Foundation Models for Causal Inference

Federico Baldo

The Third IPLESP Workshop on Causal Inference

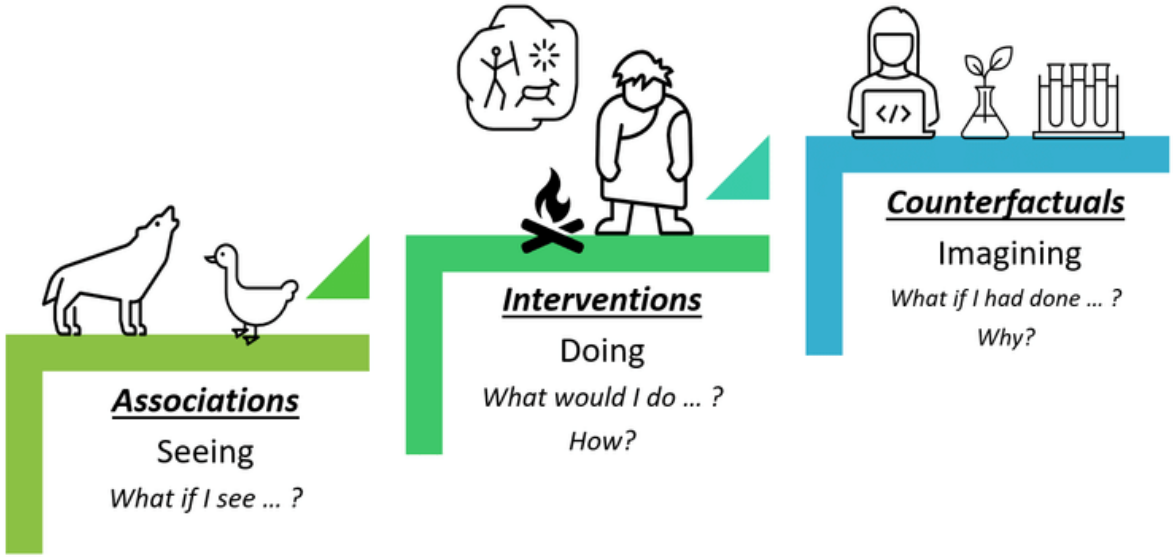
June 2026

What do you know?

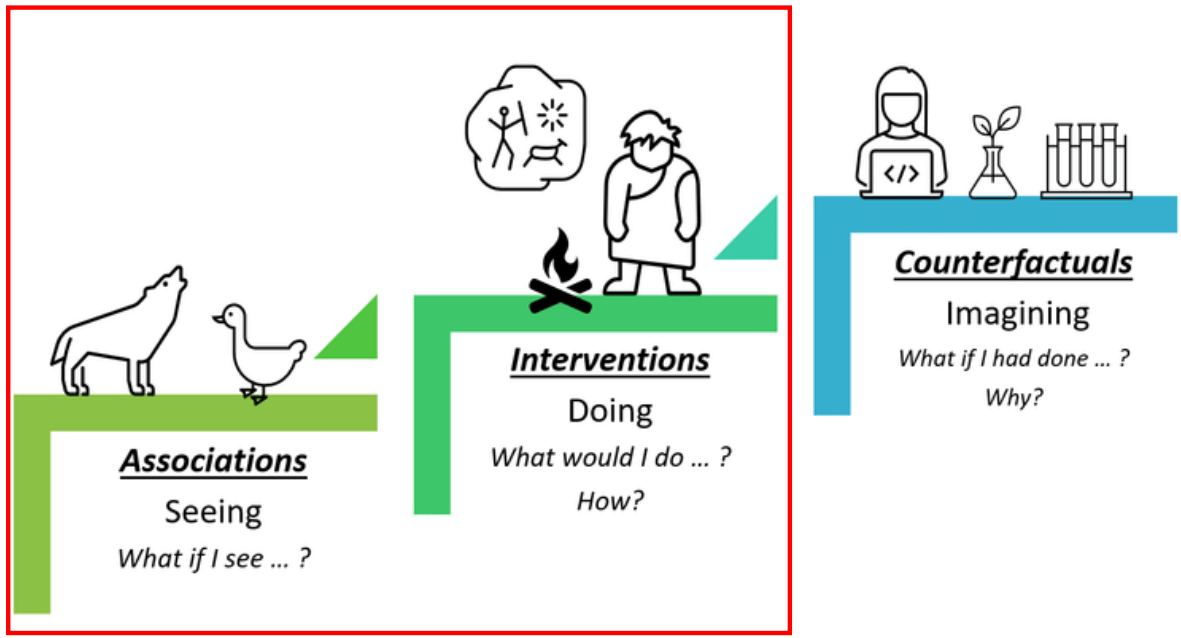
Do you know...

- What is a Neural Network? How it works?
- Transformers? Self-attention?
- What is a Foundation Model?
- TabPFN and how it works?

The Ladder of Causation



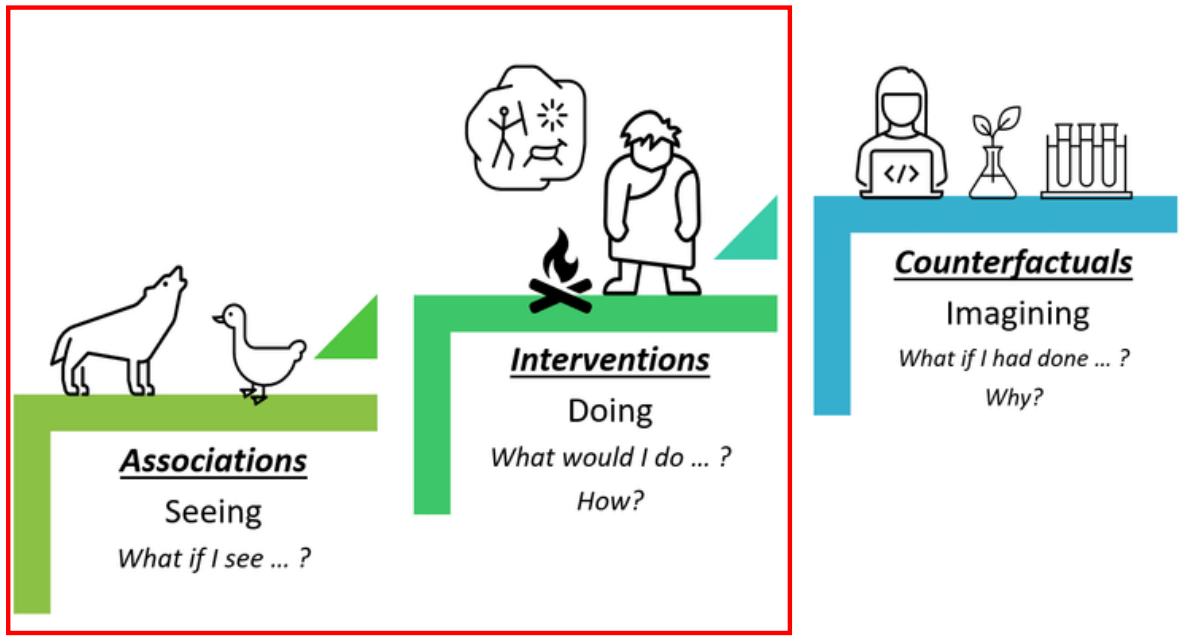
The Ladder of Causation



To climb the ladder:

- Assumptions: Causal DAG, Unconfoundedness.
- In practice, these assumptions are strong and often unverifiable.

The Ladder of Causation



To climb the ladder:

- Assumptions: Causal DAG, Unconfoundedness.
- In practice, these assumptions are strong and often unverifiable.

What if we cannot observe the DAG and we violate unconfoundedness?

Inserm

iPLesp

SANTÉ SORBONNE UNIVERSITÉ

1

Deep Learning

Problem Formulation: Machine Learning

Given data $\mathbb{D}$, find a function f_w that captures some structure of $p(\mathbb{D})$.

Problem Formulation: Machine Learning

Given data $\mathbb{D}$, find a function f_w that captures some structure of $p(\mathbb{D})$.

The structure we seek depends on the problem:

- **Supervised learning**: $\mathbb{D} = \{(x_i, y_i)\}_{i=1}^{N}$, learn $f_w : \mathcal{X} \to \mathcal{Y}$ to predict y from x

Problem Formulation: Machine Learning

Given data $\mathbb{D}$, find a function f_w that captures some structure of $p(\mathbb{D})$.

