PLDI 2024

PL4XGL: A Programming Language Approach to Explainable Graph Learning

Hakjoo Oh

Korea University

(co-work with Minseok Jeon and Jihyeok Park)

IFIP WG 2.4 Meeting @Lugano, Switzerland

PL/SE Research @Korea Univ.

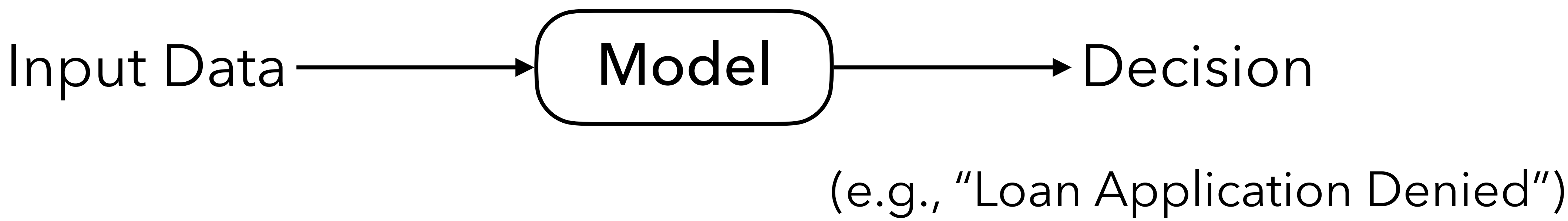
- Members: 10+ PhD and MS students
- Research area: intersection of programming languages (PL) and software engineering (SE)
  - program analysis and testing
  - program synthesis and repair

- Publication: PL, SE, and Security
  - PL: POPL('22),PLDI('12,'14,'20,'24),OOPSLA('15,'17a,'17b,'18a,'18b,'19,'20,'23)
  - SE: ICSE('17,'18,'19,'20,'21'22a,'22b,'23a,'23b,'23c), FSE('18,'19,'20,'21,'22,'23)
  - Security: IEEE S&P('17,'20), USENIX Security('21,'23)

http://kupl.github.io

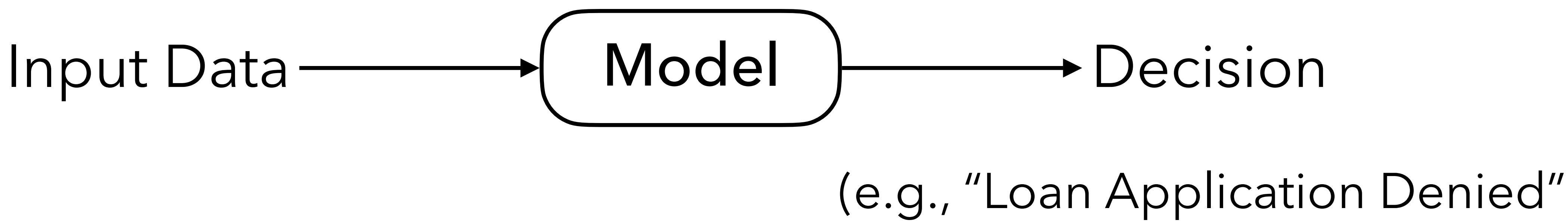
Explainable AI (XAI)

- Today: Unexplainable AI

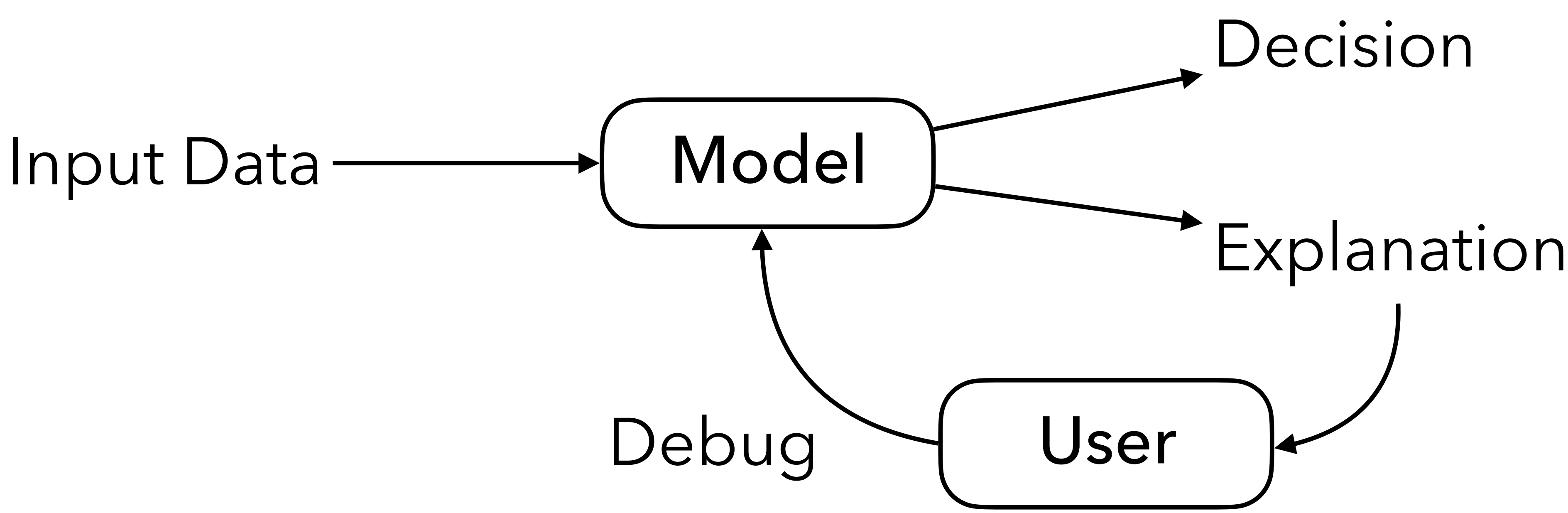


Explainable AI (XAI)

- Today: Unexplainable AI



- Tomorrow: Explainable AI



Key Challenge in XAI

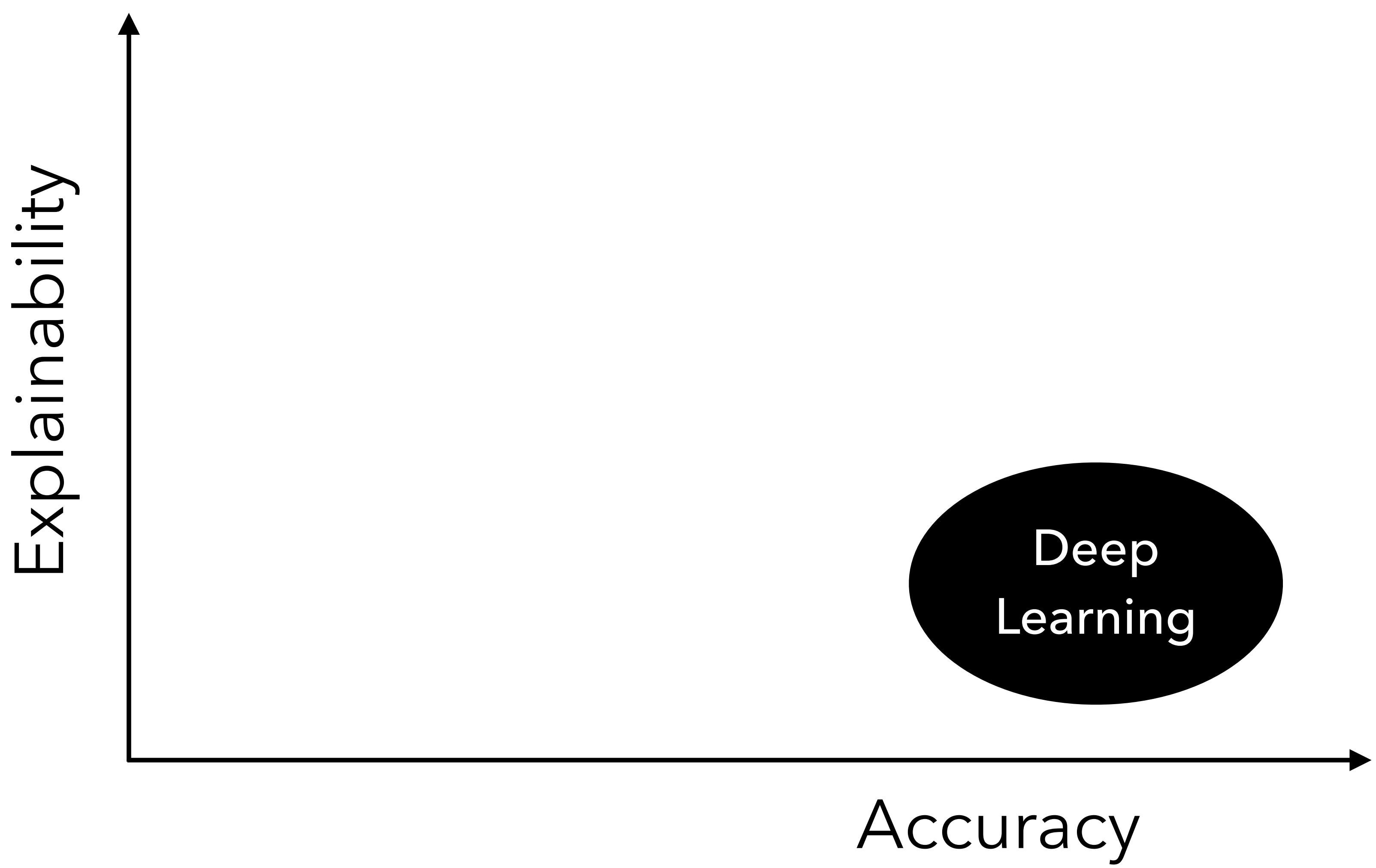
- Practical XAI should satisfy two criteria: (1) high accuracy and (2) high explainability

