

# “Easy” presentation

The logic route to strong AI

YKY

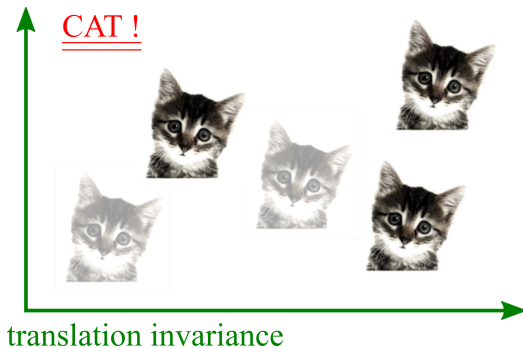
March 18, 2021

# Table of contents

1	CNN
3	Symmetry and inductive bias
4	Richard Sutton
6	
7	Structure of logic
8	Symmetric neural networks
10	BERT
11	content-addressable long-term memory
12	(knowledge graphs)
13	AGI

# The success of CNN in computer vision

- In geometry, vision is said to possess the property of **translation invariance**:



(1)

- **Convolution** is an operation invariant under translation:

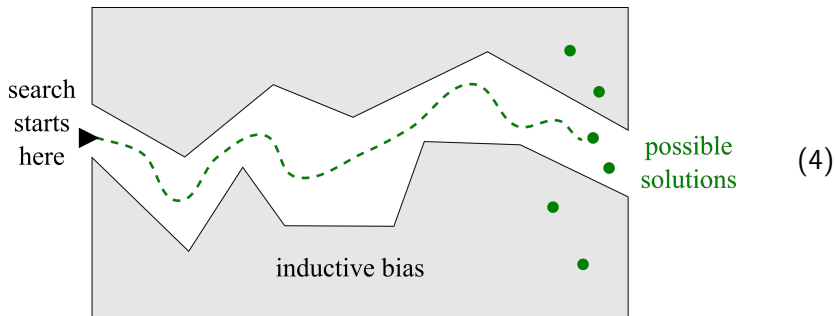
$$(T_x \circ f) * g = T_x \circ (f * g) \quad (2)$$

- Yann LeCun *et al* exploited the symmetry of CNNs to accelerate learning, successfully solved the visual recognition problem



# Symmetry and inductive bias

- In mathematics, symmetry often simplifies computation, which is why mathematicians love to study symmetries
- In machine learning, one introduces **inductive bias** to narrow down the **search space**:



- Oftentimes, if inductive bias is chosen correctly, solution is found quickly, otherwise problem becomes **intractable**

## Richard Sutton's view

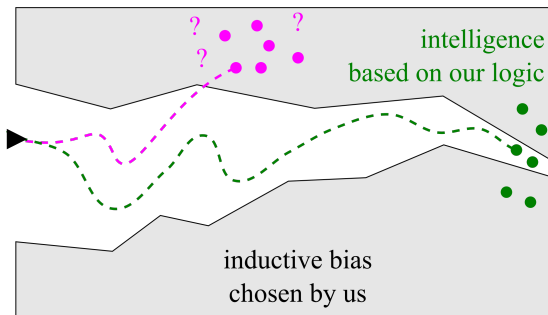
- In contrast, Sutton expressed the view that AI can be solved merely by **increasing computing power**, under the reinforcement learning framework



(5)

- Our choice is just one out of many possible **forms** of logic:

intelligence based on "alternative" logics



(6)

- This is not only a theoretical issue;  
Indeed, AI labs around the world had begun the search for AGI with various strategies!

# Doubts about logicism

- Many are doubtful: does the human brain really use **formal logic** to think?



(7)

Actually human cognition may be much closer to logic than we've thought



# Structure of logic

- The idea is: introduce symmetries of **logic** into deep learning to solve the AGI problem
- Because human cognition has logical structure, this inductive bias may help us find a solution to AGI faster
- Logic is a complicated structure, but its simplest symmetry is the **commutativity** (or permutation invariance) of **propositions**:

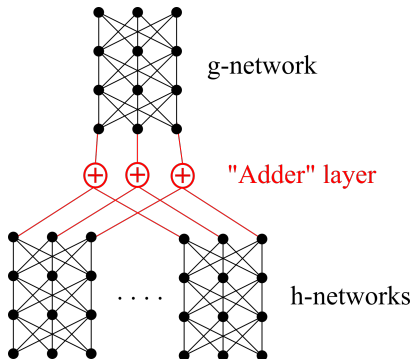
$$\begin{array}{ccc} A \wedge B & \equiv & B \wedge A \\ \text{it's raining} \wedge \text{lovesick} & \equiv & \text{lovesick} \wedge \text{it's raining} \end{array} \quad (8)$$

- Its importance may be analogous to translation invariance in vision
- The significance of commutativity is: it **decomposes** the AI system's **mental state** into individual **propositions**

# Symmetric neural networks

- Permutation invariance can be handled by **symmetric** neural networks
- I wasted 2 years trying to solve this problem, and then found out it had been solved 3 years before: [PointNet 2017] and [DeepSets 2017] and their mastery of mathematics is significantly above me!
- Any symmetric function can be represented by the following form (a special case of the Kolmogorov-Arnold representation of functions):

$$f(x, y, \dots) = g(h(x) + h(y) + \dots) \quad (9)$$



(10)

- Sym NN gives a powerful boost in efficiency  $\propto n!$  where  $n = \text{\#inputs}$
- The code for Sym NN is just a few lines of Tensorflow:

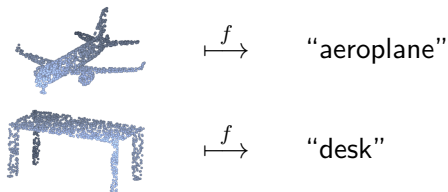
```
h = Dense(3, activation='tanh')
ys = []
for i in range(9):
    ys.append( h(xs[i]) )
y = Keras.stack(ys, axis=1)
Adder = Lambda(lambda x: Keras.sum(x, axis=1))
y = Adder(y)
g = Dense(3)
output = g(y)
```

(11)

- Very easy to adopt this to existing models such as BERT and reinforcement learning
- I have successfully tested it on the game of TicTacToe:  
<https://github.com/Cybernetic1/policy-gradient>

# For example: symmetric NN for object recognition

- Imagine objects represented as **point clouds**:

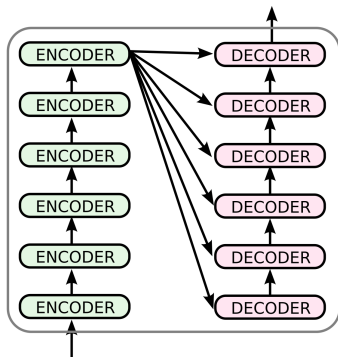


- It does not matter **in what order** the points are in a sequence; the function  $f(x_1, \dots, x_n)$  is symmetric in its arguments (the points)
- Permutation invariance is **essential** for this to work

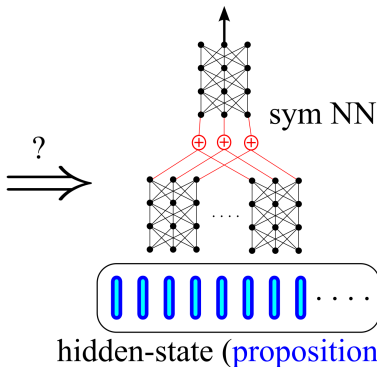
# Logicalization of BERT

- Similarly, we can convert BERT's hidden state into a set of propositions, by replacing the original **decoder** with a sym NN:

original BERT / Transformer



logic BERT ?

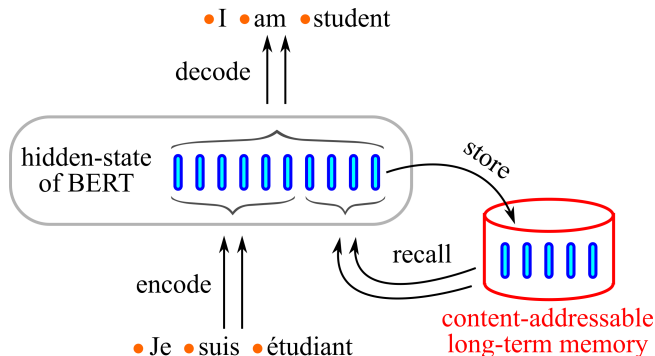


(12)

- The original **encoder** can be retained. As the **decoder** imposes symmetry on the hidden state, error propagation is expected to cause its representation to change
- Of course, this remains to be proven by experiment 🤔

# Advantages of logical AI (1)

- The original BERT hidden state lacked a logical structure and it was not clear what exactly it contains. After logicalization, propositions inside BERT can be stored into **long-term memory**:

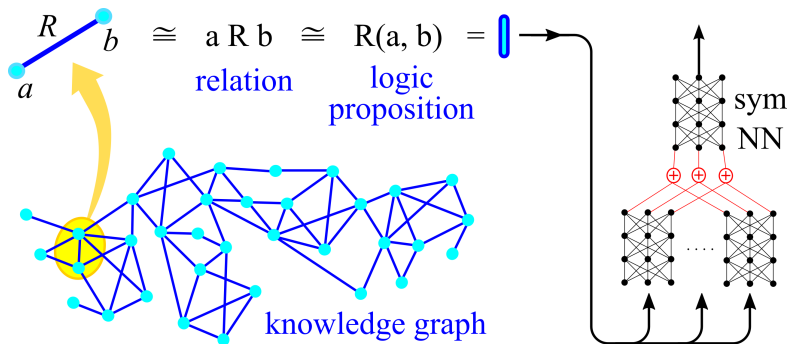


(13)

- These kind of systems are very close to strong AI, and it depends crucially on logicalization
- The content-addressable memory idea came from Alex Graves *et al*'s Neural Turing Machine [2014]

## Advantages of logical AI (2)

- Integrates seamlessly with knowledge graphs



- strong AI  
AGI  
AI
- Logic BERT   attention  
Logic BERT



# References

Thanks for watching 😊

- [1] Alex Graves, Greg Wayne, and Ivo Danihelka. “Neural Turing Machines”. In: *CoRR* abs/1410.5401 (2014). arXiv: 1410.5401. URL: <http://arxiv.org/abs/1410.5401>.
- [2] Qi et al. “Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *CVPR* (2017). <https://arxiv.org/abs/1612.00593>.
- [3] Zaheer et al. “Deep sets”. In: *Advances in Neural Information Processing Systems* 30 (2017), pp. 3391–3401.

## Illustration credits:

- Translation invariance, from Udacity Course 730, Deep Learning (L3 Convolutional Neural Networks ▷ Convolutional Networks)