Opening the black box of Deep Neural Networks via Information

(Ravid Shwartz-Ziv and Naftali Tishby)

An overview by
Philip Amortila and Nicolas Gagné

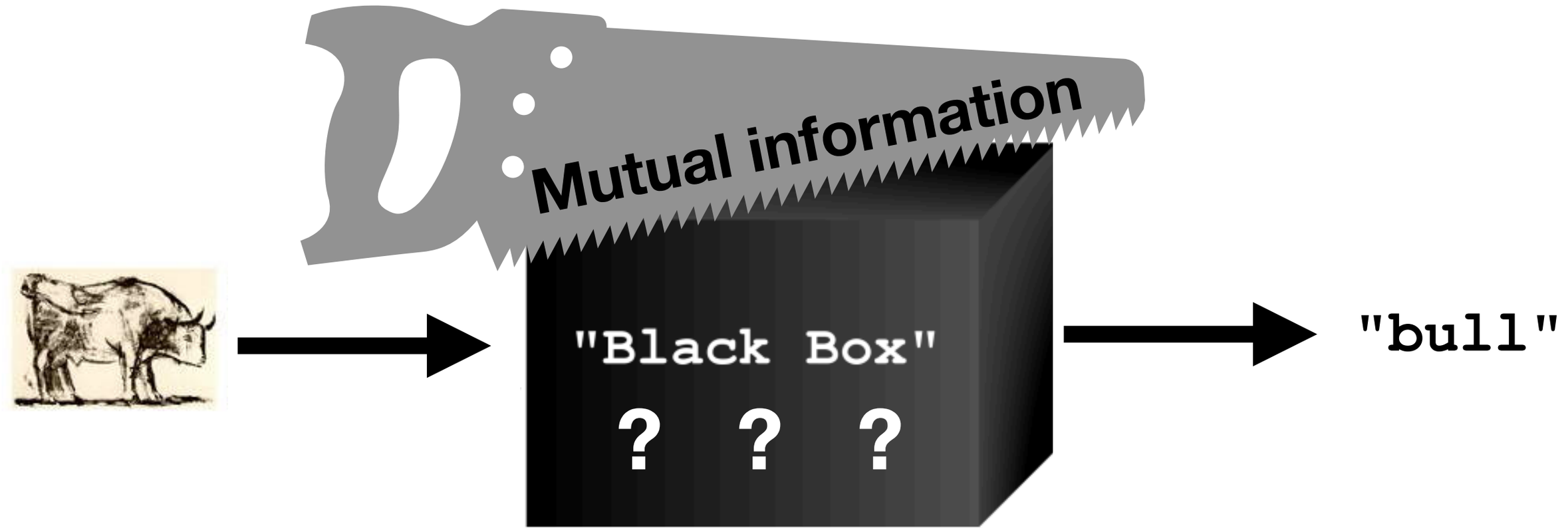
Problem:

The usual "black box" story:

"***Despite their great success, there is still <u>no comprehensive understanding</u> of the internal organization of Deep Neural Networks."***

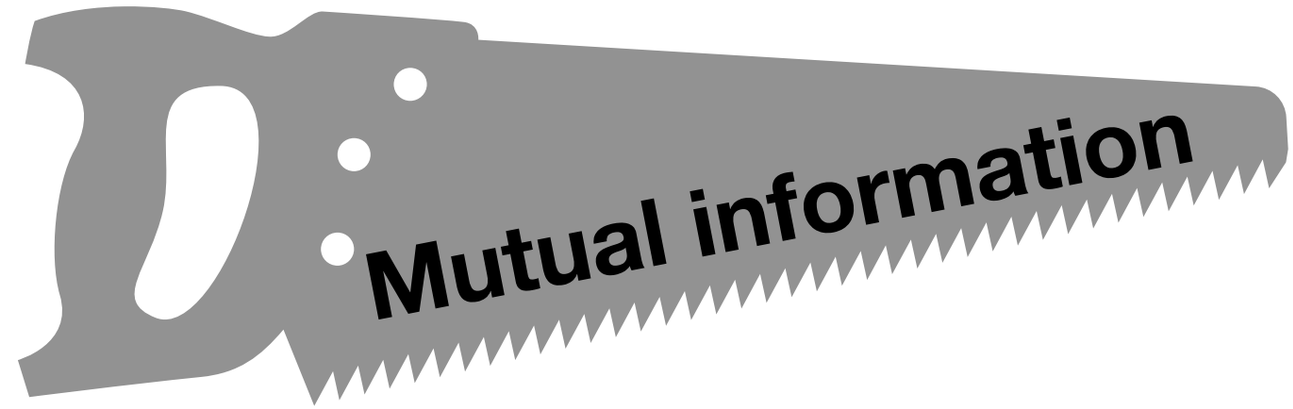
Solution:

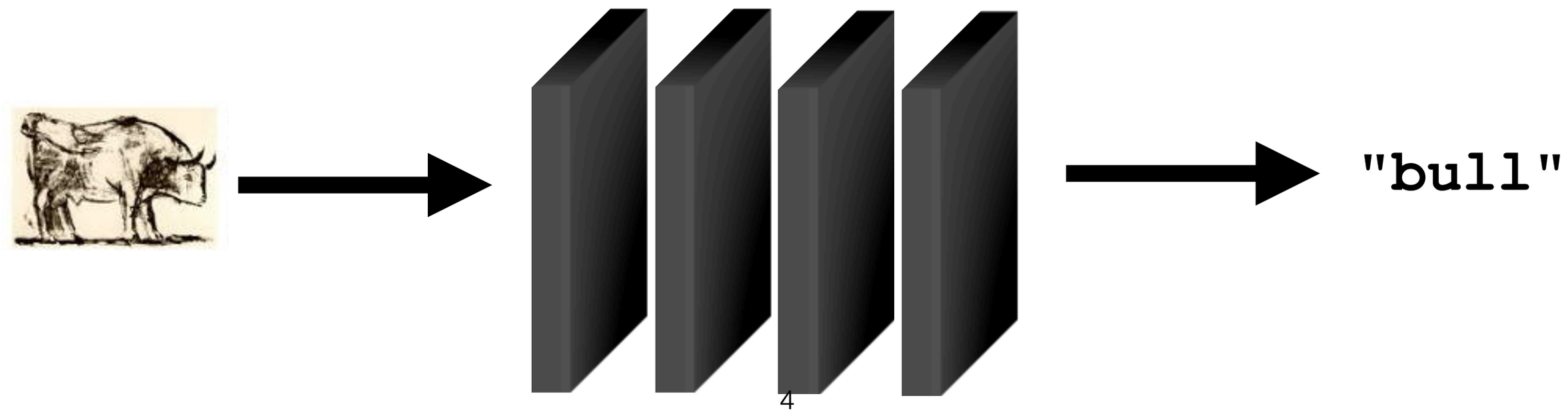
Opening the blackbox using mutual information!



