption generation was Kiros et al. (2014a), who proposed a multimodal log–bilinear model that was biased by features from the image. "–
-[from Visual Attention]

MNLM

 [Multimodal Neural Language Models.pdf](#)

This paper introduced two multimodal neural language models: models of natural language that can be conditioned on other modalities,image–text and text–image.We show that in the case of image–text modelling we can jointly learn word representations and image features by training our models together with a convolutional network. Unlike many of the existing methods, this approach can generate sentence descriptions for images without the use of templates, structured prediction, and syntactic trees.Also, it can be easily applied to other modalities such as audio.



change is in the air . what starts out as a swinging wristlet easily transforms to an adorable handbag with just two quick clicks of the trigger clasp . or , you can attach it to a larger bag in a snap . the zipper closure reveals and interior pocket and ample room for all the essentials . NUM wrist strap . trigger clasp lets yo ...



our 14k NUM / 4ct round diamond stud earrings feature two fiery , perfectly proportioned & meticulously matched diamonds . these earrings measure NUM . NUM - NUM . 8mm in diameter , and the total carat weight of the diamonds mounted in these earrings is NUM / 4ct . the diamonds are i / j / k in color and i2 in clarity . these earrings are made out ...

FOOTJOY



this product contains a variety of strategically placed peter stripes . multi - hued silk upper in pure woven red silk bow tie in navy . masculine first width . this hand - based silk ties from forzieri offers success to any wardrobe . decidedly blue . imported clean . NUM % silk .



nine children , some of them waving at the camera , others are a bit shy and are looking away ; one person is pointing to the camera ;



a room with white walls , a red tiled floor , a blue window , two double beds with reddish bed covers , two lamps , a telephone and a picture ;



in this picture there is another grey pavement on the right ; three grey clouds and a blue sky in the background ; the houses and on the left before it ; a dark green , wooded slopes behind it ; grey clouds in a light blue sky in the background ; snow covered mountains



in this picture there is another wall in the background ; a man with a white chequered waistcoat and a small books ; there is food and a white train table and a window with black waistcoat and green trousers (tables in the background) ; another man in a classroom with salt bedcover and

Figure 1. Left two columns: Sample description retrieval given images. Right two columns: description generation. Each description was initialized to ‘in this picture there is’ or ‘this product contains a’, with 50 subsequent words generated.

The Log–Bilinear Model (LBL)

The log–bilinear language model (LBL) is a deterministic model that may be viewed Multimodal Neural Language Models as a feed–forward neural network with a single linear hidden layer.

$$P(w_n = i | w_{1:n-1}) = \frac{\exp(\hat{\mathbf{r}}^T \mathbf{r}_i + b_i)}{\sum_{j=1}^K \exp(\hat{\mathbf{r}}^T \mathbf{r}_j + b_j)}$$

This may be seen as scoring the predicted representation $\hat{\mathbf{r}}$ of w_n against the actual representation \mathbf{r}_{wn} through an inner product, followed by normalization based on the inner products amongst all other word representations in the vocabulary.

in which,

$$\hat{\mathbf{r}} = \sum_{i=1}^{n-1} \mathbf{C}^{(i)} \mathbf{r}_{w_i}$$

$\mathbf{C}^{(i)}$ are $D \times D$ context parameter matrices, Each word

w in the vocabulary is represented as a D -dimensional real-valued vector $\mathbf{r}_w \in \mathbb{R}^D$.

Let R denote the $K \times D$ matrix of word representation vectors where K is the vocabulary size, the weights between the output layer and linear hidden layer is the word representation matrix R where the output layer uses a softmax activation.

Multimodal Log–Bilinear Models

Modality–Biased Log–Bilinear Model(MLBL–B)

The MLBL–B model adds an additive bias to the next predicted word representation $\hat{\mathbf{r}}$ which is computed as:

$$\hat{\mathbf{r}} = \left(\sum_{i=1}^{n-1} \mathbf{C}^{(i)} \mathbf{r}_{w_i} \right) + \mathbf{C}^{(m)} \mathbf{x}$$

Our proposed models:

Left: The predicted next word representation $\hat{\mathbf{r}}$ is a linear prediction of word features $\mathbf{r}_{w_1}, \mathbf{r}_{w_2}, \mathbf{r}_{w_3}$ (blue connections) biased by image features \mathbf{x} .

Right: The word representation matrix R is replaced by a factored tensor for which the hidden-to-output connections are gated by \mathbf{x} .

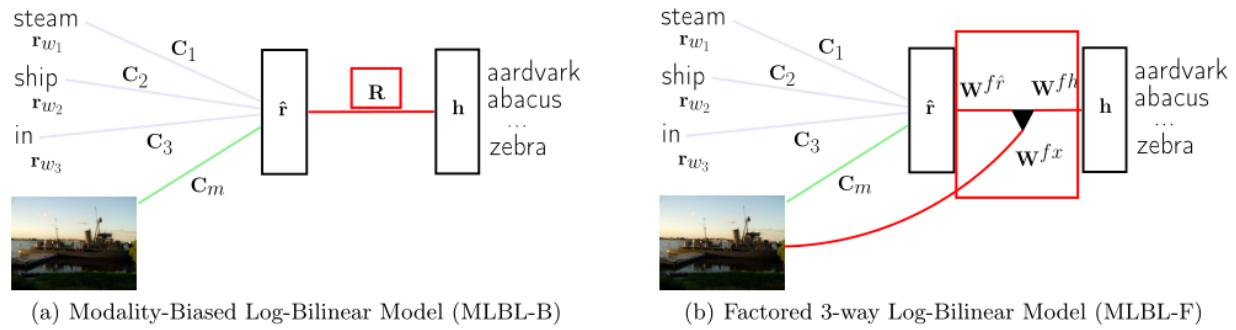


Figure 2. Our proposed models. Left: The predicted next word representation $\hat{\mathbf{r}}$ is a linear prediction of word features $\mathbf{r}_{w_1}, \mathbf{r}_{w_2}, \mathbf{r}_{w_3}$ (blue connections) biased by image features \mathbf{x} . Right: The word representation matrix \mathbf{R} is replaced by a factored tensor for which the hidden-to-output connections are gated by \mathbf{x} .