- No AI approaches can achieve them at the same time

Key Challenge in XAI

- Practical XAI should satisfy two criteria: (1) high accuracy and (2) high explainability

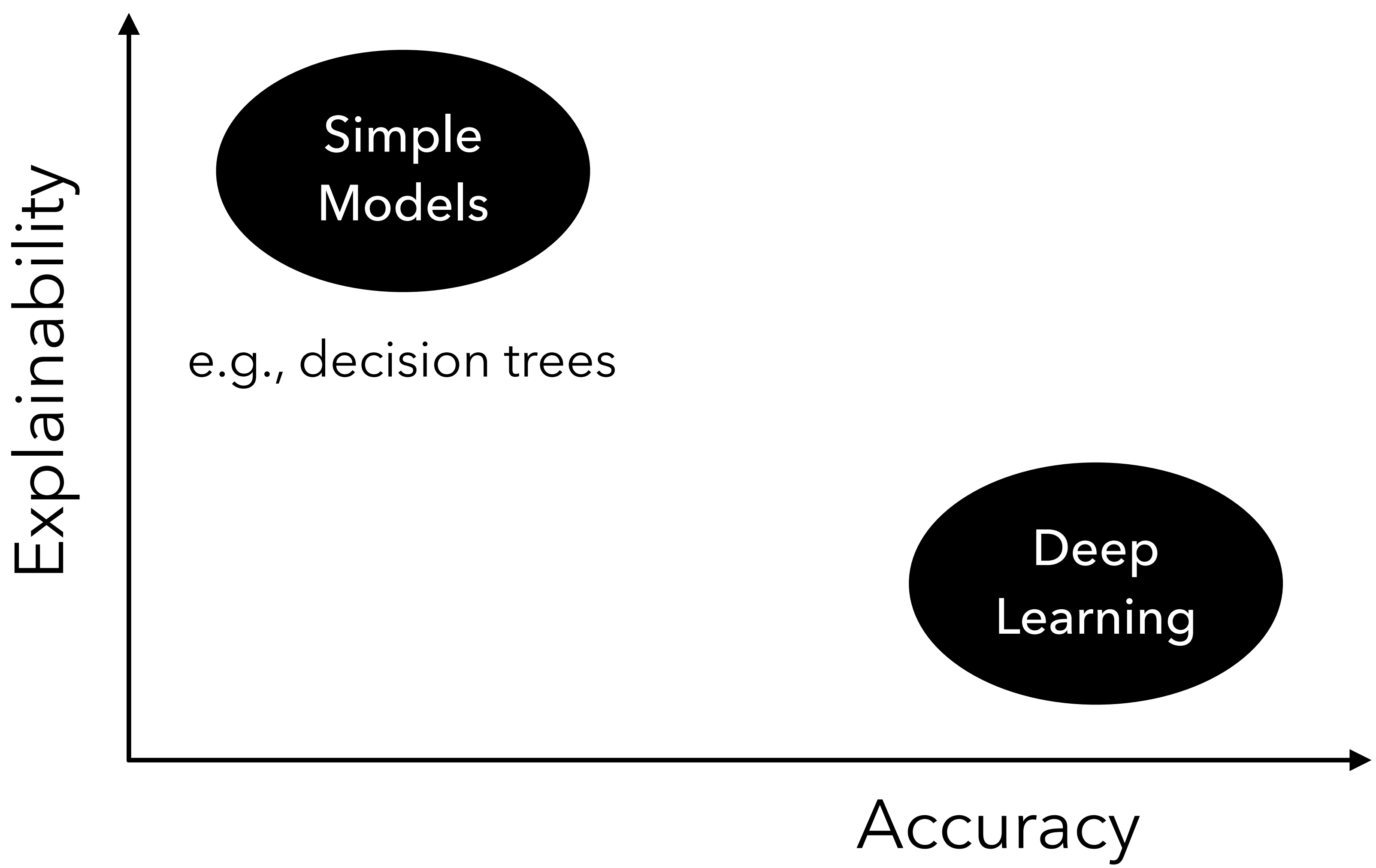
- No AI approaches can achieve them at the same time



Key Challenge in XAI

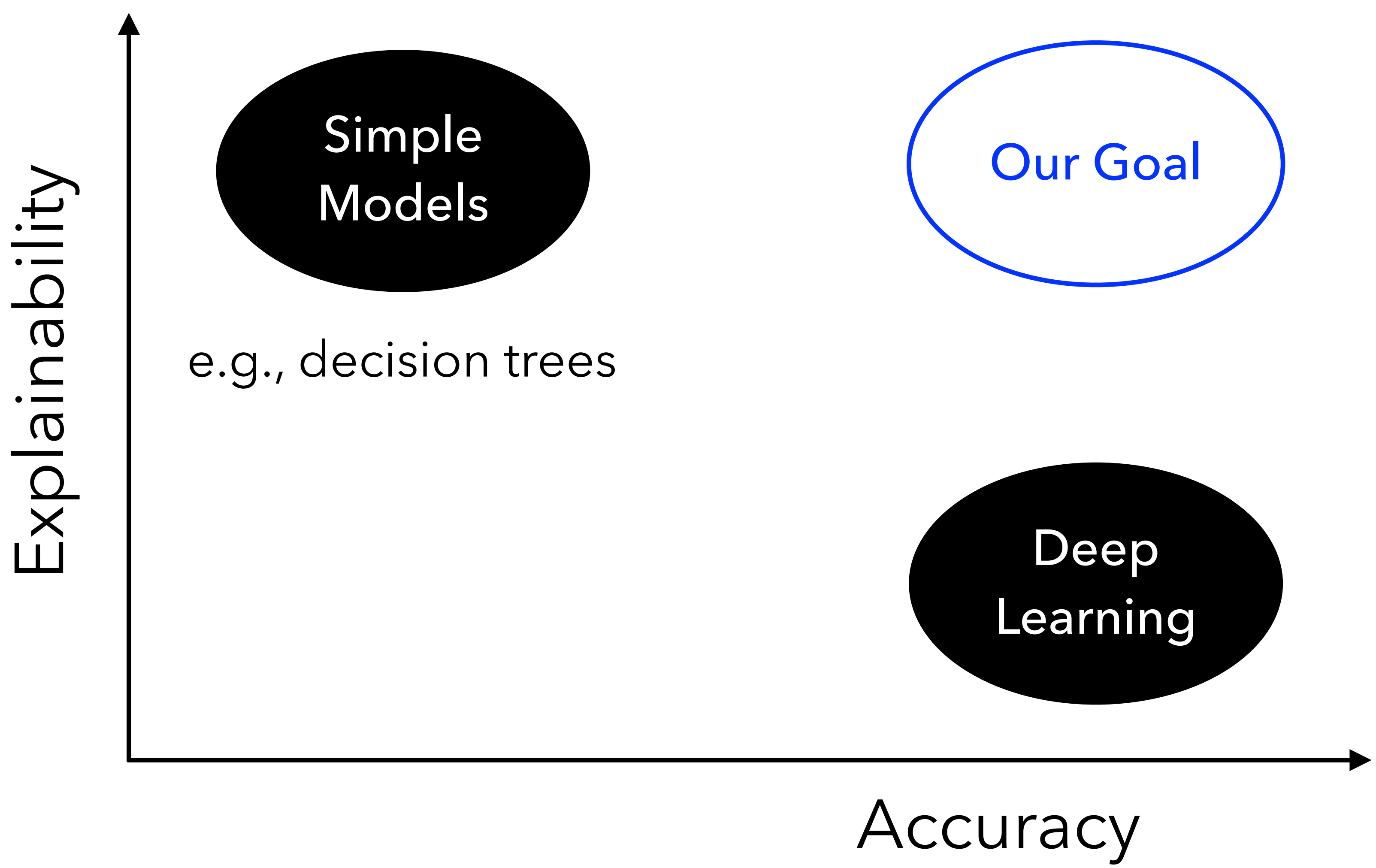
- Practical XAI should satisfy two criteria: (1) high accuracy and (2) high explainability