Solution:

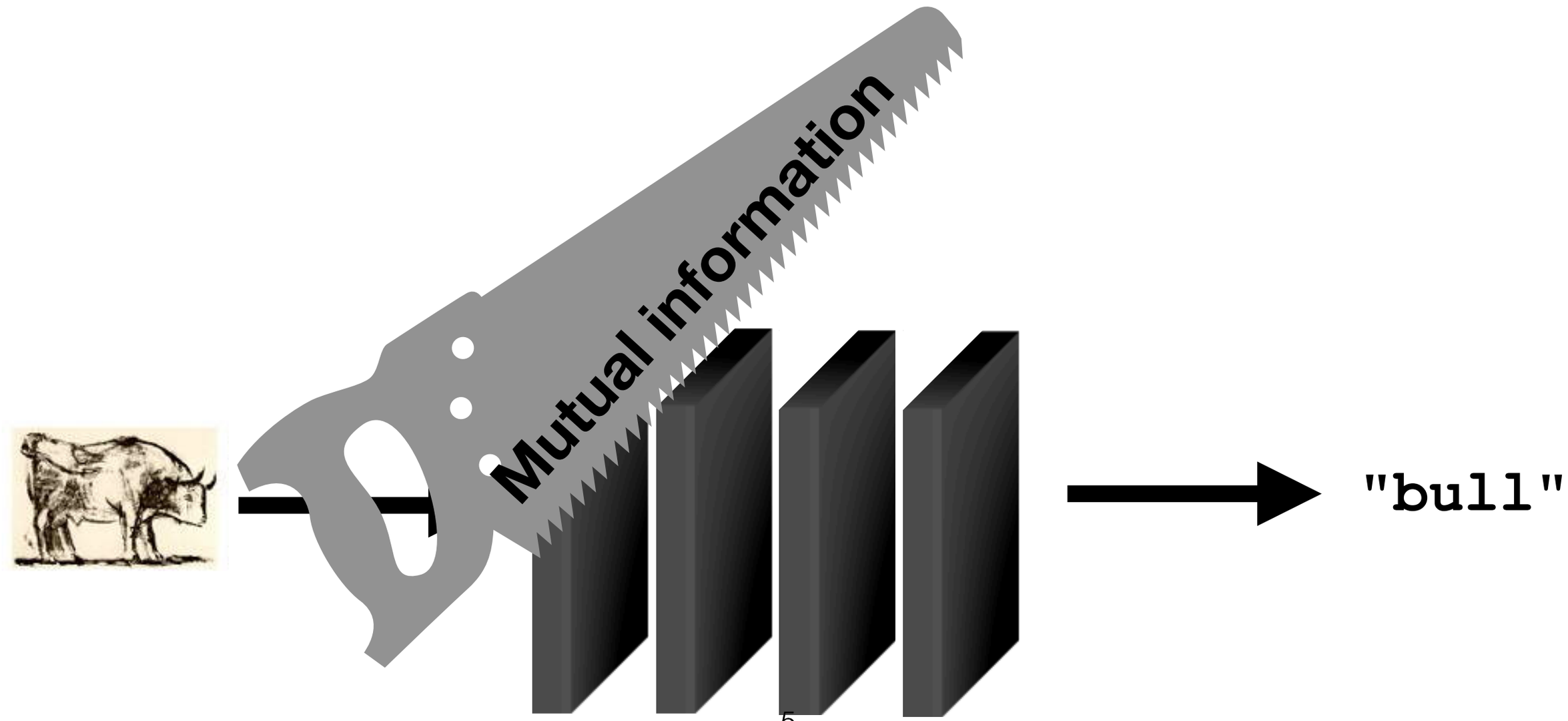
Opening the blackbox using mutual information!





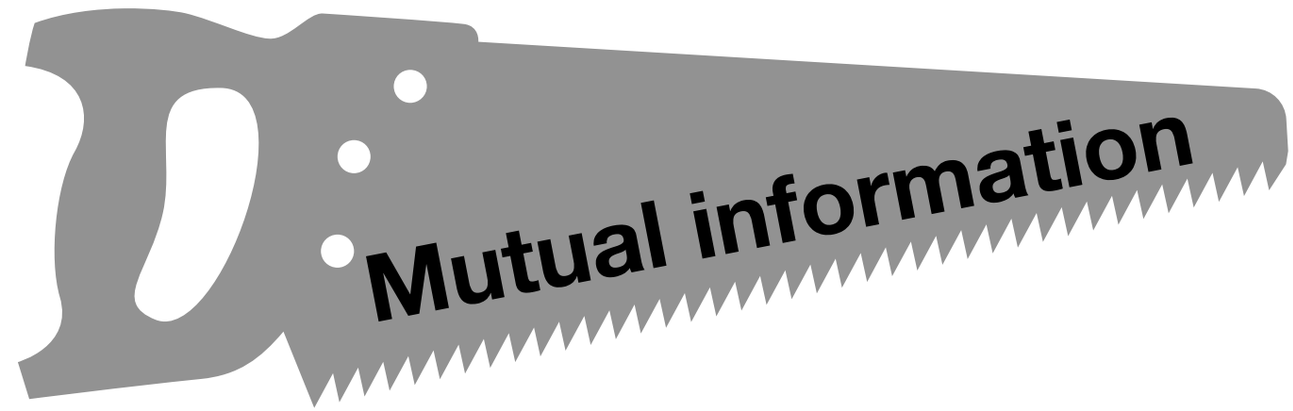
Solution:

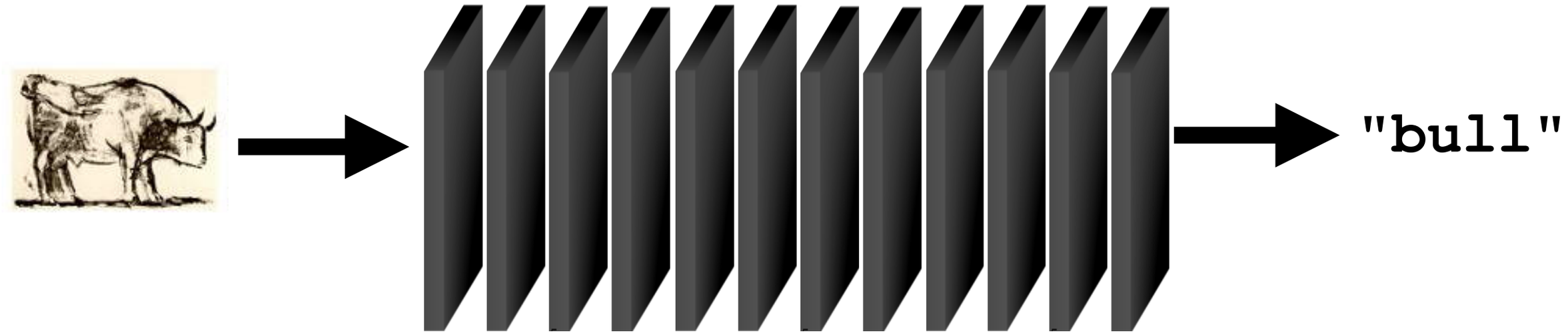
Opening the blackbox using mutual information!



