Week4 - Transformer

1 Sequence-to-sequence (seq2seq)

作用

输入一个序列, 输出一个序列. 输出的长度由模型决定.

使用方向

- 语音辨识
- 机器翻译
- 语音翻译
- Text-to-speech(TTS)
- Multi-label classification
- Object detaction

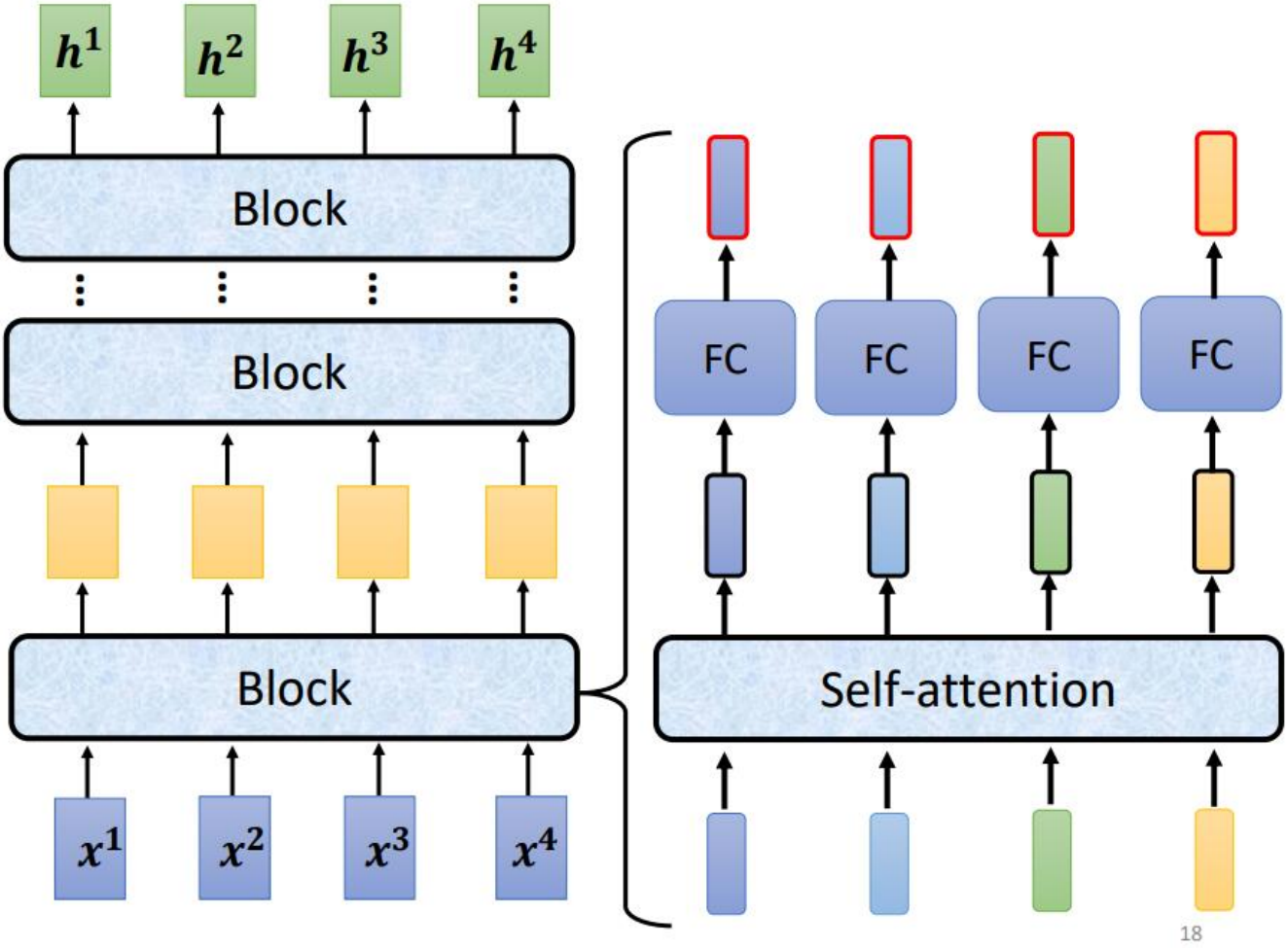
---

2 Encoder

> 输入一组向量, 输出一组同样长度的向量 (除了 `Self-attention`, 还可以使用 `RNN` 或 `CNN`).
>
> 如输入 $X(x^1, x^2, x^3, x^4)$, 输出 $H(h^1, h^2, h^3, h^4)$

