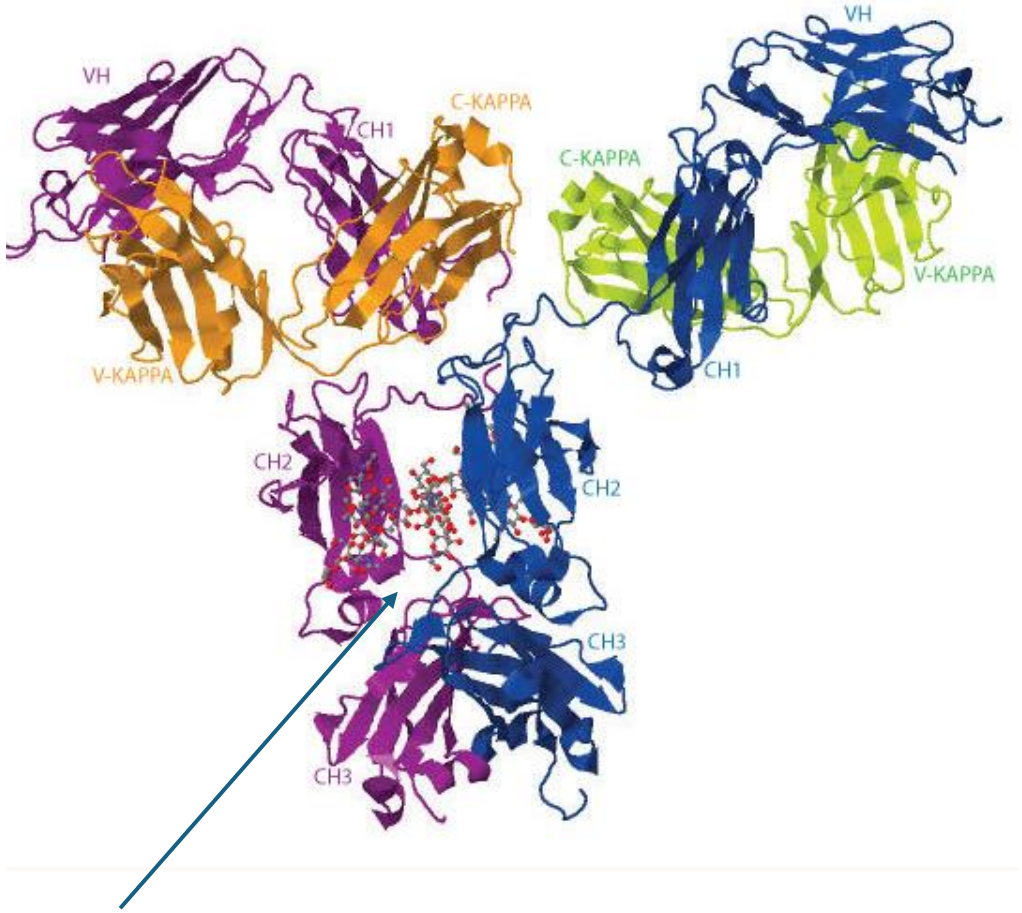
AI Basics for Bioinformatics

AI for Bioinformatics

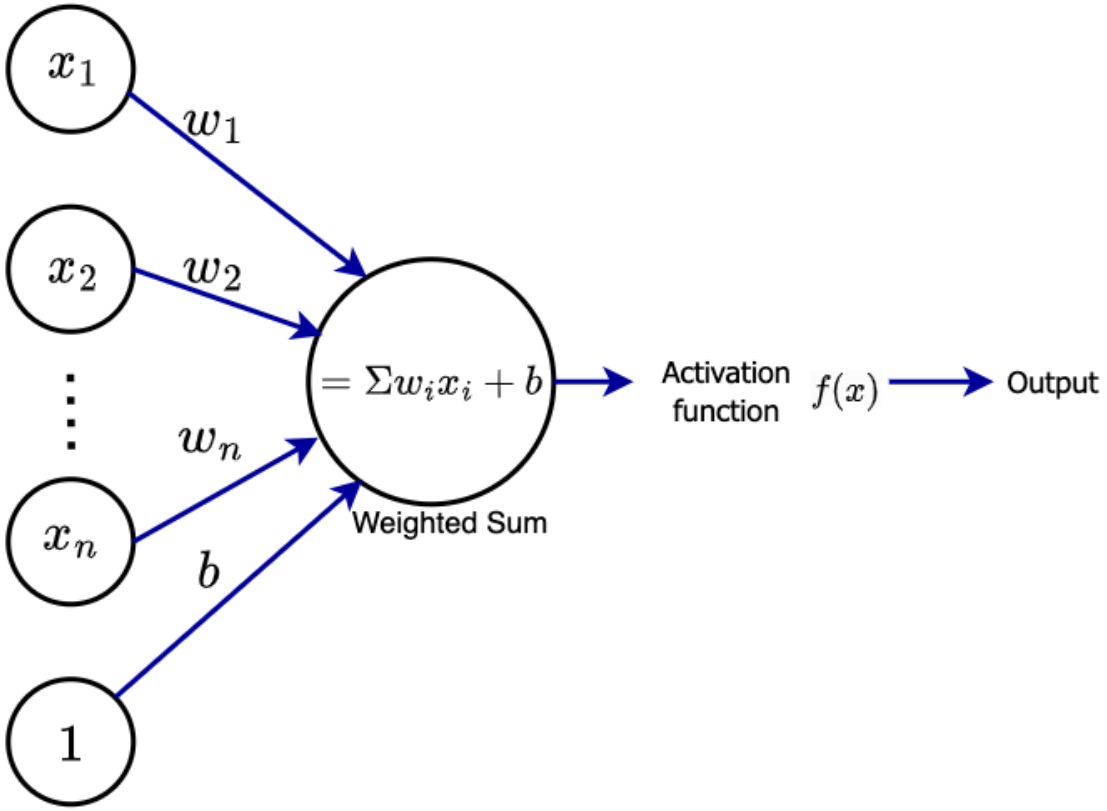
- Traditionally, bioinformatics primarily relies on combinatorial algorithms for solving biological problems.
- AI has been increasingly used in bioinformatics lately.
- We will use some examples to learn the use of AI in bioinformatics.
- This is not a thorough AI course. Our goal is to learn barely enough to use it.
- We will use a toy example: N-glycan motif detection.

N-Glycosylation Motif

- N-Glycosylation is a post-translational motification added to protein.
- Important to functions - about 50% of human proteins are glycosylated.
- It only happens at pattern N-X-S/T, where X is not a P. E.g.
  - MKT**N**RTAA**N**PS**N**FF
  - Only the first N is a potential glycosylation site due to sequence pattern.
- Toy problem: Given a peptide sequence, check whether it contains a N-glycosylation motif.