- No AI approaches can achieve them at the same time



Key Challenge in XAI

- Practical XAI should satisfy two criteria: (1) high accuracy and (2) high explainability
- No AI approaches can achieve them at the same time



Our Proposal: A PL Approach to XAI

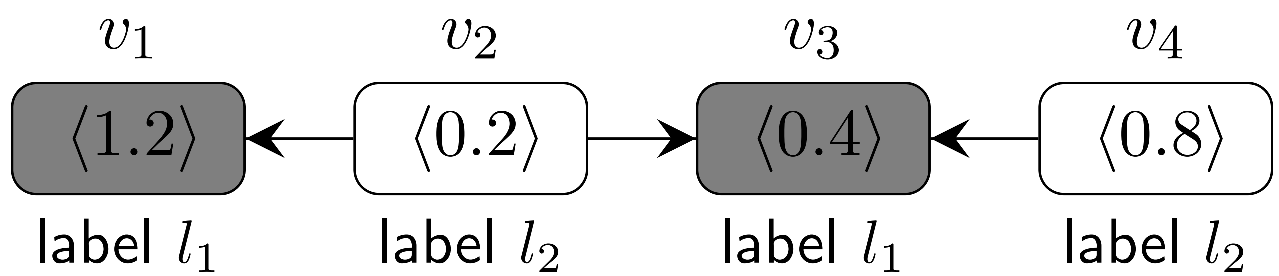
- Idea:
    1. Express AI models as programs written in a DSL
    2. Learn models (programs) from data via program synthesis
- Inherently accurate and explainable:
    - Accurate: PLs can describe any computational models
    - Explainable: DSLs are human-readable w/ high-level semantics

Our Proposal: A PL Approach to XAI

- Idea:
  1. Express AI models as programs written in a DSL
  2. Learn models (programs) from data via program synthesis
- Inherently accurate and explainable:
  - Accurate: PLs can describe any computational models
  - Explainable: DSLs are human-readable w/ high-level semantics
- This work: demonstration with a focus on graph learning
  - Graph Description Language (GDL)
  - Graph / node / edge classification

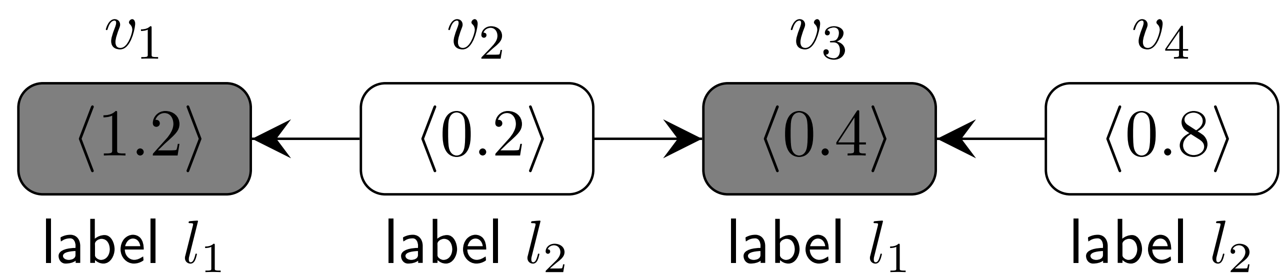
Node Classification

- Example graph

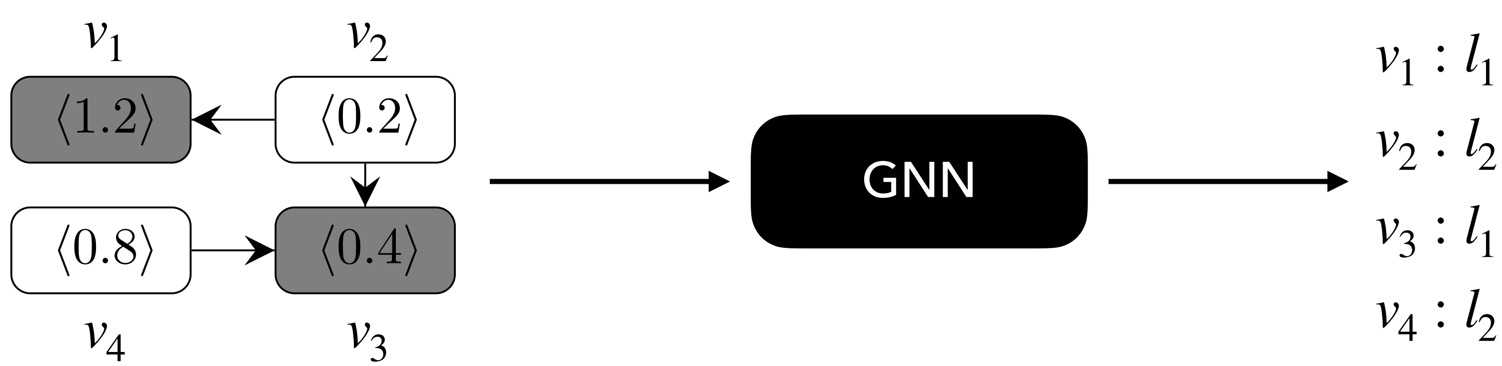


Node Classification

- Example graph

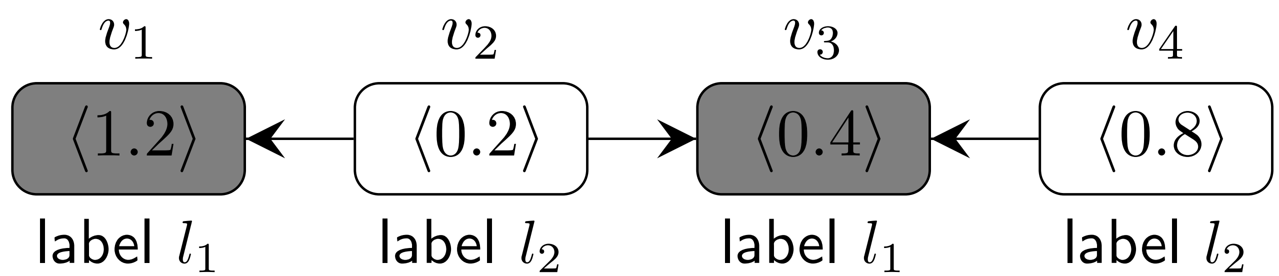


- Mainstream approach: Graph Neural Network (GNN)