The structure we seek depends on the problem:

- **Supervised learning**: $\mathbb{D} = \{(x_i, y_i)\}_{i=1}^N$, learn $f_w : \mathcal{X} \to \mathcal{Y}$ to predict y from x
- **Unsupervised learning**: $\mathbb{D} = \{x_i\}_{i=1}^N$, learn structure in $p(x)$ (clustering, density estimation)

Problem Formulation: Machine Learning

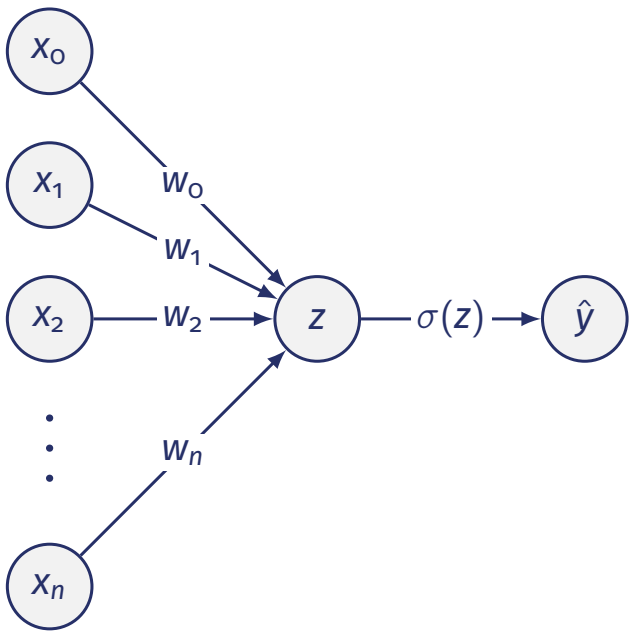
Given data $\mathbb{D}$, find a function f_w that captures some structure of $p(\mathbb{D})$.

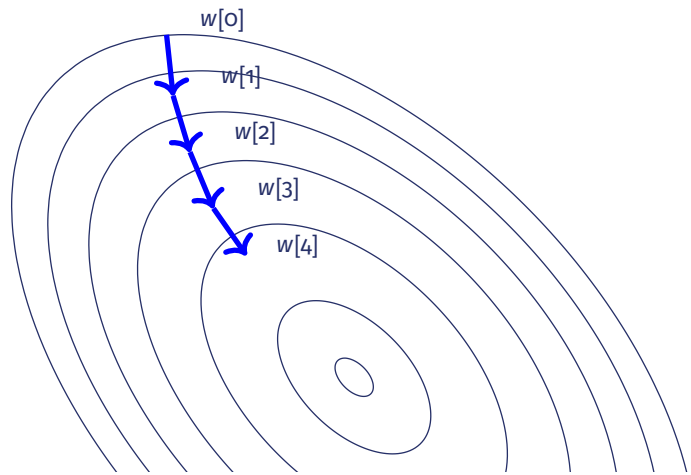
The structure we seek depends on the problem:

- **Supervised learning**: $\mathbb{D} = \{(x_i, y_i)\}_{i=1}^N$, learn $f_w : \mathcal{X} \to \mathcal{Y}$ to predict y from x
- **Unsupervised learning**: $\mathbb{D} = \{x_i\}_{i=1}^N$, learn structure in $p(x)$ (clustering, density estimation)
- **Self-supervised learning**: labels derived from x itself, learn rich representations without human annotation

Perceptron

In 1958, Frank Rosenblatt introduced the Perceptron, one of the first models of an artificial neuron.



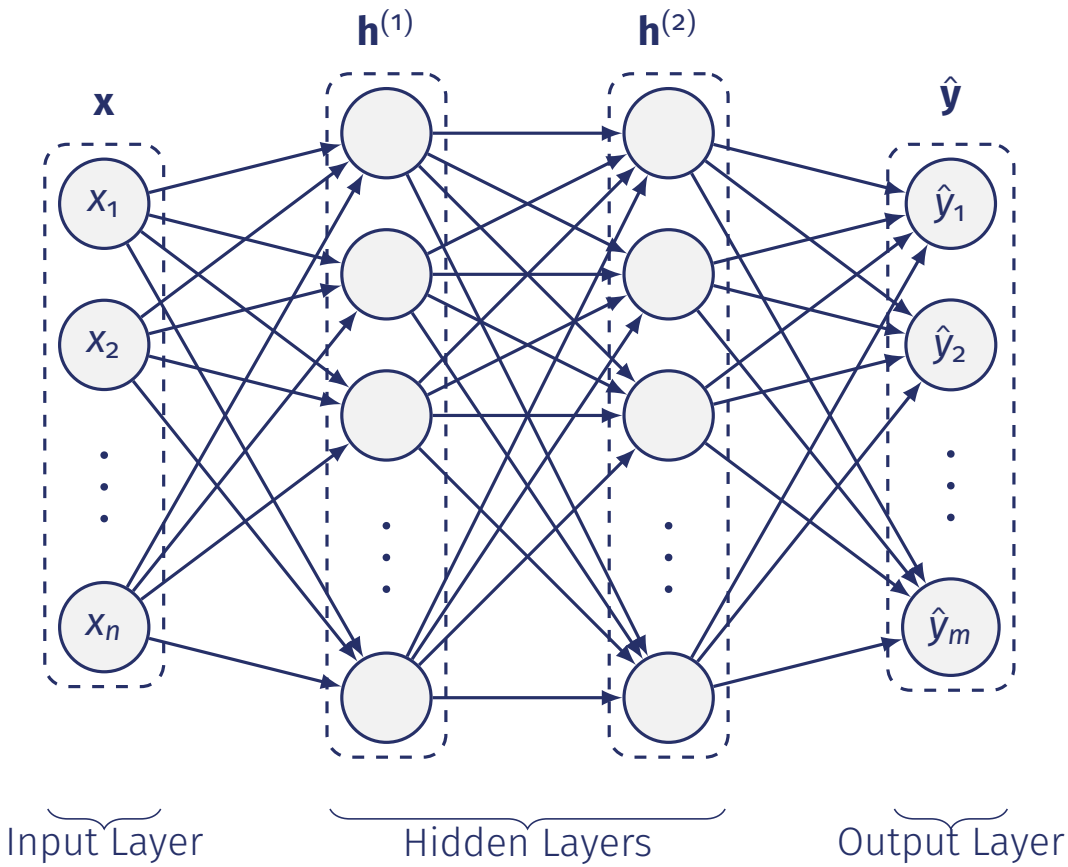


$$z = \mathbf{w}^\top \mathbf{x} + b$$

$$w \leftarrow w - \mu \nabla_w \mathcal{L}$$

Artificial Neural Networks

Artificial Neural Networks (ANNs) are organized in layers of interconnected nodes, inspired by the structure of the human brain.



Universal Approximation Theorem

Neural networks are **universal function approximators**:

Theorem (Pinkus, 1999 (Simplified))

For a neural network with a single hidden layer and a wide class of activation functions, for any continuous function $f : \mathbb{R}^n \to \mathbb{R}$ *and any* $\varepsilon > 0$*, there exists* $M \in \mathbb{N}$ *such that the network* g *with* M *hidden units satisfy:*

$$\max_{\mathbf{x} \in \Omega} |f(\mathbf{x}) - g(\mathbf{x})| < \varepsilon$$

Universal Approximation Theorem

Neural networks are **universal function approximators**:

Theorem (Pinkus, 1999 (Simplified))

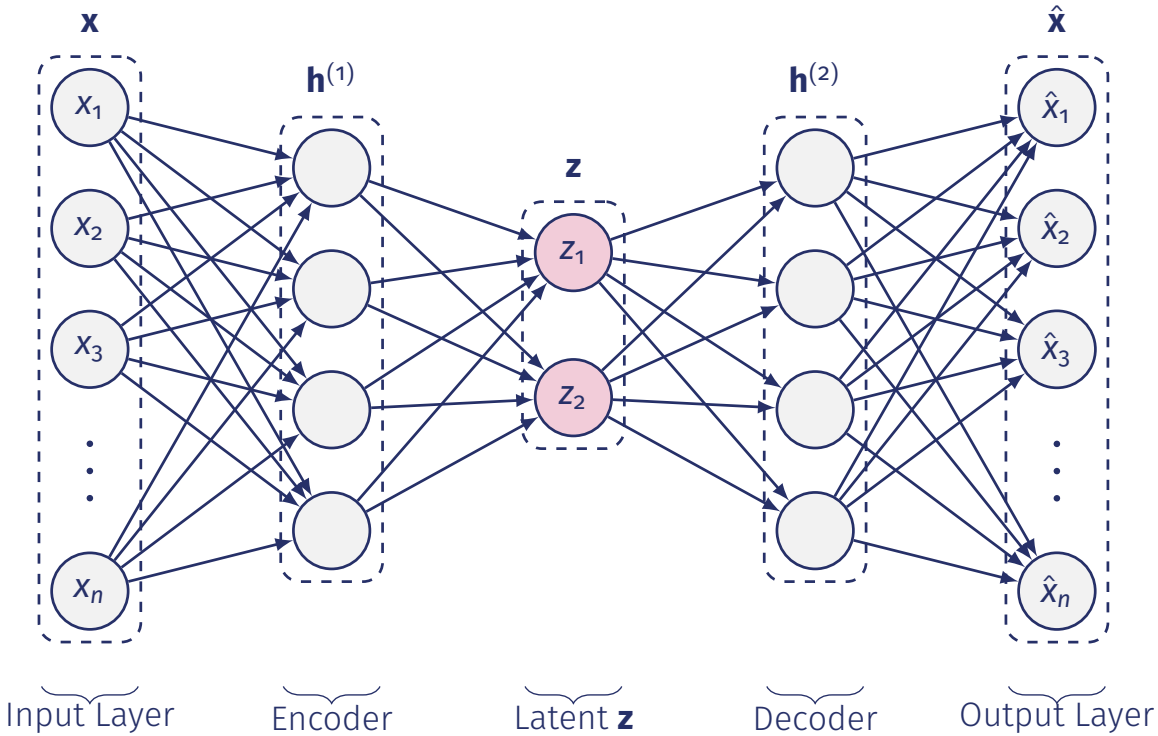
For a neural network with a single hidden layer and a wide class of activation functions, for any continuous function $f : \mathbb{R}^n \to \mathbb{R}$ *and any* $\varepsilon > 0$*, there exists* $M \in \mathbb{N}$ *such that the network* g *with* M *hidden units satisfy:*

$$\max_{\mathbf{x} \in \Omega} |f(\mathbf{x}) - g(\mathbf{x})| < \varepsilon$$

Caveat: the theorem guarantees existence, not learnability.