In the case of the LBL and MLBL–B models, each pre-trained word embedding can be used to initialize the rows of \mathbf{R} . In the case of the MLBL–E model where \mathbf{R} is a factored tensor, we can let \mathbf{E} be the $D \times K$ matrix of pre-trained embeddings. Since $\mathbf{E} = (\mathbf{W}^f \mathbf{r})^\top \mathbf{W}^h$, we can initialize the MLBL–F model with pre-trained embeddings by simply applying an SVD to \mathbf{E} .

The Factored 3-way Log-Bilinear Model(MLBL-F)

A more powerful model to incorporate modality conditioning is to gate the word representation matrix \mathbf{R} by the features \mathbf{x} .

$$\mathbf{R}^x = \sum_{i=1}^M x_i \mathbf{R}^{(i)}$$

where \mathbf{R}^x denotes the word representations with respect to features \mathbf{x} . $\mathbf{R}^{(i)}$ is $K \times D$ matrices specified by feature components $1, \dots, M$ of \mathbf{x} .

As is, the tensor \mathbf{R} requires $K \times D \times M$ parameters which makes using a general 3-way tensor impractical even for modest vocabulary sizes. A common solution to this approach (Memisevic & Hinton, 2007; Krizhevsky et

al., 2010) is to **factor** \mathbf{R} into three lower-rank matrices $\mathbf{W}^{\hat{r}} \in \mathbb{R}^{F \times D}$, $\mathbf{W}^f \in \mathbb{R}^{F \times M}$ and $\mathbf{W}^h \in \mathbb{R}^{F \times K}$, such that

$$\mathbf{R}^x = (\mathbf{W}^{fh})^\top \cdot \text{diag}(\mathbf{W}^{fx} \mathbf{x}) \cdot \mathbf{W}^{f\hat{r}}$$

Let $\mathbf{E} = (\mathbf{W}^{\hat{r}})^\top \mathbf{W}^{fh}$ denote the $D \times K$ matrix of word embeddings, the predicted next word representation is:

$$\hat{\mathbf{r}} = \left(\sum_{i=1}^{n-1} \mathbf{C}^{(i)} \mathbf{E}(:, w_i) \right) + \mathbf{C}^{(m)} \mathbf{x}$$

Given a predicted next word representation $\hat{\mathbf{r}}$, the factor outputs are:

$$\mathbf{f} = (\mathbf{W}^{f\hat{r}} \hat{\mathbf{r}}) \bullet (\mathbf{W}^{fx} \mathbf{x})$$

The conditional probability of w_n given w_1, \dots, w_{n-1} and \mathbf{x} can be written as

$$P(w_n = i | w_{1:n-1}, \mathbf{x}) = \frac{\exp((\mathbf{W}^{fh}(:, i))^\top \mathbf{f} + b_i)}{\sum_{j=1}^K \exp((\mathbf{W}^{fh}(:, j))^\top \mathbf{f} + b_j)}$$

以上：LBL, MLBL-B, MLBL-F的目的都是为了计算The conditional probability of w_n given w_1, \dots, w_{n-1} 。接下来：

Generation and Retrieval

The standard approach to evaluating language models is through perplexity:

$$\log_2 \mathcal{C}(w_{1:n} | \mathbf{x}) = -\frac{1}{N} \sum_{w_{1:n}} \log_2 P(w_n = i | w_{1:n-1}, \mathbf{x})$$

we use perplexity not only as a measure of performance but also as a link between text and the additional modality.

retrieving training images from a text query $w_{1:N}$:

compute $C(w_{1:N}|x)$ and return the images for which $C(w_{1:N}|x)$ is lowest.

Intuitively, images when conditioned on by the model that achieve low perplexity are those that are a good match to the query description.

retrieving text from an image query:

we instead look at the ratio $C(w_{1:N}|x)/C(w_{1:N}|x^{\sim})$ where x^{\sim} denotes the mean image in the training set (computed in feature space).

using the image itself as a query for other images:

first: retrieve the top k_r training images as a shortlist based on the Euclidean distance between x and images in the training set.

Then: retrieve the descriptions for which $C(w_{1:N}|x)/C(w_{1:N}|x^{\sim})$ is smallest for each description $w_{1:N}$ in the shortlist.

generate text given an image:

Suppose we are given an initialization $w_{1:n-1}$, where $n - 1$ is the context size. We compute $P(w_{n=i}|w_{1:n-1}, x)$ and obtain a sample w^{\sim} from this distribution, appending w^{\sim} to our initialization. This procedure is then repeated for as long as desired.

Conclusion

To speed up computation, we follow Wang et al.(2012); Swersky et al. (2013) and learn our convolutional networks on small feature maps learned using **k-means** as opposed to the original images. We follow the pipeline of

Coates & Ng (2011). Given training images, $r \times r$ patches are randomly extracted, contrast normalized and whitened. These are used for training a dictionary with spherical k-means. These filters are convolved with the image and a soft activation encoding is applied.

If the image is of dimensions $nV \times nH \times 3$ and k_f filters are learned, the resulting feature maps are of size $(nV - r + 1) \times (nH - r + 1) \times k_f$. Each slice of this region is then split into a $G \times G$ grid for which features within each region are max-pooled. This results in an **output of size $G \times G \times k_f$.** It is these outputs that are used as inputs **to the convolutional network.** For all of our experiments, we use $G = 9$ and $k_f = 128$.

To our surprise, we found additive biasing with high-level image features to be quite effective. A key advantage of the multiplicative model though is speed of training: even with learning rates an order of magnitude smaller these models typically required substantially fewer epochs to achieve the same performance.
Unlike MLBL-B, MLBL-F requires additional care in early stopping and learning rate selection.