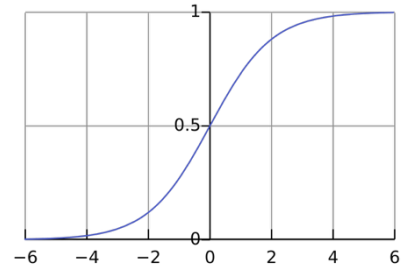


A Neuron



neuron

sigmoid

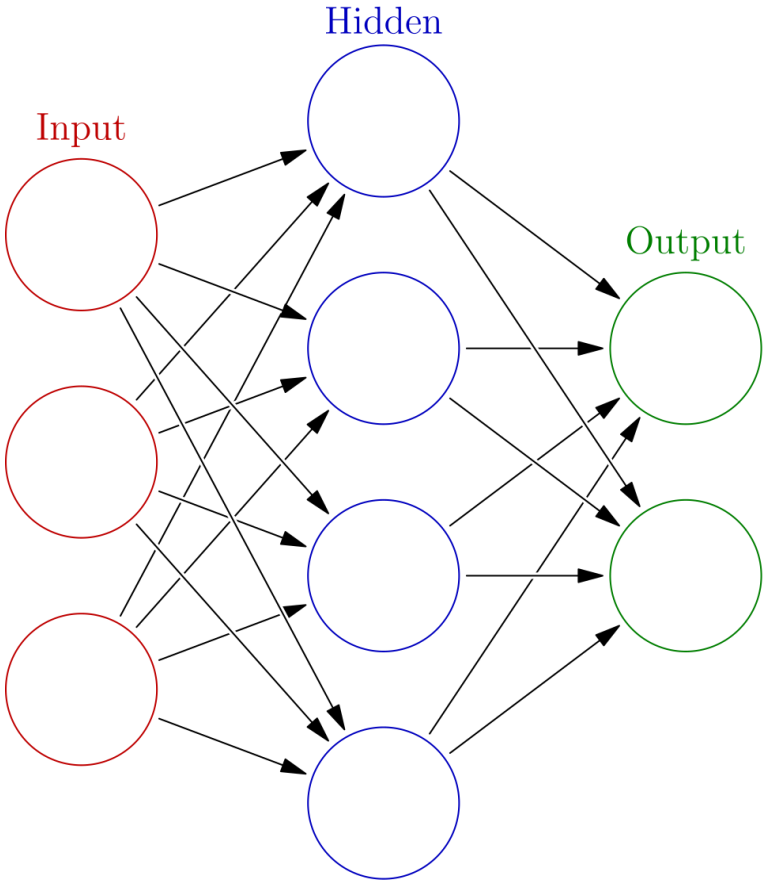


$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

ReLU

$$f(x) = \max(0, x)$$

Neural Network



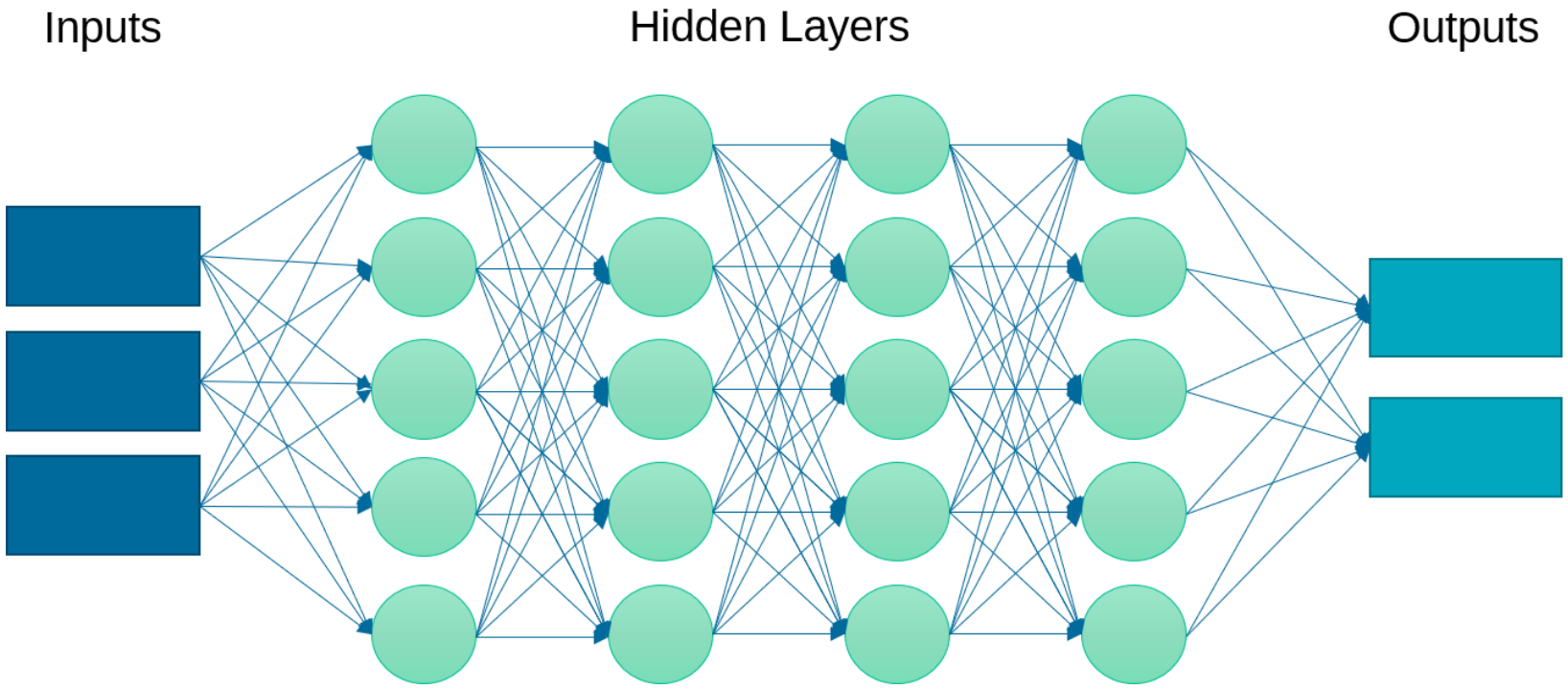
Neural network

Let $z = (z_1, \ldots, z_K)$ be the output values, the **softmax** function converts the K-tuple to a probability

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{i-1}^{K} e^{z_i}}$$

- **Training** uses labeled data to fit the parameters.