AUTOENCODERS

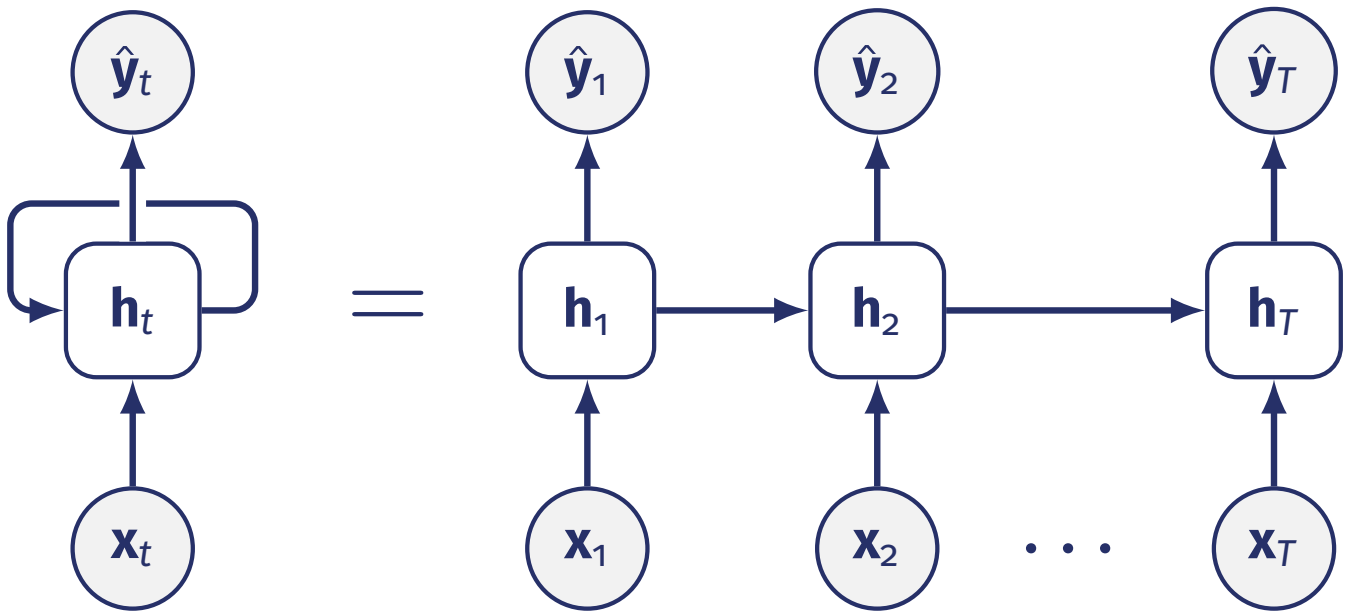
Autoencoders are a type of ANN architecture that learns to compress and reconstruct data.



$$\mathcal{L}(w) = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2$$

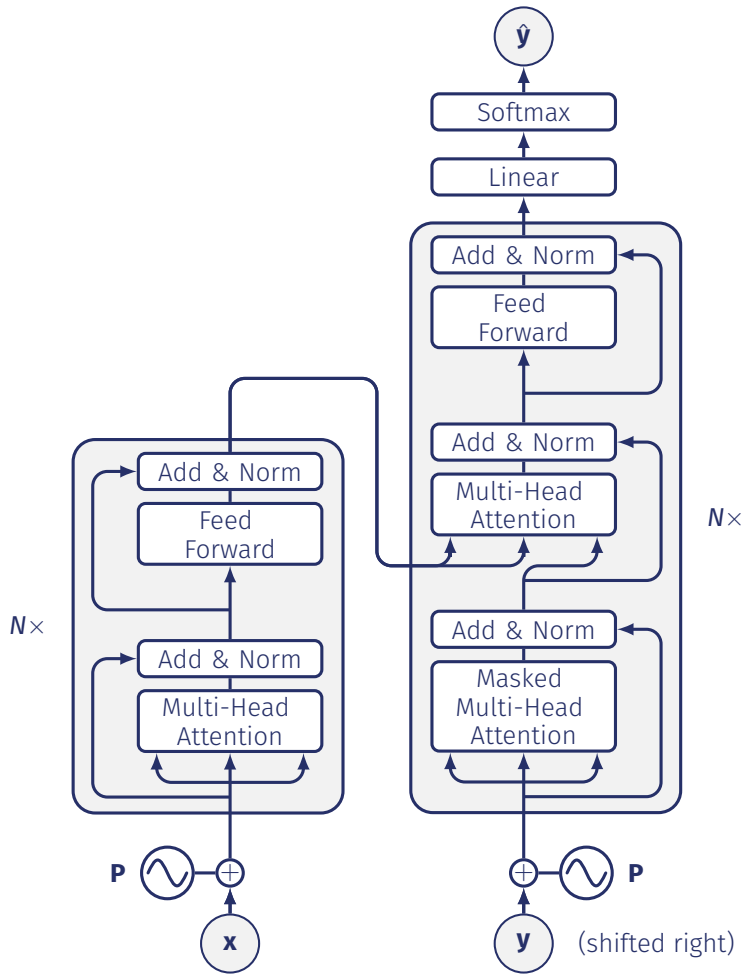
Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are designed to handle sequential data by maintaining a hidden state that captures information from previous time steps.



Transformers

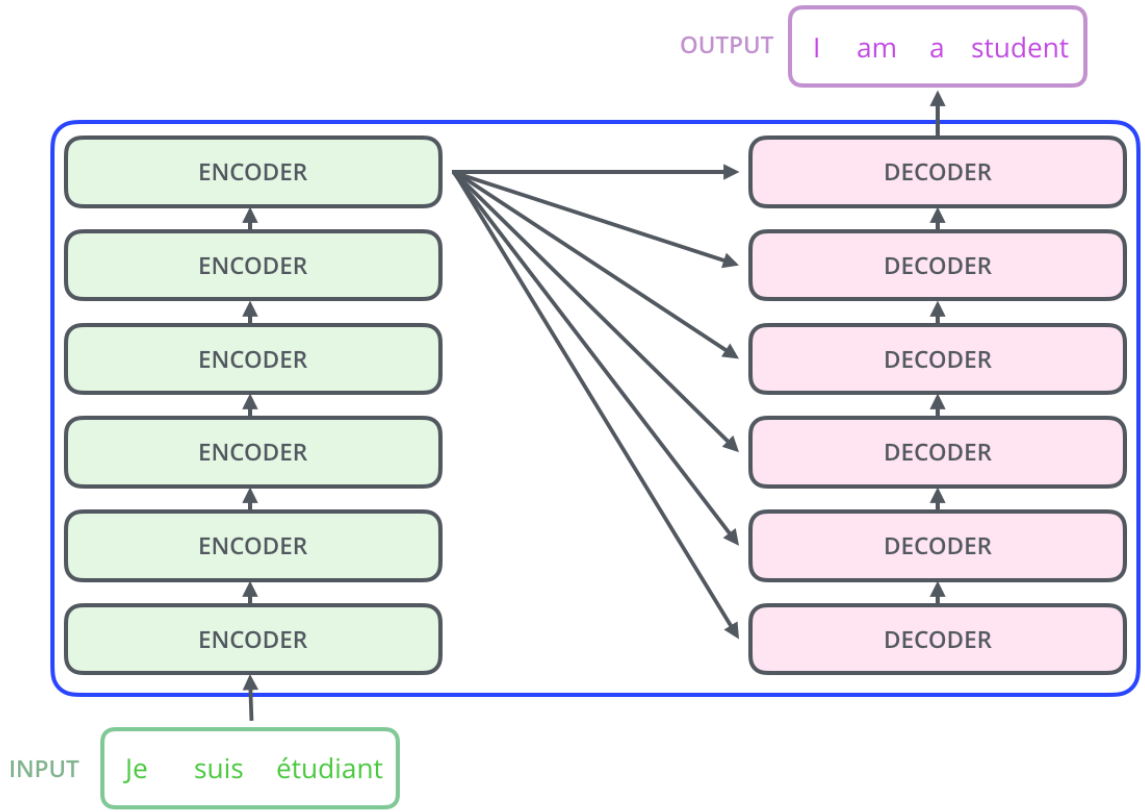
A Transformer[1] model is a neural network that learns the context of sequential data and generates new data out of it.



[1] Vaswani et al. "Attention is all you need." NeurIPS (2017).

