Explainable Deep Learning

When, Why and How

Ahmad Haj Mosa

ahmad.haj.mosa@pwc.com

LinkedIn: ahmad-haj-mosa

Fabian Schneider

schneider.fabian@pwc.com

LinkedIn: fabian-schneider-24122379

The definition of intelligence

Our minds contain processes that enable us to solve problems we consider difficult.

"Intelligence" is our name of those processes we don't yet understand.

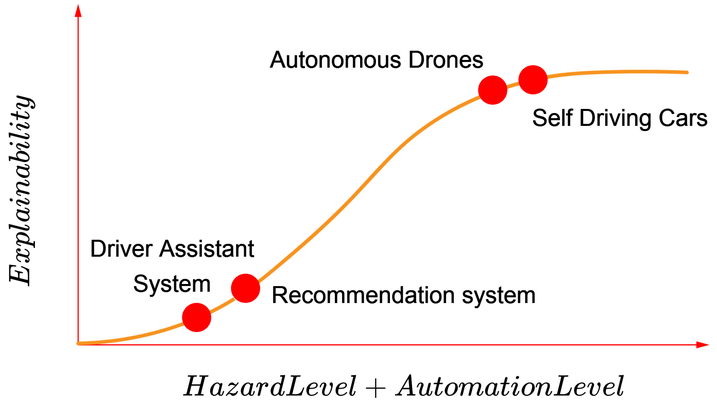
Marvin Minsky

Interpretation vs explanation

Interpretation	Explanation
Between AI and its designer	Between AI and its consumer
External analytics	Learned by the model
Model specific or agnostic	Model specific
Local or global	local
examples: LIME and TSNE	examples: Attention Mechanism

The need for XAI depends on the task, domain and process an AI is automating

The need for XAI depends on the legal, life and cost risk of the automated process



Hazard – a state or set of conditions of a system (or an object) that, together with other conditions in the environment of the system (or object), will lead inevitably to an accident (loss event).

source: USPAS

Why XAI?

Classification: wolf or husky?

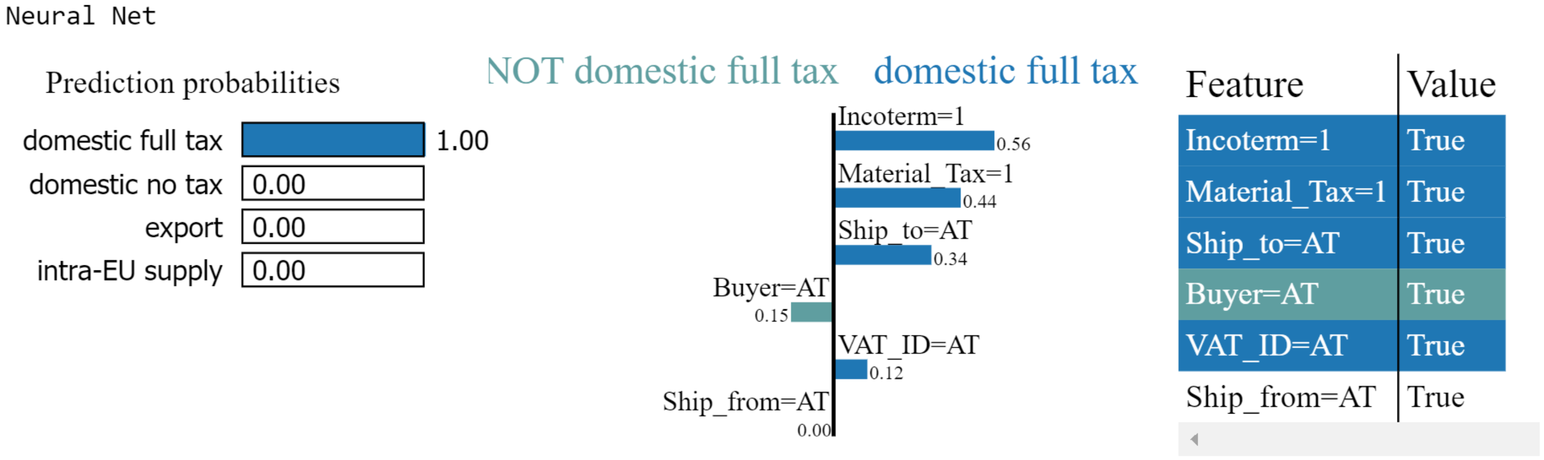
(a) Husky classified as wolf

(b) Explanation

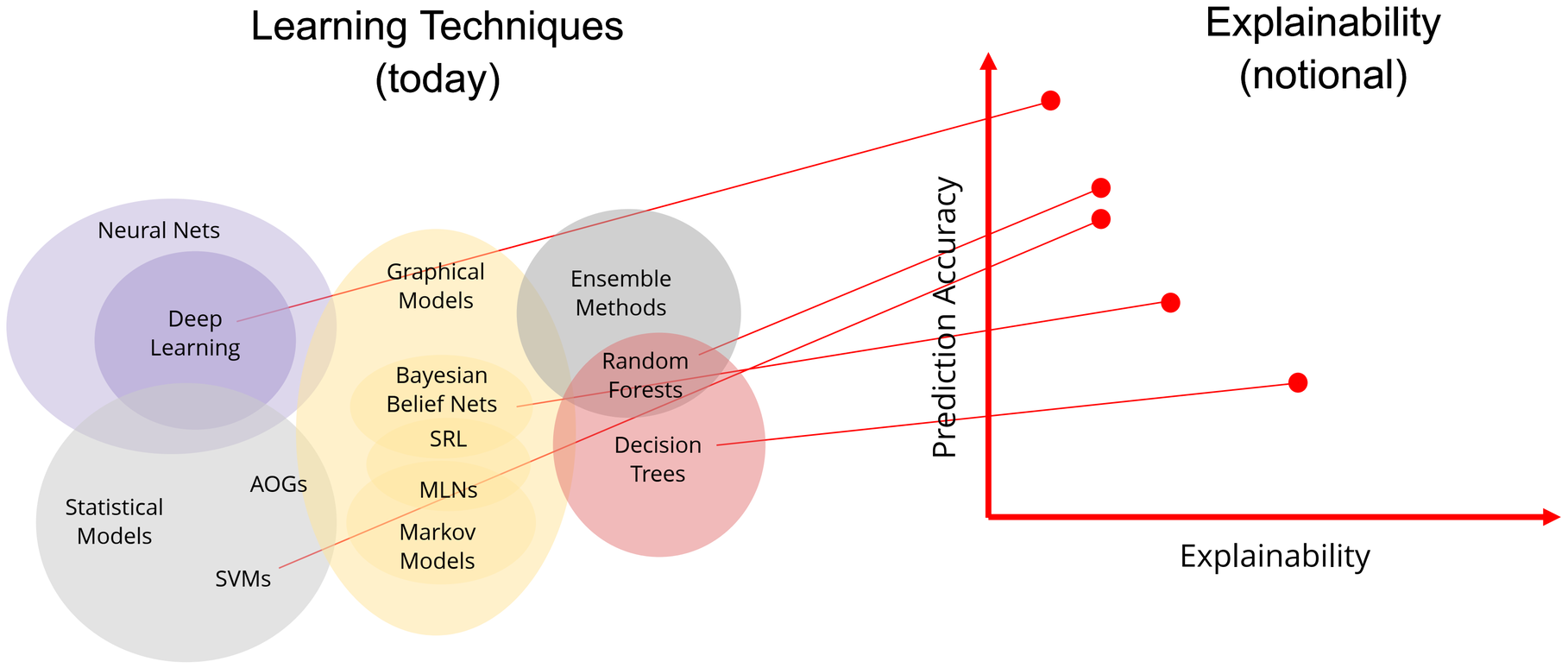
source: “Why Should I Trust You?” Explaining the Predictions of Any Classifier

Classification of Value Added Tax

The first rule is: if your are shipping a product from Austria to Austria and the shipped material is according to law taxable, then the Taxcode is: domestic full tax -> 20%

