笔记与问题

1. androguard使用androguard decompile命令，如果不生成png格式的CFG，消耗的时间确实会降低很多。
2. Susi：
    - SuSi is a tool that automatically generates a list of Android sources and sinks by analyzing the complete Android source code.
    - 为Android sources、sinks进行了分类，SuSi的输出就是以Android API方法签名形式分类好的sources和sinks。
3. tensorflow：

- 梯度：方向导数中最大的值，也就是单位步伐，函数值朝这个反向变化最快。

4. Keras:
    - 核心结构是model，其中最简单的模型是Sequential顺序模型，由多个网络层线性堆叠。
    - 模型需要知道它所期望的输入的尺寸，所以顺序模型的第一层（且只有第一层，因为下面的层可以自动推断尺寸）需要接受关于其输入尺寸的信息。
        - 关于Flatten()：实现按行降维。
        - 关于Dense，首先第一个数字，表示输出的维数，然后input_dim规定了输入的维数。

```
model = keras.Sequential()#全连接模型
model.add(keras.layers.Dense(1, input_shape=(12288,),
activation=keras.activations.sigmoid))#这里的1表示输出层的结点数
```

        - sigmoid激活函数：逻辑回归，适合于二分类问题，1/(1+e^-x)。
    - 模型编译：在训练模型之前，需要配置学习过程，通过compile完成，接收三个参数：优化器optimizer，损失函数loss，评估标准metrics。
        - 优化器optimizer：寻找最小的损失函数。常见的有：
            - adam。
        - 损失函数loss：真实值与预测值差别的一种合理表示。交叉熵能够更好地表示概率损失；softmax也是分类常用的损失函数
    - 模型训练：Keras模型在输入数据和标签的Numpy矩阵上进行训练，为了训练一个模型，通常会使用**fit**函数。
        - epochs：表示把所有的数据训练多少次。
        - batch_size：实际上应该是越大越好，但是对计算机的性能要求越高。
5. 图神经网络(GNN)
    - 先简单粗暴地大致了解GNN：GNN就是个提取特征的方法，能做分类和关联预测。
        - 聚合：找到图中某节点的邻居信息 N = a*节点1 + b*节点2 + c*节点3，其中a,b,c 的取值往往是关键。
        - 更新：把某一节点的自身特征和邻居信息以某种方式结合作为该节点的更新后的特征。
        - 多层：可以联系到多阶邻居。
    - 图的难点：图不规则，每个图都有一个大小可变的无序节点，图中的每个节点都有不同数量的相邻节点，使得卷积不再适用于图。

- 图卷积网络(Graph Convolution Networks, GCN)：将卷积运算从传统数据推广到图数据，核心思想是学习一个函数映射 f(.)，通过该映射，图中的节点 v_i 可以聚合它自己的特征 x_i 与它的邻居特征 x_j （$j \in N(v_i)$）来生成节点 v_i 的新表示；文字表述可以是利用边的信息对节点信息进行聚合而生成新的节点表示，简化公式如下，其中a(i, j)可以简化为节点i，j之间边的权值，是固定的。

$$x_i' = \Sigma_{j \in neigh(i)} a(i,j) x_j$$

- 图注意力网络(Graph Attention Networks, GAN)：注意力机制如今被广泛应用到了基于序列的任务中，它的优点是能够放大数据中最重要的部分的影响，可以类比成一种加权平均。对于上述的公式，学习i，j之间的权值a(i,j)，相当于确定对该边的attention程度，就是GAT，有两种思路，最终得到的都是注意力概率$a_{learn}(i,j)$：
  - 基于相似度的Attention： 用余弦相似度衡量节点间的差异

