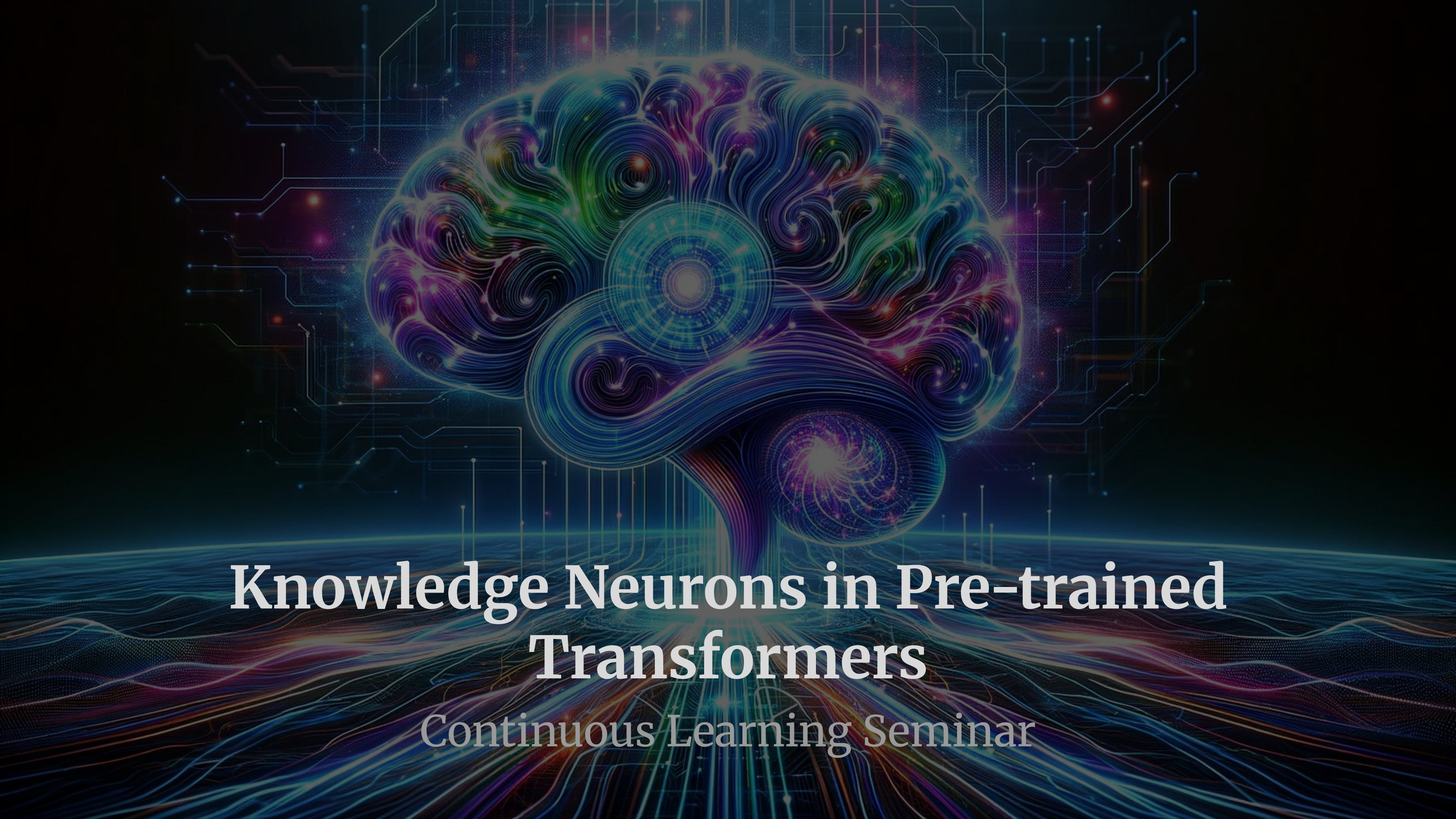
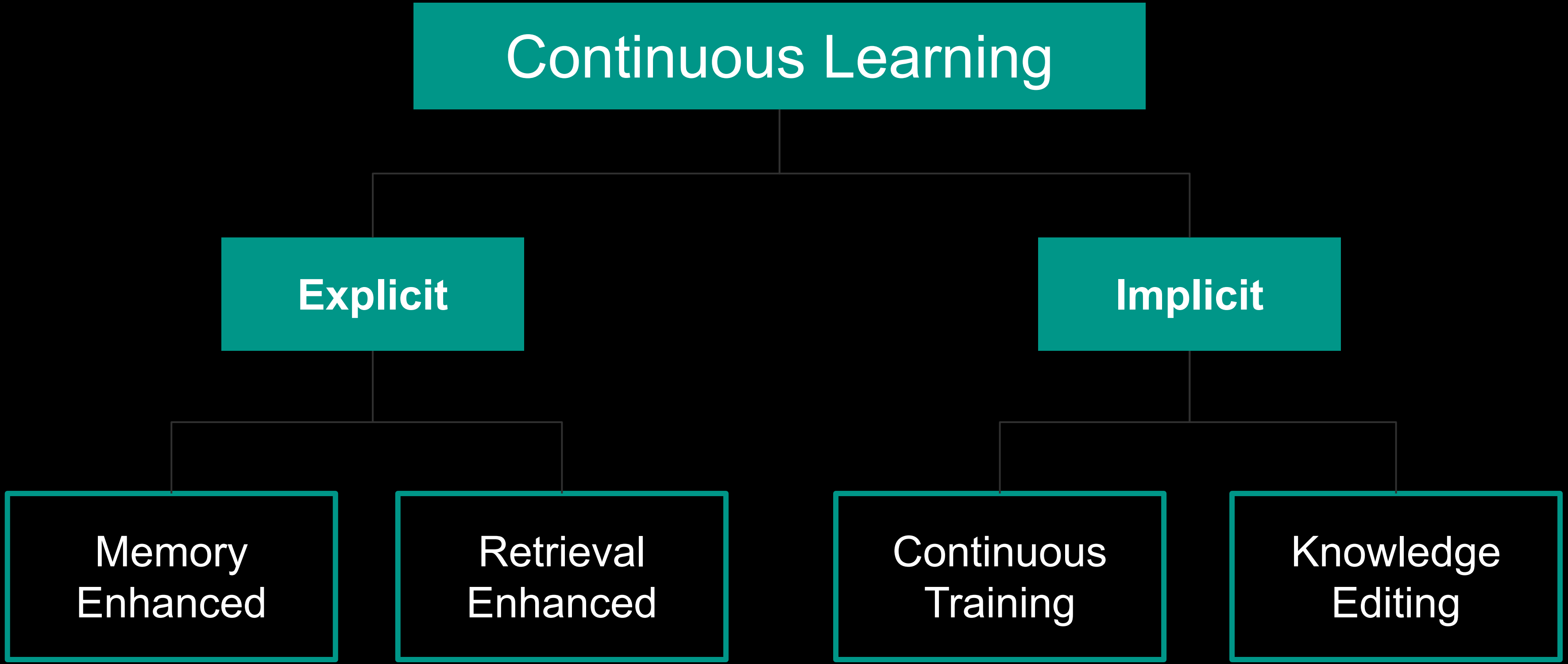