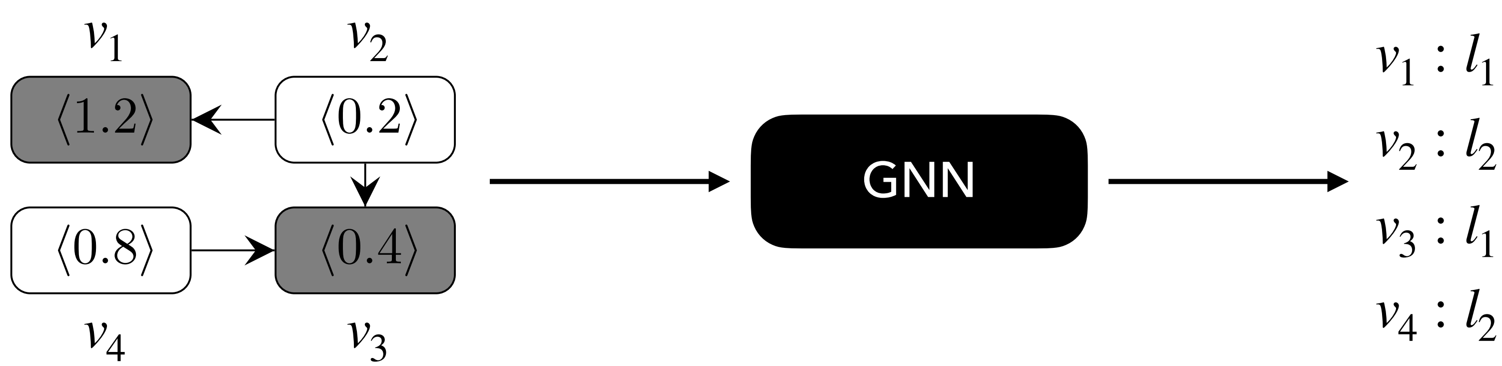


Node Classification

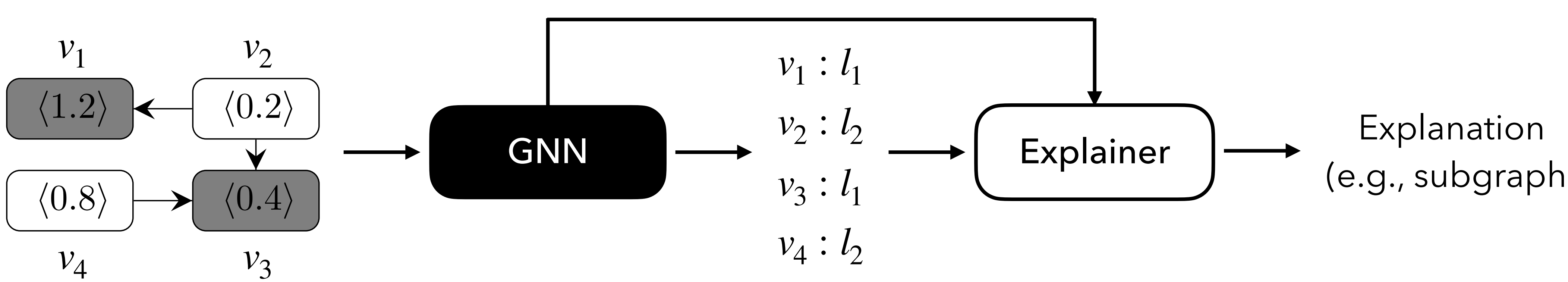
- Example graph



- Mainstream approach: Graph Neural Network (GNN)

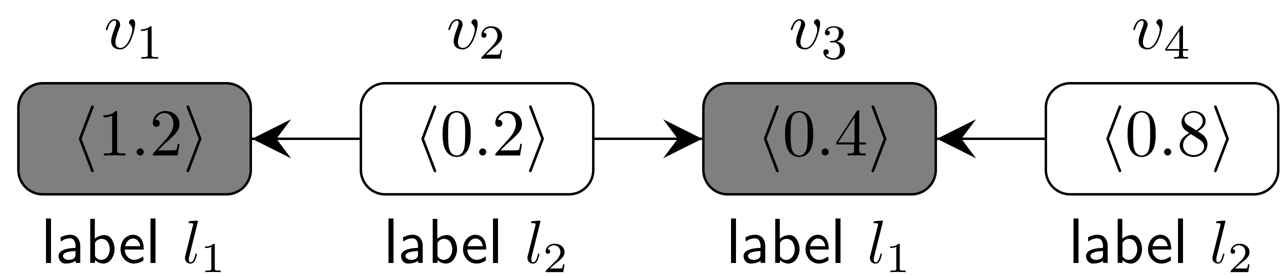


- GNNs are used with separate, post-hoc “explainers”

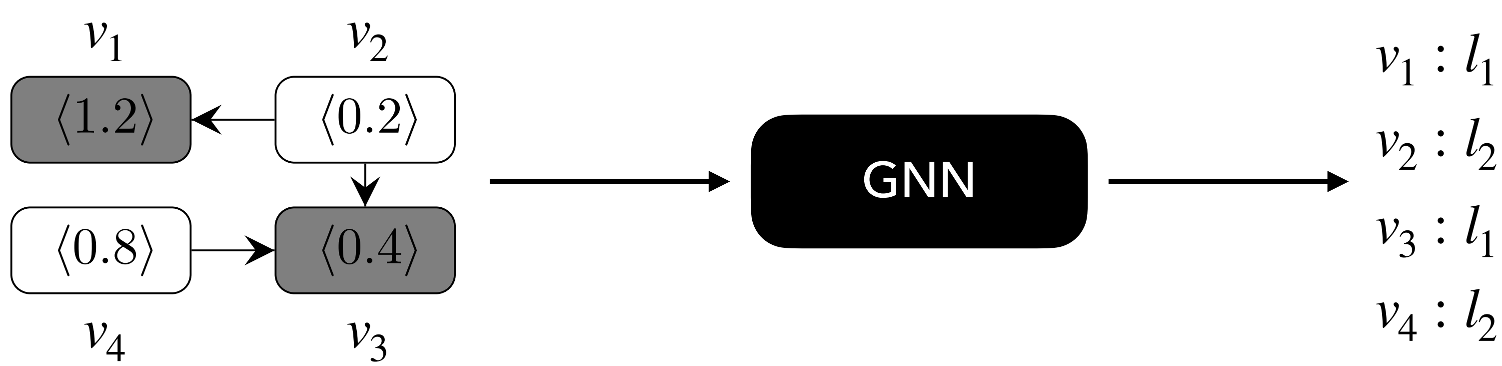


Node Classification

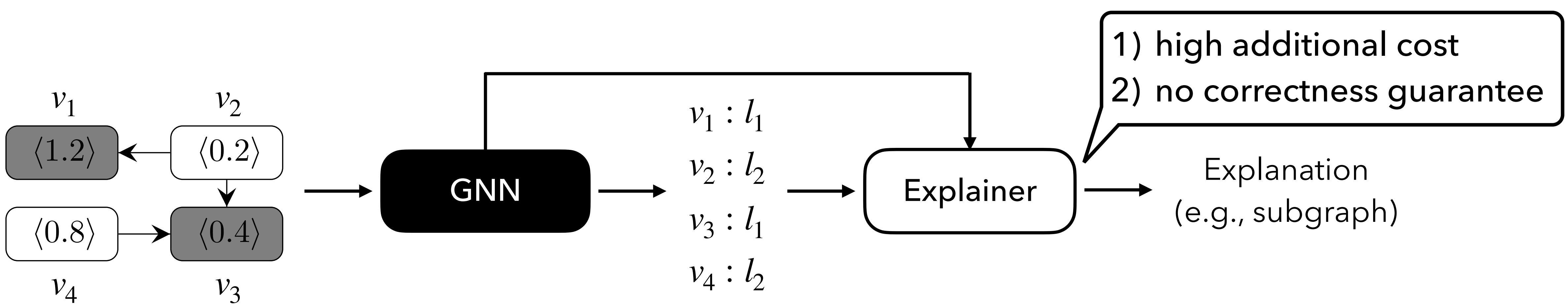
- Example graph



- Mainstream approach: Graph Neural Network (GNN)



- GNNs are used with separate, post-hoc “explainers”