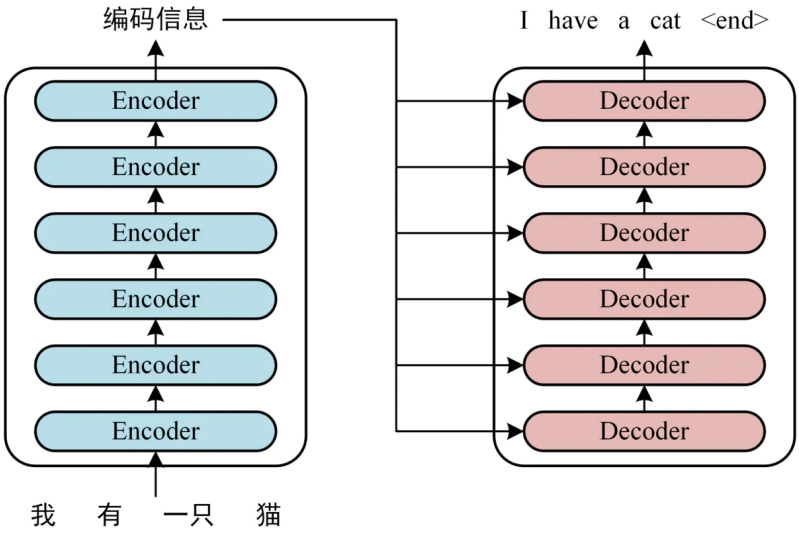
$$a_{learn}(i,j) = \frac{exp(\beta \times cos(Wx_i, Wx_j))}{\Sigma_{k \in neigh(i)} exp(\beta \times cos(Wx_i, Wx_j))}$$

  - 基于学习的Attention：

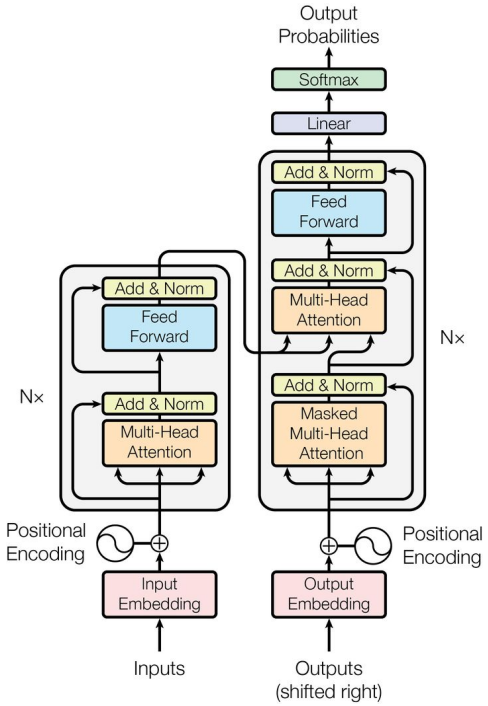
$$a_{learn}(i,j) = \frac{exp(LeakyReLU(w[Wx_i, Wx_j]))}{\Sigma_{k \in neigh(i)} exp(LeakyReLU(w[Wx_i, Wx_k]))}$$

- Transformer模型：Transformer的本质上是一个Encoder-Decoder的结构，核心是多头注意力机制(多头注意力机制就是使用多个注意力机制进行单独计算，以获取更多层面的语义信息)。是一个sequence-to-sequence(Seq2seq) model，输入一个sequence，输出一个sequence，输出长度是由model自己决定的。

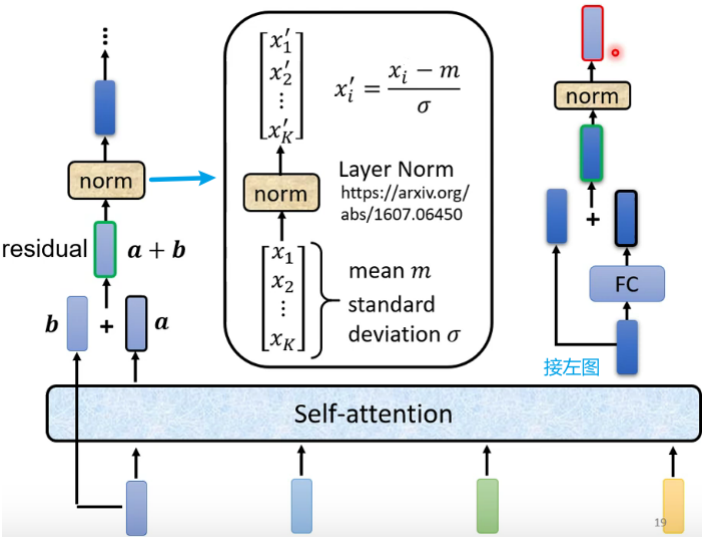
  下图是transformer用于中英文翻译的整体结构：



  下图的左侧为encoder block，右侧为decoder block，其中的multi-head attention是由多个self-attention组成的，self-attention是transform的重点。