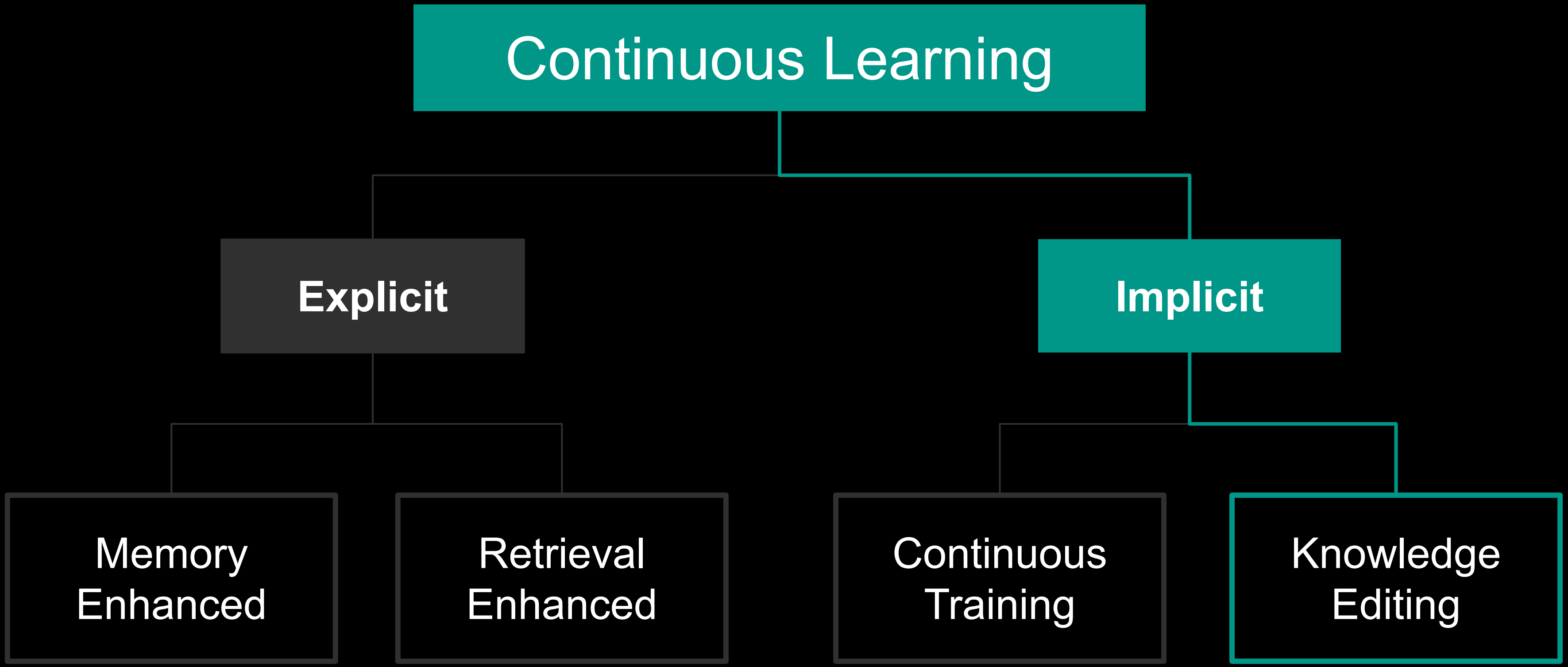
Knowledge Neurons in Pre-trained Transformers