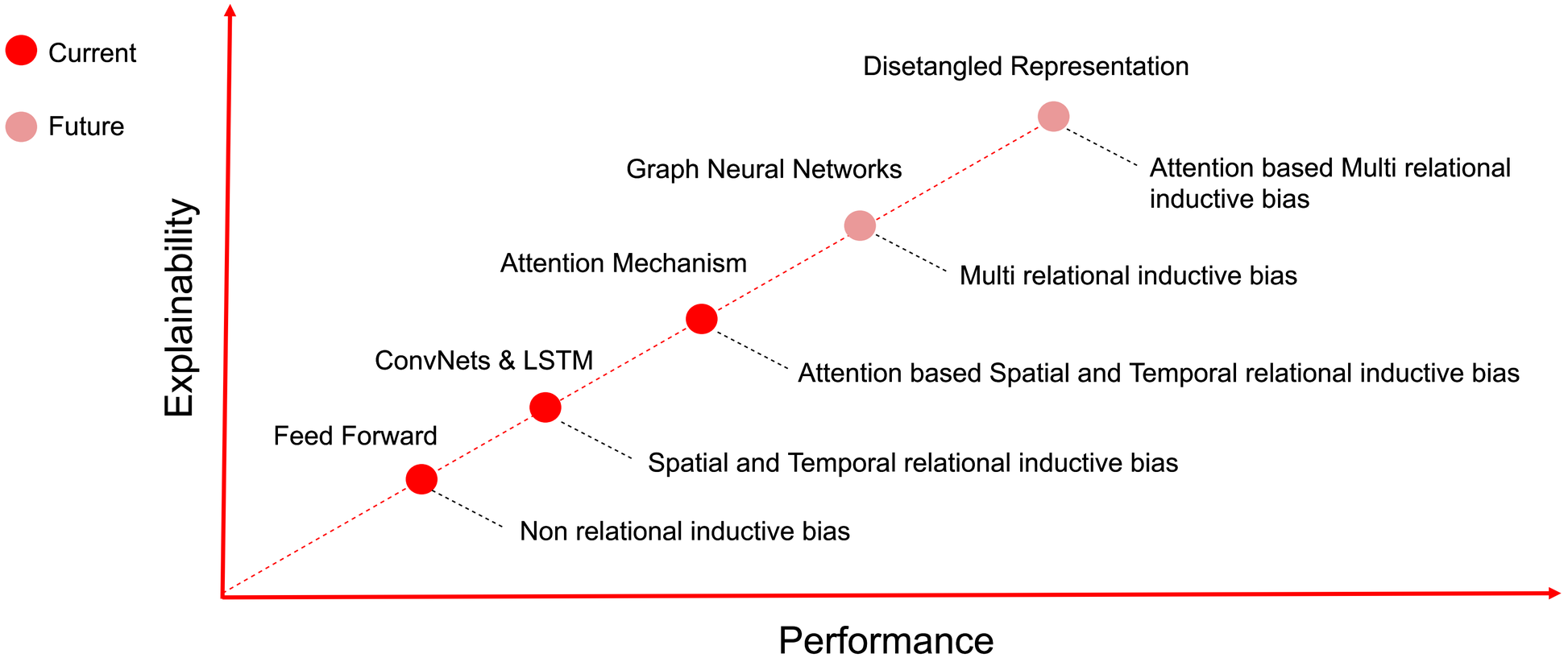


Is it explainability vs accuracy?



source: DARPA

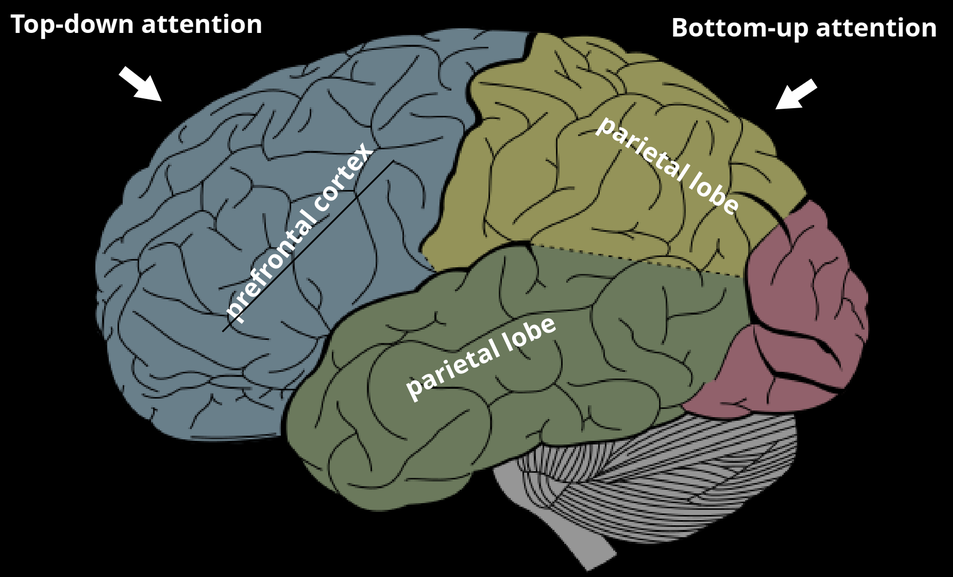
Is it explainability vs accuracy?



Relation between abstraction and attention

Attention allows us to focus on few elements out of a large set

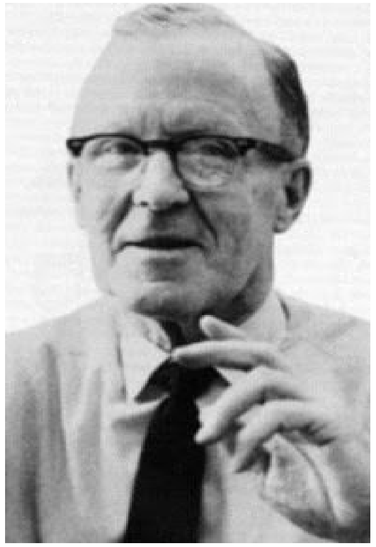
Attention focus on a few appropriate abstract or concrete elements of mental representation



source: Yoshua Bingo

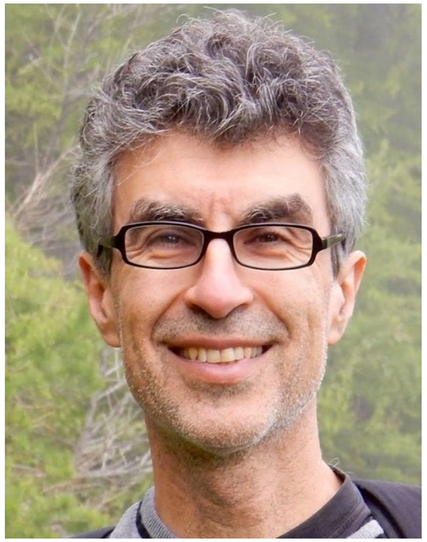
What is next in Deep Learning?

Research inspiration and references



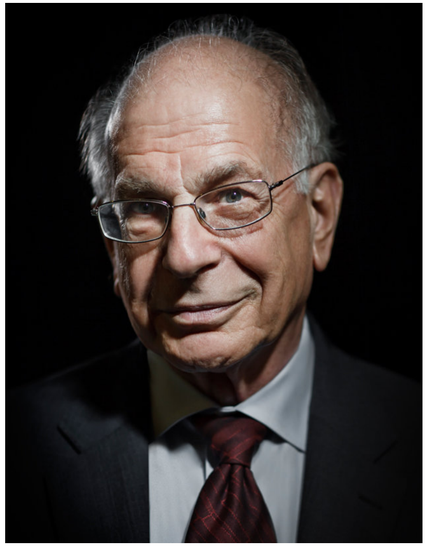
Donald O. Hebb

Hebbian Learning



Yoshua Bengio

Disetangled Representation



Daniel Kahneman

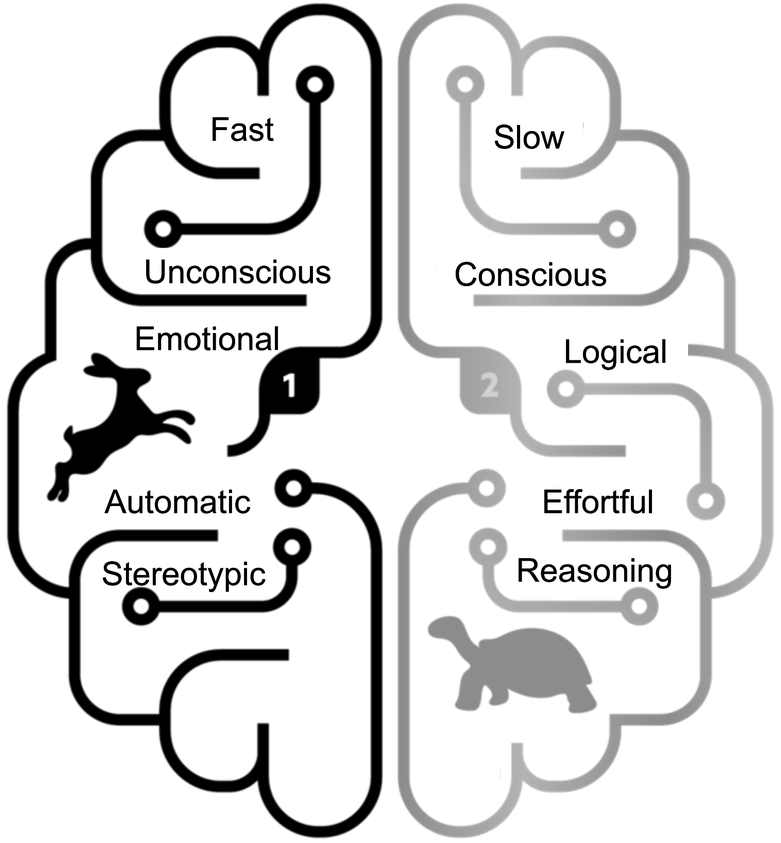
Thinking fast and slow

Marvin Minsky

Knowledge Representation

Thinking fast and slow

System 1	System 2
drive a car on highways	drive a car in cities
come up with a good chess move (if you're a chess master)	point your attention towards the clowns at the circus
understands simple sentences	understands law clauses
correlation	causation
hard to explain	easy to explain



