Assignment 2

1) A single perceptron computes a weighted sum of its inputs followed by a threshold (or activation) function. Geometrically, this means it can only learn a linear decision boundary in two dimensions: a straight line that seperates the input space into two classes.

The XOR problem is not linearly seperable. If we plot the four ~~output~~ input points (0,0), (0,1), (1,0), (1,1), the points labelled 1 lie in the opposite direction. No straight line can seperate the 1's from 0's. Thus, no matter how weights bias of perceptron are chosen, it doesn't work.

With MLPs, the hidden layers allow the network to learn ~~immediate layers and non~~ - intermediate representations of the input. This results in non-linear decision boundries.

2) Stacking linear layers without non-linear activations still results in a single linear transformations, because the composition of linear functions is still itself linear and can be collapsed into one matrix multiplication.
Gradients shrink in deep networks due to repeated multiplication by small derivatives during backpropagation, causing vanishing gradients. ReLU alleviates this because its derivative is 1 for positive inputs, allowing gradients to flow without shrinking, unlike sigmoid whose derivatives are always less than 1

3) Position encoding is necessary cause self attention is order-agnostic, without position information, a transformer cannot distinguish between different word orders.
Absolute positional encodings assign a fixed position vector to each token index, while sinusoidal encodings use deterministic sine-cosine functions that generalise to unseen lengths. ROPE (rotatory embeddings) encodes position by rotating query and key vectors preserving relative position information and enabling better extrapolation and stability for longer contexts.

4) In attention, the query represents what each token and the value is the actual information passed forward. Attention scores are scaled by $\sqrt{d_k}$ to prevent large dot products that would push softmax into saturation and shrink gradients. Diagonal values are usually the highest because a token is most similar to itself, so it attends most strongly to its own position.

5) We split the model into multiple heads so that the model can attend to different types of relationships in parallel, instead of forcing one attention mechanism to capture everything. Each head learns a different projection of the same input making attention more expressive.

- d_{model} : total embedding dimension of the model.
- h : number of attention heads
- d_{head} : dimension per head, where

$$d_{head} = \frac{d_{model}}{h}.$$

Solving

1). $d_{model} = 768$, $h = 12$

(a) $d_{head} = 768/12 = 64$

(b) Parameters for Q, K, V (one layer).

Each matrix: 768×768

Total: $3 \times 768 \times 768 = 1769472$ parameters

2) Softmax.

Input: $[2, 1, 0]$.

Steps: $[7.39, 2.72, 1.0]$

sum: 11.11.

Final softmax: $[0.665, 0.245, 0.090]$