One may argue that most DL models are minor variations of a few core architectures, such as the Transformer [VSP+17], so a reference augmented by a description of the changes should suffice. This would be true if (a) the changes were described precisely, (b) the reference architecture has been described precisely elsewhere, and (c) a reference is given to this description.

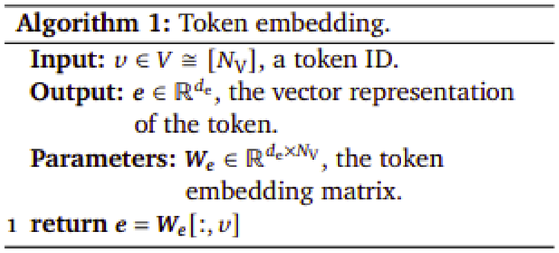
Transformers- good at modeling sequential data

- Sequence modeling (learn an estimation of the distribution)
  - Learn a distribution of a single token given its preceding tokens
  - Language modeling, music generation
- Sequence to sequence prediction (learn an estimation of the conditional distribution)
  - Text2speech, translation
- Classification (learn an estimate of the conditional distribution)
  - Sentiment classification, spam filtering

Tokenization

- Character level, word level, subword (most common)

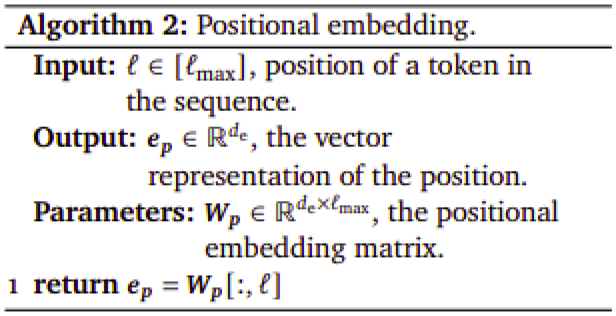
Token embedding



Algorithm 1: Token embedding.

```
Input: $v \in V \cong [N_V]$, a token ID.
Output: $e \in \mathbb{R}^{d_e}$, the vector representation
        of the token.
Parameters: $W_e \in \mathbb{R}^{d_e \times N_V}$, the token
            embedding matrix.
1 return $e = W_e[:, v]$
```

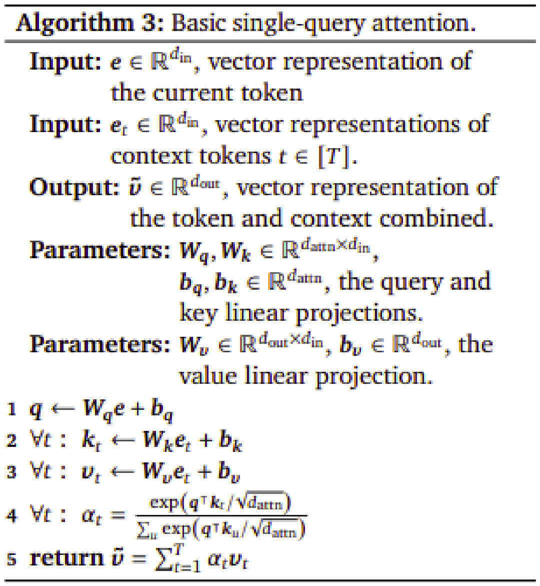
Positional embedding (learned require fixed max length while hard coded don't such as original transformer paper)



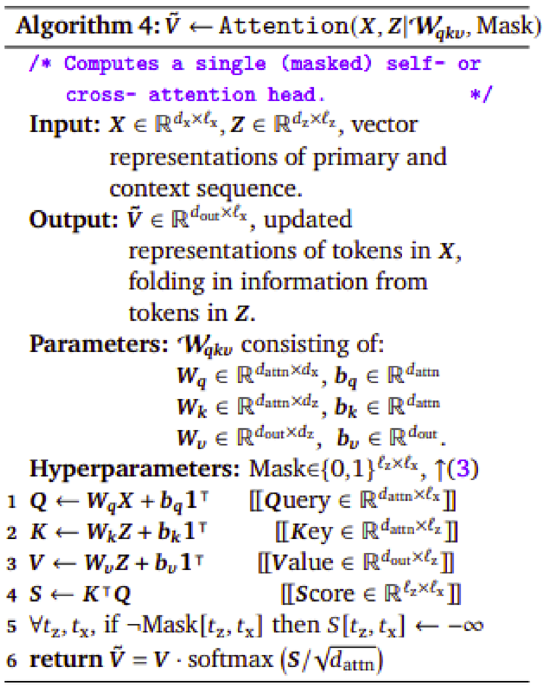
Algorithm 2: Positional embedding.

```
Input: $\ell \in [\ell_{max}]$, position of a token in
       the sequence.
Output: $e_p \in \mathbb{R}^{d_e}$, the vector
        representation of the position.
Parameters: $W_p \in \mathbb{R}^{d_e \times \ell_{max}}$, the positional
            embedding matrix.
1 return $e_p = W_p[:, \ell]$
```