`Encoder` 中, 输入需要经过多个 `Block` 的处理得到输出

原理

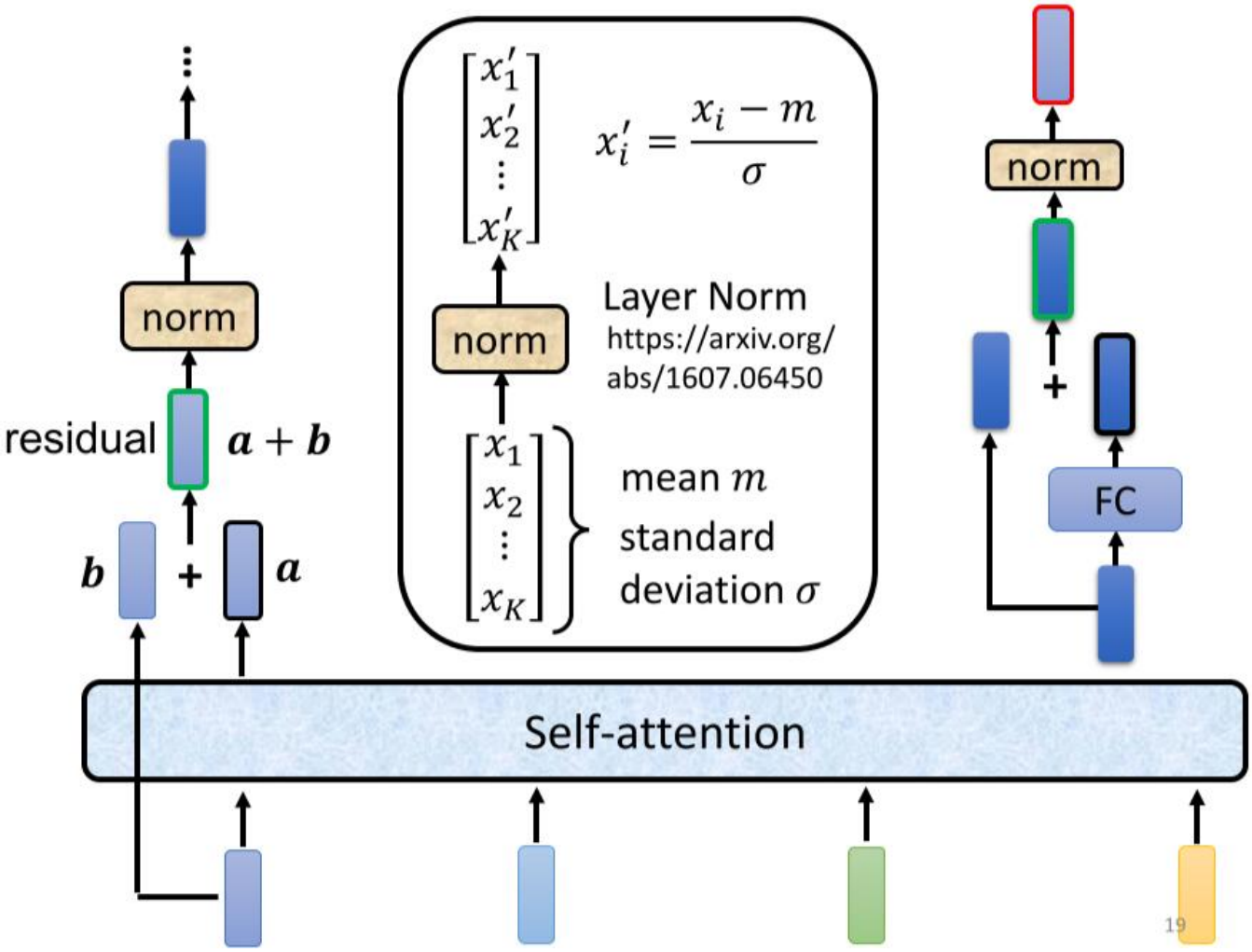


如 `Pic 4-1`:

1. 对输入 $X(x^1, x^2, x^3, x^4)$ 进行一次 `Self-attention`;
2. 将 `Self-attention` 的输出丢入 `Fully connect` 的 `Fead Forward Network` 中, 输出结果.