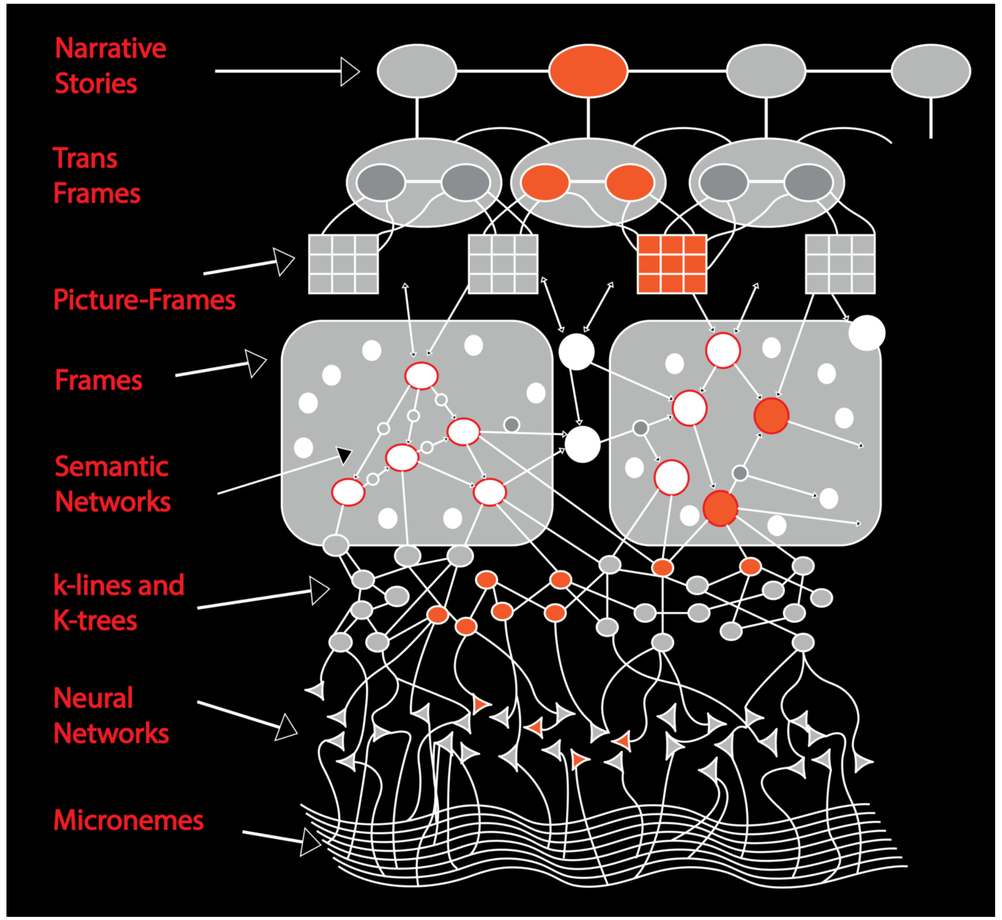
source: Thinking fast and slow by Daniel Kahneman

A framework for representing knowledge

"A frame is a data structure with typical knowledge about a particular object or concept."

"When you "get an idea," or "solve a problem" ... you create what we shall call a K-line. ... When that K-line is later "activated", it reactivates ... mental agencies, creating a partial mental state "resembling the original."

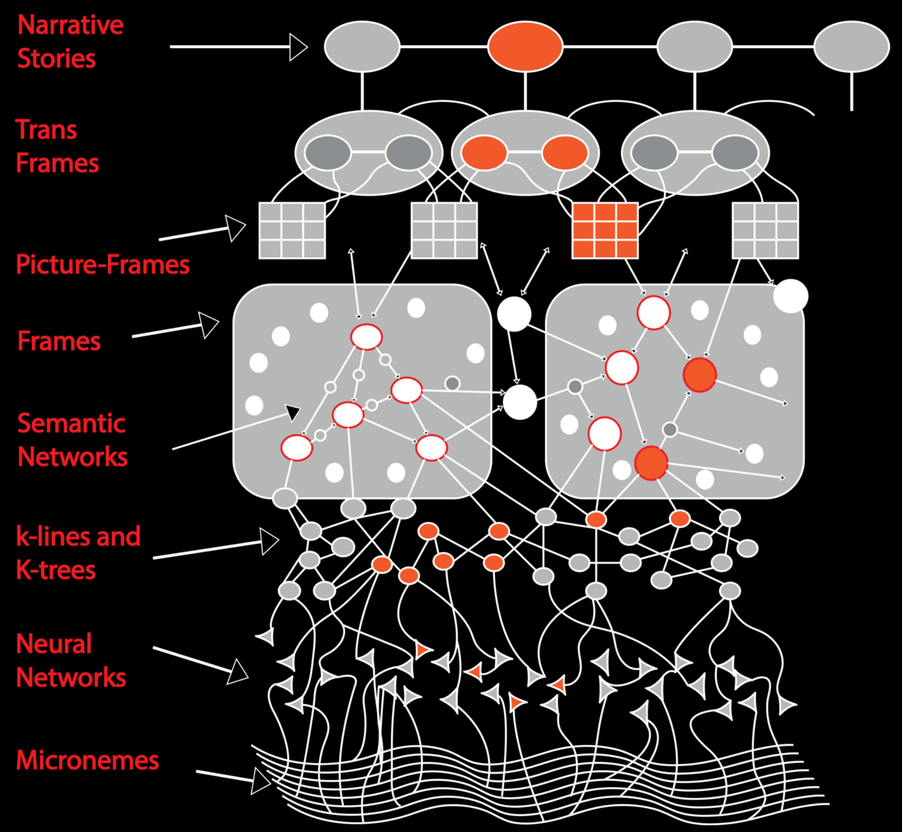
source: The Emotion Machine, Marvin Minsky



A framework for representing knowledge

easy
explanation
learning fast

hard
explanation
learning slow

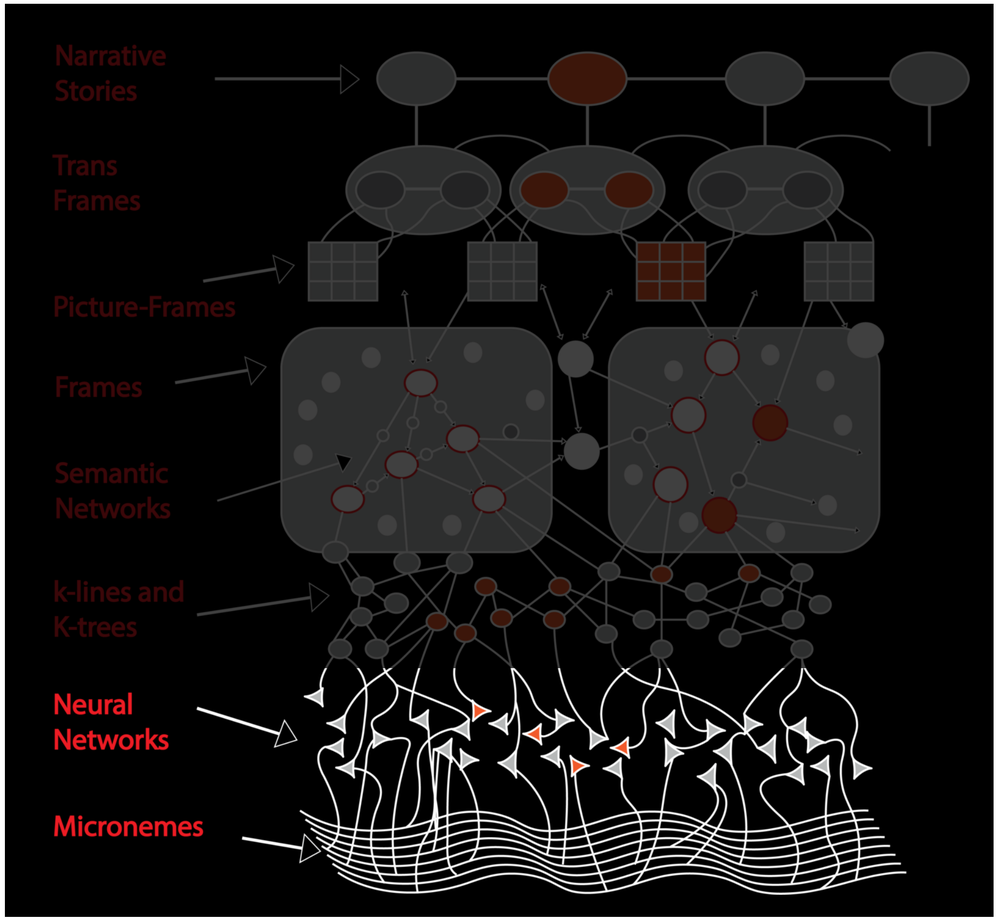


thinking slow

consciousness prior

thinking fast

A framework for representing knowledge



Trend 1 for XDL

Train a readable/visible representation before training a decision.

