Heinrich Heine
Universität Düsseldorf

Training Transformer Networks

Self-Supervised Training

Which parameters can be adjusted?

- Typical initializations of the learnable parameters:
  - Bias vectors $\vec{b}$ are often initialized to zero
  - Entries of matrices W: small gaussian (or uniformly) distributed values
  - Entries of γ to 1 and entries of β to 0

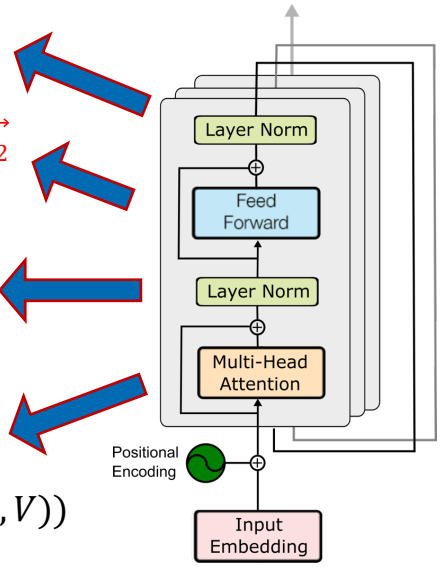
$$\tilde{\gamma} \odot \left(\frac{\vec{e}_i - \mu_i}{\sigma_i}\right) + \tilde{\beta}$$

$$FFN(\vec{e}_i) = W_2 \cdot \mathrm{ReLU}(W_1 \cdot \vec{e}_i + \vec{b_1}) + \vec{b_2}$$

$$\gamma \odot \left(\frac{\vec{e}_i - \mu_i}{\sigma_i}\right) + \beta$$

$$head_j(\vec{e}_i, K, V) = Attention\left(W_E^j \cdot \vec{e}_i, W_K^j \cdot K, W_V^j \cdot V\right)$$

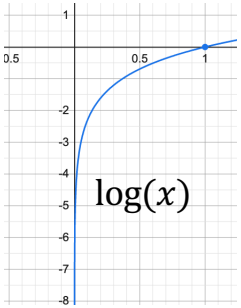
$$MultiHead(\vec{e}_i, K, V) = W_O \cdot Concat(head_1(\vec{e}_i, K, V), \ldots, head_h(\vec{e}_i, K, V))$$



Self-Supervised Training

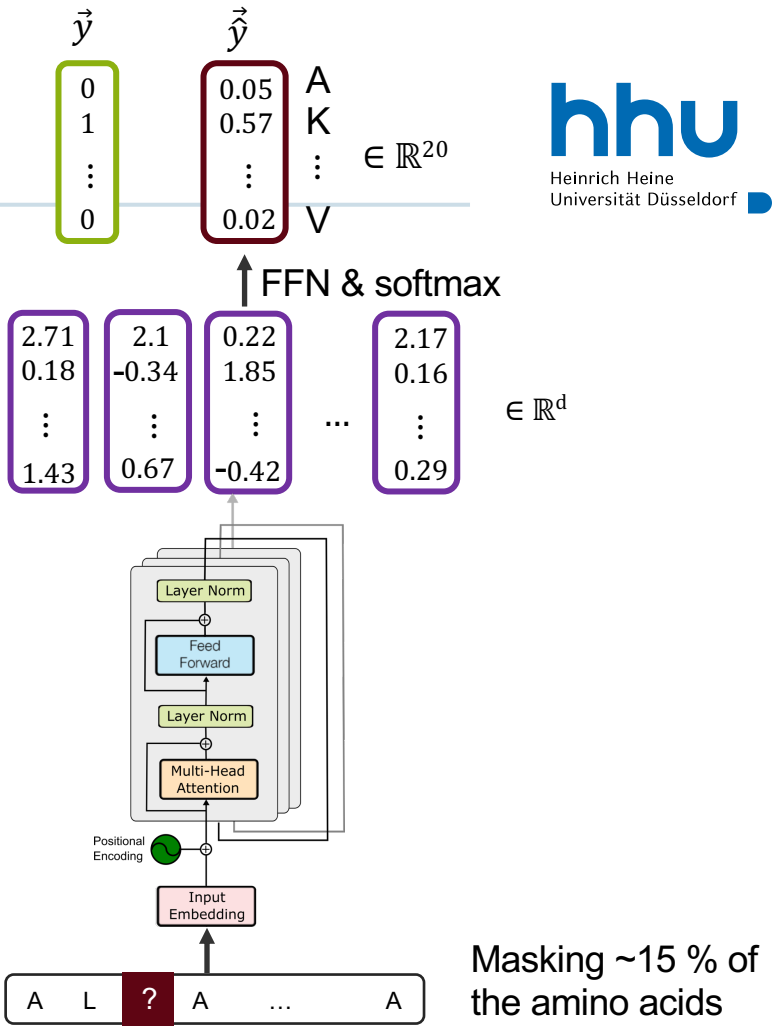
- Categorical cross-entropy:

$$C(\vec{y}, \vec{\hat{y}}) = -\sum_{i=1}^{20} y_i \log(\hat{y}_i)$$
$$= -\log(0.57)$$



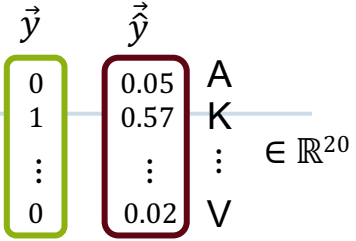
- Taking mean across batch and across all masked positions