Attention

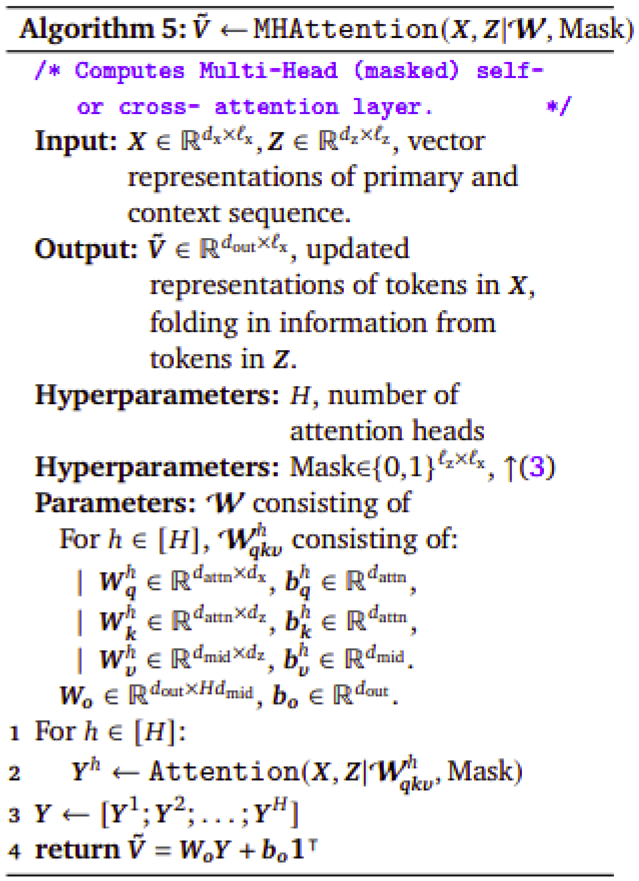


```
Algorithm 3: Basic single-query attention.
  Input: $e \in \mathbb{R}^{d_{\text{in}}}$, vector representation of
         the current token
  Input: $e_t \in \mathbb{R}^{d_{\text{in}}}$, vector representations of
         context tokens $t \in [T]$.
  Output: $\tilde{v} \in \mathbb{R}^{d_{\text{out}}}$, vector representation of
          the token and context combined.
  Parameters: $W_q, W_k \in \mathbb{R}^{d_{\text{attn}} \times d_{\text{in}}}$,
              $b_q, b_k \in \mathbb{R}^{d_{\text{attn}}}$, the query and
              key linear projections.
  Parameters: $W_v \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, $b_v \in \mathbb{R}^{d_{\text{out}}}$, the
              value linear projection.
1 $q \leftarrow W_q e + b_q$
2 $\forall t: \; k_t \leftarrow W_k e_t + b_k$
3 $\forall t: \; v_t \leftarrow W_v e_t + b_v$
4 $\forall t: \; \alpha_t = \frac{\exp(q^\top k_t / \sqrt{d_{\text{attn}}})}{\sum_u \exp(q^\top k_u / \sqrt{d_{\text{attn}}})}$
5 return $\tilde{v} = \sum_{t=1}^{T} \alpha_t v_t$
```



```
Algorithm 4: $\tilde{V} \leftarrow$ Attention($X, Z | \mathcal{W}_{qkv}$, Mask)
  /* Computes a single (masked) self- or
     cross- attention head.                    */
  Input: $X \in \mathbb{R}^{d_x \times \ell_x}$, $Z \in \mathbb{R}^{d_z \times \ell_z}$, vector
         representations of primary and
         context sequence.
  Output: $\tilde{V} \in \mathbb{R}^{d_{\text{out}} \times \ell_x}$, updated
          representations of tokens in $X$,
          folding in information from
          tokens in $Z$.
  Parameters: $\mathcal{W}_{qkv}$ consisting of:
      $W_q \in \mathbb{R}^{d_{\text{attn}} \times d_x}$, $b_q \in \mathbb{R}^{d_{\text{attn}}}$
      $W_k \in \mathbb{R}^{d_{\text{attn}} \times d_z}$, $b_k \in \mathbb{R}^{d_{\text{attn}}}$
      $W_v \in \mathbb{R}^{d_{\text{out}} \times d_z}$, $b_v \in \mathbb{R}^{d_{\text{out}}}$.
  Hyperparameters: Mask$\in \{0,1\}^{\ell_z \times \ell_x}$, ↑(3)
1 $Q \leftarrow W_q X + b_q 1^\top$      [[Query $\in \mathbb{R}^{d_{\text{attn}} \times \ell_x}$]]
2 $K \leftarrow W_k Z + b_k 1^\top$      [[Key $\in \mathbb{R}^{d_{\text{attn}} \times \ell_z}$]]
3 $V \leftarrow W_v Z + b_v 1^\top$      [[Value $\in \mathbb{R}^{d_{\text{out}} \times \ell_z}$]]
4 $S \leftarrow K^\top Q$               [[Score $\in \mathbb{R}^{\ell_z \times \ell_x}$]]
5 $\forall t_z, t_x$, if $\neg$Mask$[t_z, t_x]$ then $S[t_z, t_x] \leftarrow -\infty$
6 return $\tilde{V} = V \cdot \text{softmax}(S / \sqrt{d_{\text{attn}}})$
```