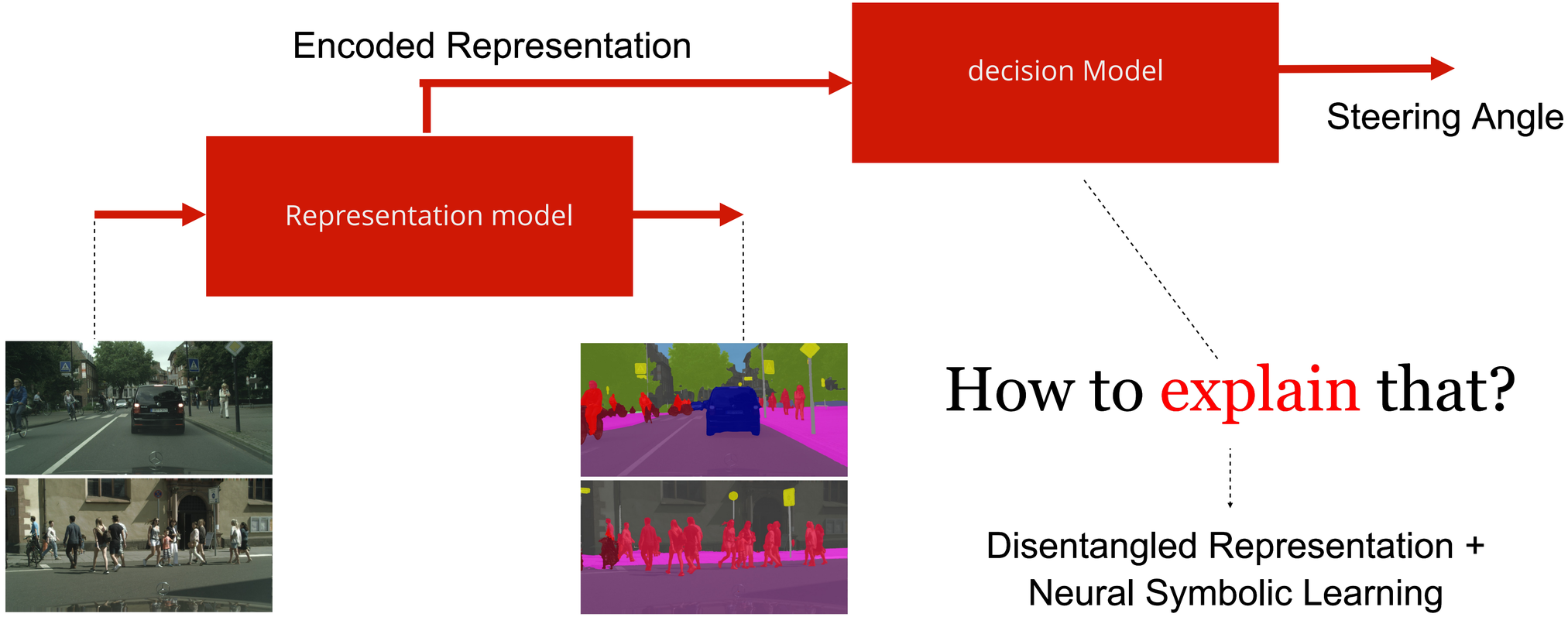
Representation learning

Representation model

Decision model

Steering Angle

Representation then decision



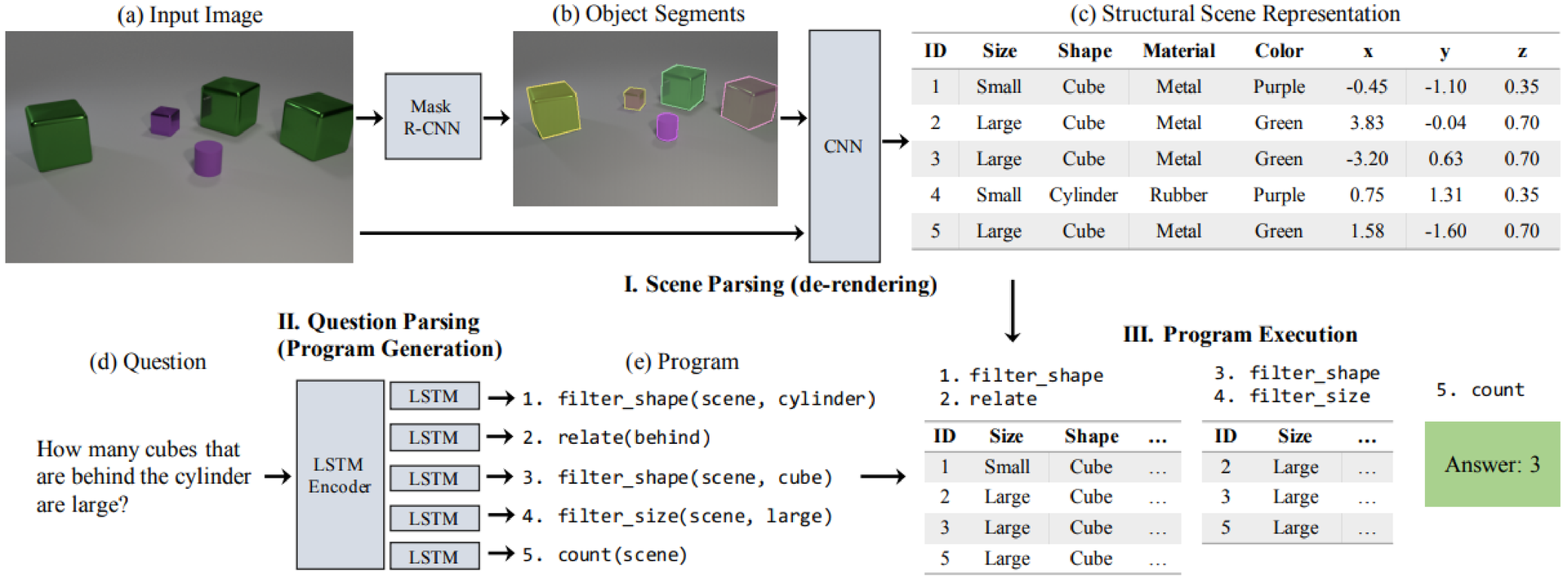
Trend 2 for XDL

Object-oriented-learning

instead of

End-end learning.

Neural symbolic and disentangled reasoning



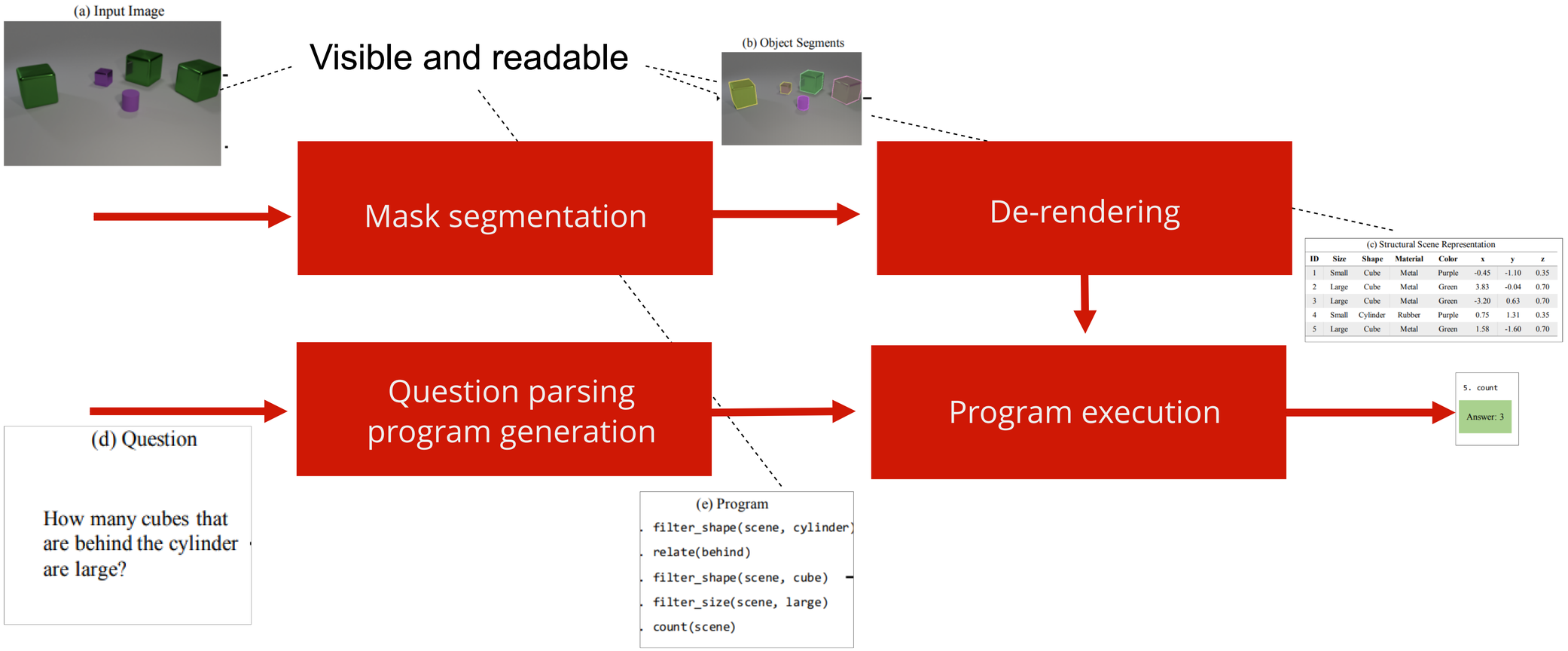
ID	Size	Shape	Material	Color	x	y	z
1	Small	Cube	Metal	Purple	-0.45	-1.10	0.35
2	Large	Cube	Metal	Green	3.83	-0.04	0.70
3	Large	Cube	Metal	Green	-3.20	0.63	0.70
4	Small	Cylinder	Rubber	Purple	0.75	1.31	0.35
5	Large	Cube	Metal	Green	1.58	-1.60	0.70

ID	Size	Shape	...
1	Small	Cube	...
2	Large	Cube	...
3	Large	Cube	...
5	Large	Cube	...

ID	Size	...
2	Large	...
3	Large	...
5	Large	...