Deep Visual-Semantic Alignments for Generating Image Descriptions

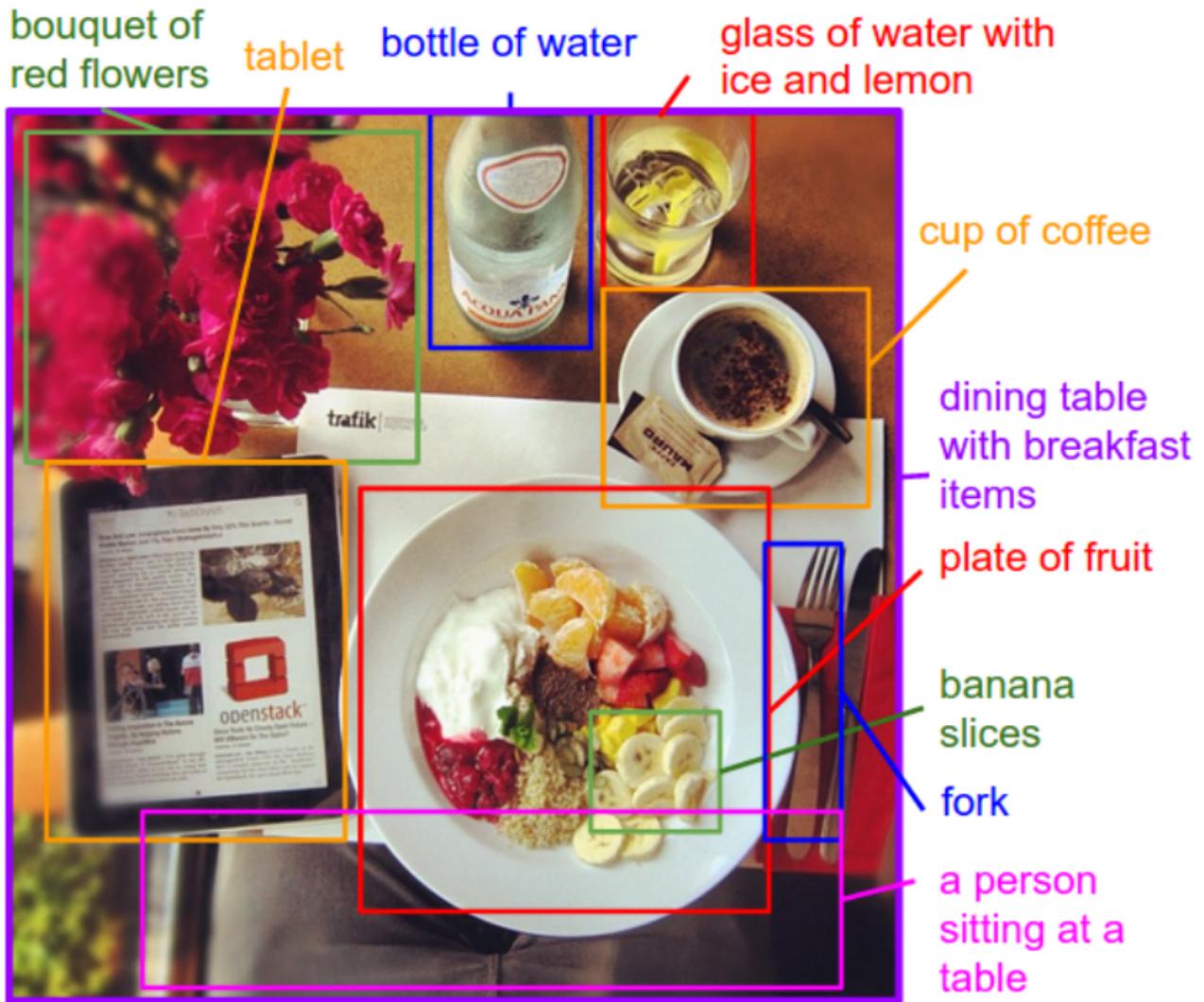
 [devisagen.pdf](#)

Previous Work

上一篇 'Multimodel Neural Language Models' 通过MLBL-B和MLBL-F两种方法比较好的实现了字幕生成，但是这个模型使用固定的窗口上下文，如本文所说：

“Most closely related to us, Kiros et al. [26] developed a log-bilinear model that can generate full sentence descriptions for images, but their model **uses a fixed window context** while our Recurrent Neural Network (RNN) model conditions the probability distribution over the next word in a sentence **on all previously generated words.**”

本文想要实现的目标是：



概念图：我们的模型将语言视为一个丰富的标签空间，并生成图像区域的描述

Some pioneering approaches that address the challenge

of generating image descriptions have been developed. However, these models often rely on hard-coded visual concepts and sentence templates, which imposes **limits on their variety**.

“Our approach is inspired by [Frome et al. \[16\]](#) who associate words and images through a semantic embedding. More closely related is the work of [Karpathy et al. \[24\]](#), who decompose images and sentences into fragments and infer their inter-modal alignment using a ranking objective. In contrast to their model which is based on grounding dependency tree relations, our model aligns contiguous segments of sentences which are more meaningful, interpretable, and not fixed in length.”

Our alignment model is based on a novel combination of Convolutional Neural Networks over image regions, bidirectional Recurrent Neural Networks over sentences, and a structured objective that aligns the two modalities through a multimodal embedding.

What we do?

The ultimate goal of our model is to generate descriptions of image regions. During training, the input to our model is a set of images and their corresponding sentence descriptions (如图). “Tabby cat is leaning” refer to the cat, the words “wooden table” refer to the table, etc.