Transformers

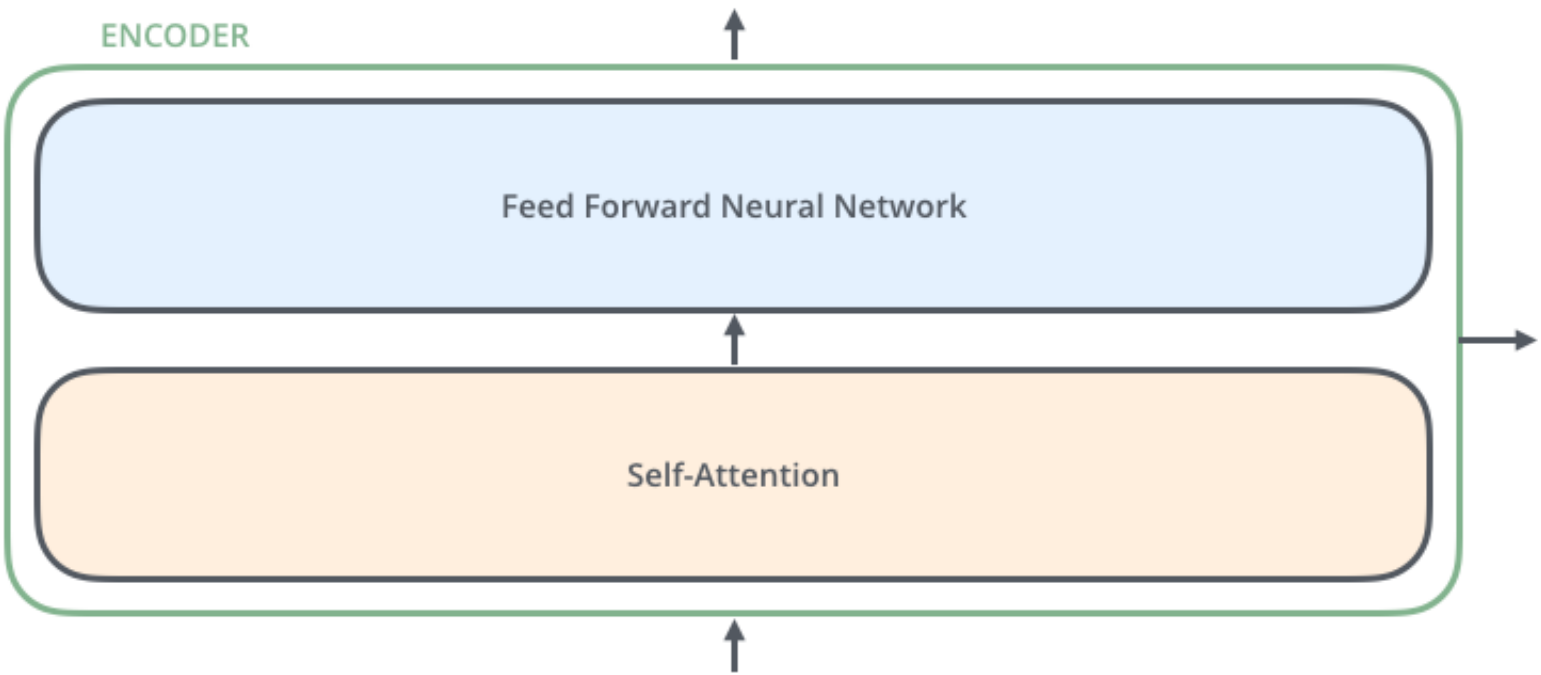
A Transformer[1] model is a neural network that learns the context of sequential data and generates new data out of it.



[1] Vaswani et al. "Attention is all you need." NeurIPS (2017).

Transformers - Encoder

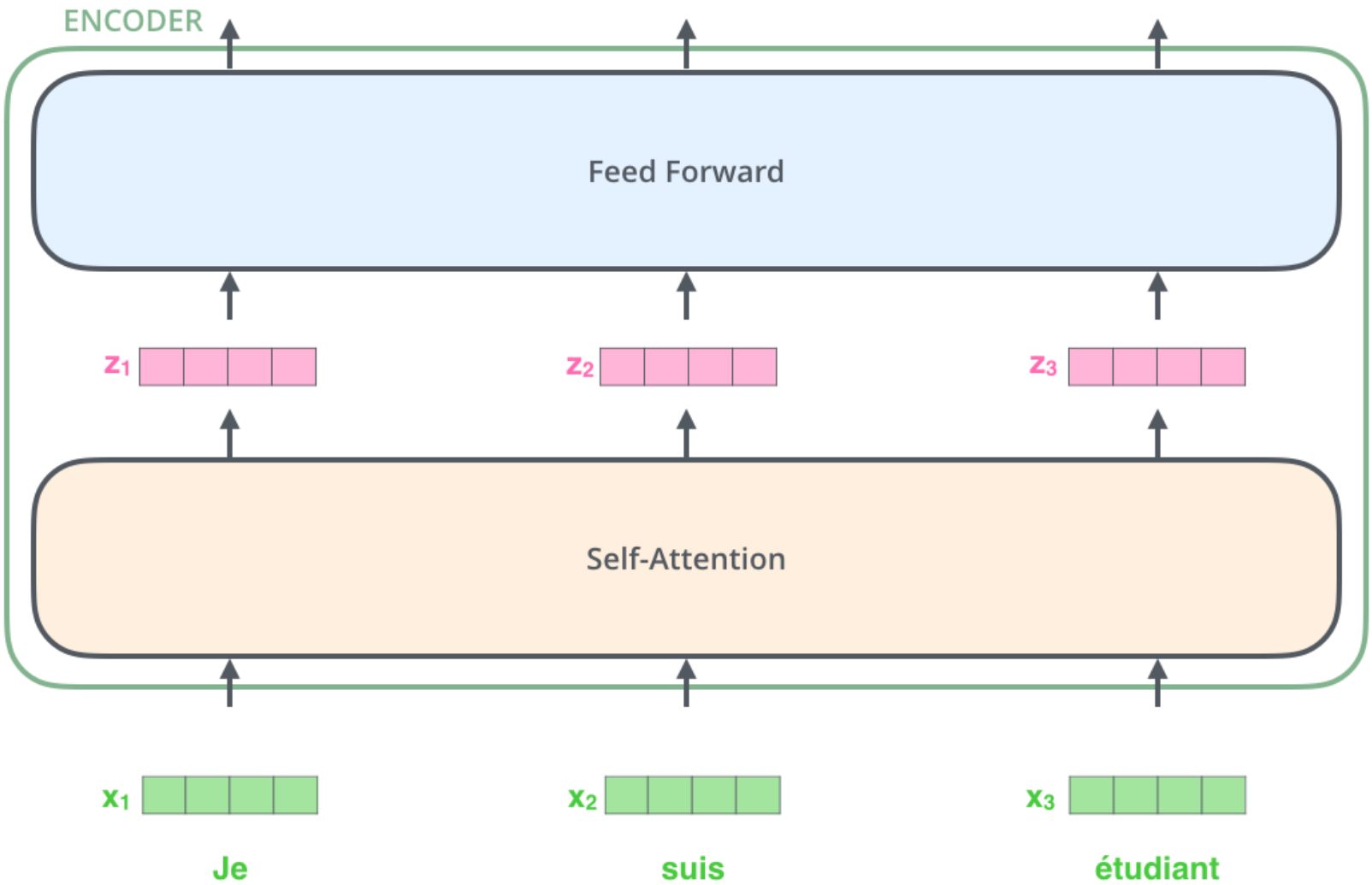
We can simplify removing residual connections and layer normalization, but they are crucial for training deep transformers.



Tokenization and Embeddings

- The input sequence is tokenized and converted into embeddings, i.e., dense vector representations.
  - Frequency-based Emdeddings (e.g., Bag of Words, TF-IDF)
  - Predictive Embeddings (e.g., Word2Vec, GloVe)
  - Contextual Embeddings (e.g., BERT, GPT)

Transformers - Encoder



Self-Attention: Query, Key, Value

Intuition: You are in a library looking for information.

- **Query**: the question you bring to the library
- **Key**: the label on each book's spine, what it is about
- **Value**: the content inside, what you actually read

You compare your question against every label, read the most relevant books more carefully, and aggregate what you learned.

Self-Attention: Query, Key, Value

Intuition: You are in a library looking for information.

- **Query**: the question you bring to the library
- **Key**: the label on each book's spine, what it is about
- **Value**: the content inside, what you actually read

You compare your question against every label, read the most relevant books more carefully, and aggregate what you learned.

In self-attention, each token plays all three roles simultaneously:

- **Q**: what this token is looking for
- **K**: what this token advertises about itself
- **V**: what this token contributes if selected

Self-Attention: Query, Key, Value

Intuition: You are in a library looking for information.

- **Query**: the question you bring to the library
- **Key**: the label on each book's spine, what it is about
- **Value**: the content inside, what you actually read

You compare your question against every label, read the most relevant books more carefully, and aggregate what you learned.