Continuous Learning Seminar



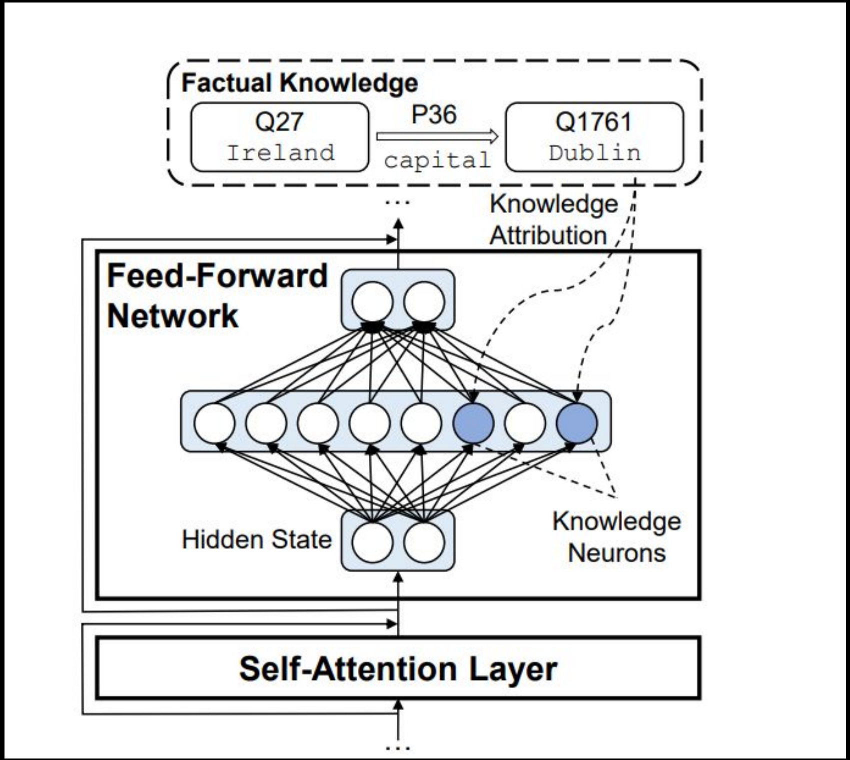


Zhang et al, “How Do Large Language Models Capture the Ever-changing World Knowledge? A Review of Recent Advances” (2023)



Zhang et al, “How Do Large Language Models Capture the Ever-changing World Knowledge? A Review of Recent Advances” (2023)

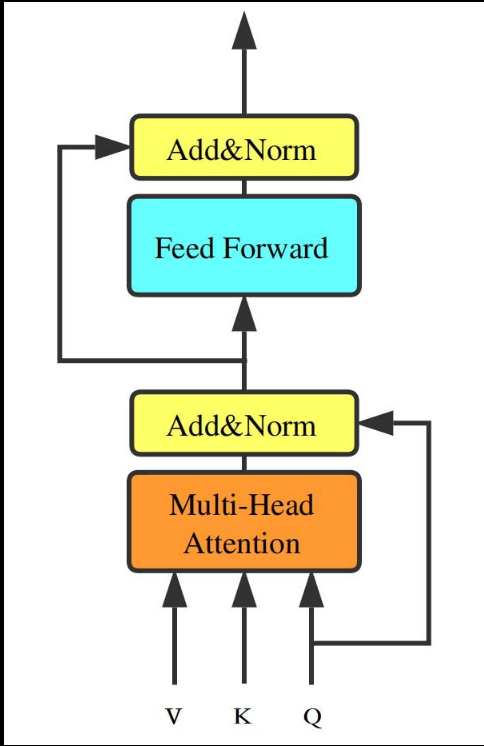
Knowledge Neurons in Pre-Trained Transformers



- Knowledge attribution method to identify the neurons that express a specific fact.
- Leverage this “knowledge neurons” (KN) to edit (such as update and erase) specific factual knowledge without fine-tuning.