- Encoder：输入一排向量，输出另一排向量，上图左边部分的详细如下图：



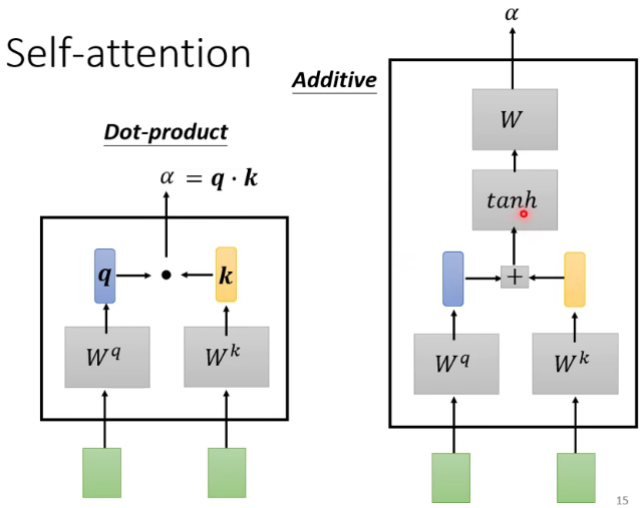
- Decoder：Decoder的输出，往往是穷举所有可能性的集合（包含很多的类）。
- 如何做训练的?

  在Decoder训练的时候要给它正确答案(Teacher Forcing)，

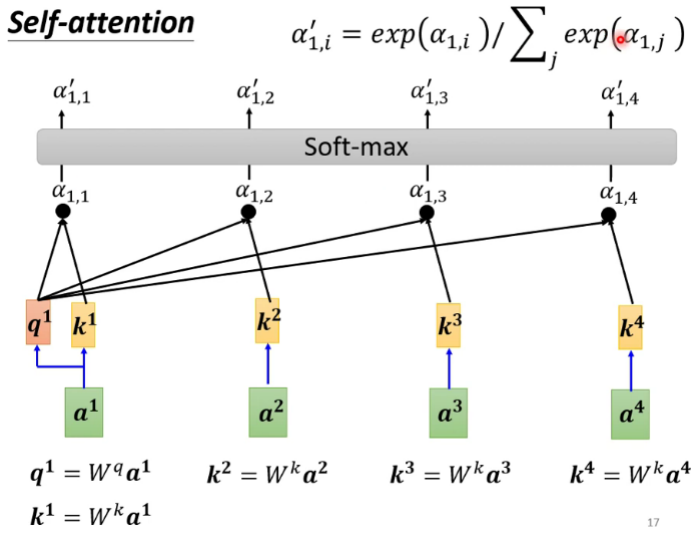
- self-attention： [Transformer模型详解（图解最完整版） - 知乎 (zhihu.com)]