> 1个 `Block` 做的事, 是好几个Layer在做的事情, 因此 `Block` 并不是 `Neural Network` 中的一层.

`Block` 具体过程



> - `Residual Connection`: 将输出的向量加上之前输入的向量作为新的输出
> - `Layer Normalization`:
>   1. 输入一个向量, 计算其平均值 m, 标准差 σ
>   2. 不同于 `Batch Normalization`
>   - `Batch Normalization`: 对不同的样例, 不同特征的同一维度去计算 m 和 σ
>   - `Layer Normalization`: 对同一样例, 同一特征的不同维度去计算 m 和 σ
>   3. 根据公式 $x_i' = \frac{x_i - m}{\sigma}$ 得到新的向量

如 `Pic 4-2`:

1. 对向量位置有要求, 需要先对输入进行 `Position Embedding`
2. 对输入向量做 `Self-attention`, 得到 a
3. 做 `Residual Connection`, 得到 $(a+b)$
4. 对向量 $(a+b)$, 做 `Layer Norm`
5. 再对上一步得到的结果, 在做 `Fully connect`
6. 再对上一步的结果做 `Residual` 和 `Layer Norm`, 得到该 `Block` 的输出

重复多次 `Block` 过程.

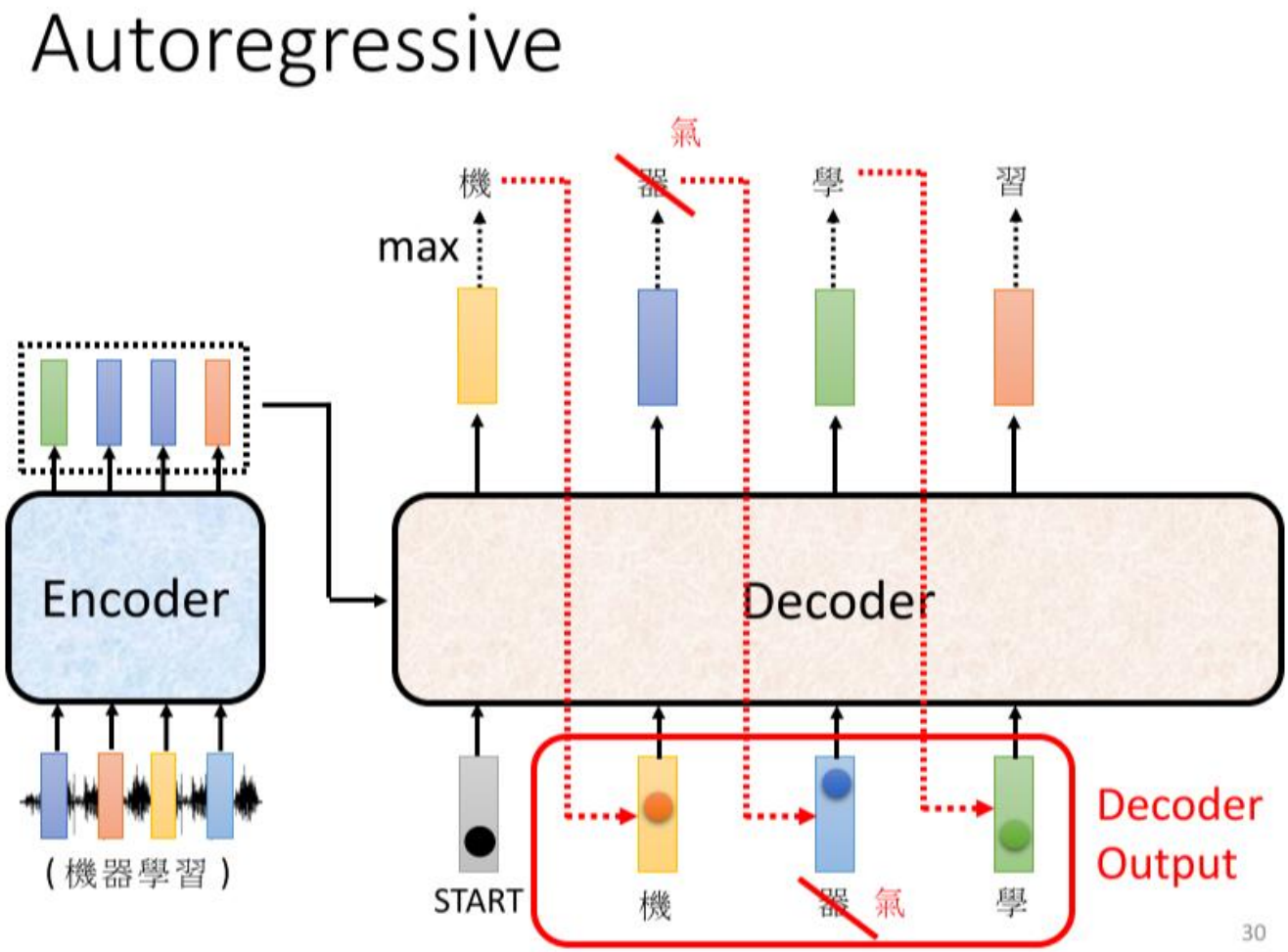
---

3 Decoder

> `Decoder` 完成模型的最终输出

Autoregression (**自回归**, AT)

过程



1. 开始给定一个Begin标记(`BOS`)