Solution:

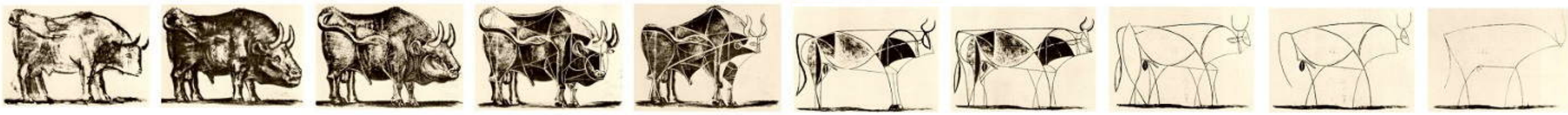
Opening the blackbox using mutual information!





Spoiler alert!

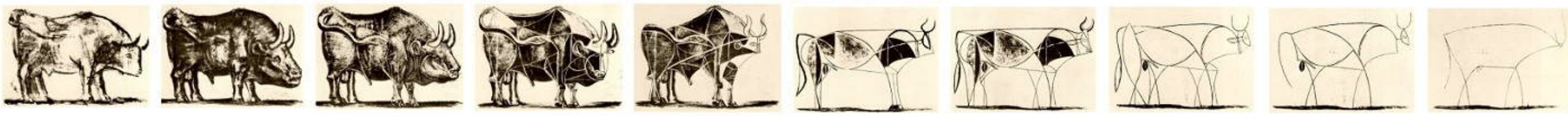
As we train a deep neural network, its layers will



Spoiler alert!

As we train a deep neural network, its layers will

1. Gain "information" about the true label and the input.



Spoiler alert!

As we train a deep neural network, its layers will

1. Gain "information" about the true label and the input.

2. Then, will gain "information" about the true label but will lose "information" about the input!

Spoiler alert!

As we train a deep neural network, its layers will

1. Gain "information" about the true label and the input.

2. Then, will gain "information" about the true label but will lose "information" about the input!

During that second stage, it forgets the details of the input that are irrelevant to the task.

Spoiler alert!

As we train a deep neural network, its layers will

1. Gain "information" about the true label and the input.

2. Then, will gain "information" about the true label but will lose "information" about the input!

During that second stage, it forgets the details of the input that are irrelevant to the task.

(Not unlike Picasso's deconstruction of the bull.)

Spoiler alert!

Before delving in, we first need to define what we mean by "information".

Spoiler alert!

Before delving in, we first need to define what we mean by "information".

By information, we mean mutual information.

What is mutual information?

For discrete random variables *X* and *Y* :

- Entropy of X given nothing

$$H(X) := -\sum_{i=1}^{n} p(x_i) \log \left(p(x_i)\right)$$

What is mutual information?

For discrete random variables *X* and *Y* :

- Entropy of X given nothing

$$H(X) := -\sum_{i=1}^{n} p(x_i) \log \left(p(x_i)\right)$$

- Entropy of X given y

$$H(X|y) := -\sum_{i=1}^{n} p(x_i|y) \log \left(p(x_i|y)\right)$$

What is mutual information?

For discrete random variables *X* and *Y* :

- Entropy of X given nothing

$$H(X) := -\sum_{i=1}^{n} p(x_i) \log \left(p(x_i)\right)$$

- Entropy of X given y

$$H(X|y) := -\sum_{i=1}^{n} p(x_i|y) \log \left(p(x_i|y)\right)$$

- Entropy of X given Y

$$H(X|Y) := \sum_{j=1}^{m} p(y_j) H(X|y_j)$$

What is mutual information?

For discrete random variables *X* and *Y* :

- Entropy of X given nothing

$$H(X) := -\sum_{i=1}^{n} p(x_i) \log \left(p(x_i)\right)$$

- Entropy of X given y

$$H(X|y) := -\sum_{i=1}^{n} p(x_i|y) \log \left(p(x_i|y)\right)$$

- Entropy of X given Y

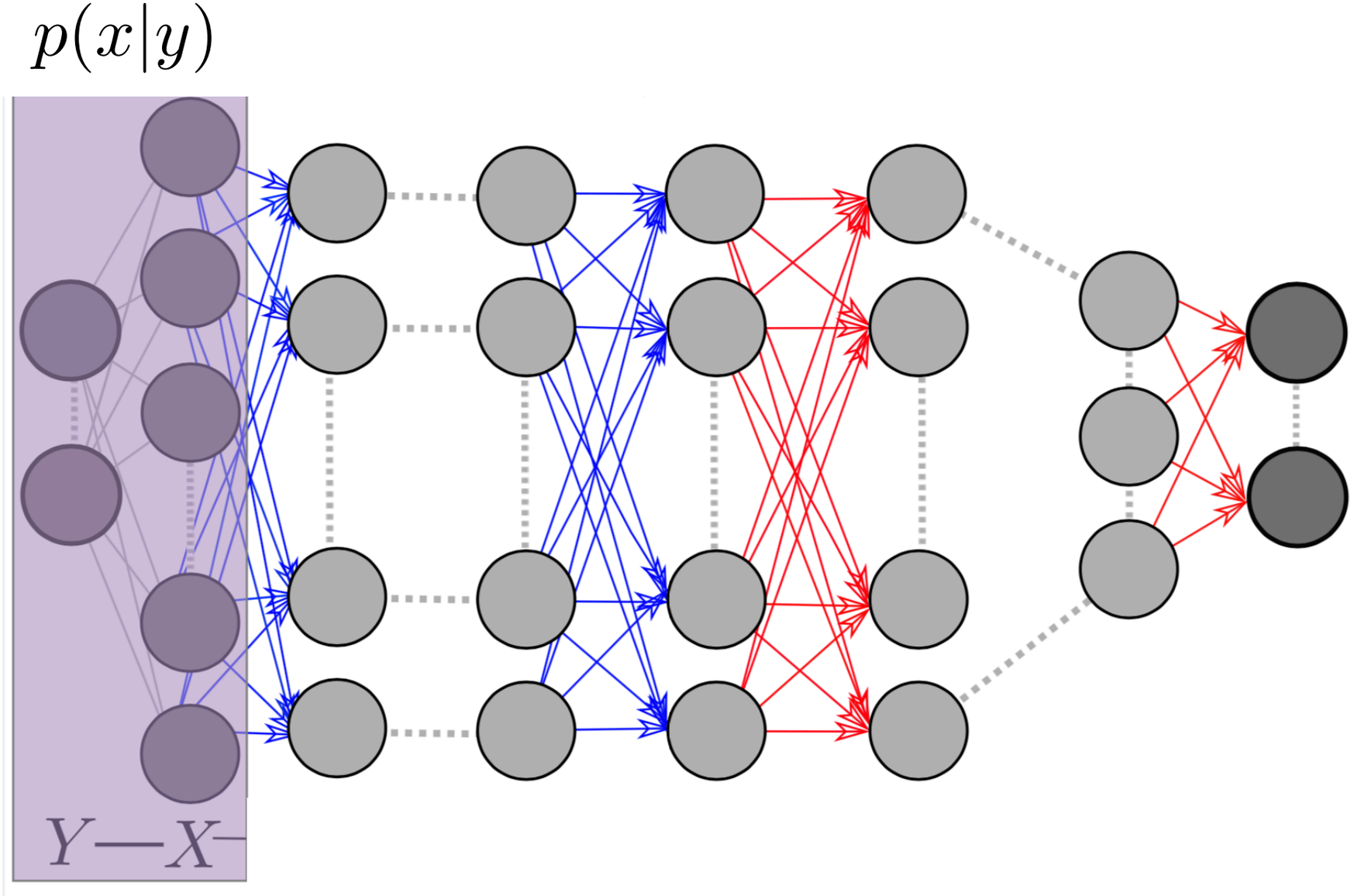
$$H(X|Y) := \sum_{j=1}^{m} p(y_j) H(X|y_j)$$