Dai et al, “Knowledge Neurons in Pretrained Transformers” (2022)

Knowledge Neurons in Pre-Trained Transformers



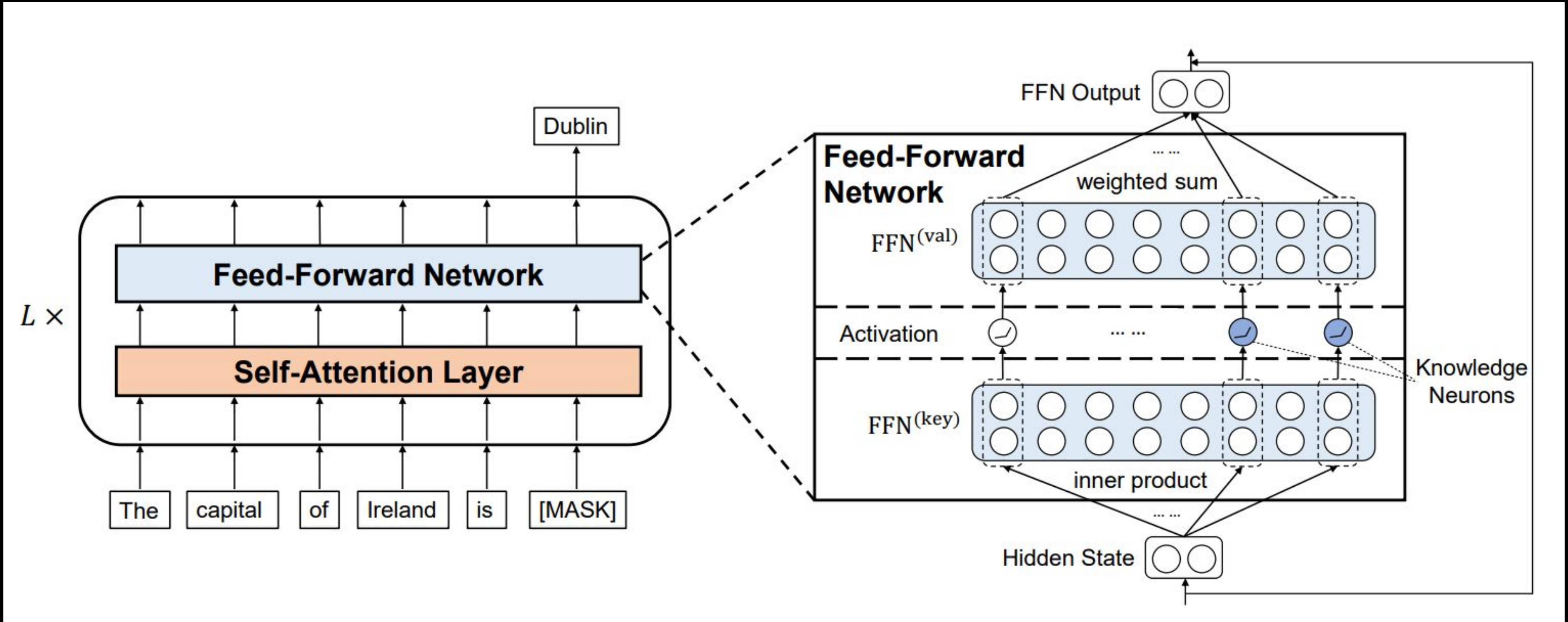
$$Q_h = XW_h^Q, K_h = XW_h^K, V_h = XW_h^V, \quad (1)$$