- **Inference** makes predictions from input using the learned parameters.
- NN was popularly used around 1990s (1990-2000)
- Exercise: use NN to solve the target/decoy problem.

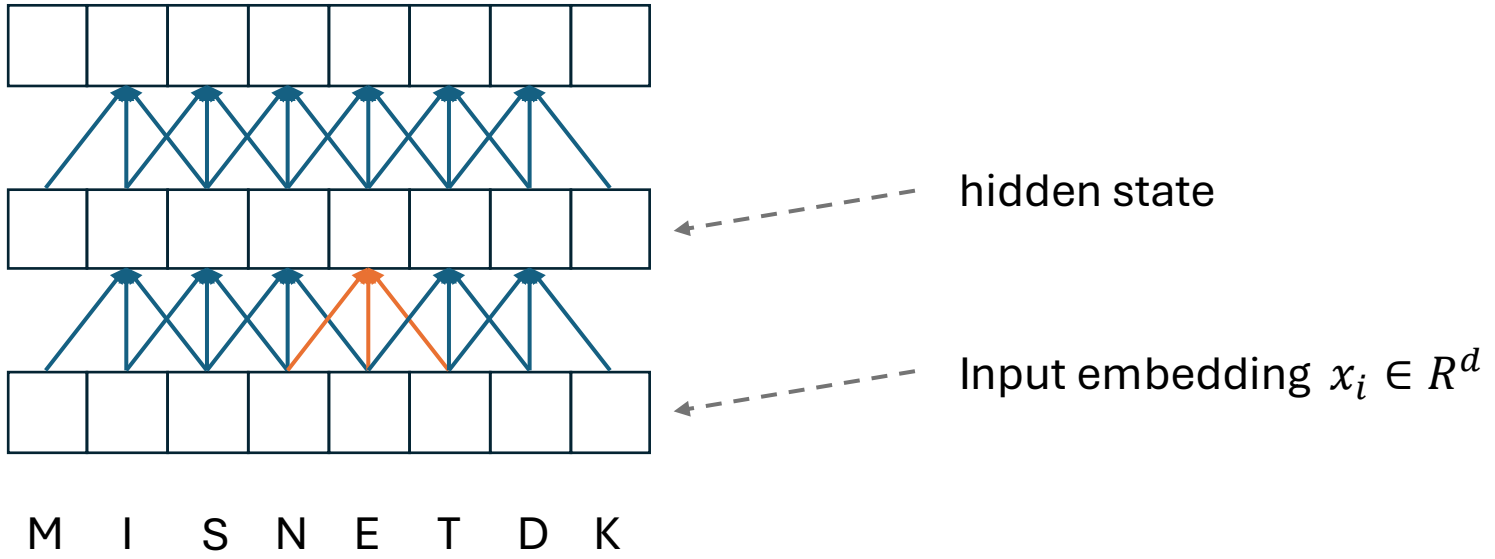
Deep Neural Network



- Essentially DNN uses more layers.
- Often fully connected between layers. Significant increase of number of parameters.
- Hard to train: requires more training data, more compute, and better learning algorithm.
- Studied and used between 2000-2010.