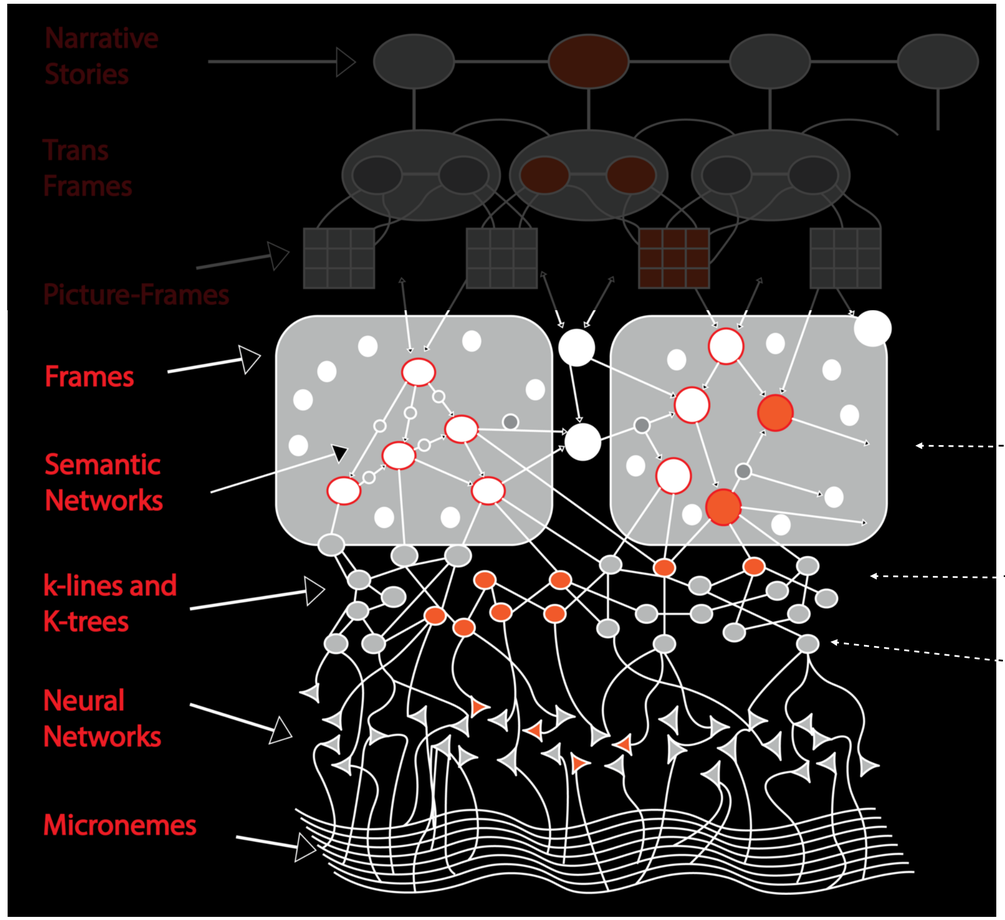
source: arXiv:1810.02338: Neural-Symbolic VQA: Disentangling Reasoning from Vision and Language Understanding

Neural symbolic and disentangled reasoning



A framework for representing knowledge

"When you "get an idea," or "solve a problem" ... you create what we shall call a K-line. ... When that K-line is later "activated", it reactivates ... mental agencies, creating a partial mental state "resembling the original."



Consciousness priors

Program generation

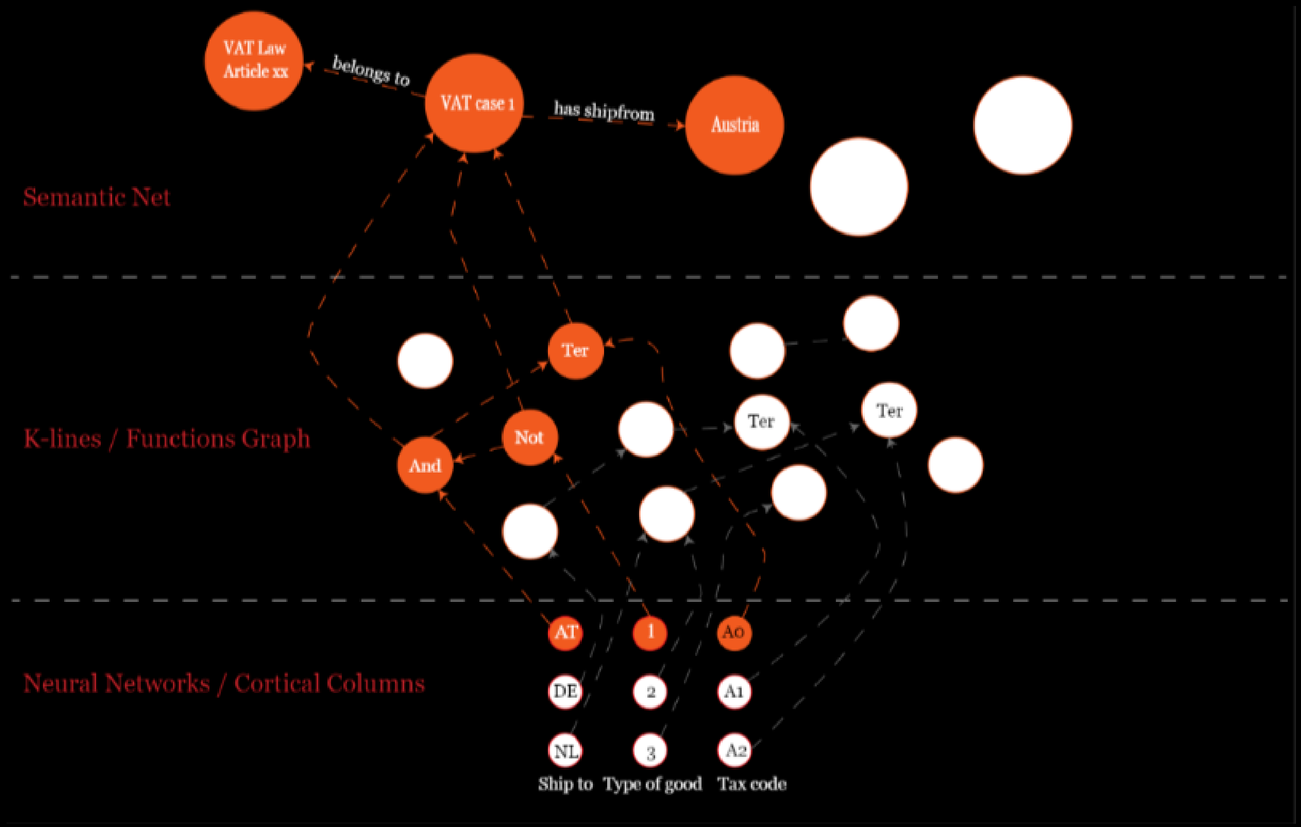
Disentangled, distributed or semantic representation

Tax-VAT data

Explaining easy

Thinking slow

Learning fast



Consciousness priors / attention

Program generation

Representation

Explaining hard

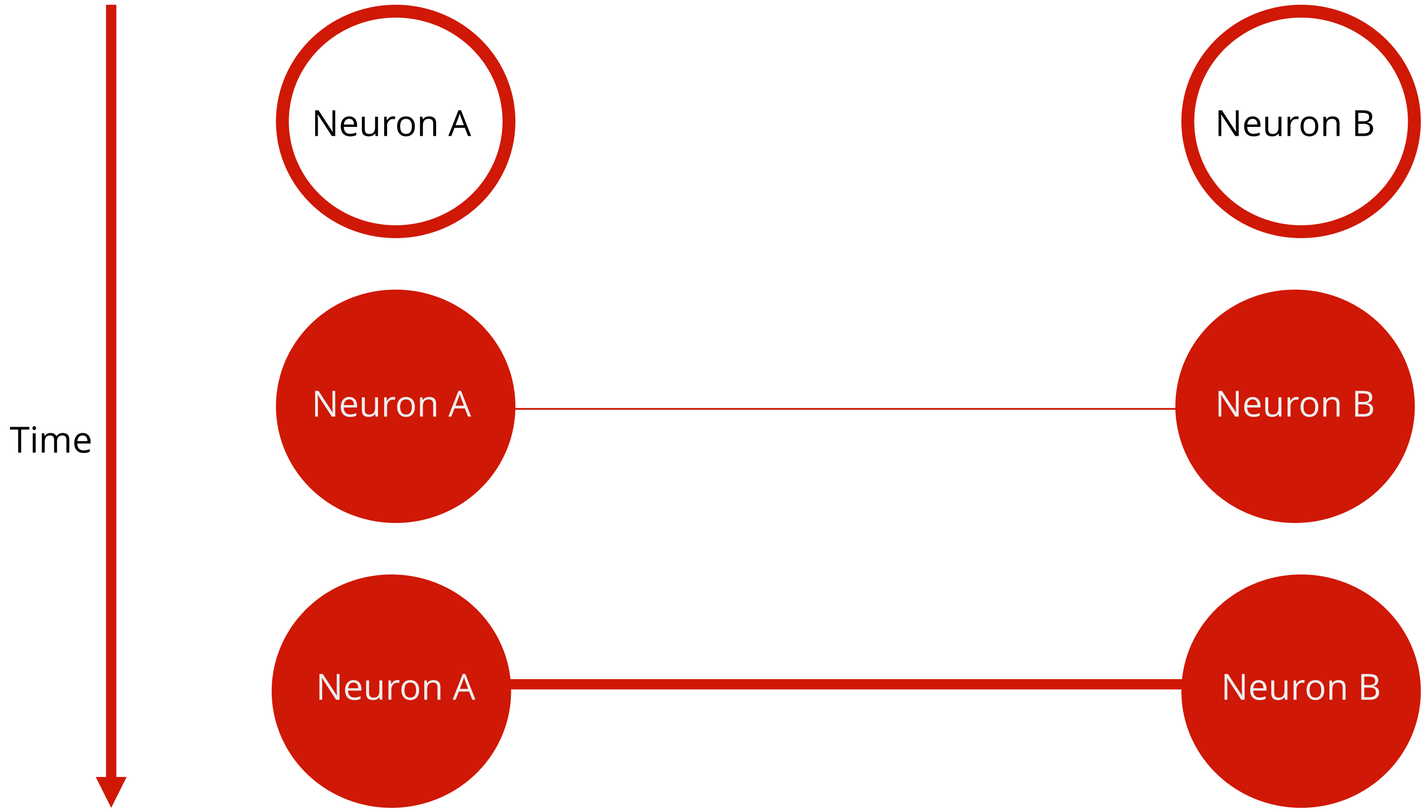
Thinking fast

Learning slow

Program generation using Hebbian Theory

"Any two cells or systems of cells that are repeatedly active at the same time will tend to become 'associated', so that activity in one facilitates activity in the other."