Mutual information between X and Y

$$I(X;Y) := H(X) - H(X|Y)$$

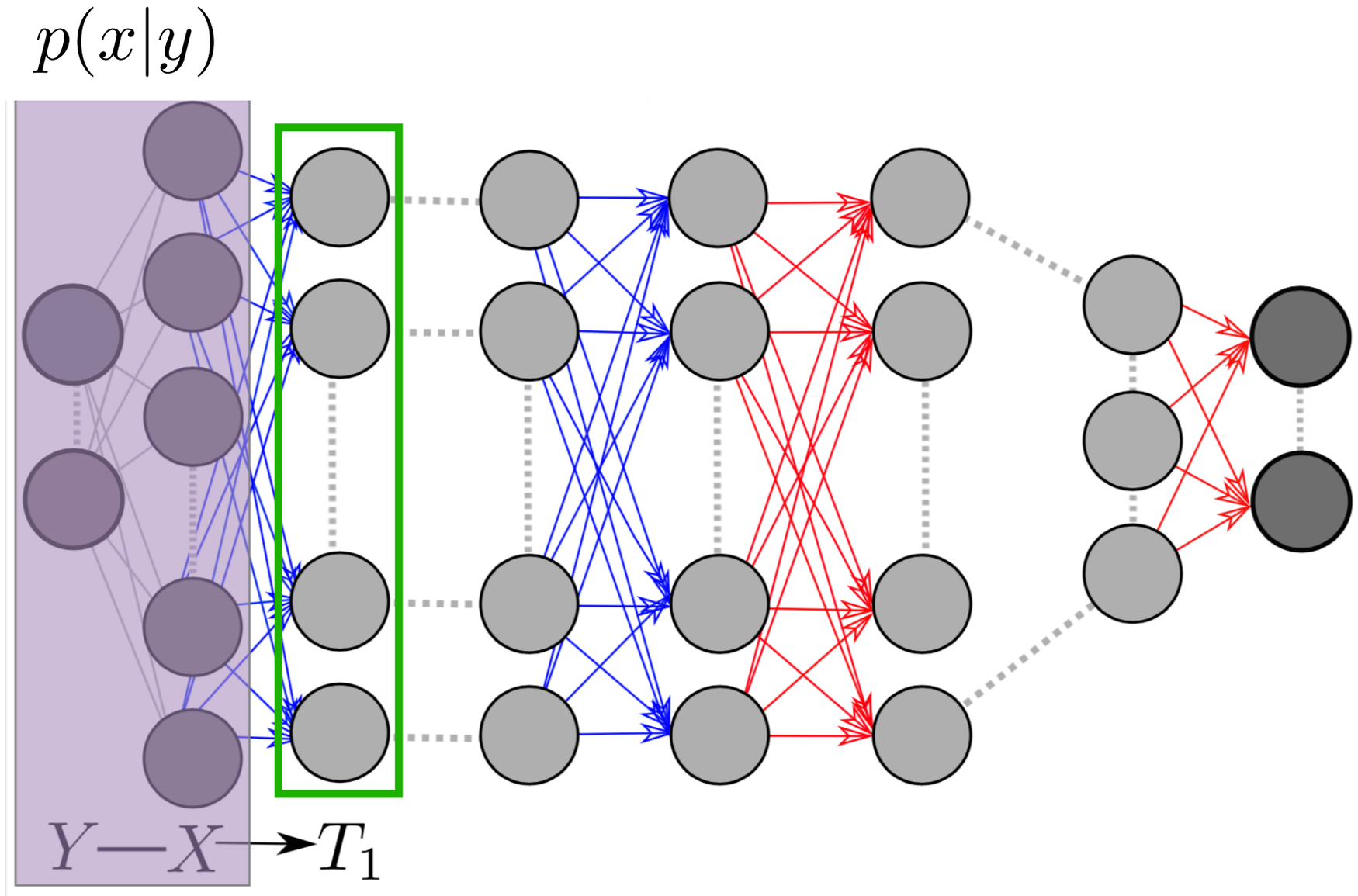
Setup:

Given a deep neural network where *X* and *Y* are discrete random variables:



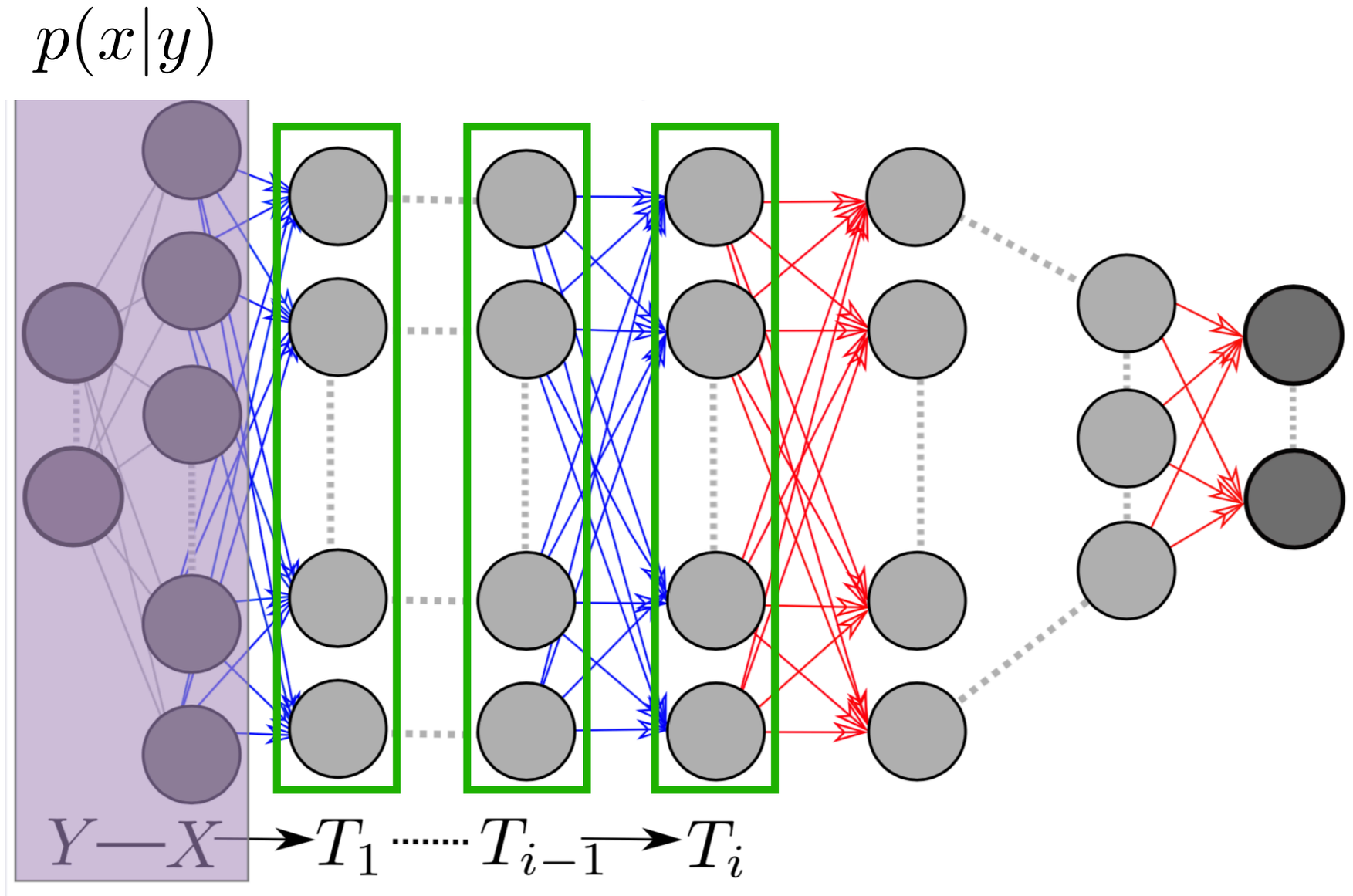
Setup:

We identify the propagated values at layer *1* with the vector T_1



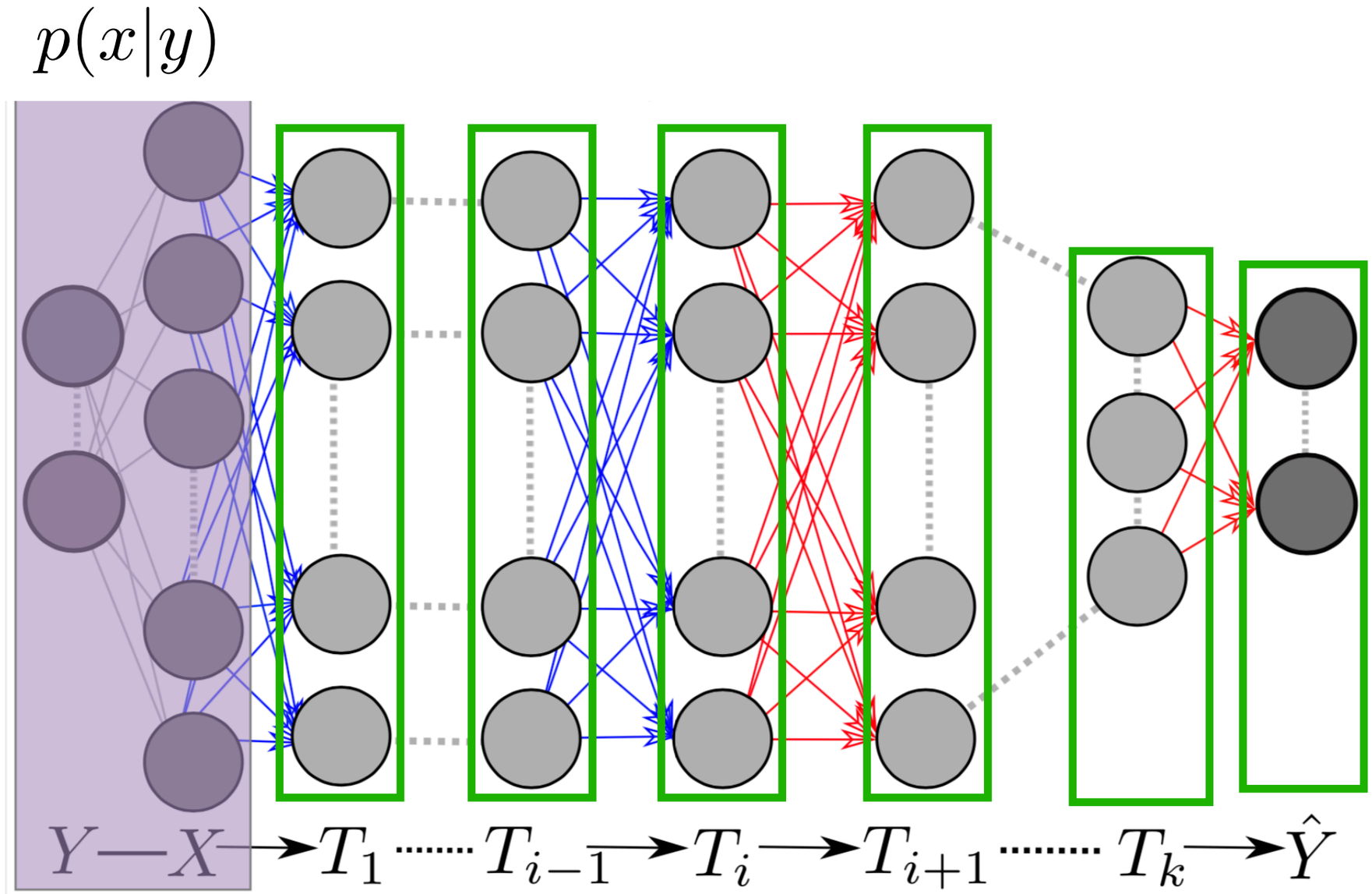
Setup:

We identify the propagated values at layer *i* with the vector T_i



Setup:

We identify the values for every layer with their corresponding vector.



Setup:

We identify the values for every layer with their corresponding vector.

And doing so, we get the following Markov chain.

$$Y \text{—} X \rightarrow T_1 \cdots\cdots T_{i-1} \rightarrow T_i \rightarrow T_{i+1} \cdots\cdots T_k \rightarrow \hat{Y}$$

Setup:

We identify the values for every layer with their corresponding vector.

And doing so, we get the following Markov chain.