Convolutional Neural Network (CNN)

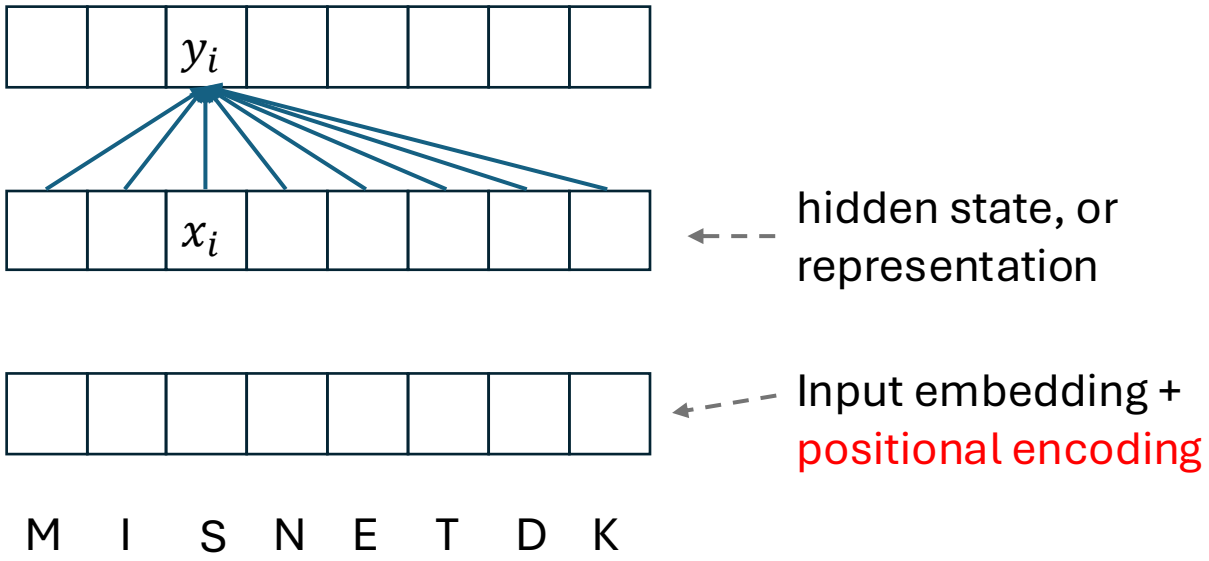
- Core ideas: when input is structured data (e.g. image, sequence), then
  - Some calculation may be based on local context
  - Parameters for local calculation may be reused
- Additional techniques are needed for a successful CNN.
- Mostly studied after 2012 – exemplified by the success of AlexNet.



Transformer

- Threre concepts borrowed from database: Query, Key, Value (QKV).
- These correspond to three matrices W_Q, W_K, W_V.

$$q_i = W_Q x_i$$
$$k_i = W_K x_i$$
$$v_i = W_V x_i$$



$$s_{i,j} = \frac{q_i \cdot k_j}{\sqrt{d_k}}$$

"similarity" between query i and key j.