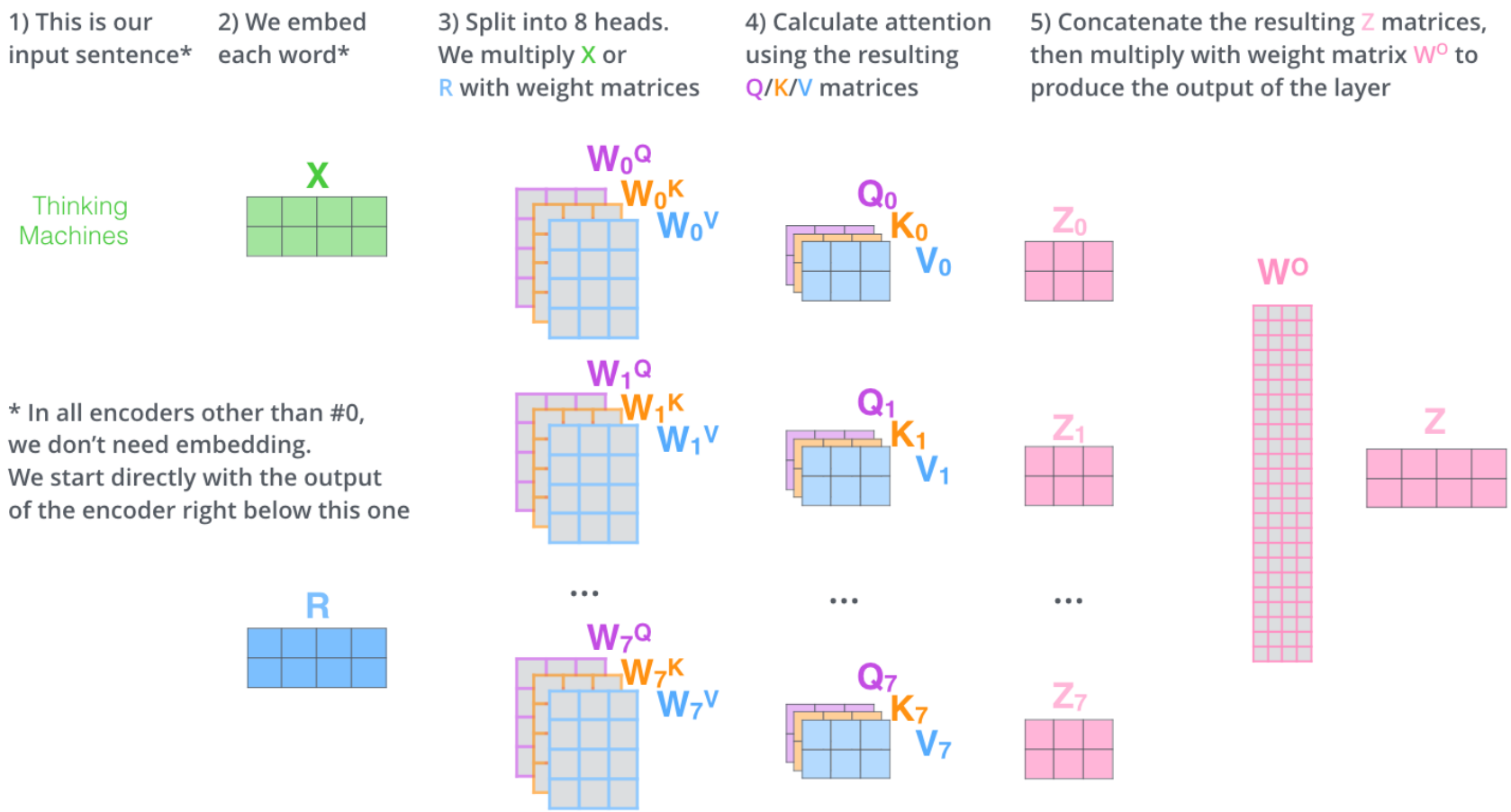
In self-attention, each token plays all three roles simultaneously:

- **Q**: what this token is looking for
- **K**: what this token advertises about itself
- **V**: what this token contributes if selected

"The animal didn't cross the street because **it** was too tired."
it queries all tokens → animal has the highest matching key → its value gets the most weight.

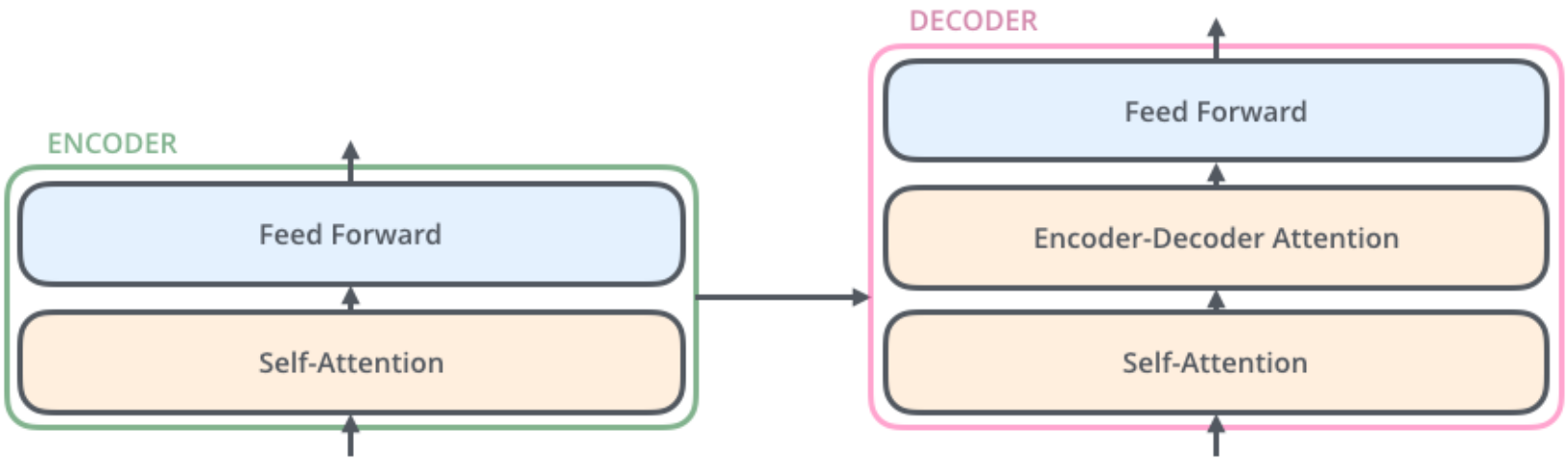
Multi-Head Attention

Multi-head attention captures different types of relationships within a sequence simultaneously.



Transformers - Decoder

The decoder block is composed of layers similar to those in the encoder, plus a cross-attention layer.



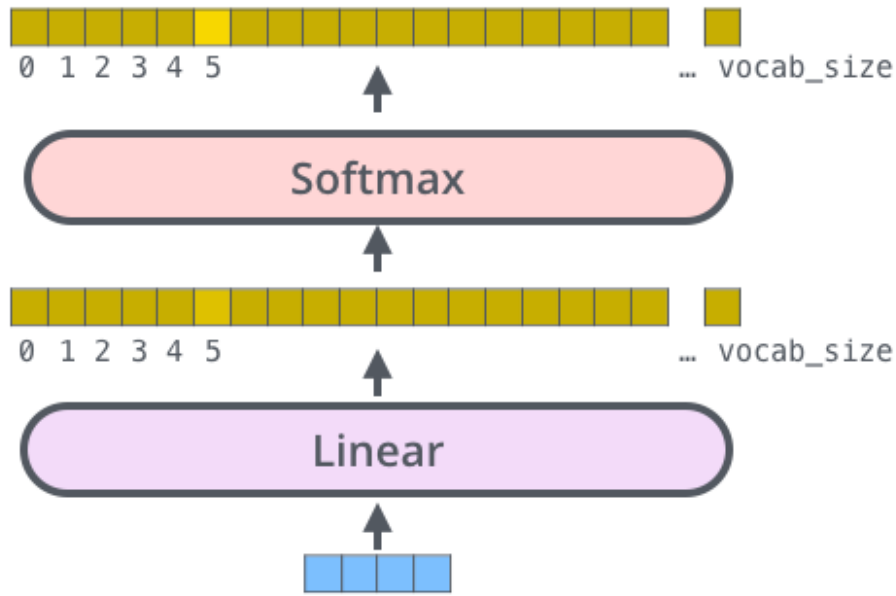
Transformers - Output

Which word in our vocabulary is associated with this index?

am

Get the index of the cell with the highest value (**argmax**)

5



Transformers for multimodal data

A transformer only sees a sequence of vectors, any modality (data type) works as long as it can be tokenized and embedded into a common space.

- Each modality gets its own tokenization and encoder (patches, frames, rows…).
- A shared or aligned latent space connects modalities.
- Cross-attention lets the decoder attend to the encoded source modality.

2

Foundation Model for Tabular Data

Foundation Models

A foundation model is a large neural network trained on broad, large-scale data that can be adapted to a wide range of downstream tasks.

- **Emergent capabilities**: abilities not explicitly trained for (reasoning, in-context learning) arise beyond certain scale thresholds.
- **Transfer**: a single pretrained model serves as the starting point for many tasks, drastically reducing the cost of specialisation.

Foundation Model for Tabular Data (?)