- During training:
  - Calculate gradient of loss with respect to learnable parameters
  - Adjust parameters slightly such that the loss is decreasing (direction of negative gradient)



Masking ~15 % of the amino acids

Self-Supervised Training (2)



- Categorical cross-entropy: $C(\vec{y}, \hat{\vec{y}}) = -\sum_{i=1}^{20} y_i \log(\hat{y}_i)$

- Full loss function across a batch with multiple sequences and a total number of M masked tokens:

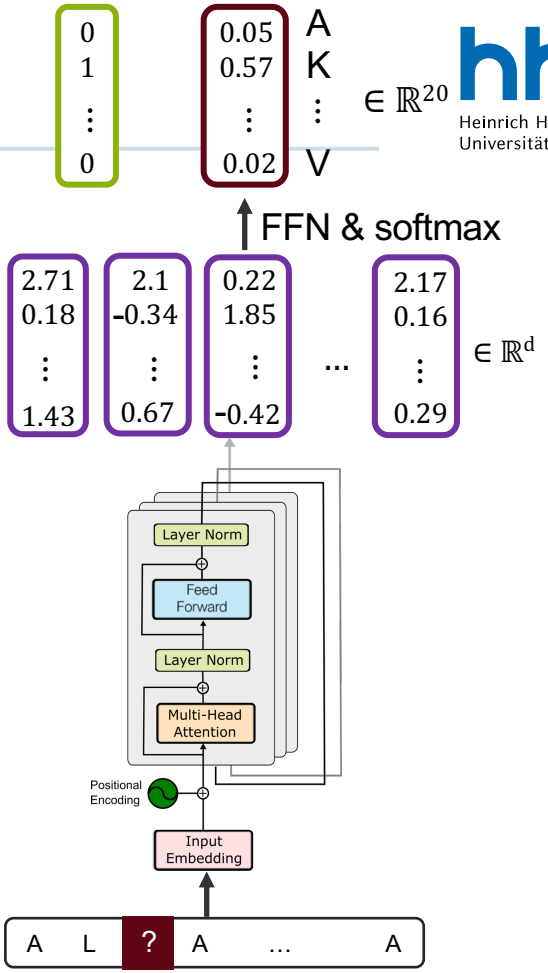
$$L = \frac{1}{M} \sum_{m=1}^{M} C(\vec{y}_m, \overrightarrow{\hat{y}_m})$$

- Updating parameters (using backpropagation):
  - Calculating gradients towards all adjustable parameters w (entries of matrices W, bias vectors b, layer norm param. γ and β) $\frac{\partial L}{\partial w}$
  - Moving in the direction of the negative gradient reduces the loss function:

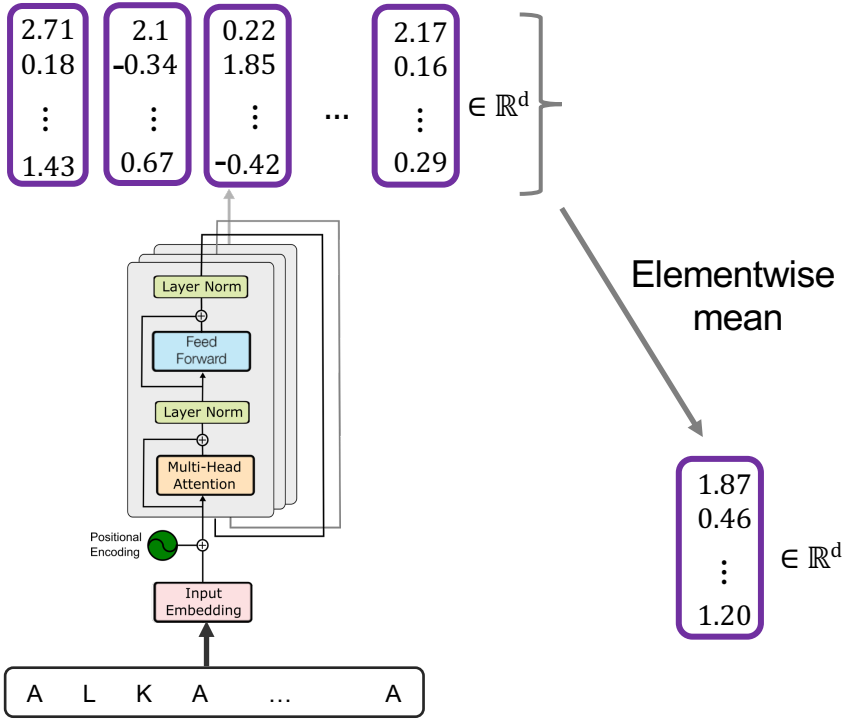
$$w = w - lr \cdot \frac{\partial L}{\partial w}$$

Why Self-Supervised Training?

- The encoder has to extract information from the other amino acids in the sequence
  - Learning context of amino acids
  - Learning dependencies between amino acids
- The encoder not only learns good representations for the masked amino acids, but for all tokens
  - Useful for generating protein representations after training
- The encoder learns implicitly structural and functional insights about the protein



Whole Protein Representations



- Computing element-wise mean across all token representations
  - Aggregating contextual information across the entire sequence
  - Loss of information through averaging
    - Still useful for many prediction tasks
- Alternatively, we can train the encoder for a specific task
  - Results directly in a single protein representation