  可以理解为一个预处理阶段，输入一排向量，输出是考虑了所有输入向量的联系的另一组输出向量。

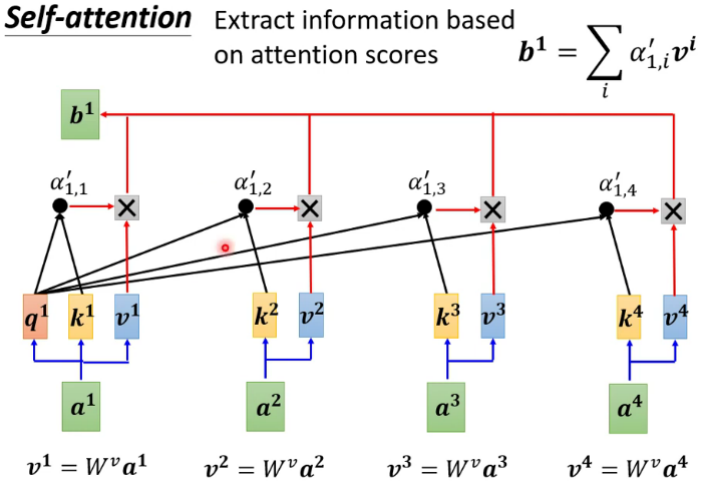
  - 给两个向量，计算出他们的关系α（常用的是dot-product、additive）：