Hebbian Theory



Program generation using Hebbian Theory

"When one cell repeatedly assists in firing another, the axon of the first cell develops synaptic knobs (or enlarges them if they already exist) in contact with the soma *of the second cell."*

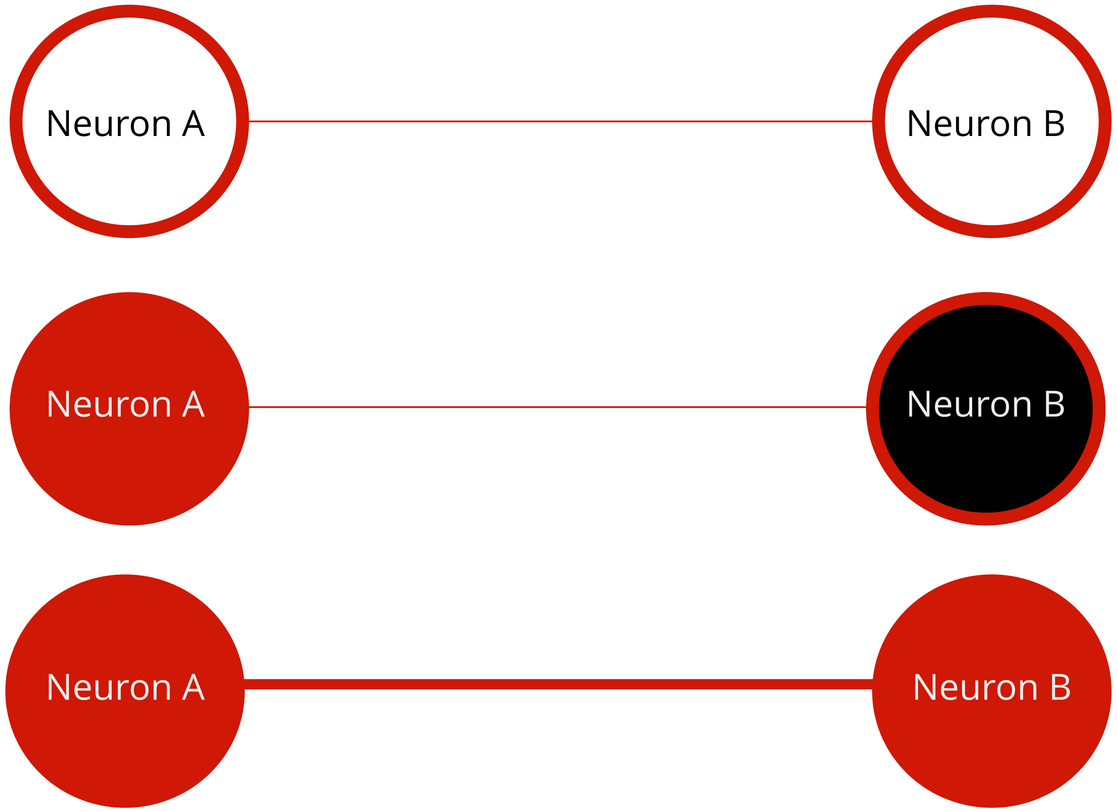
Program generation using Hebbian Theory

If Neuron A is active and conntected to Neuron B, then Neuron A will put Neuron B in a predicted state

If the prediction is correct (Neuron B is observed in an active state), then reward/increase the connection between them

If the prediction is wrong (Neuron B is not observed in an active state), then weaken/decrease the connections between them

Time



Hebbian Reinforce Theory

Find the optimum programs path using the REINFORCE algorithm

The state represents the graph and the cells state (active, predictive and inactive)

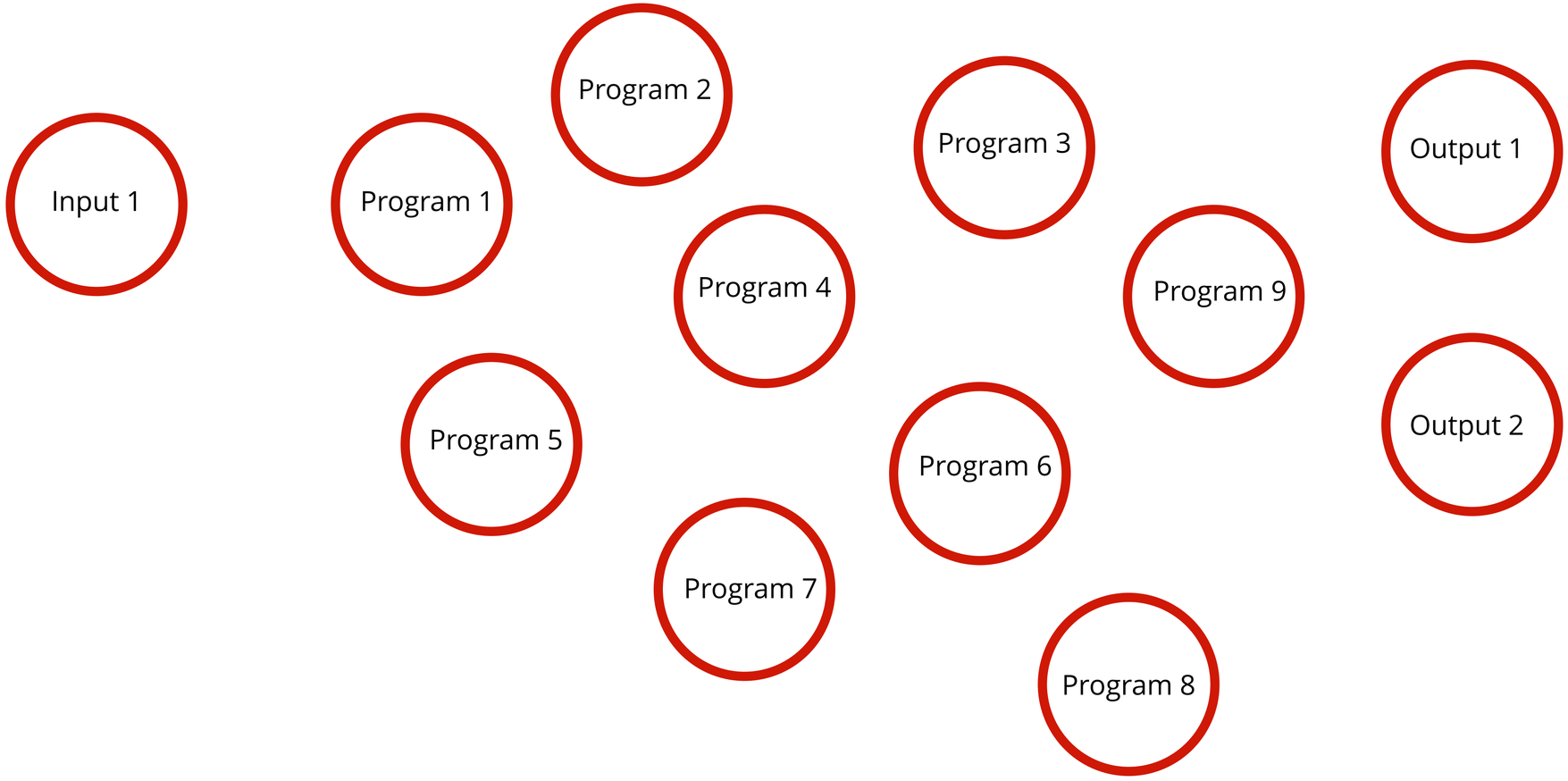
Actions are binaries (connect or disconnet two programs)