Our Approach: PL4XGL

- GDL: A declarative language for describing graphs
  - Syntax

Programs	P	$::= \overline{\delta}$ `target` t	$\in$	$\mathbb{P}$	$= \mathbb{D}^* \times \mathbb{T}$
Descriptions	δ	$::= \delta_V \mid \delta_E$	$\in$	$\mathbb{D}$	$= \mathbb{D}_V \uplus \mathbb{D}_E$
Node Descriptions	δ_V	$::=$ `node` x `<`$\overline{\phi}$`>`$^?$	$\in$	$\mathbb{D}_V$	$= \mathbb{X} \times \Phi^d$
Edge Descriptions	δ_E	$::=$ `edge` `(`x`,`x`)` `<`$\overline{\phi}$`>`$^?$	$\in$	$\mathbb{D}_E$	$= \mathbb{X} \times \mathbb{X} \times \Phi^c$
Target Symbols	t	$::=$ `node` x $\mid$ `edge` `(`x`,`x`)` $\mid$ `graph`	$\in$	$\mathbb{T}$	$= \mathbb{X} \uplus (\mathbb{X} \times \mathbb{X}) \uplus \{\epsilon\}$
Intervals	ϕ	$::=$ `[`$n^?$`,`$n^?$`]`	$\in$	Φ	$= (\mathbb{R} \uplus \{-\infty\}) \times (\mathbb{R} \uplus \{\infty\})$
Real Numbers	n	$::=$ `0.2` $\mid$ `0.7` $\mid$ `6` $\mid$ `-8` $\ldots$	$\in$	$\mathbb{R}$	
Variables	x	$::=$ `x` $\mid$ `y` $\mid$ `z` $\mid \ldots$	$\in$	$\mathbb{X}$	

  - Semantics

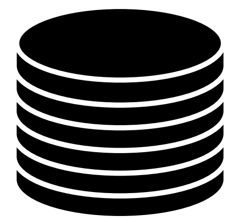
$$
\begin{array}{lllll}
[\![\texttt{<}\phi_1,\ldots,\phi_k\texttt{>}]\!] & : \wp(\mathbb{R}^k) & = \{ & \mathbf{f} & \mid \mathbf{f} = (f_1,\ldots,f_k) \wedge \forall i.\ f_i \in \gamma(\phi_i)\} \\
[\![\texttt{node}\ x\ \texttt{<}\overline{\phi}\texttt{>}]\!] & : \wp(\mathbb{G} \times \mathbb{H}) & = \{ & (G,\eta) & \mid v = \eta(x) \wedge \mathbf{f}_v^G \in [\![\texttt{<}\overline{\phi}\texttt{>}]\!]\} \\
[\![\texttt{edge}\ (x,y)\ \texttt{<}\overline{\phi}\texttt{>}]\!] & : \wp(\mathbb{G} \times \mathbb{H}) & = \{ & (G,\eta) & \mid e \in E \wedge e = (\eta(x),\eta(y)) \wedge \mathbf{f}_e^G \in [\![\texttt{<}\overline{\phi}\texttt{>}]\!]\} \\
[\![\delta_1\delta_2\ldots\delta_k]\!] & : \wp(\mathbb{G} \times \mathbb{H}) & = \{ & (G,\eta) & \mid \forall i.\ (G,\eta) \in [\![\delta_i]\!]\} \\
[\![\overline{\delta}\ \texttt{target node}\ x]\!] & : \wp(\mathbb{G} \times V) & = \{ & (G,v) & \mid \exists (G,\eta) \in [\![\overline{\delta}]\!].\ v = \eta(x)\} \\
[\![\overline{\delta}\ \texttt{target edge}\ (x,y)]\!] & : \wp(\mathbb{G} \times E) & = \{ & (G,e) & \mid \exists (G,\eta) \in [\![\overline{\delta}]\!].\ e = (\eta(x),\eta(y))\} \\
[\![\overline{\delta}\ \texttt{target graph}]\!] & : \wp(\mathbb{G}) & = \{ & G & \mid \exists (G,\eta) \in [\![\overline{\delta}]\!]\}
\end{array}
$$

- A GDL program denotes a set of nodes (or edges, graphs)

$$[\![P]\!] \subseteq \text{Nodes}$$

How Our Approach Works

Training Data
(graphs w/ node labels)



How Our Approach Works

Training Data
(graphs w/ node labels)

Learning via program synthesis

$(l_1, P_1, 0.9)$
$(l_2, P_2, 0.8)$
$(l_1, P_3, 0.0)$

Model = A set of GDL programs

```
// GDL program P1
node x <[0.0, 0.5]>
node y
edge (x, y)
target node y
```

[0.0, 0.5] → target

```
// GDL program P2
node x
node y <[0.2, 0.7]>
edge (x, y)
target node x
```

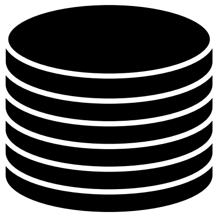
target → [0.2, 0.7]

```
// GDL program P3
node x
target node x
```

target

How Our Approach Works

Training Data
(graphs w/ node labels)



Learning via program synthesis

$(l_1, P_1, 0.9)$
$(l_2, P_2, 0.8)$
$(l_1, P_3, 0.0)$

The set of nodes having a predecessor whose feature value is between 0.0 and 0.5

odel = A set of GDL programs

```
// GDL program P1
node x <[0.0, 0.5]>
node y
edge (x, y)
target node y
```

[0.0, 0.5] → target

```
// GDL program P2
node x
node y <[0.2, 0.7]>
edge (x, y)
target node x
```

target → [0.2, 0.7]

```
// GDL program P3


node x
target node x
```

target

How Our Approach Works

Training Data
(graphs w/ node labels)

Learning via program synthesis

The set of nodes having a predecessor whose feature value is between 0.0 and 0.5

The set of nodes having a successor whose feature value is between 0.2 and 0.7

```
// GDL program P1
node x <[0.0, 0.5]>
node y
edge (x, y)
target node y
```

[0.0, 0.5] → target

```
// GDL program P2
node x
node y <[0.2, 0.7]>
edge (x, y)
target node x
```

target → [0.2, 0.7]

```
// GDL program P3
node x
target node x
```

target

How Our Approach Works

Training Data
(graphs w/ node labels)

Learning via program synthesis

The set of nodes having a predecessor whose feature value is between 0.0 and 0.5

```
// GDL program P1
node x <[0.0, 0.5]>
node y
edge (x, y)
target node y
```

[0.0, 0.5] → target

The set of nodes having a successor whose feature value is between 0.2 and 0.7

```
// GDL program P2
node x
node y <[0.2, 0.7]>
edge (x, y)
target node x
```