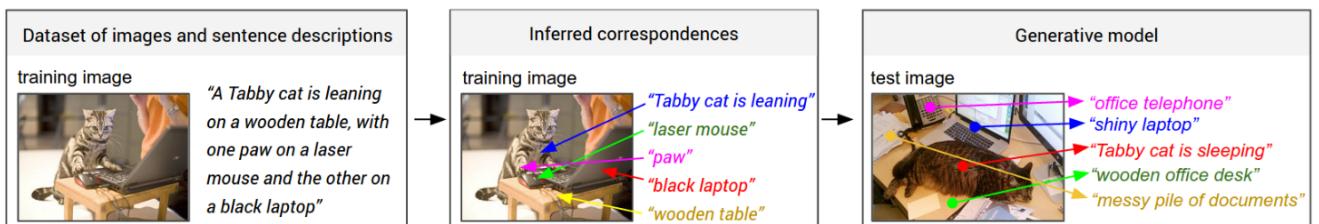


Figure 2. Overview of our approach. A dataset of images and their sentence descriptions is the input to our model (left). Our model first infers the correspondences (middle, Section 3.1) and then learns to generate novel descriptions (right, Section 3.2).

First, present a model that **aligns sentence snippets to the visual regions** that they describe through a multimodal embedding. Then, treat these correspondences as training data for a second, multimodal Recurrent Neural Network model that learns to generate the snippets.

Learning to align visual and language data

“猫猫在学习”指的是猫，“木桌”几个字指的是桌子等，我们想要推断这些潜在的对应关系，最终目标是稍后学习从图像区域生成这些片段。Karparis等人学习将依赖树关系与具有排名目标的图像区域建立联系。本文的贡献在于使用双向递归神经网络来计算句子中的单词表示，消除了计算依存关系树的需要，并允许单词与句子中的上下文进行无界交互，同时大大简化了它们的目标。

Representing images

Follow the method of Girshick et al. to detect objects in every image with a Region Convolutional Neural Network (**RCNN**). Following Karpathy et al., use the top 19 detected locations in addition to the whole image and compute the representations based on the pixels I_b inside each bounding box as follows:

$$v = W_m[\text{CNN}_{\theta_c}(I_b)] + b_m, \quad (1)$$

where $\text{CNN}(I_b)$ transforms the pixels inside bounding box I_b into 4096-dimensional activations of the fully connected layer immediately before the classifier. The CNN parameters θ_c contain approximately 60 million parameters. The matrix W_m has dimensions $h \times 4096$, where h is the size of the multimodal embedding space (h ranges from 1000–1600 in the experiments). Every image is thus represented as a set of h -dimensional vectors $\{v_i | i = 1 \dots 20\}$.

Representing sentences

为了建立跨模态关系，我们希望将句子中的词表示在图像区域占据的相同的多维嵌入空间中。最简单的方法可能是将每个单词直接投射到这个嵌入中。然而，这种方法没有考虑句子中的任何排序和单词上下文信息。这一想法的一个扩展是使用单词二元语法，或先前提出的依存关系树关系。但是，这仍然限制了上下文窗口的任意最大大小，并且需要使用可能针对无关文本语料库进行训练的依赖关系树解析器。

本文使用双向递归神经网络(BRNN)来计算单词表示，使用索引 $t=1 \dots N$ 表示单词在句子中的位置，BRNN的精确形式如下：

$$x_t = W_w \mathbb{I}_t \quad (2)$$

$$e_t = f(W_e x_t + b_e) \quad (3)$$

$$h_t^f = f(e_t + W_f h_{t-1}^f + b_f) \quad (4)$$

$$h_t^b = f(e_t + W_b h_{t+1}^b + b_b) \quad (5)$$

$$s_t = f(W_d(h_t^f + h_t^b) + b_d). \quad (6)$$

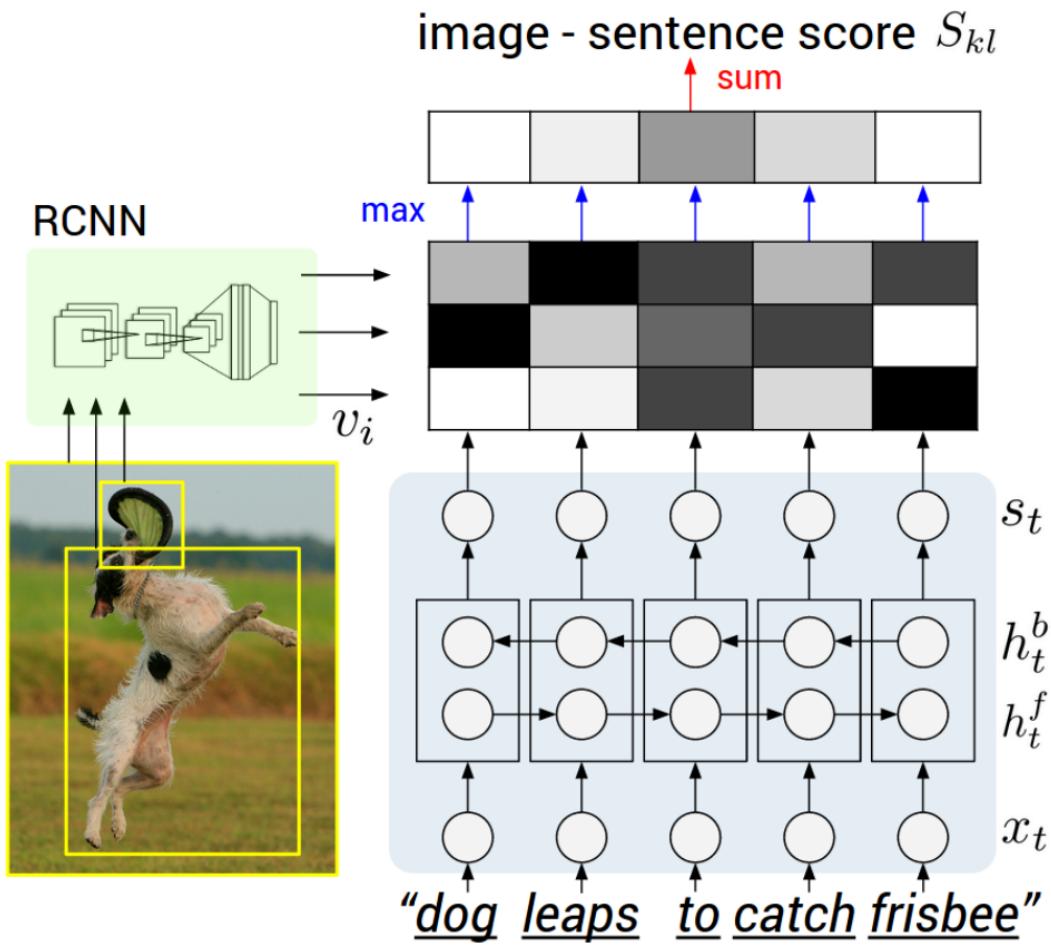
(1) \mathbb{I}_t is an indicator column vector that has a single one at the index of the t -th word in a word vocabulary.