2. 输入向量经过 `Decoder` 得到输出的向量
3. 将输出的向量, 经过 `Soft-max` 得到每个字的分数, 将最高分数的字符作为结果
4. 将该次生成的结果作为下一次的输入, 重复第2步, 直到遇到End标记(`EOS`)

> `Pic 4-3` 的 `Decoder Output` 可能产生 `Error Propagation`, 即一步错步步错.

`Autoregression` 的 `Decoder` 结构

在 `Decoder` 中, 通常使用 `Masked Multi-head Self-attention`. 原本的 `Self-attention` 需要考虑所有输入向量才能计算出 (q, k, v), 之后才能得到输出向量.

而 `Masked Multi-head Self-attention` 只考虑自己与自己之前的向量即可, 例如需要输出向量 b^3 时, 只需要考虑 a^1, a^2, a^3 所产生的 (q, k, v), 而无需考虑 a^4

> Q1: **为什么使用** `Masked Multi-head Self-attention` ?
>
> 在 `AutoRegression` 中, 输出时一个个根据前一次的输出所产生的, 并不会考虑之后的事情.

> Q2: **什么时候Encoder停下来?**
>
> 准备一个特殊符号 `EOS`, 当结束时, 让End标记的几率最大时, 结束Encoder. 即让机器自己训练决定何时该结束.

Non-autoregression (NAT)

`NAT` 与 `AT` 的差别, 在于AT需要上一步的输出作为下一步的输入, 并一步步这样循环. 而 `AT` 则直接一次性输入, 直接得到输出.

如何决定 NAT 输出的长度

有两种思路:

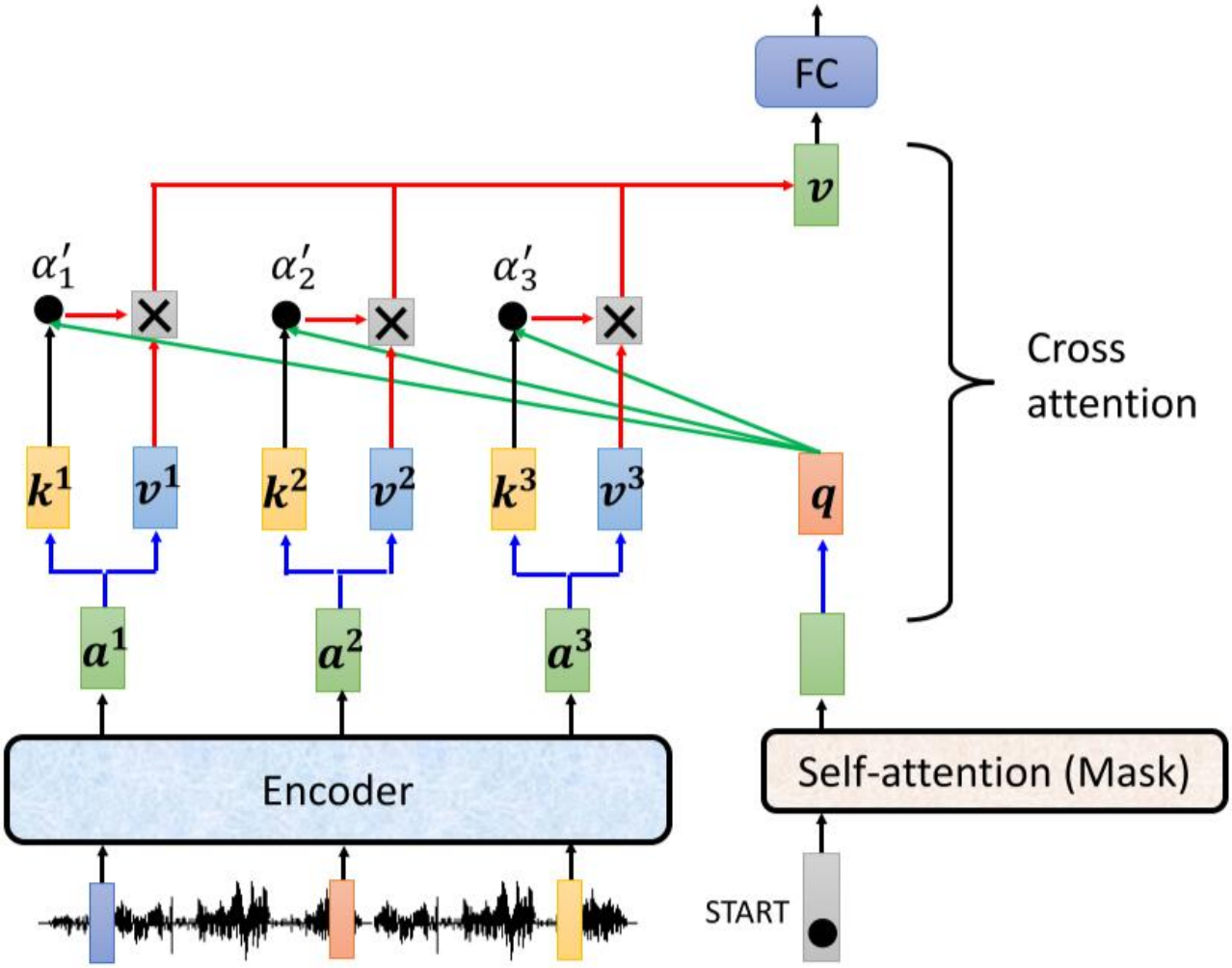
- 另外学习一个Classifier, 输入 Encoder 的输入, 输出一个值表示该Decoder输出的长度;
- 设定一个较大上限值, 观察何处出现End标记, 丢弃End标记之后的所有输出.

NAT 优缺点

- 优点: 平行处理, 可以控制输出的长度
- 缺点: 效率与性能不如 `AT`

---

4 Encoder 与 Decoder 之间的通信



1. 在 Encoder 部分, 输入一组向量后输出了 (a^1, a^2, a^3)
2. 在 Decoder 部分, 得到一个输入(如 `BOS`), 经过 `Masked Multi-head Self-attention` 得到向量, 乘上矩阵做 `Transformer`, 得到 query - q
3. 在 Encoder 部分, 各个向量一次产生 key - $(k^1, k^2, k^3, \cdots)$, 将 k 与 $(k^1, k^2, k^3, \cdots)$ 进行计算得到分数 - $(\alpha^1, \alpha^2, \alpha^3, \cdots)$, 进行 `Soft-max`, `Norm` 等操作, 得到 $(\alpha'_1, \alpha'_2, \alpha'_3, \cdots)$
4. 在 Encoder 部分, 产生向量 $(a^1, a^2, a^3, \cdots)$ 的 value - $(v^1, v^2, v^3, \cdots)$, 与 $(\alpha'_1, \alpha'_2, \alpha'_3, \cdots)$ 相乘. $v = \sum_h (v^i \alpha'_i)$
5. 将 v 做 `Fully-connection`, 丢入 Network 进行接下来的处理

---

5 Training

训练方式

当将 `BOS` 丢入 Encoder 时, 希望将输出与字符 x 越接近越好. x 会被表示成一个 `one-hot vector`, 只有 x 对应的那个维度是 1, 其余皆为 0.

此时 Decoder 输出一个 `Distribution` (几率的分布), 此时希望纪律分布与该 `one-hot vector` 越接近越好.

> 即计算输出的 `Ground Truth` 与 `Distribution` 之间的 `Cross-entropy`, 越小越好. 近似于分类问题.

在 Decoder 训练过程中, 在输入的时候基于其正确答案, 这个过程称为 `Teacher Forcing`.

在实际使用模型时, Decoder 因为仅仅时看见自己的输入, 中间存在 Mismatch.

解决Mismatch的一些方法

- Copy Mechanism
- Summarization
- Guide-attention
- Scheduled Sampling