$$\text{Self-Att}_h(X) = \text{softmax}\left(Q_h K_h^T\right) V_h, \quad (2)$$

$$\text{FFN}(H) = \text{gelu}\left(HW_1\right) W_2, \quad (3)$$

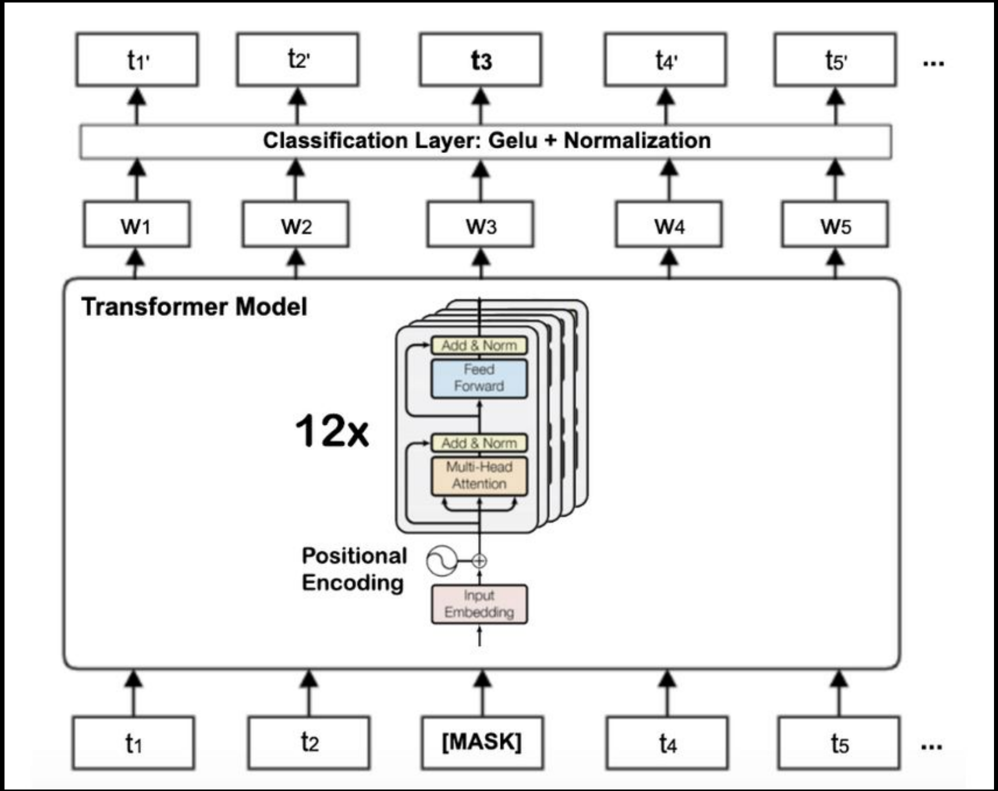
Dai et al, “Knowledge Neurons in Pretrained Transformers” (2022)

FFNs in Transformers as key-value memories



Dai et al, “Knowledge Neurons in Pretrained Transformers” (2022)

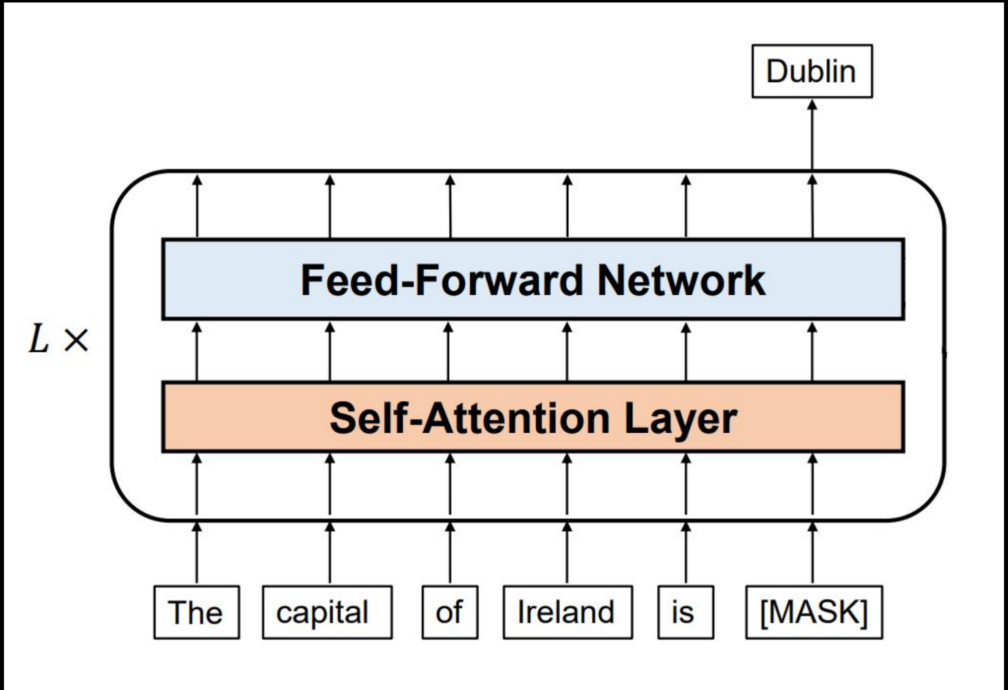
Remembering BERT's Architecture



- BERT Pre-trained Transformer: 12 layers, 768 model's dimension, 3072 FFN hidden size.
- Masked Language Model.

Devlin et al, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" (2018)

How do they assess knowledge?