Reward the winner path and weaken the loser paths

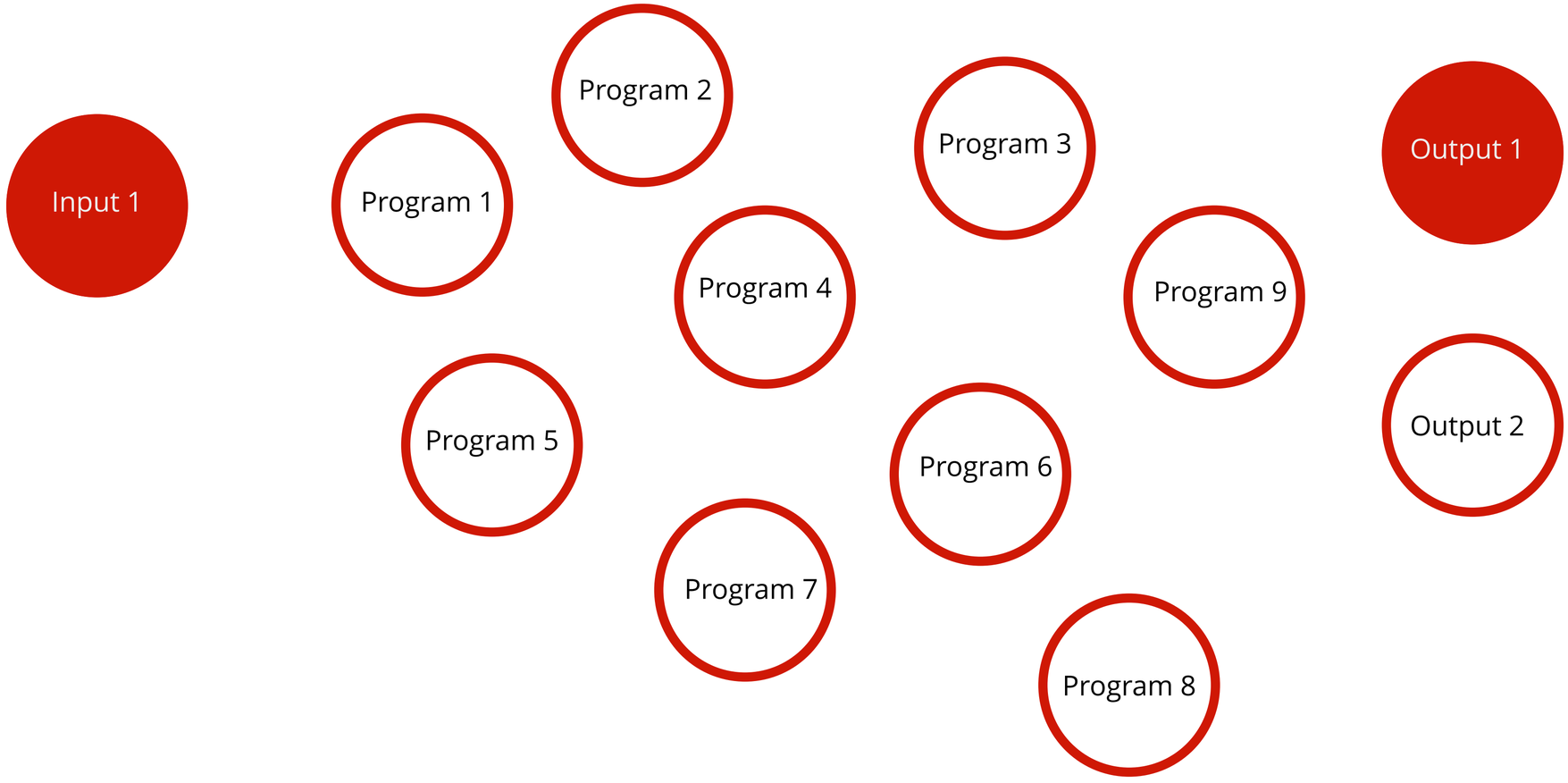
(e) Program

```
filter_shape(scene, cylinder)
relate(behind)
filter_shape(scene, cube)
filter_size(scene, large)
count(scene)
```