Tabular data: Structured data organized in rows and columns, where each row is an observation and each column is a feature.

[2] Hollmann et al. "TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second." ICLR 2023.

Foundation Model for Tabular Data (?)

Tabular data: Structured data organized in rows and columns, where each row is an observation and each column is a feature.

Can we train Transformers with tabular data?

[2] Hollmann et al. "TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second." ICLR 2023.

Foundation Model for Tabular Data

Tabular data: Structured data organized in rows and columns, where each row is an observation and each column is a feature.

Can we train Transformers with tabular data?

- Yes[2], if we treat each row as a sequence of (feature, value) tokens, i.e. $\mathbb{D} = \{(x_i, y_i)\}_{i=1}^{N}$.

[2] Hollmann et al. "TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second." ICLR 2023.

Foundation Model for Tabular Data

Tabular data: Structured data organized in rows and columns, where each row is an observation and each column is a feature.

Can we train Transformers with tabular data?

- Yes[2], if we treat each row as a sequence of (feature, value) tokens, i.e. $\mathbb{D} = \{(x_i, y_i)\}_{i=1}^{N}$.

Where can we find a large sample of tabular datasets?

[2] Hollmann et al. "TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second." ICLR 2023.

Foundation Model for Tabular Data

Tabular data: Structured data organized in rows and columns, where each row is an observation and each column is a feature.

Can we train Transformers with tabular data?

- Yes[2], if we treat each row as a sequence of (feature, value) tokens, i.e. $\mathbb{D} = \{(x_i, y_i)\}_{i=1}^{N}$.

Where can we find a large sample of tabular datasets?

- We can generate them !

[2] Hollmann et al. "TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second." ICLR 2023.

Foundation Model for Tabular Data

Tabular data: Structured data organized in rows and columns, where each row is an observation and each column is a feature.

Can we train Transformers with tabular data?

- Yes[2], if we treat each row as a sequence of (feature, value) tokens, i.e. $\mathbb{D} = \{(x_i, y_i)\}_{i=1}^{N}$.

Where can we find a large sample of tabular datasets?

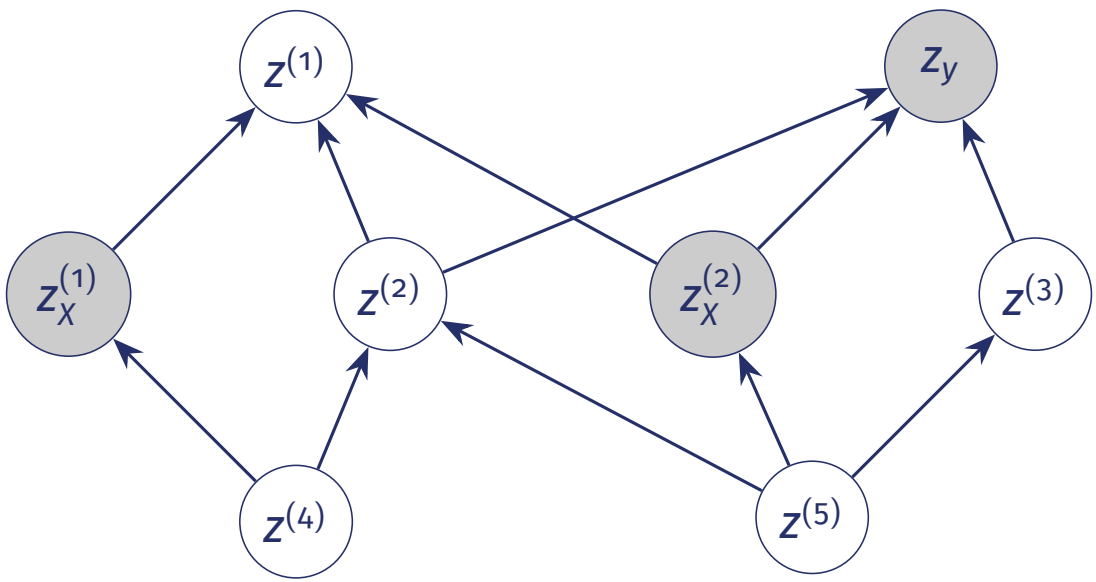
- We can generate them ! ...but we need a prior

[2] Hollmann et al. "TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second." ICLR 2023.

Synthetic Datasets

We generate synthetic datasets by sampling from a prior over data-generating processes:

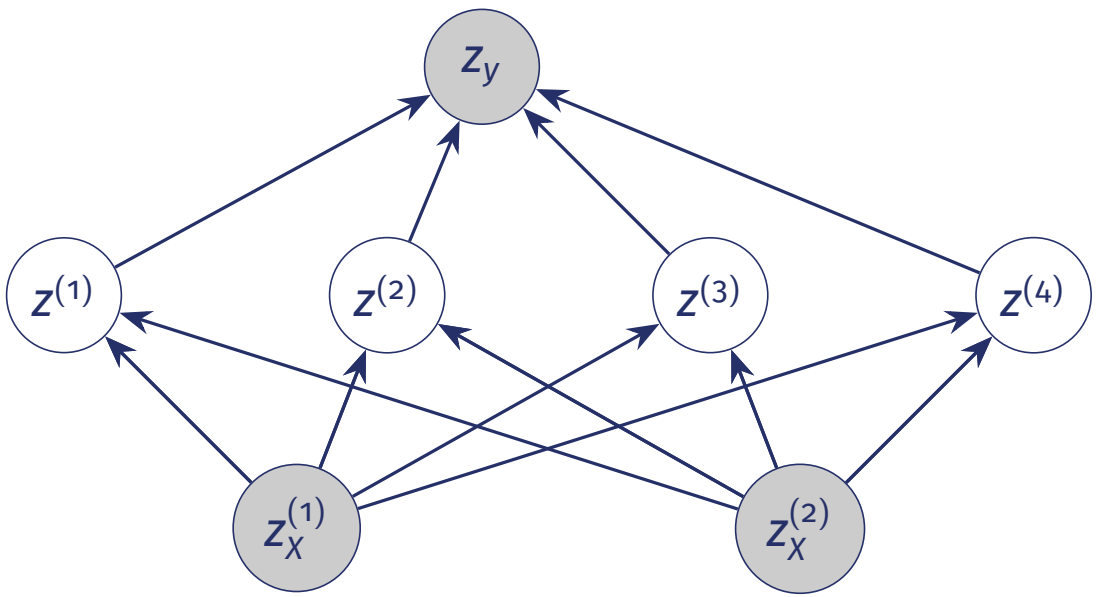
1. Random Structural Causal Models (SCMs) as a generative process:
   - Select a subset of nodes as covariates, $z_X^{(i)}$, and a downstream node as target, z_y.



Synthetic Datasets

We generate synthetic datasets by sampling from a prior over data-generating processes:

2. Bayesian Neural Networks (BNNs) are neural models in which each weight has a distribution.



Synthetic Datasets

We generate synthetic datasets by sampling from a prior over data-generating processes:

- The generated SCMs are not training examples in the usual ML sense.

Synthetic Datasets

We generate synthetic datasets by sampling from a prior over data-generating processes:

- The generated SCMs are not training examples in the usual ML sense.
- Collectively, these generated SCMs define a prior over data-generating mechanisms.

Posterior Predictive Distribution (PPD)

We have a dataset, $\mathbb{D} = \{(x_i, y_i)\}_{i=1}^{N}$, and we have an additional point x. We want to predict y:

Posterior Predictive Distribution (PPD)

We have a dataset, $\mathbb{D} = \{(x_i, y_i)\}_{i=1}^{N}$, and we have an additional point x. We want to predict y:

$$p(y \mid x, \mathbb{D}) = \int p(y \mid x, \mathcal{M}) p(\mathcal{M} \mid \mathbb{D}), d\mathcal{M}$$

- $\mathcal{M}$ is an SCM.
- The PPD weight all plausible models according to the data.

Posterior Predictive Distribution (PPD)

We have a dataset, $\mathbb{D} = \{(x_i, y_i)\}_{i=1}^{N}$, and we have an additional point x. We want to predict y:

$$p(y \mid x, \mathbb{D}) = \int p(y \mid x, \mathcal{M}) p(\mathcal{M} \mid \mathbb{D}), d\mathcal{M}$$

- $\mathcal{M}$ is an SCM.
- The PPD weight all plausible models according to the data.

Given everything I have seen in $\mathbb{D}$, and given all the causal mechanisms that could have generated it, what is my best guess for y?