- Fill-in-the-blank cloze task.
- Each relational fact is in the form of a triplet <h, r, t>.
- Query the model with *knowledge-expressing prompt*:

<Ireland, capital, Dublin>

"The capital of Ireland is __"

Dai et al, "Knowledge Neurons in Pretrained Transformers" (2022)

Example of a dataset instance (ParaRel)

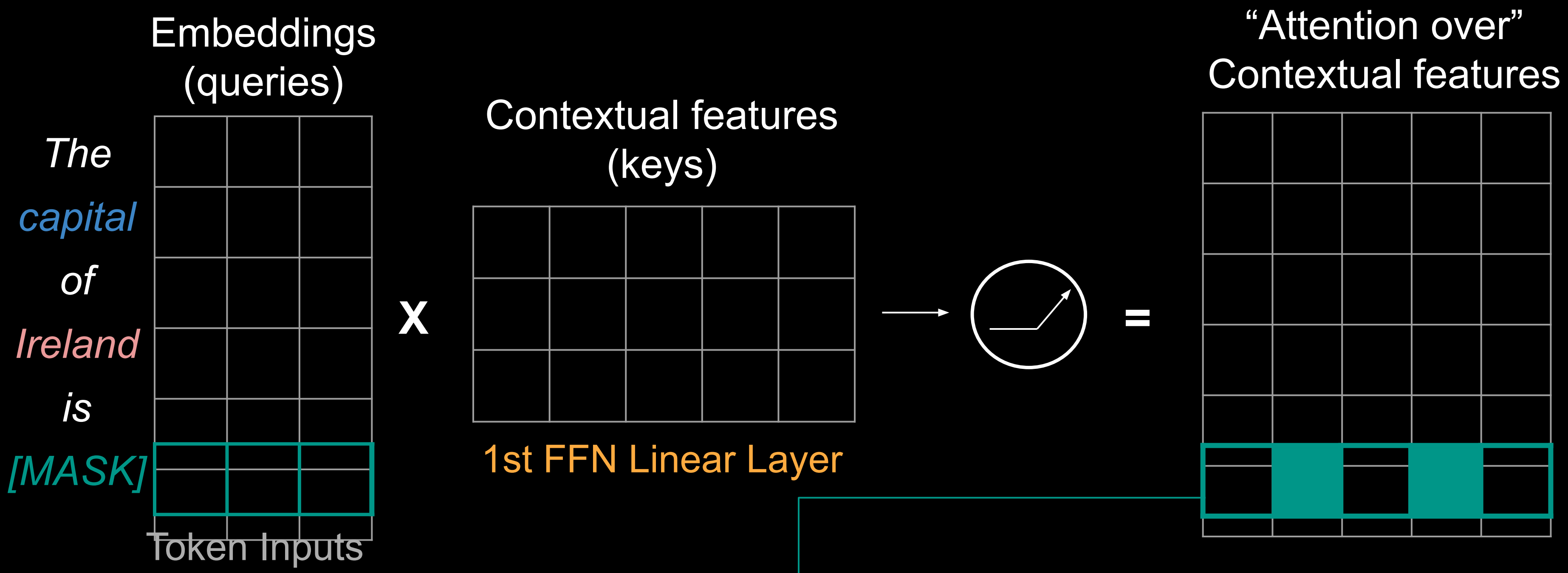
data > PARAREL > {} data_all_allbags.json > [ ] P101 > [ ] 0


```
{
  "P101": [
    [
      [
        "Alan Turing works in the field of [MASK].",
        "logic",
        "P101(field of work)"
      ],
      [
        "Alan Turing specializes in [MASK].",
        "logic",
        "P101(field of work)"
      ],
      [
        "[MASK] is the specialization of Alan Turing.",
        "logic",
        "P101(field of work)"
      ],
      [
        "The expertise of Alan Turing is [MASK].",
        "logic",
        "P101(field of work)"
      ],
      [
        "[MASK] is the expertise of Alan Turing.",
        "logic",
        "P101(field of work)"
      ],
      [
        "The domain of activity of Alan Turing is [MASK].",
        "logic",
        "P101(field of work)"
```


data > PARAREL > {} data_all_allbags.json > [ ] P101 > [ ] 1


```
        "Alan Turing works in the area of [MASK].",
        "logic",
        "P101(field of work)"
      ]
    ],
    [
      [
        "John Vincent Atanasoff works in the field of [MASK].",
        "mathematics",
        "P101(field of work)"
      ],
      [
        "John Vincent Atanasoff specializes in [MASK].",
        "mathematics",
        "P101(field of work)"
      ],
      [
        "[MASK] is the specialization of John Vincent Atanasoff.",
        "mathematics",
        "P101(field of work)"
      ],
      [
        "The expertise of John Vincent Atanasoff is [MASK].",
        "mathematics",
        "P101(field of work)"
      ],
      [
        "[MASK] is the expertise of John Vincent Atanasoff.",
        "mathematics",
        "P101(field of work)"
      ],
      [
```

How do they attribute knowledge to neurons?