(Hint: Markov chain rhymes with data processing inequality)

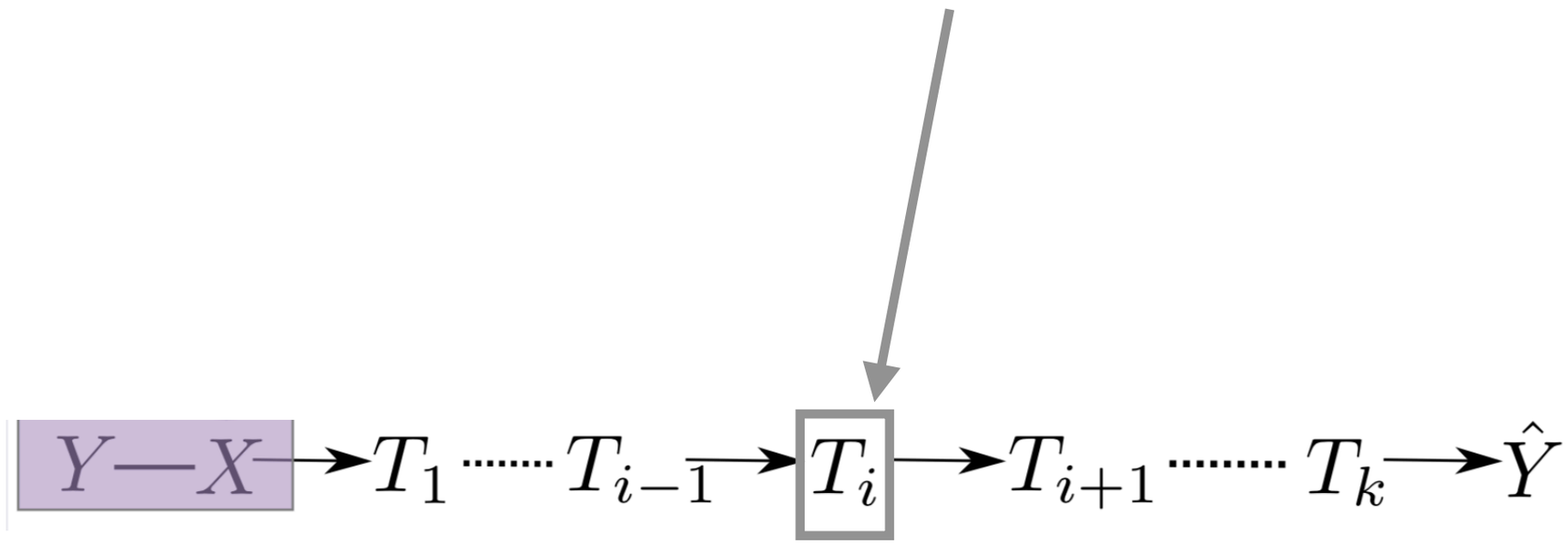
$$I(Y, T_i) \geq I(Y, T_{i+n}) \text{ and } I(X, T_i) \geq I(X, T_{i+n})$$

$$Y \text{—} X \rightarrow T_1 \cdots\cdots T_{i-1} \rightarrow T_i \rightarrow T_{i+1} \cdots\cdots T_k \rightarrow \hat{Y}$$

Setup:

And doing so, we get the following Markov chain.

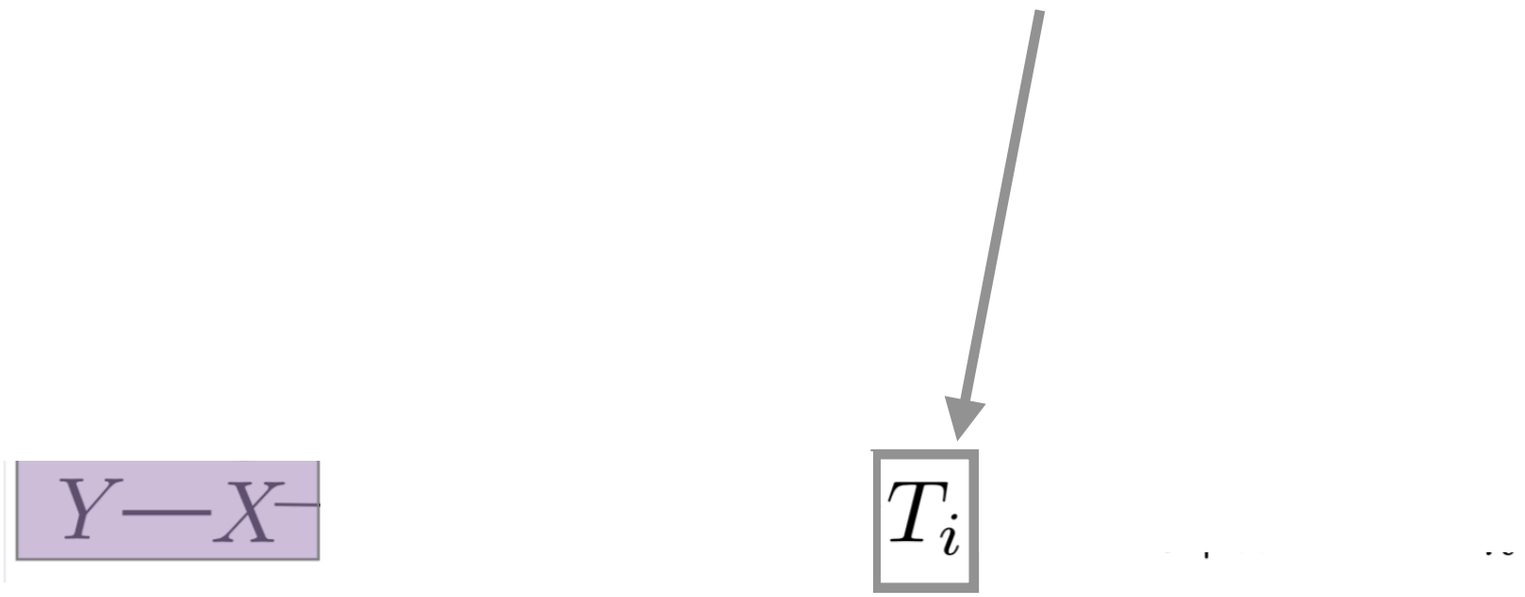
Next, pick your favourite layer, say T_i



Setup:

And doing so, we get the following Markov chain.