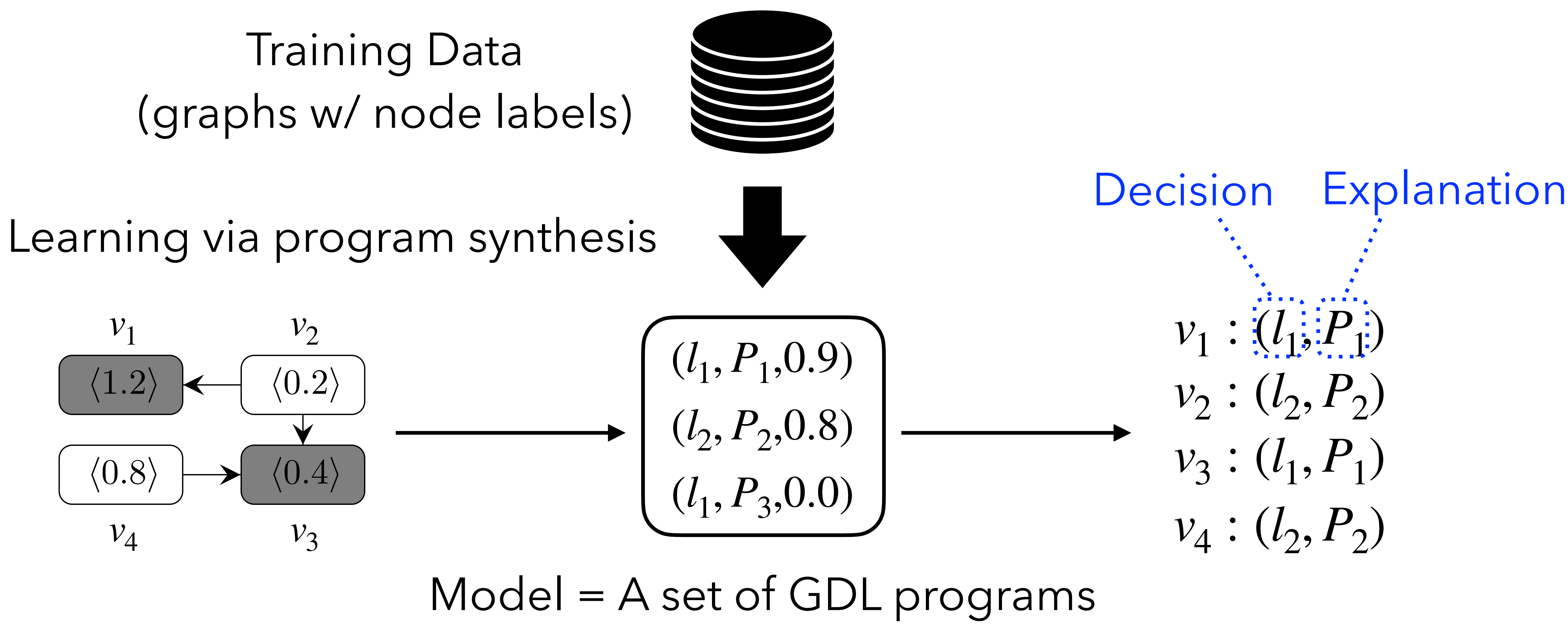
target → [0.2, 0.7]

All nodes in the given graph

```
// GDL program P3
node x
target node x
```

target

How Our Approach Works

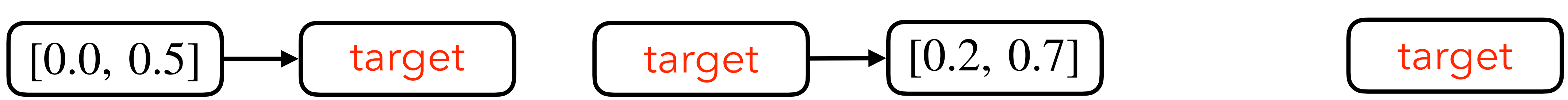



```
// GDL program P1
node x <[0.0, 0.5]>
node y
edge (x, y)
target node y
```

```
// GDL program P2
node x
node y <[0.2, 0.7]>
edge (x, y)
target node x
```

```
// GDL program P3


node x
target node x
```



Evaluation

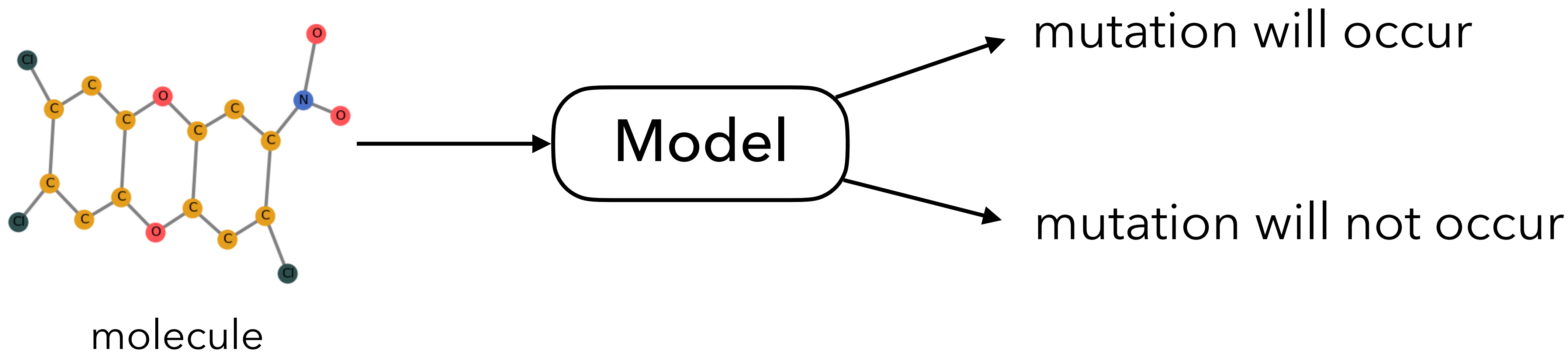
- Compared PL4XGL with
  - representative GNNs: GCN, GAT, GIN, etc
  - state-of-the-art GNN explainer, SubgraphX*
- Research questions:
  1. Classification accuracy
  2. Explanation quality
- Machines used:
  - GNNs trained and evaluated using a GPU (RTX A6000)
  - PL4XGL trained and evaluated using a 64-core CPU

---

*Yuan et al. On explainability of graph neural networks via subgraph explorations. ICML 2021

Datasets

- Four datasets for graph classification:
  - e.g., the MUTAG dataset (a set of molecule graphs)



- Eight datasets for node classification
  - e.g., the citation network datasets: Cora, Citeseer, Pubmed
- Each dataset is split into 8:1:1 for training, validation, and evaluation

Datasets

- Four datasets for graph classification:
    - e.g., the MUTAG dataset (a set of molecule graphs)

mutation will occur

	Graph classification				Node classification							
	Molecular datasets				Synthetic datasets		Web page datasets			Citation networks		
	MUTAG	BBBP	BACE	HIV	BA-Shapes	Tree-Cycles	Wisconsin	Texas	Cornell	Cora	Citeseer	Pubmed
# Graphs	188	2,039	1,513	41,127	1	1	1	1	1	1	1	1
# Nodes (avg)	17.9	24.0	34.0	25.5	700	871	183	183	251	2,708	3,327	19,717
# Edges (avg)	19.7	25.9	36.8	27.5	2,055	971	450	279	277	5,278	4,552	44,324
# Labels	2	2	2	2	4	2	5	5	5	7	6	3
# Node features	1	9	9	9	1	1	1,703	1,703	1,703	1,433	3,703	500
# Edge features	1	3	3	3	0	0	0	0	0	0	0	0

-
    - e.g., the citation network datasets: Cora, Citeseer, Pubmed
- Each dataset is split into 8:1:1 for training, validation, and evaluation

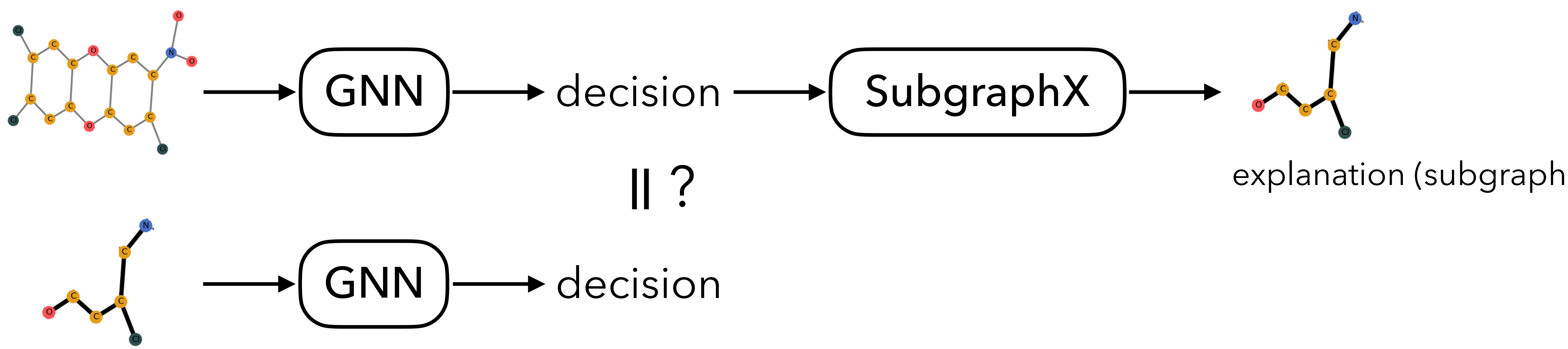
(1) Classification Accuracy

- Overall, PL4XGL can compete with GNNs
  - For 5 datasets, achieved the best accuracy (e.g., 100% for MUTAG)
- For the largest benchmark (HIV), PL4XGL did not scale (48 hours)