Change these neurons for the [MASK] token and calculate how much that affects the output probability of the target label ("Dublin").

Knowledge Attribution Method

1. Gradually change neurons' "weights" and integrate the gradients. This will accumulate the output probability change caused by the change.

$$\tilde{\mathrm{Attr}}(w_i^{(l)}) = \frac{\overline{w}_i^{(l)}}{m} \sum_{k=1}^{m} \frac{\partial\, \mathrm{P}_x\left(\frac{k}{m}\overline{w}_i^{(l)}\right)}{\partial w_i^{(l)}}$$

2. Refine KN, by filter out False Positives.

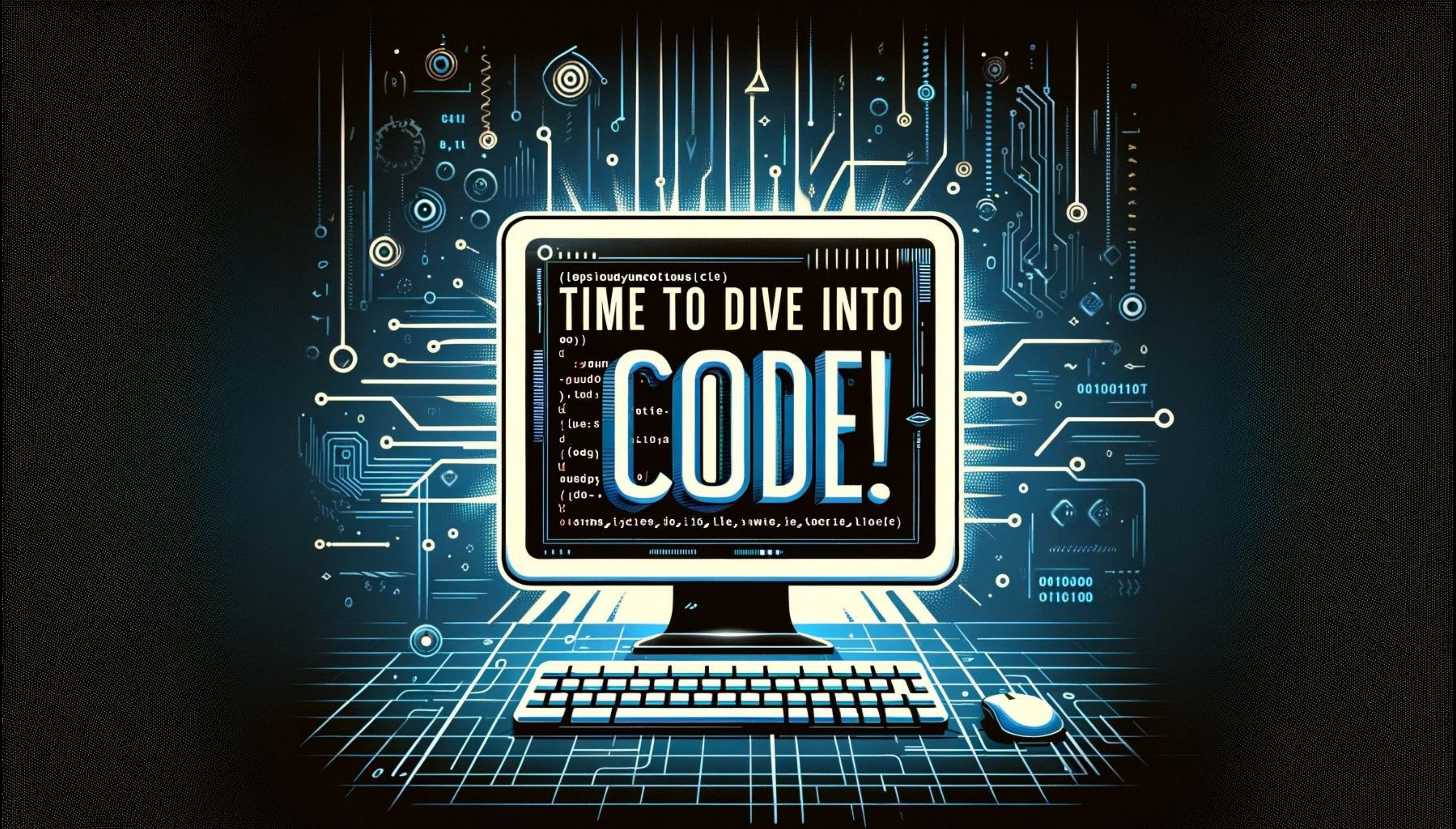
Dai et al, "Knowledge Neurons in Pretrained Transformers" (2022)