Prior-Data Fitted Networks (PFNs)

We can use a Transformer to approximate the PPD:

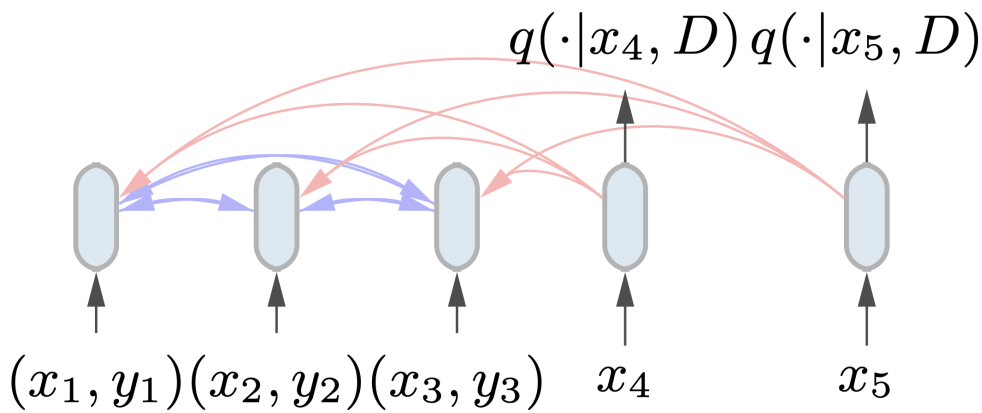
$$\mathcal{L}_{\text{PFN}} = \mathbb{E}_{\{(x,y)\}\cup\mathbb{D}\sim p_{\mathcal{M}}}[-\log q_w(y \mid x,\ \mathbb{D}_{\text{train}})]$$

- q_w is the predictive distribution computed by the network with parameters w.
- Prediction requires only a **single forward pass**: $q_w(y \mid x_{\text{test}},\ \mathbb{D}_{\text{train}})$ — the entire training set is the context.

Why Transformers?

Because of self-attention!

- Each (x_i, y_i) pair is treated as a single **token**, so representations can attend to each other across the full dataset.
- The dataset represents the context to identify the right SCMs.
  - This does not require retrainig!



Can We Use TabPFN for Causal Effect Estimation?

"Our work can be considered as rung 1.5 [...] we do not perform causal reasoning, but make association-based predictions on observational data assuming SCMs model common datasets well."

Can We Use TabPFN for Causal Effect Estimation?

"Our work can be considered as rung 1.5 [...] we do not perform causal reasoning, but make association-based predictions on observational data assuming SCMs model common datasets well."

Since TabPFN is trained on data generated from SCMs, can we try estimate directly causal effects?

In-Context Causal Effect Estimation

Do-PFN[3] is a transformer that estimates causal effects in one forward pass from observational data alone, with no causal graph required.

- It approximates the **conditional interventional distribution (CID)** from observational data $\mathbb{D}^{\text{obs}} = \{(t_i, x_i, y_i)\}_{i=1}^{N}$

[3] Robertson, Reuter et al. "Do-PFN: In-Context Learning for Causal Effect Estimation." NeurIPS 2025.

In-Context Causal Effect Estimation

Do-PFN[3] is a transformer that estimates causal effects in one forward pass from observational data alone, with no causal graph required.

- It approximates the **conditional interventional distribution (CID)** from observational data $\mathbb{D}^{\text{obs}} = \{(t_i, x_i, y_i)\}_{i=1}^{N}$

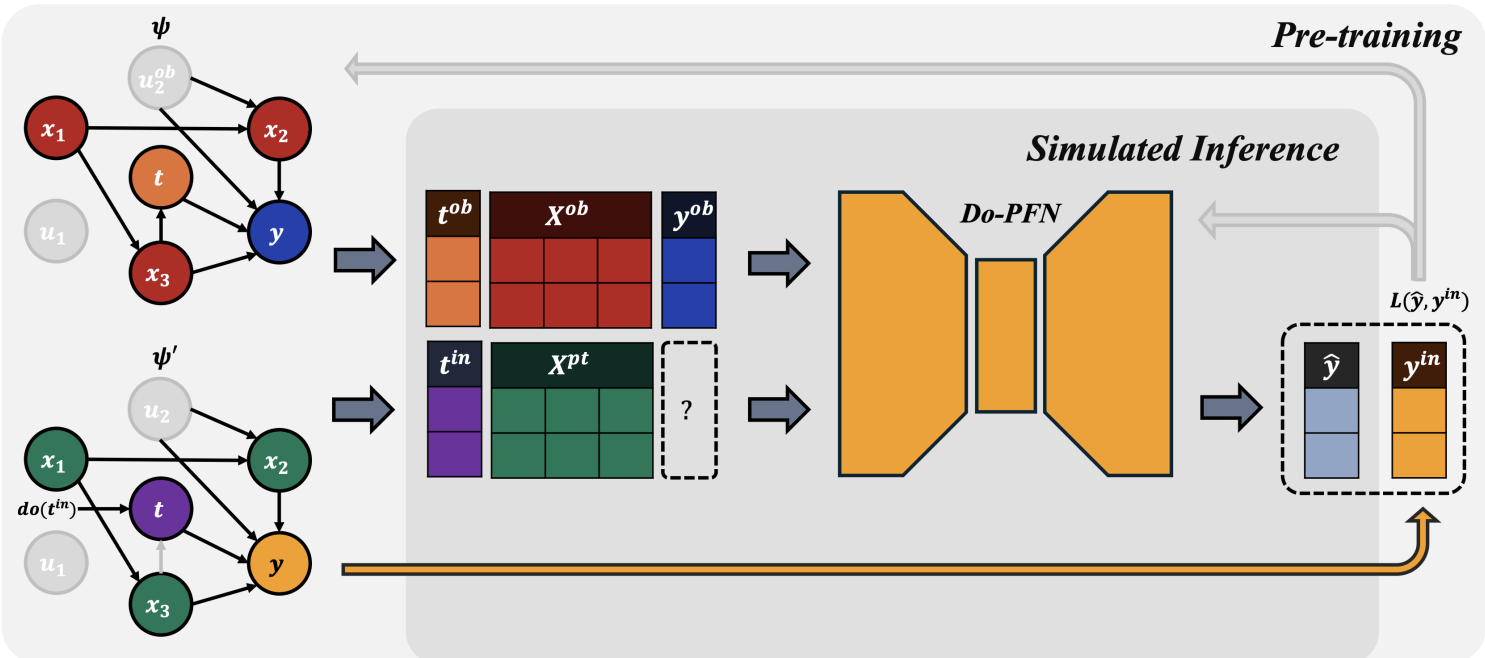
$$p(\tilde{y} \mid do(\tilde{t}), x, \mathbb{D}^{\text{obs}}) = \int p(\tilde{y} \mid do(\tilde{t}), x, \mathcal{M})\, p(\mathcal{M} \mid \mathbb{D}^{\text{obs}})\, d\mathcal{M}$$

[3] Robertson, Reuter et al. "Do-PFN: In-Context Learning for Causal Effect Estimation." NeurIPS 2025.

In-Context Causal Effect Estimation

Do-PFN[3] is a transformer that estimates causal effects in one forward pass from observational data alone, with no causal graph required.

- It approximates the **conditional interventional distribution (CID)** from observational data $\mathbb{D}^{\text{obs}} = \{(t_i, x_i, y_i)\}_{i=1}^{N}$



[3] Robertson, Reuter et al. "Do-PFN: In-Context Learning for Causal Effect Estimation." NeurIPS 2025.

In-Context Causal Effect Estimation

We approximate the CID by training a Transformer on pairs of observational and interventional datasets:

$$\mathcal{L}(w) = \mathbb{E}_{(\tilde{t},x,\tilde{y})\sim p^{do}_{\mathcal{M}} \cup \mathbb{D}^{obs} \sim p_{\mathcal{M}}} \left[-\log q_w\left(\tilde{y} \mid do(\tilde{t}),\, x,\, \mathbb{D}^{\text{obs}}\right) \right]$$

In-Context Causal Effect Estimation

We approximate the CID by training a Transformer on pairs of observational and interventional datasets:

$$\mathcal{L}(w) = \mathbb{E}_{(\tilde{t},x,\tilde{y})\sim p^{do}_{\mathcal{M}} \cup \mathbb{D}^{obs} \sim p_{\mathcal{M}}} \left[-\log q_w\left(\tilde{y} \mid do(\tilde{t}),\, x,\, \mathbb{D}^{\text{obs}}\right) \right]$$

Performing Gradient Descent on $\mathcal{L}(w)$ is equivalent to minimising the KL-divergence between q_w and the true CID $p(\tilde{y} \mid x, do(\tilde{t}), \mathcal{M})$ (Proposition 1).

In-Context Causal Effect Estimation

We approximate the CID by training a Transformer on pairs of observational and interventional datasets:

$$\mathcal{L}(w) = \mathbb{E}_{(\tilde{t},x,\tilde{y})\sim p^{do}_{\mathcal{M}} \cup \mathbb{D}^{obs} \sim p_{\mathcal{M}}} \left[-\log q_w\left(\tilde{y} \mid do(\tilde{t}),\, x,\, \mathbb{D}^{\text{obs}}\right) \right]$$

Performing Gradient Descent on $\mathcal{L}(w)$ is equivalent to minimising the KL-divergence between q_w and the true CID $p(\tilde{y} \mid x, do(\tilde{t}), \mathcal{M})$ (Proposition 1).

Important caveat

This does **not** mean every causal effect is identifiable. When hidden confounders are present, Do-PFN cannot resolve the ambiguity.

In-Context Causal Effect Estimation

Does q_w recover the true interventional distribution as we observe more data?

In-Context Causal Effect Estimation

Does q_w recover the true interventional distribution as we observe more data?