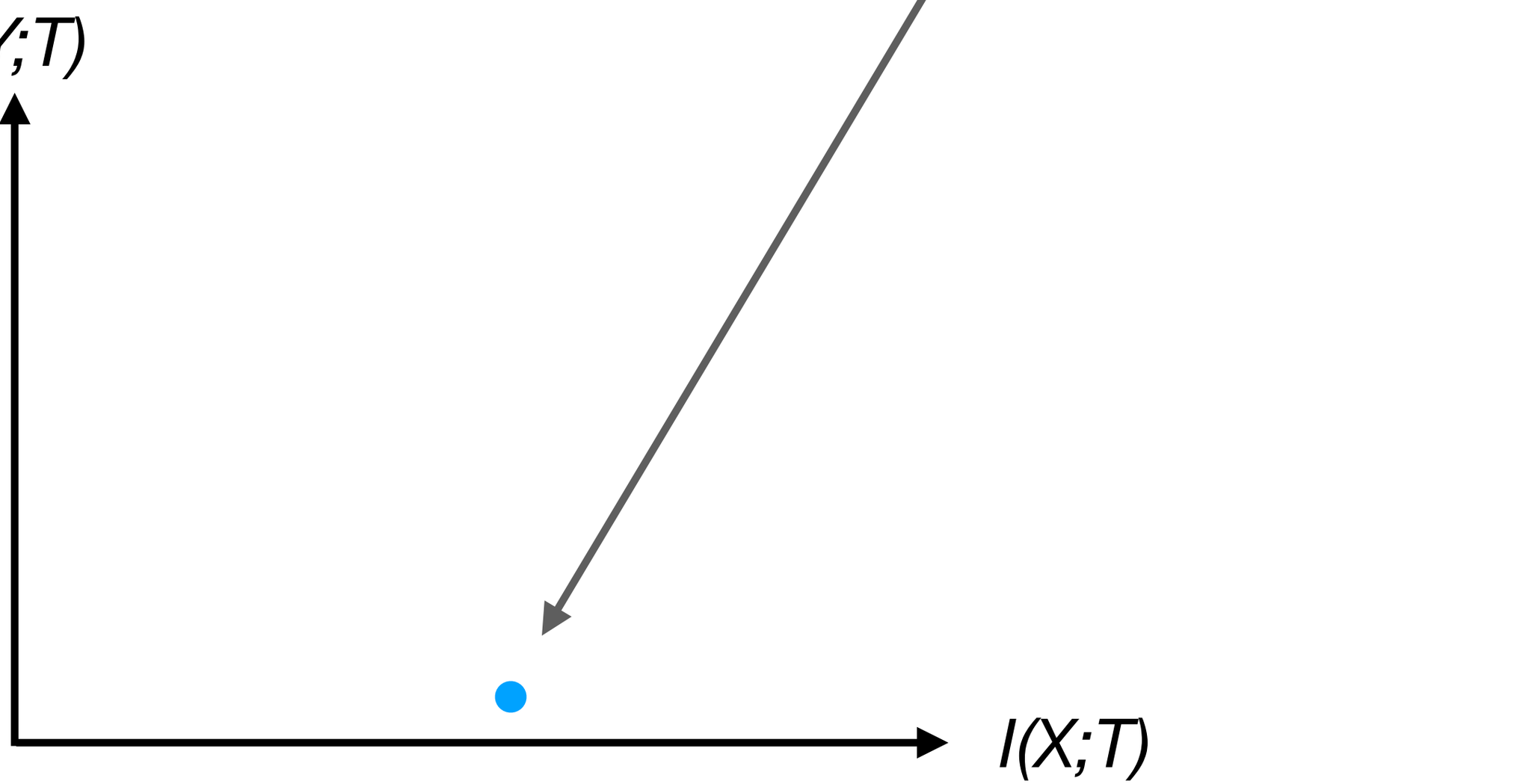
Next, pick your favourite layer, say T_i



Next, pick your favorite layer, say *T.*

We will plot *T*'s current location on the "information plane".

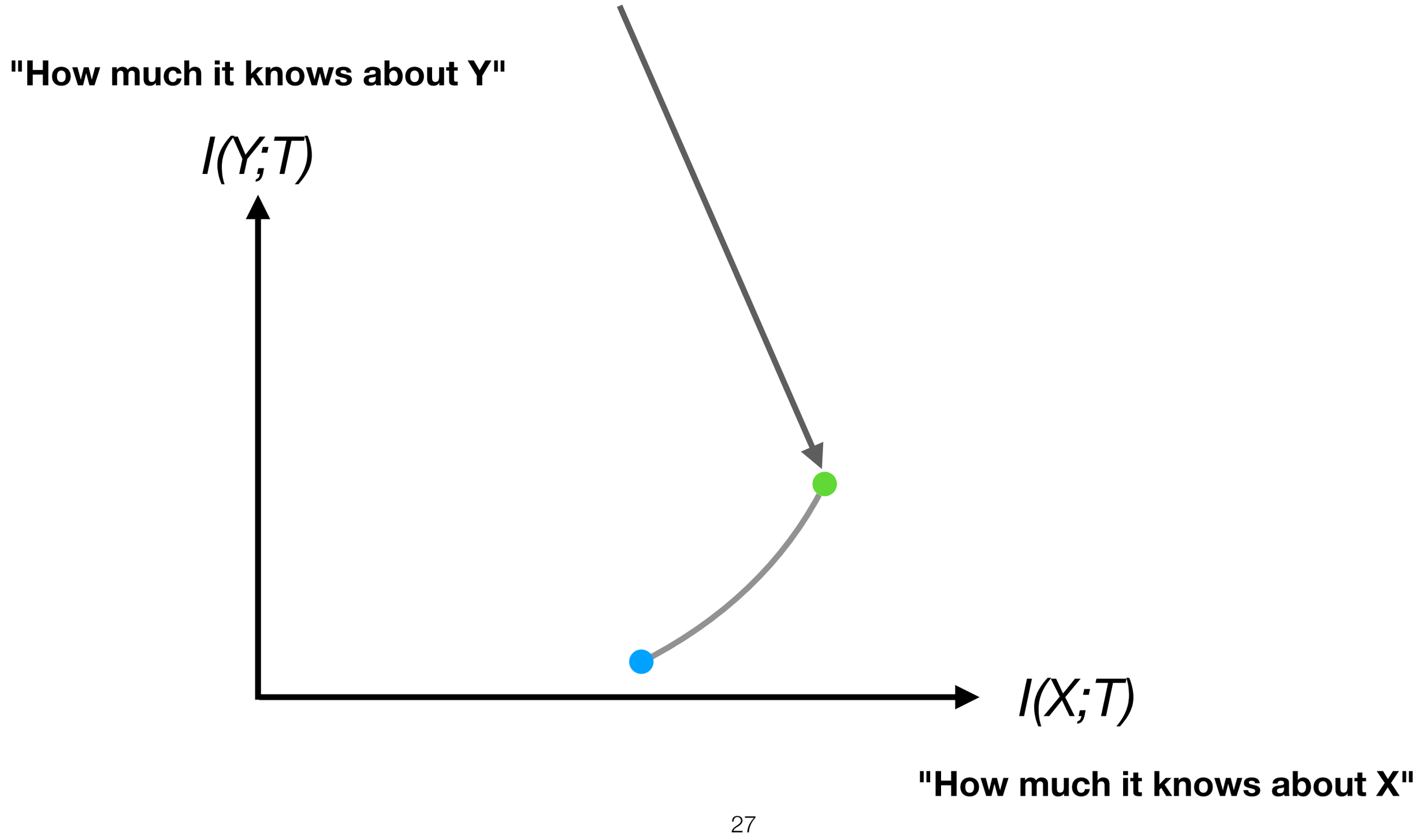
"How much it knows about Y"