Algorithm for identifying KNs

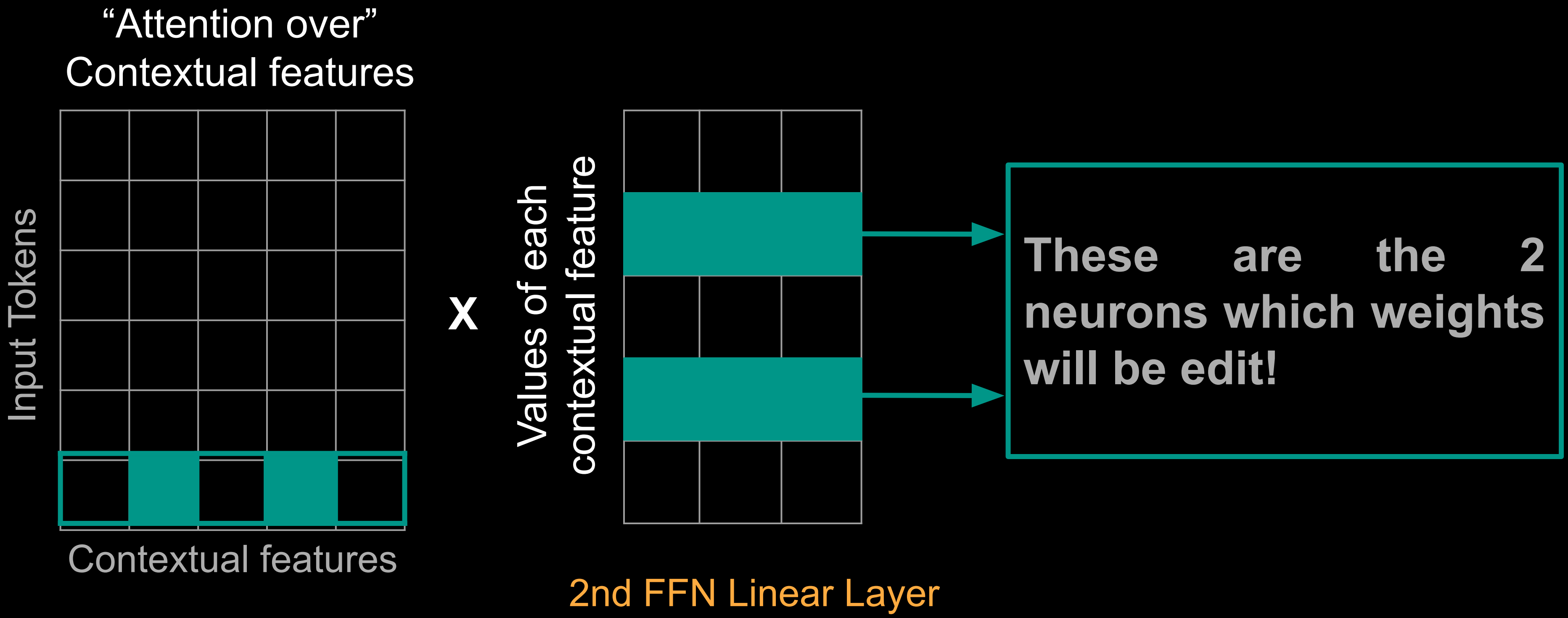
Given a relational fact:

1. Produce *n* diverse prompts;
2. For each prompt, calculate the knowledge attribution scores of neurons;
3. For each prompt, retain the neurons with attribution scores greater than the attribution threshold t, obtaining the coarse set of knowledge neurons;
4. Considering all the coarse sets together, retain the knowledge neurons shared by more than p% prompts.

Dai et al, “Knowledge Neurons in Pretrained Transformers” (2022)


TIME TO DIVE INTO
CODE!

Edit Knowledge Task



Results and Conclusions

1. On average: 4.13 KNs for each relational fact using the knowledge attribution method and 3.96 for the baseline method.

2. Identified KN by this method tend to notably affect knowledge expression: suppressing them decreases the correct output probability, while enhancing them increasis it.

3. KNs can be used to edit or erase specific relational facts.

4. We can manipulate the model's prediction for a specific relational fact by just tweaking the values of a few neurons.

Dai et al, "Knowledge Neurons in Pretrained Transformers" (2022)

Problems

1. Changing KNs' values for one specific fact affects the prediction probability of other facts.

2. Needs data and diverse examples of the same fact to be able to infer the KN.

3. Manually and directly changing the KN value requires hyperparameter tuning and may be forcing the model towards some target.

4. Not generalizable to multiple types of knowledge.

5. Specific task and prompt style.

6. Says nothing about the interactions between KNs.

Dai et al, "Knowledge Neurons in Pretrained Transformers" (2022)

Possible Solutions?

1. Train a hypernetwork to learn shifts for these KNs.

2. Explore the representation meaning of these "contextual features" (How?).

3. …