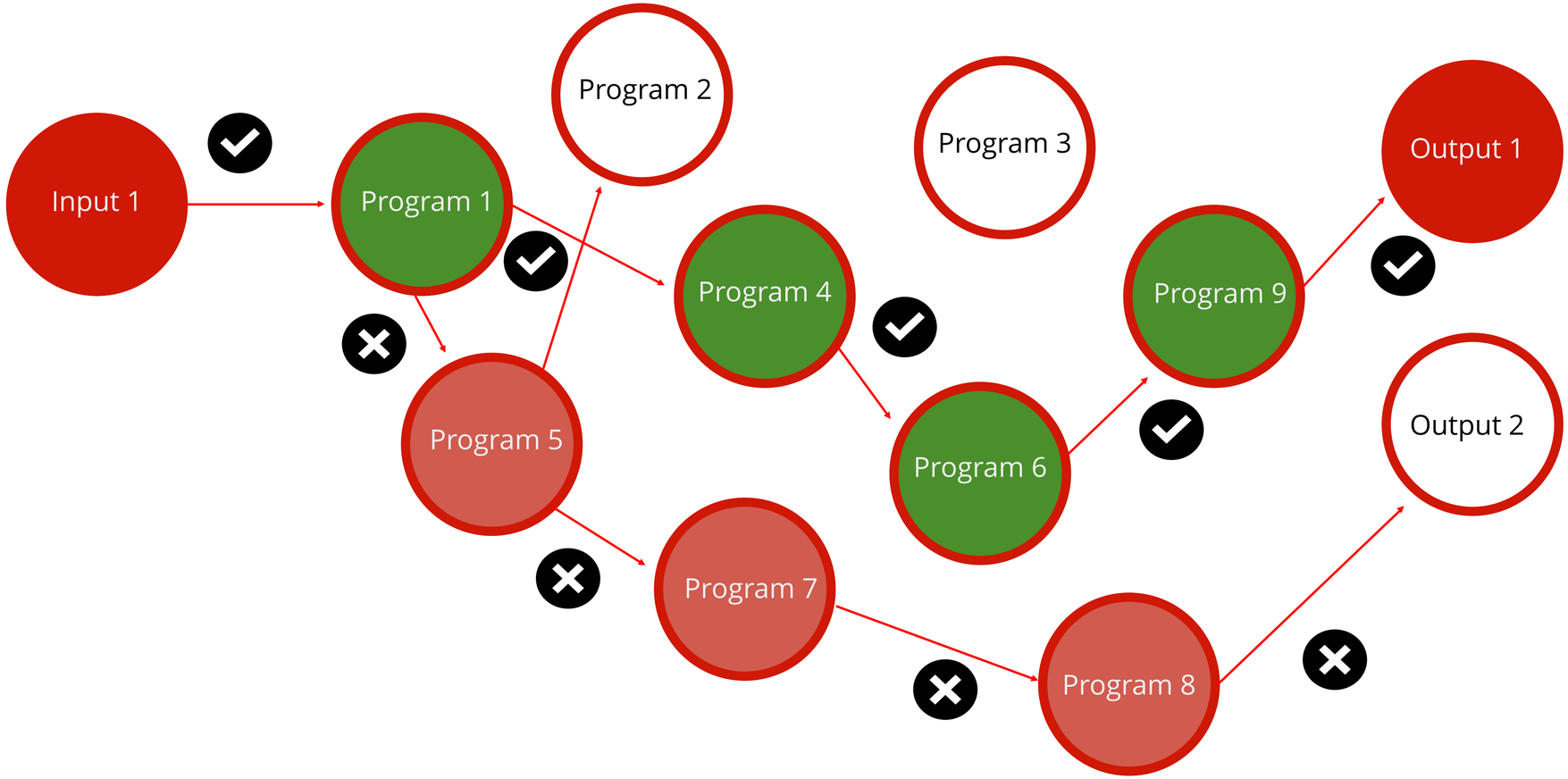
Hebbian Reinforce Theory



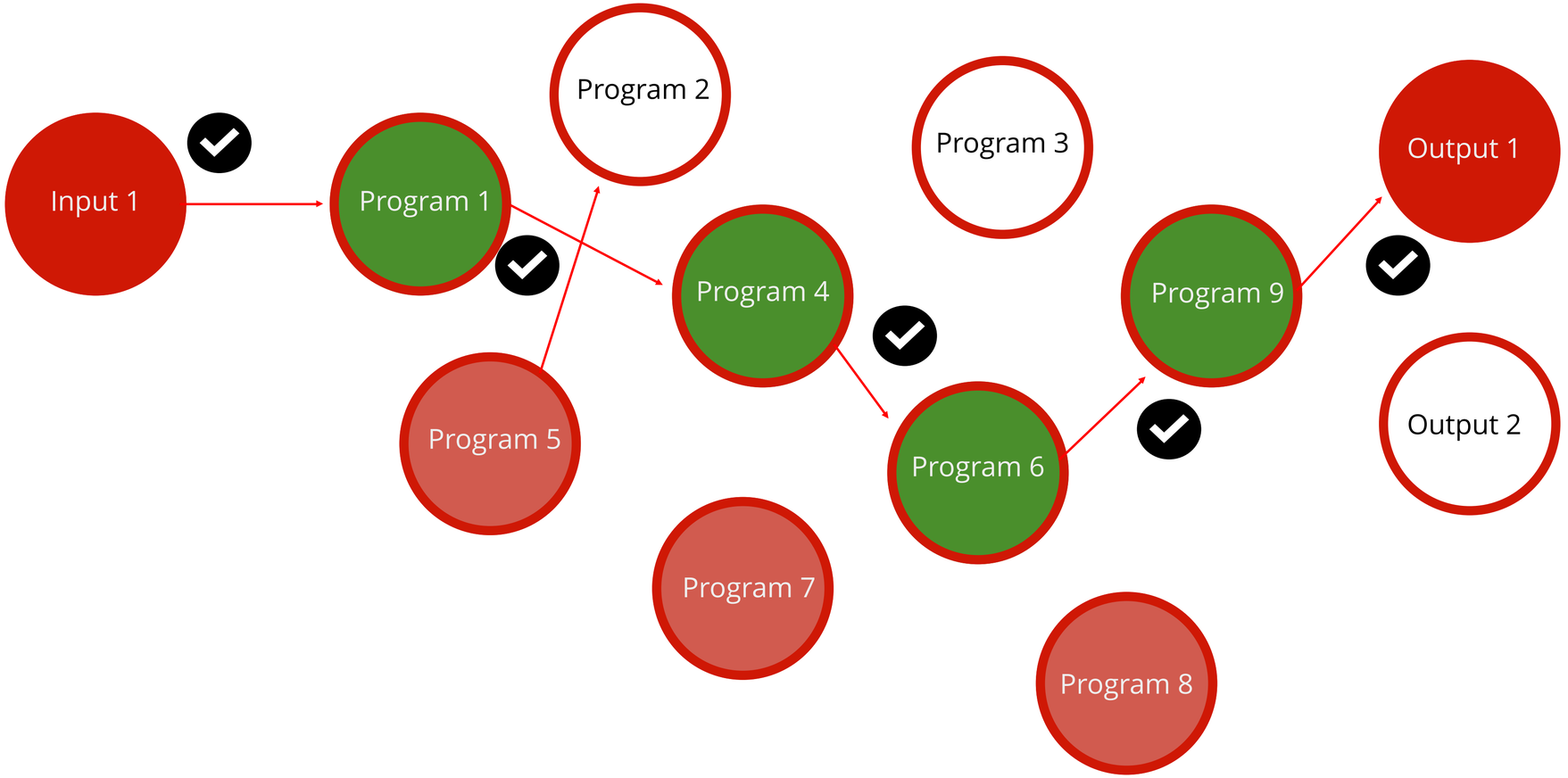
Hebbian Reinforce Theory



Hebbian Reinforce Theory



Hebbian Reinforce Theory



Function Composition

Function Composition

$$f \circ g \; x = f\,(\,g\,(\,x\,)\,)$$

Function Composition

$$f \circ g \; x = f\,(g\,(x))$$

$$\varphi = f \circ g = f\,(g\,(?))$$

φ not applied to input yet

Function Composition

$$f \circ g \; x = f(g(x))$$

$$\varphi = f \circ g = f(g(?))$$

φ not applied to input yet

$$\varphi(x) = f \circ g \; x = f(g(x))$$

apply φ to input so we get result

Function Composition

Function Composition

$$\varphi\,(x) \;=\; f \circ g \;\; (x, y)\, y \;=\; f\,(\,g\,(\,x, y\,)\,)\, y$$

Function Composition

$$\varphi\,(x) \;=\; f \circ g \;\; (x, y)\, y \;=\; f\,(\,g\,(\,x, y\,)\,)\, y$$

$$\varphi\,(x) \;=\; f \circ g \;\; (x, ?)\, ? \;=\; f\,(\,g\,(\,x, ?\,)\,)\, ?$$

apply φ to input so we get result

Function Composition

$$\varphi\,(x) \;=\; f \circ g \;\; (x, y)\, y \;=\; f\,(\,g\,(\,x, y\,)\,)\, y$$

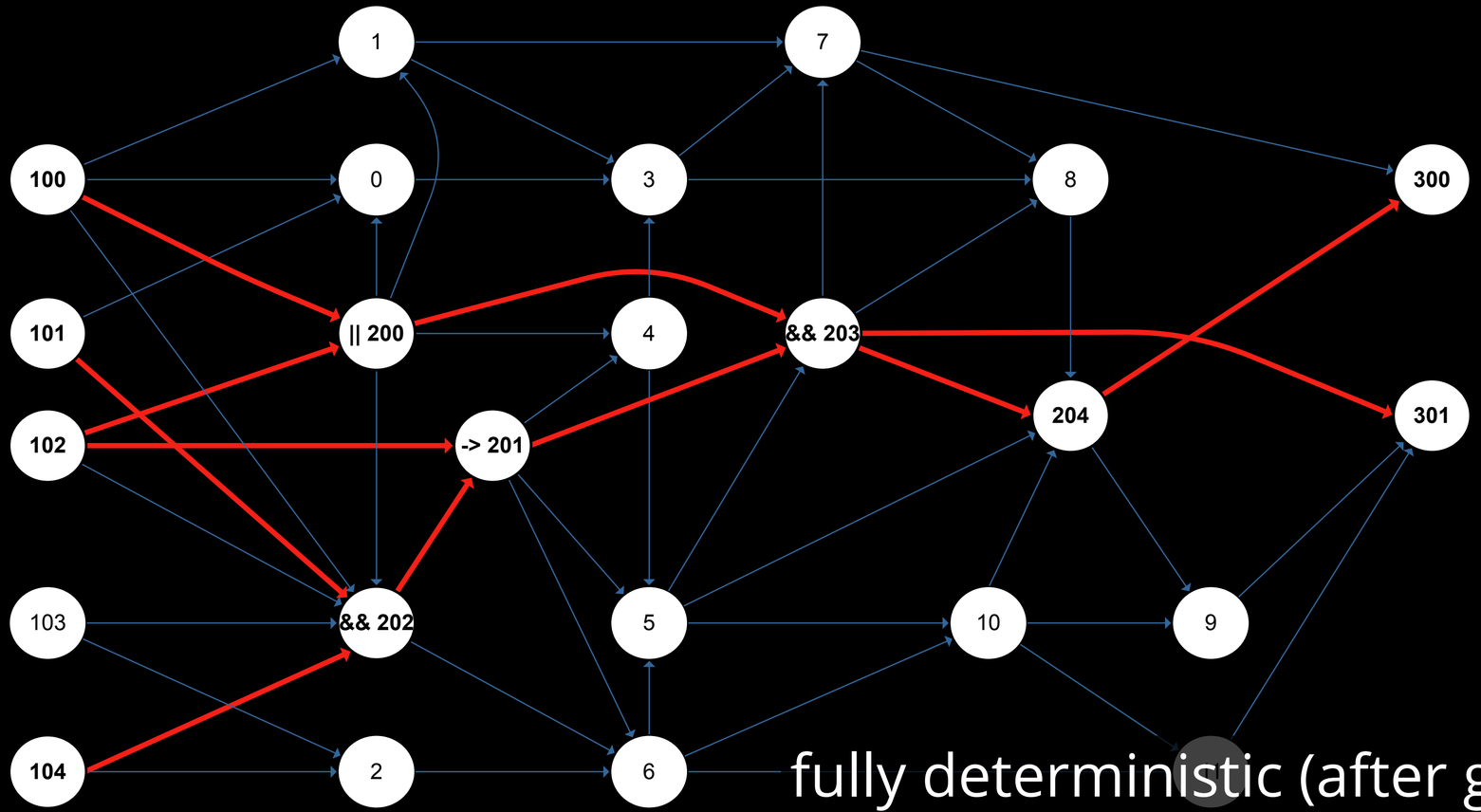
$$\varphi\,(x) \;=\; f \circ g \;\; (x, ?)\, ? \;=\; f\,(\,g\,(\,x, ?\,)\,)\, ?$$

apply φ to input so we get result

$$(\varphi\,(x)\, y) \;=\; f \circ (g \;\; (x))\, y \;=\; f\,(\,g\,(\,x, y\,)\,)\, y$$

apply φ to input so we get result

Computational Graph Network



modular
interpretable
parallelizable
fully deterministic (after graph generation)

Combinatorial Explosion

```
[ not ( A || C ) || ( C && not ( B && E
) )
, f7  ( f1 ( A ) )
, not ( f8 ( f7 ( f1 ( A ) ) ) )
, not ( A || B )  || implication( C, D
)
, ... ]
```

Combinatorial Explosion

```
[ not ( A || C ) || ( C && not ( B && E
) )
, f7  ( f1 ( A ) )
, not ( f8 ( f7 ( f1 ( A ) ) ) )
, not ( A || B )  || implication( C, D
)
, ... ]

--List of Parameters for Node Function f (x, y
[ 0, 1, 2 ]

--Parameter Permutation:
[ (0,1), (1,2), (2,3), (3,2), (2,1), (1,0) ]
```

Commutativity as Complexity Barrier

(using ingenuity instead of computational power)

Commutativity as Complexity Barrier

(using ingenuity instead of computational power)

$$a + b == b + a$$

$$(+)\, a\, b == (+)\, b\, a$$

Commutativity as Complexity Barrier

(using ingenuity instead of computational power)

$$a + b == b + a$$

$$(+)\, a\, b == (+)\, b\, a$$

```
--List of Parameters for Node Function f (x, y):
[ 0, 1, 2 ]

--Parameter Permutation:
[ (0,1), (1,2), (2,3), (3,2), (2,1), (1,0) ]

--Parameter Permutation with commutative function f (x,
[ (0,1), (1,2), (2,3) ]
```

Haskell

$$\varphi(x) = f \circ g\ (x,?)? = f(g(x,?))?$$

apply φ to input so we get result

```
(.-) [[a] -> [b]] -> ([[b]] -> [a] -> [c]) -> [a] -> [c]
(.-) funcList          newFunc                  inp =
                       newFunc (map ($inp) funcList) inp
-- where
--    funcList = Previous Node Function List
--    newFunc  = Node Function
--    inp = Sample Input
```

Formal Software Verification

$$Theorem: \;\; Int \to Int \to Int$$
$$Proof: \;\;\;\; (+) \;\; Q.E.D.$$

~Curry-Howard-Lambek Isomorphism

a proof is a program, and the formula it proves is the type for the program

Conclusion

- Accuracy is not enough to trust an AI
- Accuracy vs Explainability is not a trade-off

Look for solutions in weird places:

- Try Functional Programming & Category Theory

Trends in XAI:

- closing the gap between Symbolic AI and DL
- training DL representation models not decision models
- using Object-Oriented-Learning
- using Computational Graph Networks

+ Don't let your robot read legal texts ;)

Thank you for your attention

Interested in more AI projects?

- Visit us at the ***PwC booth*** (ground floor)
- Join our workshop: ***"Open the Blackbox using local interpretable model agnostic explanation"*** (5.12. 14:30-16:30, Room LAB 1.0)

Ahmad Haj Mosa
AI Researcher
ahmad.haj.mosa@pwc.com

Fabian Schneider
PoC-Engineer & Researcher
schneider.fabian@pwc.com