(2) W_w specify a word embedding matrix that we initialize with 300-dimensional word2vec weights and keep fixed due to overfitting concerns.

(3) (4) Note that the BRNN consists of two independent streams of processing, one moving left to right (h_t^f) and the other right to left (h_t^b).

(5) The final h -dimensional representation s_t for the t -th word is a function of both the word at that location and also its surrounding context in the sentence.

We learn the parameters W_e , W_f , W_b , W_d and the respective biases b_e , b_f , b_b , b_d . A typical size of the hidden representation in our experiments ranges between 300–600 dimensions. We set the activation function f to the rectified linear unit (ReLU), which computes $f : x \rightarrow \max(0, x)$.



Object regions are embedded with a CNN (left). Words are embedded in the same multimodal space with a BRNN (right).

Alignment objective

The model of [Karpathy et al.](#) interprets the dot product $v_i^T s_t$ between the i -th region and t -th word as a measure of similarity and use it to define the score between image k and sentence L as:

$$S_{kl} = \sum_{t \in g_l} \sum_{i \in g_k} \max(0, v_i^T s_t). \quad (7)$$

g_k is the set of image fragments in image k and g_l is the set of sentence fragments in sentence L . The indices k, L range over the images and sentences in the training set.

We can interpret the quantity $v_i^T s_t$ as the **unnormalized log probability** of the t-th word describing any of the bounding boxes in the image. 每当点积为正时，句子片段就会与图像区域的子集对齐。

简化 (7) :

$$S_{kl} = \sum_{t \in g_l} \max_{i \in g_k} v_i^T s_t. \quad (8)$$

Pairwise similarities are computed with **inner products** (灰色部分) and finally reduced to image–sentence score with Equation 8. 每个单词都对齐到单个最佳图像区域，这种简化的模型还导致了最终排名性能的提高。

Assuming that $k = l$ denotes a corresponding image and sentence pair, the final max–margin, structured loss remains:

$$\begin{aligned} \mathcal{C}(\theta) = & \sum_k \left[\underbrace{\sum_l \max(0, S_{kl} - S_{kk} + 1)}_{\text{rank images}} \right. \\ & \left. + \underbrace{\sum_l \max(0, S_{lk} - S_{kk} + 1)}_{\text{rank sentences}} \right]. \end{aligned} \quad (9)$$

This objective **encourages aligned image–sentences pairs to have a higher score** than misaligned pairs, by a margin.

Decoding text segment alignments to images

为了将扩展的、连续的单词序列与单个边界框对齐，本文将 true alignments 视为马尔可夫随机场(MRF)中的潜在变量，其中相邻词之间的二元相互作用鼓励对同一区域的对齐。具体来说，给定一个有N个单词的句子和一个有M个边界框的图像，我们引入了潜在的排列变量 $a_j \in \{1, \dots, M\}$, $j=1, \dots, N$, 并沿着句子以链式结构表示 MRF，如下所示：

$$E(\mathbf{a}) = \sum_{j=1\dots N} \psi_j^U(a_j) + \sum_{j=1\dots N-1} \psi_j^B(a_j, a_{j+1}) \quad (10)$$

$$\psi_j^U(a_j = t) = v_i^T s_t \quad (11)$$

$$\psi_j^B(a_j, a_{j+1}) = \beta \mathbb{1}[a_j = a_{j+1}]. \quad (12)$$

β 是一个超参数，它控制对较长单词短语的亲和力。此参数允许我们在单个单词对齐($\beta=0$)和当 β 较大时将整个句子对齐到单个得分最高的区域之间进行插值，使用动态规划来最小化寻找最佳比对 a 的能量。

Multimodal Recurrent Neural Network for generating descriptions

During training our Multimodal RNN takes the imagepixels I and a sequence of input vectors (x_1, \dots, x_T) . It then computes a sequence of hidden states (h_1, \dots, h_T) and a sequence of outputs (y_1, \dots, y_T) by iterating the following recurrence relation for $t = 1$ to T :

$$b_v = W_{hi}[\text{CNN}_{\theta_c}(I)] \quad (13)$$

$$h_t = f(W_{hx}x_t + W_{hh}h_{t-1} + b_h + \mathbb{1}(t=1) \odot b_v) \quad (14)$$

$$y_t = \text{softmax}(W_{oh}h_t + b_o). \quad (15)$$

RNN training.

The RNN is trained to combine a word (x_t), the previous context (h_{t-1}) to predict the next word (y_t). We condition the RNN's predictions on the image information (b_v) via bias interactions on the first step. The training proceeds as follows (refer to Figure 4): We set $h_0=0^-$, x_1 to a special START vector, and the desired label y_1 as the first word in the sequence. Analogously, we set x_2 to the word vector of the first word and expect the network to predict the second word, etc. Finally, on the last step when x_T represents the last word, the target label is set to a special END token.

The cost function is to maximize the log probability assigned to the target labels (i.e. Softmax classifier).