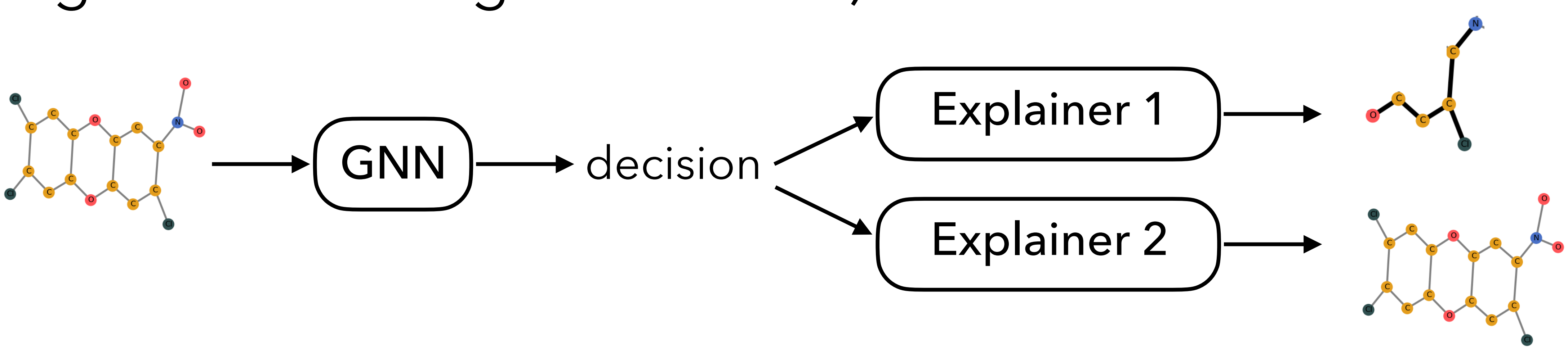
	GCN	GAT	ChebyNet	JKNet	GraphSage	GIN	DGCN	PL4XGL
MUTAG	80.0±0.0	89.0±2.2	86.0±4.1	68.0±7.5	78.0±4.4	91.0±5.4	N/A	**100.0±0.0**
BBBP	83.6±1.4	82.3±1.6	84.6±1.0	85.6±1.9	86.6±0.9	86.2±1.4	N/A	**86.8±0.0**
BACE	78.4±2.8	52.4±3.3	78.9±1.4	79.9±1.9	79.8±0.8	**80.9±0.4**	N/A	**80.9±0.0**
HIV	96.4±0.0	96.4±0.0	96.8±0.2	96.8±0.1	**96.9±0.2**	96.8±0.1	N/A	N/A
BA-Shapes	95.1±0.6	76.8±2.3	**97.1±0.0**	94.3±0.0	**97.1±0.0**	92.0±1.1	95.1±0.7	95.7±0.0
Tree-Cycles	97.7±0.0	90.9±0.0	**100.0±0.0**	98.9±0.0	**100.0±0.0**	93.2±0.0	99.2±0.5	**100.0±0.0**
Wisconsin	64.0±0.0	49.6±3.1	86.4±3.9	64.8±1.5	92.8±2.9	56.0±0.0	**96.0±0.0**	88.0±0.0
Texas	67.7±5.3	50.0±0.0	87.7±2.1	68.8±4.3	86.6±2.6	50.0±0.0	**86.6±2.6**	83.3±0.0
Cornell	58.9±2.6	61.1±0.0	81.0±6.5	61.1±0.0	87.7±2.1	61.1±0.0	86.6±2.6	**88.8±0.0**
Cora	85.6±0.3	86.4±1.8	86.5±5.2	84.9±3.5	86.3±3.2	**86.7±0.0**	83.2±5.9	80.0± 0.0
Citeseer	75.2±0.0	74.3±0.7	**79.1±0.9**	73.7±4.2	75.9±2.3	75.2±0.0	71.3±6.0	63.8± 0.0
Pubmed	82.8±1.1	84.7±1.2	**88.7±1.0**	83.2±0.4	88.0±0.4	86.1±0.6	85.1±0.6	81.4±0.0

(2) Explanation Quality

- **Fidelity** quantifies the correctness of explanations (in range 0 and 1 – lower is better)

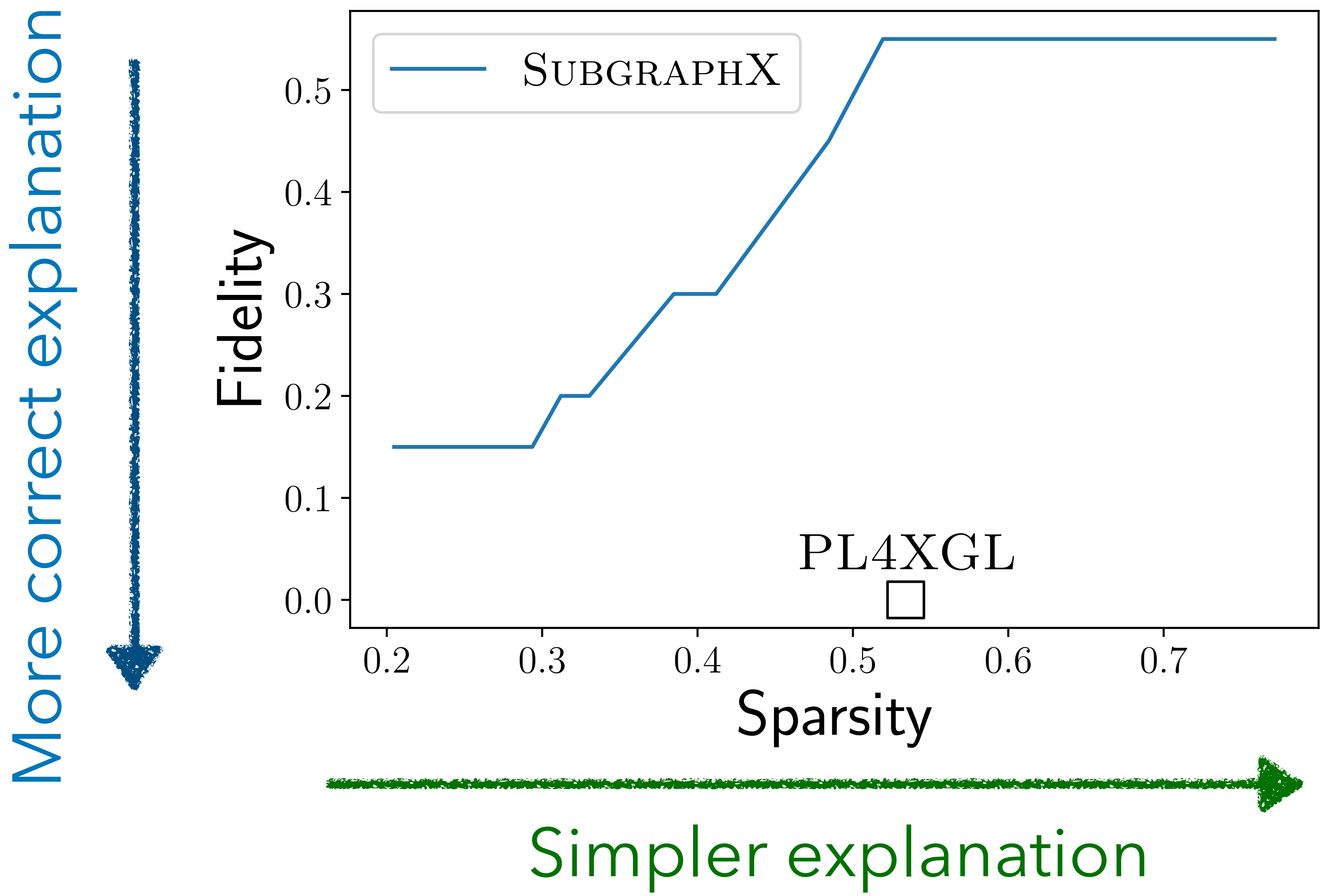


- **Sparsity** quantifies the simplicity (size) of explanations (in range 0 and 1 – higher is better)



