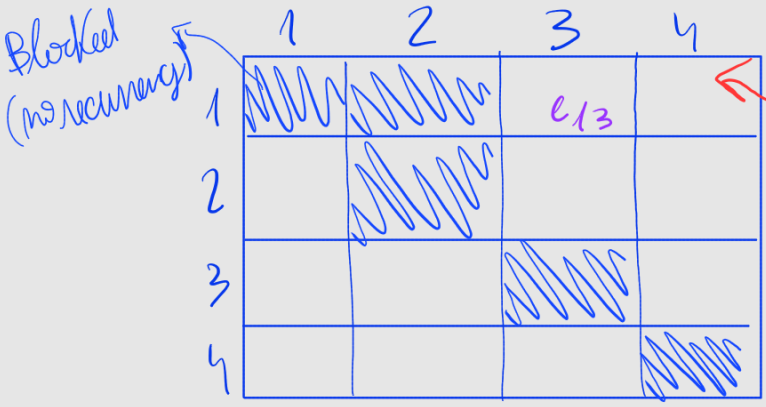
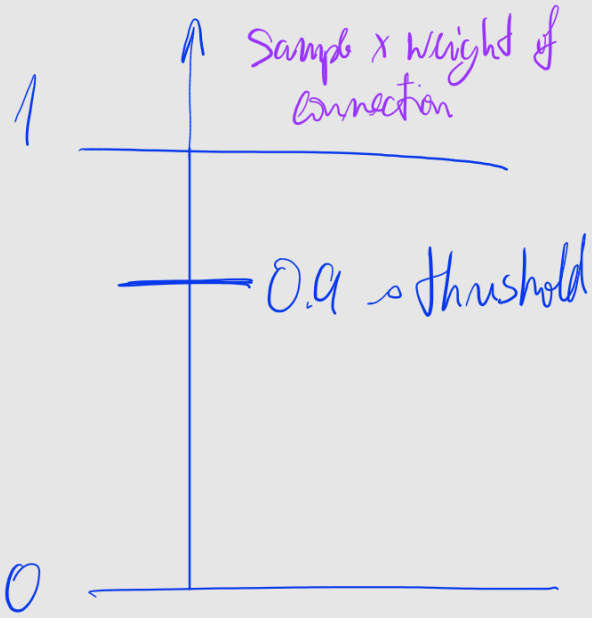
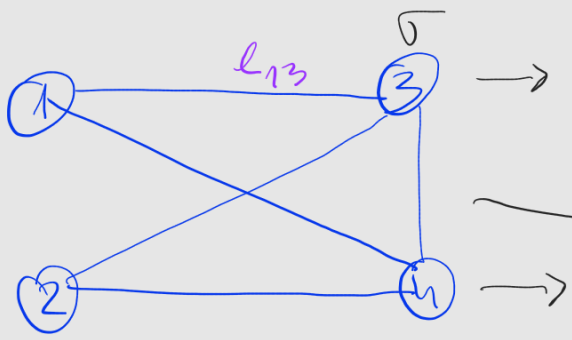
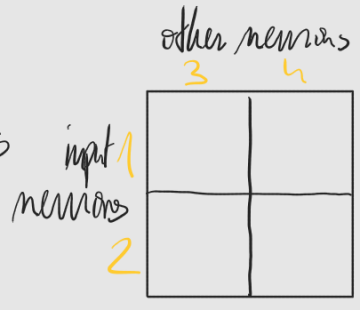






Train connections by training attention weights of connections

- Weights/biases of MLP don't change! → fixed
  - ↳ Attention matrices for connections

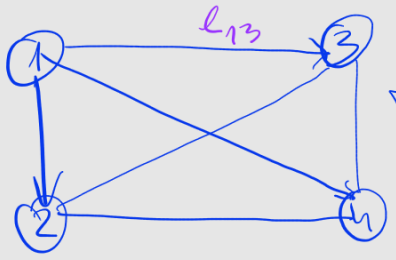


- We need to train our attention head
  - Metric → weight of connection × input

Input embeddings would be the "weightened connections"