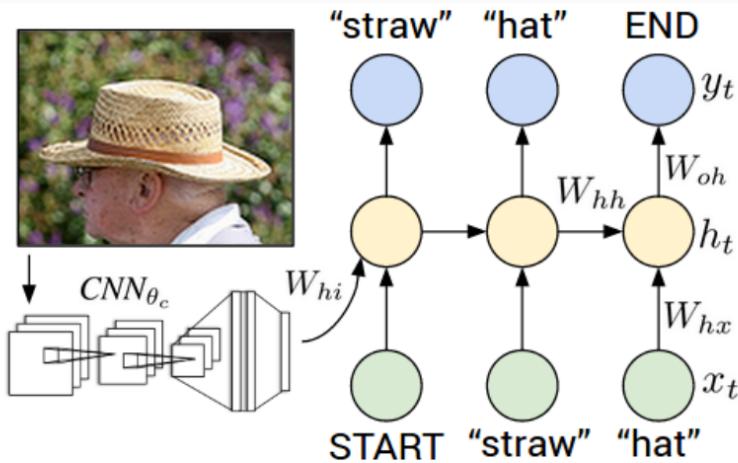


Figure 4. Diagram of our multimodal Recurrent Neural Network generative model. The RNN takes a word, the context from previous time steps and defines a distribution over the next word in the sentence. The RNN is conditioned on the image information at the first time step. START and END are special tokens.

RNN at test time.

To predict a sentence, we compute the image representation b_v , set $h_0 = 0$, x_1 to the START vector and compute the distribution over the first word y_1 . We sample a word from the distribution (or pick the argmax), set its embedding vector as x_2 , and repeat this process until the END token is generated. In practice , found that beam search (e.g. beam size 7) can improve results.

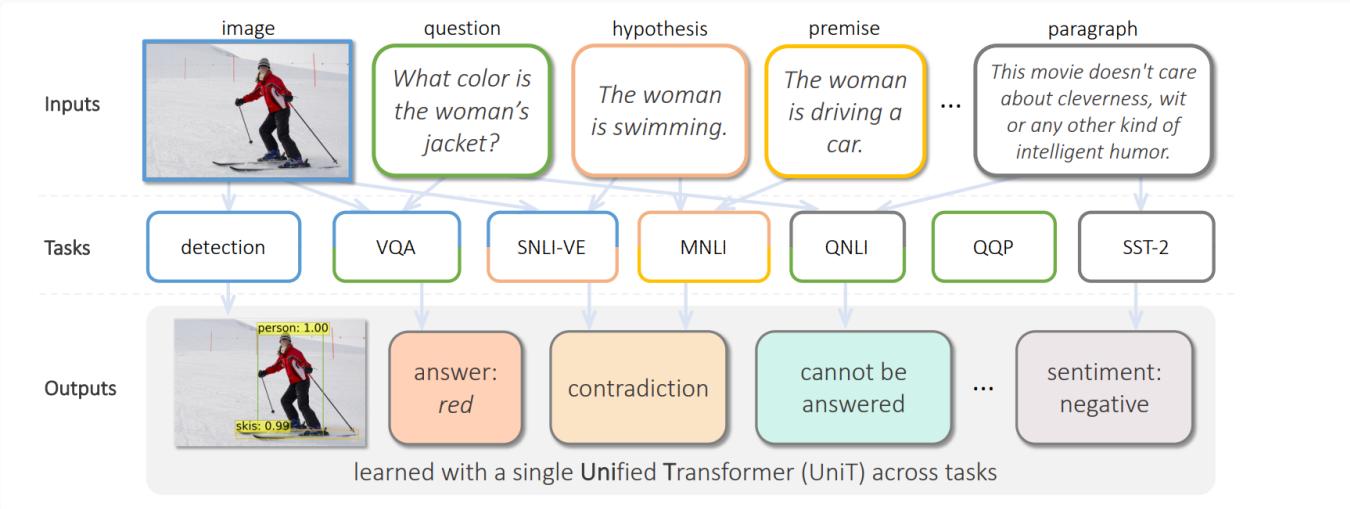
Optimization

1. 使用小批量100个图句对的SGD和0.9的动量来优化对齐模型;
2. 对学习率和权值衰减进行了交叉验证;
3. 在除递归层之外的所有层中使用Dropout正则化;
4. 在重要处以元素方式裁剪梯度;

5. 由于稀有词和常用词(如“a”或结尾标记)之间的词频差异，生成性RNN更难优化。使用RMSprop，这是一种自适应步长方法，它通过梯度范数的运行平均值来调整每个权重的更新。

UniT: Multimodal Multitask Learning with a Unified Transformer

<https://arxiv.org/pdf/2102.10772.pdf>



Introduction

以前的工作试图解决其中一些问题，但范围有限：

1. 只适用于单个领域或特定多模态领域的任务；ViT和DETR只关注视觉任务，BERT及其衍生作品只处理语言任务，而VisualBERT，ViLBERT及其他多模态Transformer只工作于特定的多模态视觉和语言领域。
2. 涉及每个任务的特定于任务的微调，而不是利用所有任务之间的任何共享参数，通常最终得到N个任务的N倍的参数，例如，必须使用BERT分别微调每个任务的模型。
3. 仅对单个领域的相关或类似任务执行多任务，有时使用硬编码的训练策略；例如，T5仅适用于语言领域的任务，而ViLBERT-MT仅适用于相关的视觉和语言任务

我们提出了一种统一的Transformer编解码器体系结构--UNIT，它以较少的参数在单一模型中处理多个任务和域，向通用智能迈进了一步。

1. UNIT共同学习视觉和文本领域及其交集中最突出的任务，即对象检测、视觉问答(VQA)、视觉蕴涵和GLUE benchmark 中的自然语言理解任务，包括QNLI、MNLI、QQP和SST-2。
2. UNIT表明，在我们的训练方案下，这些不同的任务可以同时学习并正确收敛。通过对各种任务的分析，我们发现，VQA和视觉蕴涵等多通道任务受益于单通道任务的多任务训练。