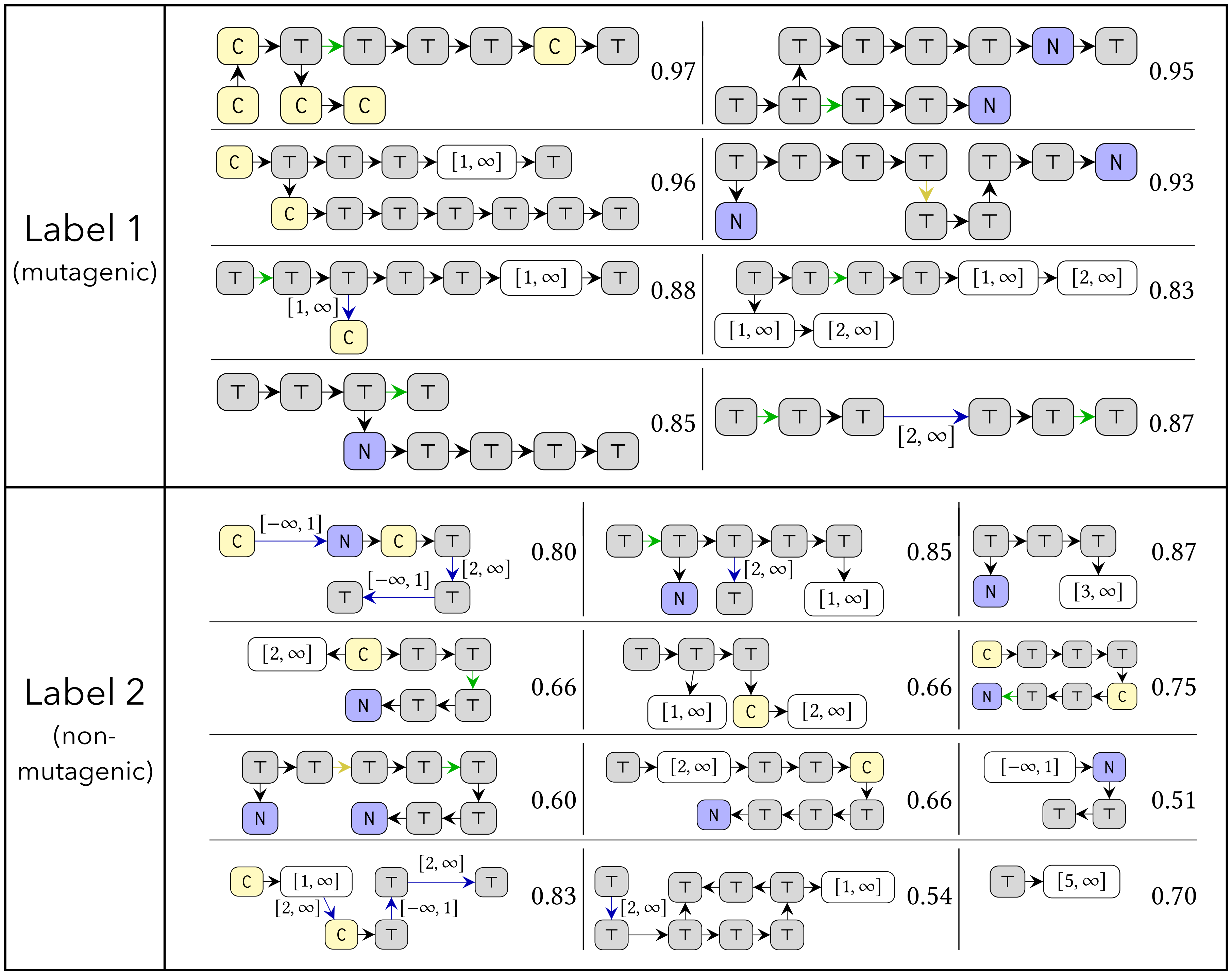
(2) Explanation Quality

- PL4XGL produced better explanations than SubgraphX
- E.g., graph classification on the MUTAG dataset



Human-Readable Models

- E.g., the learned model for MUTAG (20 GDL programs)



Summary

- Problem: Accurate and explainable graph learning
- Solution: A purely PL-based approach to XAI
  - Domain-specific languages for defining AI models
  - Program synthesis for learning model programs from data
- Result:
  - Accuracy can compete with GNNs
  - Better explainability than GNNs with post-hoc explainer

Summary

- Problem: Accurate and explainable graph learning
- Solution: A purely PL-based approach to XAI
  - Domain-specific languages for defining AI models
  - Program synthesis for learning model programs from data
- Result:
  - Accuracy can compete with GNNs
  - Better explainability than GNNs with post-hoc explainer

Conclusion: PL techniques are useful even for AI!

Backup Slides

Training / Inference Cost

Dataset	Cost (minutes)	GNN+ SubgraphX	PL4XGL	Dataset	Cost (minutes)	GNN+ SubgraphX	PL4XGL
MUTAG	Training	**0.2**	12.3	Wisconsin	Training	**0.4**	8.0
	Classification	**0.1**	**0.1**		Classification	**0.1**	**0.1**
	Explanation	8.4	**0.0**		Explanation	69.3	**0.0**
	Total	**8.7**	12.4		Total	69.5	**8.1**
BBBP	Training	**1.0**	34.3	Texas	Training	**0.4**	5.0
	Classification	**0.1**	0.7		Classification	**0.1**	**0.1**
	Explanation	160.0	**0.0**		Explanation	52.1	**0.0**
	Total	161.1	**35.0**		Total	52.3	**5.1**
BACE	Training	**1.0**	60.6	Cornell	Training	**0.3**	5.0
	Classification	**0.1**	4.0		Classification	**0.1**	**0.1**
	Explanation	141.1	**0.0**		Explanation	95.8	**0.0**
	Total	142.2	**69.9**		Total	96.0	**5.1**
HIV	Training	12.2	timeout	Cora	Training	**0.4**	61.6
	Classification	0.1	N/A		Classification	**0.1**	0.9
	Explanation	2887.8	N/A		Explanation	timeout	**0.0**
	Total	2900.1	timeout		Total	timeout	62.5
BA-Shapes	Training	**0.1**	0.2	Citeseer	Training	**0.4**	245.2
	Classification	**0.1**	**0.1**		Classification	**0.1**	2.0
	Explanation	4756.0	**0.0**		Explanation	timeout	**0.0**
	Total	4756.2	**0.2**		Total	timeout	**247.2**
Tree-Cycles	Training	**0.1**	0.2	Pubmed	Training	**0.6**	2702.9
	Classification	**0.1**	**0.1**		Classification	**0.1**	17.0
	Explanation	3.4	**0.0**		Explanation	timeout	**0.0**
	Total	3.6	**0.2**		Total	timeout	**2719.9**

General Methodology

- In principle, applicable to general classification tasks
  - $\mathbb{I}$: instances (e.g., nodes)
  - $\mathbb{L}$: labels (e.g., node labels)
  - $D \in \wp(\mathbb{I} \times \mathbb{L})$: training data

Goal: Learn a classifier $f : \mathbb{I} \rightarrow \mathbb{L}$ from D

General Methodology

- Model = Programs in domain-specific language $\mathbb{P}$
  - A program $P \in \mathbb{P}$ denotes a set of instances:

$$[\![P]\!] \in \wp(\mathbb{I})$$

  - Our language-based model:

$$\mathcal{M} \in \mathbb{M} = \wp(\mathbb{L} \times \mathbb{P} \times [0,1])$$

  - Our classifier:

$$f_{\mathcal{M}} : \mathbb{I} \to \mathbb{L} \times \mathbb{P}$$

Given $i \in \mathbb{I}$, $f_{\mathcal{M}}(i)$ returns $(l, P, \psi) \in \mathcal{M}$ with highest ψ

General Methodology

- Learning is formulated as program synthesis

$$Learn : \wp(\mathbb{I} \times \mathbb{L}) \rightarrow \mathbb{M}$$

- Goal is to synthesize programs in $\mathscr{M}$ from D, maximizing classification accuracy over the training data

- We use a variant of search-based synthesis algorithms