- 然后输出a_1的新表示的α系数：

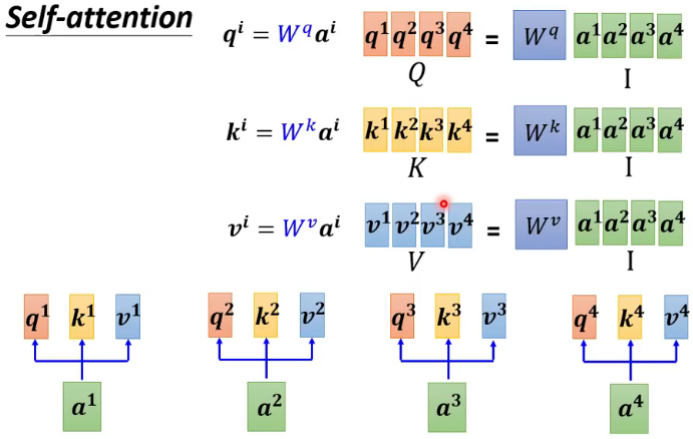


- 再引入V，得到最终的新表示b。

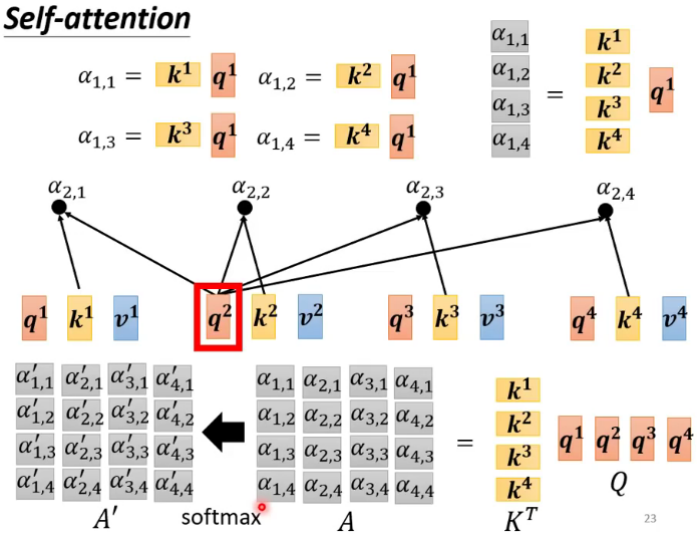


可以从矩阵的角度再来看看上面的过程：

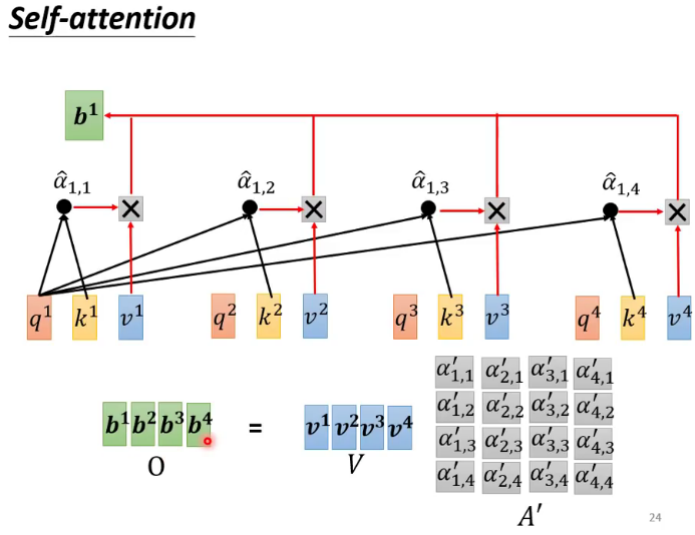
- 矩阵并行得到q、k、v：



- 得到系数α：

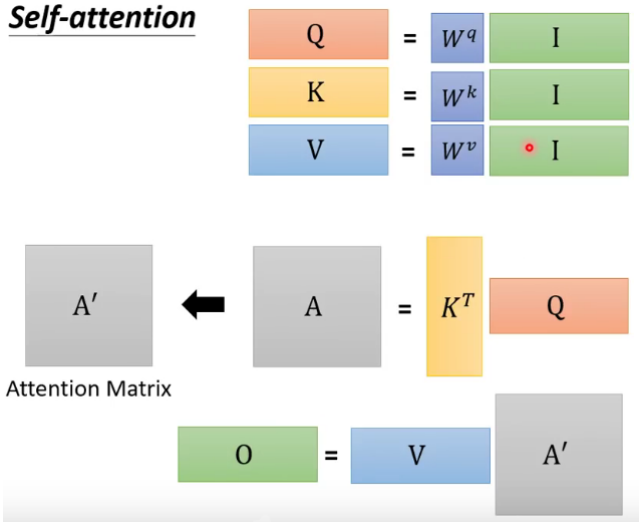


- 得到新表示：

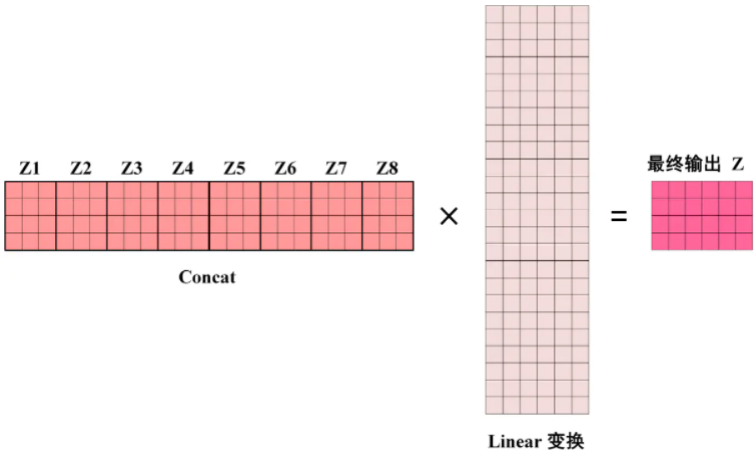


- 总结如下：

  只有W^q、W^k、W^v需要透过训练资料找出来。self-attention可以被理解为扩展版本的CNN。

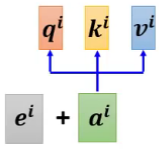


- Multi-head attention：将输入X经过多层（并行）self-attention后的结果拼接在一起后进行线性变换，得到最终的输出矩阵Z。



- positional information:

  为了考虑到某个vector在上下文中的位置，可以在α中加上e(e表示位置信息，有不同的表示)。



- 问题：
  - androguard反汇编出来的就是.java文件吗（虽然和自己写的源代码有些不同）？
  - tensorflow网站上对于过拟合的描述感觉不是很准确：过拟合是指机器学习模型在新的、以前未曾见过的输入上的表现不如在训练数据上的表现。过拟合的模型会“记住”训练数据集中的噪声和细节，从而对模型在新数据上的表现产生负面影响。
  - 是否要学习下BERT?
  - https://github.com/microsoft/tf-gnn-samples 是多篇论文关于GNN方法论文的实现

    https://github.com/microsoft/CodeXGLUE 关于code summary、code translation等的功能的实现。

    https://github.com/microsoft/CodeBERT CodeBERT这篇论文的复现， code search 和 code summary。

    后两个是有联系的，然后以什么样的顺序进行?