UniT: Unified Transformer across domains

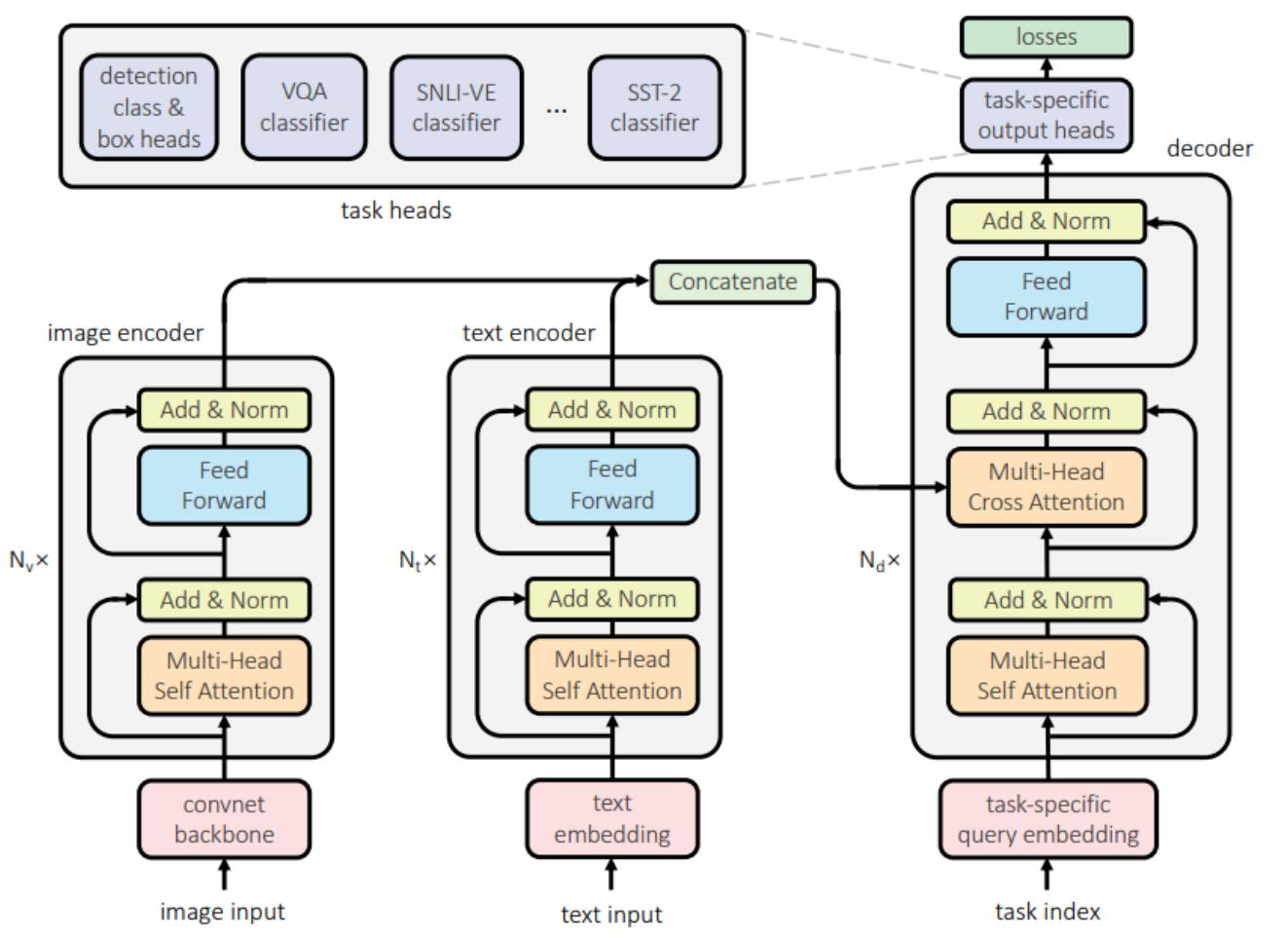


Image encoder

本文将Transformer强大的表达能力运用到视觉特征的提取中，由于图片像素点数量巨大，首先通过基于卷积神经网络的ResNet-50提取卷积特征，极大程度上地降低了特征数量，最终得到的feature map大小为 $W \times H \times d^b$ ，然后用全联接层调整单个特征的维度到 d^e ，再利用多层Transformer中的注意力机制提取各个feature之间的关系，由于Transformer的输入是序列，文章将 $W \times H$ 拉成一条长为 $L=WH$ 的序列，另外和文本编码器类似，同样添加了与下游任务相关的 W^{task} 。

$$\mathbf{x}^v = B(I). \quad B \text{ follows the structure of ResNet-50}$$

$$\mathbf{h}^v = \{h_1^v, h_2^v, \dots, h_L^v\} = E_v(P_{b \rightarrow e}(\mathbf{x}^v), w_v^{task}) \quad (2)$$

其中 $P_{b \rightarrow e}$ 是调整维度的全联接层， E 是多层Transformer编码器。

Text encoder

用BERT的结构提取文本内容，为了解决多个任务，在文本序列前添加了一个针对不同任务的参数向量 W^{task} ，在最后输出隐藏状态到解码器时再去掉。

$$\mathbf{h}^t = \{h_1^t, h_2^t, \dots, h_S^t\} = \text{BERT}(\{w_1, \dots, w_S\}, w_t^{task}) \quad (3)$$