Bidirectional/ unmasked self- attention (all tokens as context)
Unidirectional/ masked self- attention (all preceding tokens as context)
Cross attention (attention to each token of primary token sequence, treating second token sequence as context)



```
Algorithm 5: $\tilde{V} \leftarrow$ MHAttention($X, Z | \mathcal{W}$, Mask)
  /* Computes Multi-Head (masked) self-
     or cross- attention layer.        */
  Input: $X \in \mathbb{R}^{d_x \times \ell_x}$, $Z \in \mathbb{R}^{d_z \times \ell_z}$, vector
         representations of primary and
         context sequence.
  Output: $\tilde{V} \in \mathbb{R}^{d_{out} \times \ell_x}$, updated
          representations of tokens in $X$,
          folding in information from
          tokens in $Z$.
  Hyperparameters: $H$, number of
                   attention heads
  Hyperparameters: Mask$\in \{0,1\}^{\ell_z \times \ell_x}$, ↑(3)
  Parameters: $\mathcal{W}$ consisting of
    For $h \in [H]$, $\mathcal{W}^h_{qkv}$ consisting of:
      | $W^h_q \in \mathbb{R}^{d_{attn} \times d_x}$, $b^h_q \in \mathbb{R}^{d_{attn}}$,
      | $W^h_k \in \mathbb{R}^{d_{attn} \times d_z}$, $b^h_k \in \mathbb{R}^{d_{attn}}$,
      | $W^h_v \in \mathbb{R}^{d_{mid} \times d_z}$, $b^h_v \in \mathbb{R}^{d_{mid}}$.
    $W_o \in \mathbb{R}^{d_{out} \times H d_{mid}}$, $b_o \in \mathbb{R}^{d_{out}}$.
1 For $h \in [H]$:
2     $Y^h \leftarrow$ Attention($X, Z | \mathcal{W}^h_{qkv}$, Mask)
3 $Y \leftarrow [Y^1; Y^2; \ldots; Y^H]$
4 return $\tilde{V} = W_o Y + b_o \mathbf{1}^\intercal$
```

Layer norm
Unembedding: convert a vector representation of a token and its context into a distribution over the vocabulary elements (sometimes learned, sometimes fixed)

BERT and GPT: main difference is attention masking

- Different activation
- Layer norms positioned differently

Encoder- decoder (seq2seq transformer) (original): used for machine translation which is why it is more complicated than its successors

1. Context sequence is encoded using bidirectional multi head attention
2. Output of this encoder is a vector representation of each context token, taking into account entire context sequence
3. Primary sequence is then encoded
4. Each token in this sequence uses information from encoded context sequence and primary sequence tokens that precede it

BERT (Encoder only)

- Bidirectional transformer trained on masked language modeling
    - Given text with some tokens masked out, recover the masked out tokens

- Doesn’t use mask parameter but each input token is replaced with probability by dummy token and evaluation is based on reconstruction to probability of tokens

GPT-2, GPT-3 Gopher (Decoder only)

- Given incomplete sentence or paragraph, predict next token (autoregressive language modeling)
- Unidirectional attention instead of bidirectional
- GPT-2 and 3 are similar except 3 is larger and other small differences

Gato (multi domain decoder only transformer)

- Each modality converted into a sequence prediction problem by a separate tokenization and embedding method

Tricks for improving performance

- Data preprocessing: cleaning, augmentation, adding noise, shuffling (besides tokenization and chunking)
- Architecture: sparse layers, weight sharing (besides attention)
- Training: improved optimizers, minibatches, batch normalization, learning rate scheduling, weight initialization, pretraining, ensembling, multi-task, adversarial (besides layer normalization)
- Regularization: weight decay, early stopping, cross-validation, dropout, adding noise
- Inference: scratchpad prompting, few-shot prompting, chain of thought, majority voting