↳ $W^{13} \cdot x_1^{(1)}$ ← task 1

→ Rows need to be size of network



feed forward no backwards connections

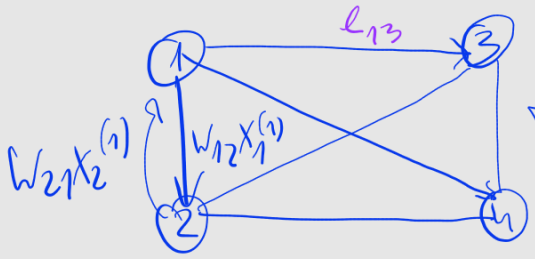
Continued Attention

Input embedding: X

$T_1 : S_1, S_2, \dots$

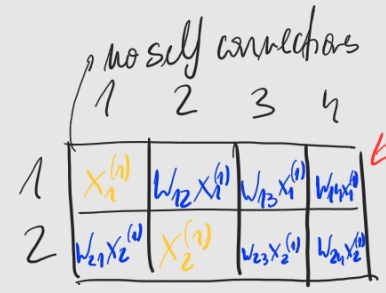
↳ Task ↳ Sample

→ Rows need to be size of network



feed forward no backwards connections

neurons



Signature of Task 1 Sample 1

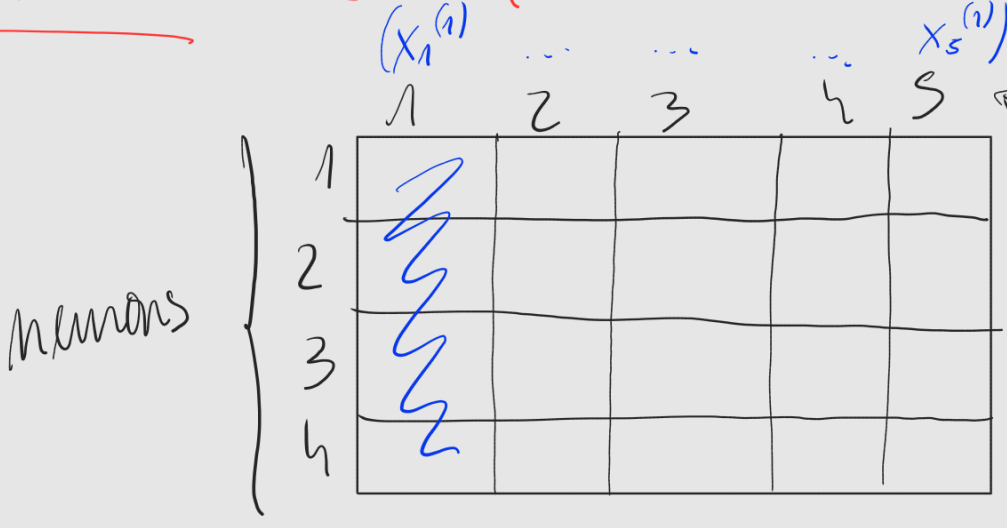
$X = 2 \times 4$

NOTE: We could allow self connections

or

bidirectional connections either { between layers / in-between layer

Attention head $Q = W_q$



$$Q = X * W_q$$

$$(2 \times 4) \cdot (4 \times N) = (2 \times N)$$

Attention weight

$Q = X * W_q$

$K = X * W_k$

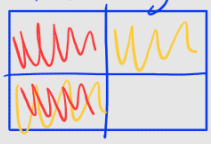
$(2 \times N)\ (2 \times 4)\ (4 \times N)$

Attention weights

$Q K^T$

$(2 \times N)(N \times 2) \rightarrow 2 \times 2$

Attention weights



T_1

T_2

} "trace of task i"

mask

Every time we see the same sample, we will see the same mask.

Values

- Distinguish cars & cats
  - Attention weights gives us tasks
  - 2^4 possible activation patterns (1/0 for 4 cells)
- Multiply: attention weights × values

$W_v \rightarrow 4 \times N_v$

$V = X * W_v$

$= (2 \times 4) \cdot (4 \times N_v)$

$= 2 \cdot N_v$

N_v → # classes × # tasks? (or) Number of classes within the task!?

We just need to select N_v suitable for the largest number of classes within ALL tasks (zero padding)

⚠ Define loss for attention scores carefully!

↳ Classification loss whose output is a GRAPH or vectorize it

↳ CROSS-ENTROPY loss!