$$\alpha_{i,j} = softmax_j(s_{i,j})$$

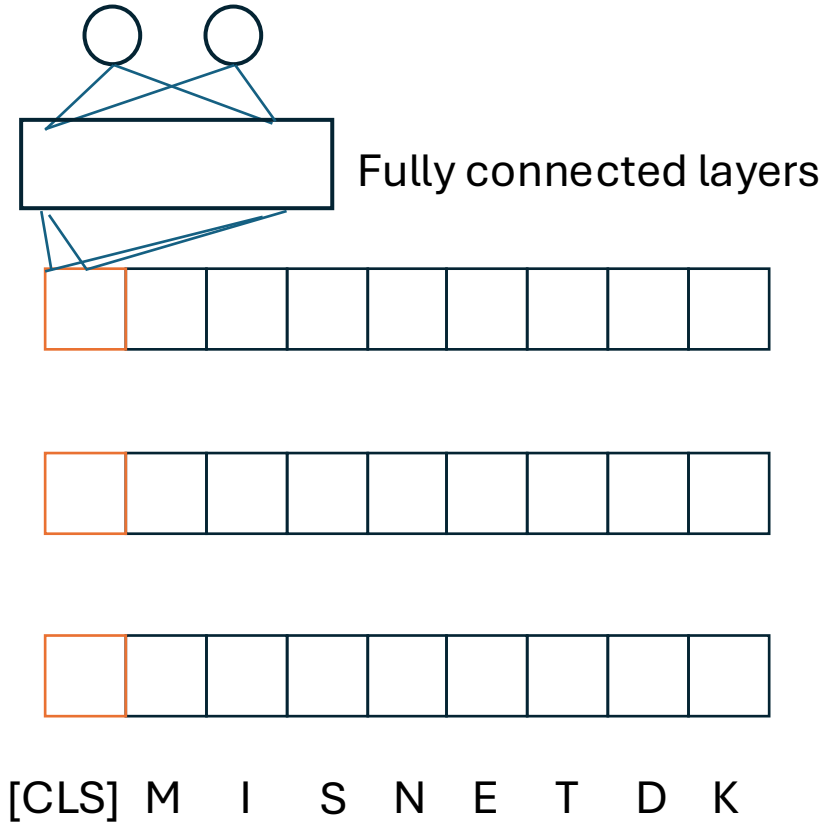
attention weight.

$$y_i = \sum_{j=1}^{n} \alpha_{i,j} v_j$$

update representation with weighted sum of values

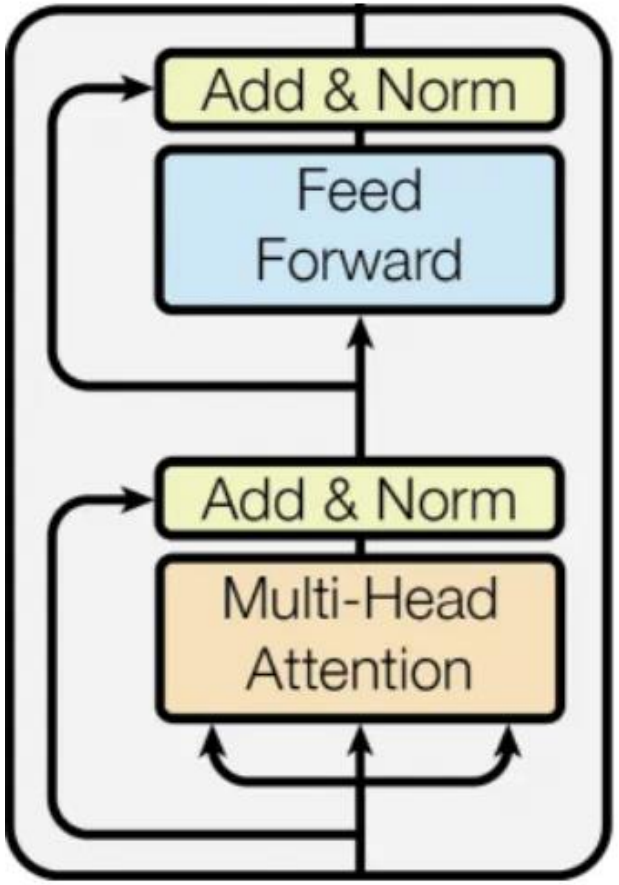
This is called self-attention.

Make Predictions with Transformer



- Often one can make prediction by connecting the [CLS] representation at final layer to a fully connected NN.
- Other ways include pooling the final layer together, such as: concatenation, max pool, average pool, etc..