Domain-agnostic UniT decoder

多模态的关键之一就在于怎么同时利用多个模态，在本文中是通过Transformer的解码器实现的，这个解码器首先将任务相关的query做self-attention，再将结果与文本编码器和视觉编码器的结果做cross-attention，针对单一模态的任务，选取对应编码器的输出即可，针对多模态的任务，取两个编码器输出的拼接。

$$\{\mathbf{h}^{dec,l}\} = D(\mathbf{h}^{enc}, \mathbf{q}^{task})$$

Task-specific output heads

之前多模态预训练模型往往只针对某一项任务，而本文提出的一个模型可以解决多个文本+视觉任务，与BERT可以解决多个文本任务类似，本文的模型在模态融合解码器的结果上添加为每个任务设计的处理器，这个处理器相对简单，用于从隐藏状态中提取出与特定任务相匹配的特征。

- **目标检测：**添加box_head和class_head两个前馈神经网络从最后一层隐藏状态中提取特征用来确定目标位置和预测目标类型。

$$\mathbf{c}^l = \text{class_head}(\mathbf{h}^{dec,l}) \quad (5)$$

$$\mathbf{b}^l = \text{box_head}(\mathbf{h}^{dec,l}) \quad (6)$$

$$\mathbf{a}^l = \text{attr_head}(\mathbf{h}^{dec,l}, \mathbf{c}^l) \quad (7)$$

- **自然语言理解、视觉问答：**通过基于全联接层的分类模型实现，将模态融合解码器结果的第一位隐藏状态输入到两层全联接层并以GeLU作为激活函数，最后计算交叉熵损失。

$$\mathbf{p} = W_1 \cdot \text{GeLU}(W_2 \cdot \mathbf{h}_1^{dec,N_d} + b_2) + b_1 \quad (8)$$

$$\text{loss} = \text{CrossEntropyLoss}(\mathbf{p}, \mathbf{t}) \quad (9)$$

Experiments

本文提出的多模态预训练模型各个板块划分明确，通过多层Transformer分别提取特征，再利用解码器机制融合特征并完成下游任务，同时借助最后一层任务相关的处理器，可以通过一个模型解决多个任务，同时也让多任务预训练成为可能，并在实验中的各个数据集上得到了论文主要进行了两部分实验：

多任务学习：

多任务涉及目标检测和视觉问答两个任务，在目标检测上运用COCO和VG两个数据集，在视觉问答上运用VQAv2数据集。对比了单一任务和多任务同时训练的结果，同时对比了不同任务共用解码器的结果，

#	decoder setup	COCO det. mAP	VG det. mAP	VQAv2 accuracy	#	training data	COCO det. mAP	VG det. mAP	VQAv2 accuracy
1	single-task training	40.6 / –	3.87	66.38 / –	1	single-task training	40.6	3.87	66.38
2	separate	40.8 / –	3.91	68.84 / –	2	COCO + VQAv2	40.2	–	66.88
3	shared	37.2 / –	4.05	68.79 / –	3	VG + VQAv2	–	3.83	68.49
4	shared (COCO init.)	40.8 / 41.1	4.53	67.30 / 67.47	4	COCO + VG + VQAv2	40.8	4.53	67.30

多模态学习：

这一实验中，为了体现所提出模型能够有效解决多个多种模态的不同任务，论文作者在之前COCO、VG、VQAv2的基础上，增加了单一文本任务GLUE的几个数据集（QNLI、QQP、MNLI、SST-2）和视觉推断数据集SNLI-VE，从数据集的数量上可以看出本文模型的全能性。与本文对比的有纯文本的BERT、基于Transformer的视觉模型DETR、多模态预训练模型VisualBERT

#	decoder setup	COCO det. mAP	VG det. mAP	VQAv2 accuracy	SNLI-VE accuracy	QNLI accuracy	MNLI-mm accuracy	QQP accuracy	SST-2 accuracy
1	UniT – single-task training	40.6	3.87	66.38 / –	70.52 / –	91.62 / –	84.23 / –	91.18 / –	91.63 / –
2	UniT – separate	32.2	2.54	67.38 / –	74.31 / –	87.68 / –	81.76 / –	90.44 / –	89.40 / –
3	UniT – shared	33.8	2.69	67.36 / –	74.14 / –	87.99 / –	81.40 / –	90.62 / –	89.40 / –
4	UniT – separate (COCO init.)	38.9	3.22	67.58 / –	74.20 / –	87.99 / –	81.33 / –	90.61 / –	89.17 / –
5	UniT – shared (COCO init.)	39.0	3.29	66.97 / 67.03	73.16 / 73.16	87.95 / 88.0	80.91 / 79.8	90.64 / 88.4	89.29 / 91.5
6	UniT – per-task finetuning	42.3	4.68	67.60 / –	72.56 / –	86.92 / –	81.53 / –	90.57 / –	88.06 / –
7	DETR [5]	43.3	4.02	–	–	–	–	–	–
8	VisualBERT [31]	–	–	67.36 / 67.37	75.69 / 75.09	–	–	–	–
9	BERT [14] (bert-base-uncased)	–	–	–	–	91.25 / 90.4	83.90 / 83.4	90.54 / 88.9	92.43 / 93.7

Table 3: **Performance of our UniT model on 7 tasks across 8 datasets**, ranging from vision-only tasks (object detection on COCO and VG), vision-and-language reasoning tasks (visual question answering on VQAv2 and visual entailment on SNLI-VE), and language-only tasks from the GLUE benchmark (QNLI, MNLI, QQP, and SST-2). For the line 5, 8 and 9, we also show results on VQAv2 test-dev, SNLI-VE test, and from GLUE evaluation server. See Sec. 4.2 for details.

本文提出的模型其实并不能在所有数据集上刷出SOTA，比如COCO上逊色于DETR，SNLI-VE逊色于VisualBERT，SST-2逊色于BERT，其他数据集上都有一定的提高，但是模型却胜在一个“全”字，模型的结构十分清晰明了，各个板块的作用十分明确，同时针对不同任务的处理器也对后续多模态任务富有启发性。

算法

参考: https://blog.csdn.net/weixin_37947156/article/details/94885690?spm=1001.2014.3001.5501

代码: https://github.com/KaiyuanGao/Kick_Algorithm

RE

<http://wulc.me/>