As $|\mathbb{D}^{obs}| \to \infty$, the posterior $p(\mathcal{M} \mid \mathbb{D}^{\text{obs}})$ concentrates on the true SCM, $\mathcal{M}_0$:

In-Context Causal Effect Estimation

Does q_w recover the true interventional distribution as we observe more data?

As $|\mathbb{D}^{obs}| \to \infty$, the posterior $p(\mathcal{M} \mid \mathbb{D}^{\text{obs}})$ concentrates on the true SCM, $\mathcal{M}_{\text{o}}$:

$$p\left(\tilde{y} \mid do(\tilde{t}), x, \mathbb{D}^{\text{obs}}\right) \xrightarrow{|\mathbb{D}^{obs}| \to \infty} p\left(\tilde{y} \mid do(\tilde{t}), x, [\mathcal{M}_{\text{o}}]\right)$$

where $[\mathcal{M}_{\text{o}}]$ is the Markov-equivalence class of $\mathcal{M}_{\text{o}}$.

In-Context Causal Effect Estimation

Does q_w recover the true interventional distribution as we observe more data?

As $|\mathbb{D}^{obs}| \to \infty$, the posterior $p(\mathcal{M} \mid \mathbb{D}^{\text{obs}})$ concentrates on the true SCM, $\mathcal{M}_{\text{o}}$:

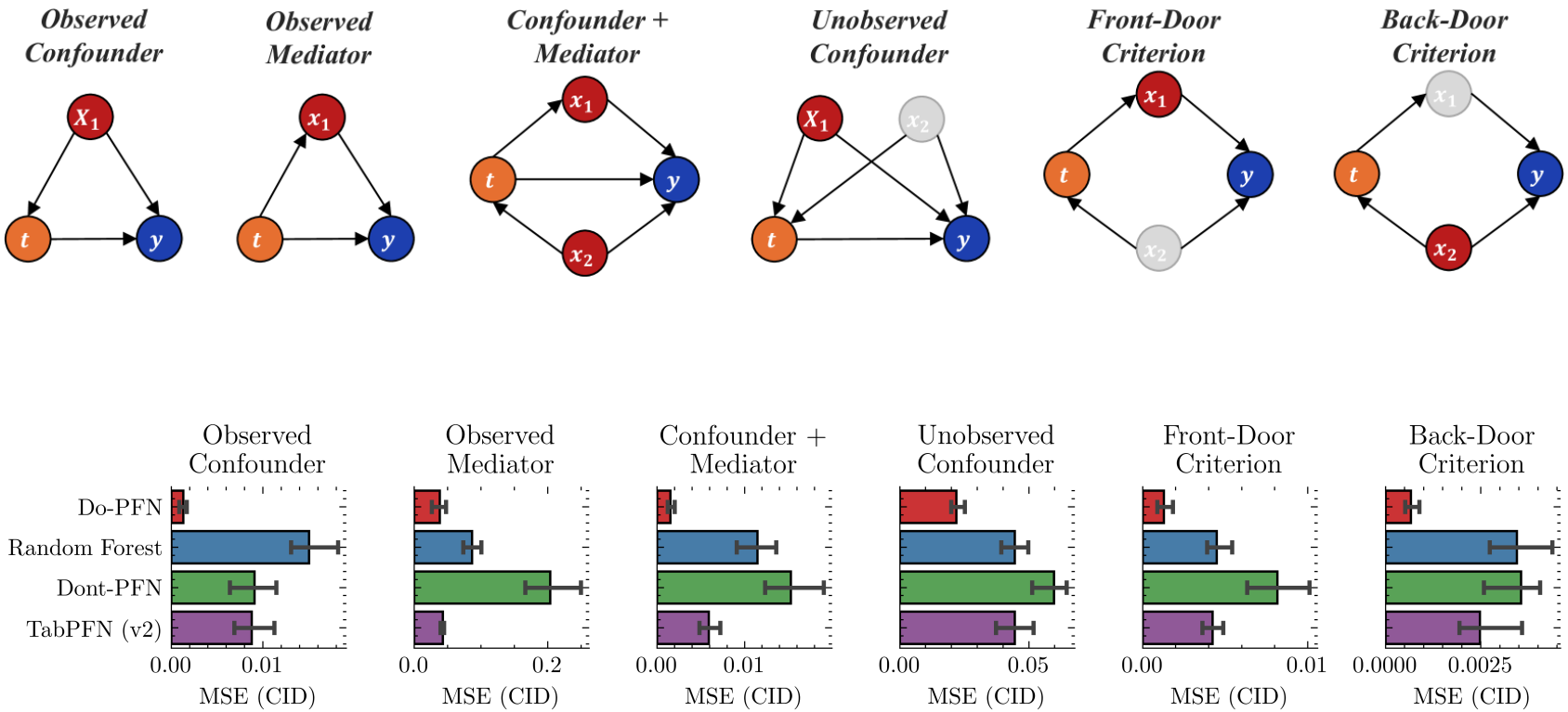
$$p(\tilde{y} \mid do(\tilde{t}), x, \mathbb{D}^{\text{obs}}) \xrightarrow{|\mathbb{D}^{obs}| \to \infty} p(\tilde{y} \mid do(\tilde{t}), x, [\mathcal{M}_{\text{o}}])$$

where $[\mathcal{M}_{\text{o}}]$ is the Markov-equivalence class of $\mathcal{M}_{\text{o}}$.

Why? Observational data alone cannot distinguish between SCMs in the same Markov-equivalence class

In-Context Causal Effect Estimation

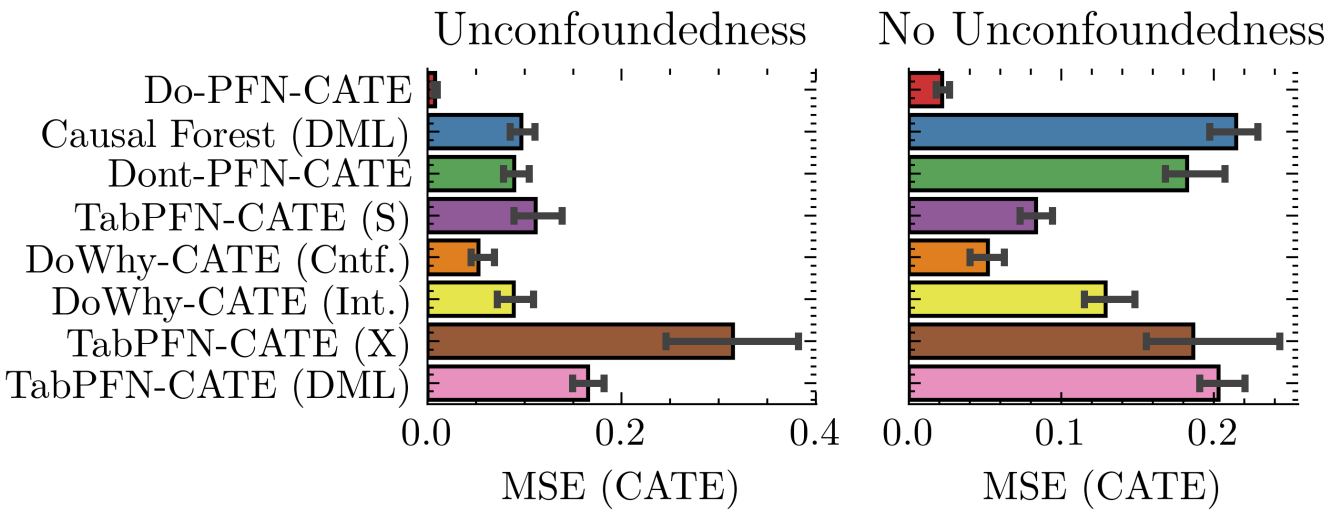
The graph structures of six causal case studies, requiring Do-PFN to automatically perform adjustment.



Do-PFN Under Confoundedness

Do-PFN is trained on SCMs with and without hidden confounders.

- It cannot resolve unidentifiability.
- But when confounding is plausible, it outputs a wider predictive distribution rather than a falsely confident estimate.



From Deep Learning to Causal Inference

- Transformers can process tabular data as **(feature, value)** token sequences.
- **TabPFN** generalizes across unseen datasets by training on synthetic SCMs.
- **Do-PFN** estimates causal effects from observational data with no causal graph required.
  - Non-identifiable instances produce higher uncertainty in the results

From Deep Learning to Causal Inference

- Transformers can process tabular data as **(feature, value)** token sequences.
- **TabPFN** generalizes across unseen datasets by training on synthetic SCMs.
- **Do-PFN** estimates causal effects from observational data with no causal graph required.
  - Non-identifiable instances produce higher uncertainty in the results

Limitations

- Sensitive to the choice of synthetic prior.
- Evaluation mostly on synthetic data, no real-world ground truth.
- Amortized inference does not provide formal guarantees.
- Many configurations were not considered, e.g., non-binary interventions.

Resources

Books & Papers:

- "Deep Learning", Goodfellow and Bengio
- "Attention is all you need", Vaswani et al.
- "The Illustrated Transformer", Jay Alamar
- "TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second", Hollmann et al.
- "Do-PFN: In-Context Learning for Causal Effect Estimation", Robertson et al.

Code:

- `tabpfn python` package
- `tabpfn R` code
- `dopfn python` code

QUESTIONS?