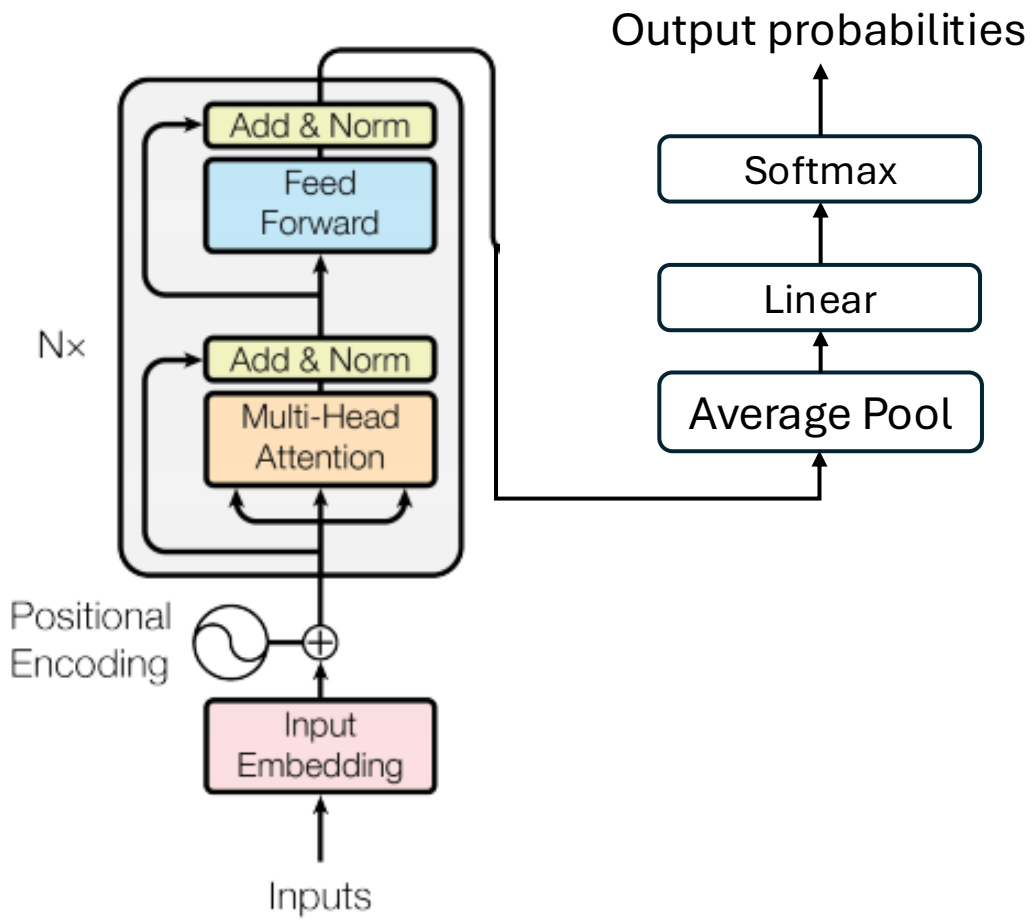
Transformer Encoder Layer



Encoder layer

- There are standard transformer implementations in libraries such as PyTorch and TensorFlow.
- An encoder layer include a few additional concepts/operations:
  - Multi-head attention: each head attend separately then combine results.
  - Add: equivalently, the attention computes the incremental difference (concept of residual neural network).
  - Norm: control the scale of representations.
  - Feed forward: per-token nonlinear projection of the representation.
- These additional operations are necessary for training and inference stability and performance.
- We do not get into details in this course, but treat them as configurable standard components.
- We will only discuss decoder in later lectures. Currently focus on encoder only architecture.

Architecture of Transformer Model for the N-Glycan Motif Problem



Training

- Training data: Input sequences and their labels (0/1).
- Loss function: Use the model to make a prediction, loss measures the deviation between the prediction and the label.
- Learning: A learning algorithm repeatedly uses the training data to adjust the parameters to reduce the overall loss.
- Batch: Each batch contains a limited number of training samples. Each batch is trained together in GPU. Large batch size usually gives faster and more stable training. But GPU memory is a limit.
- Epoch: One round of training using all training data.
- Training-validation split: Use a portion (e.g. 20%) of training data to validate the current model after each epoch. Stop early to avoid overfitting.