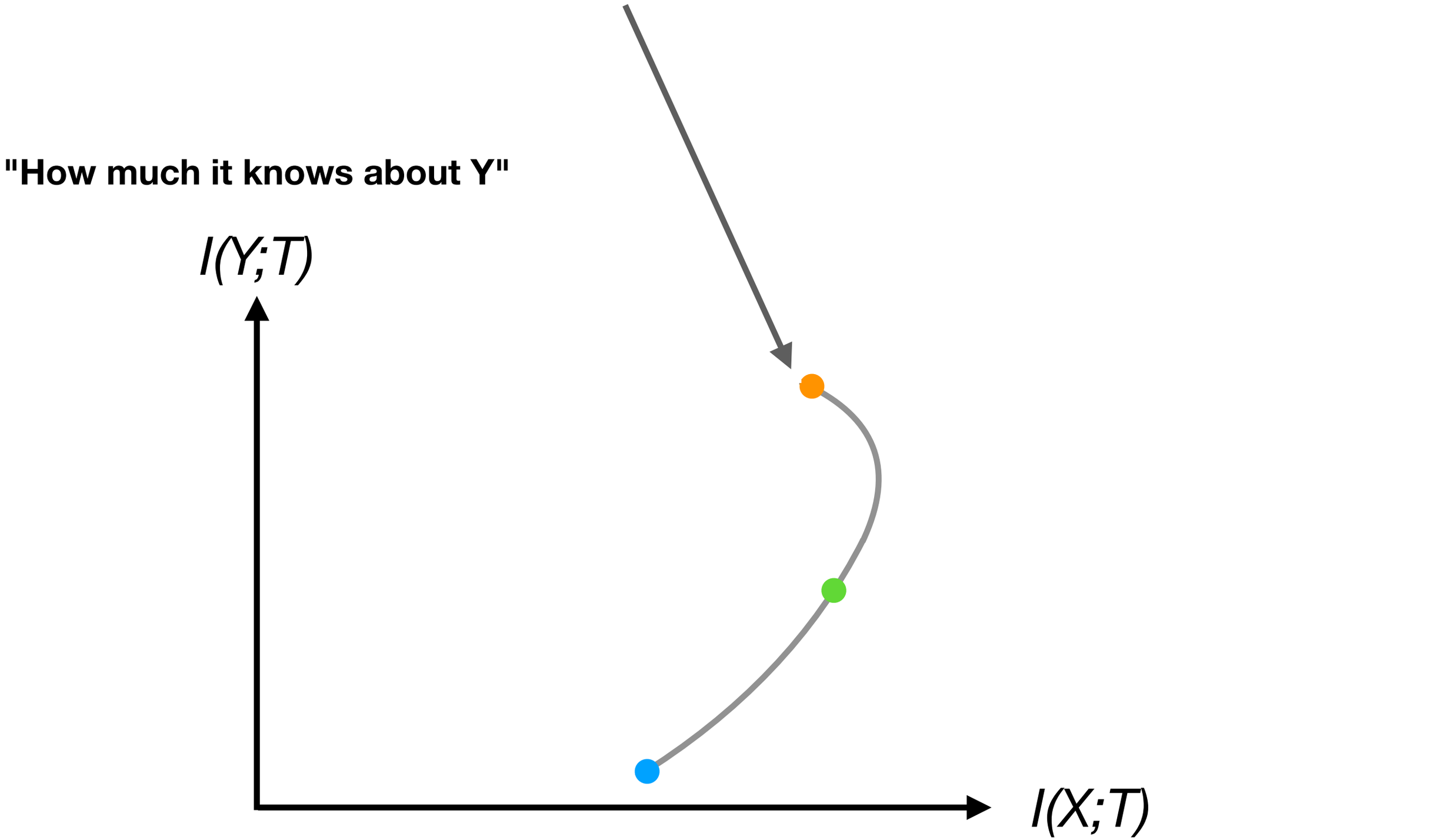
"How much it knows about X"

Then, we train our deep neural network for a bit.

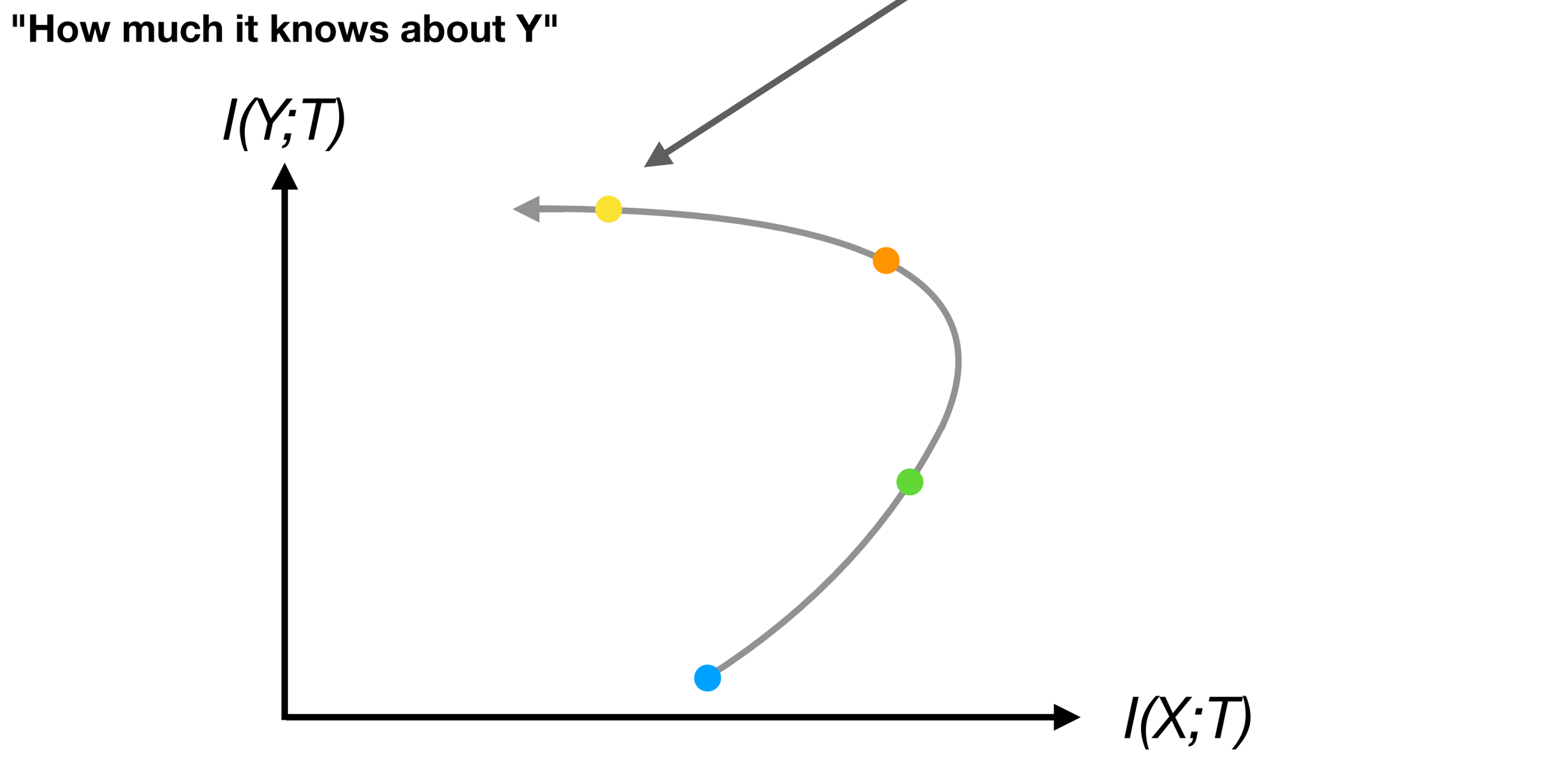
And plot the new corresponding distribution of *T*



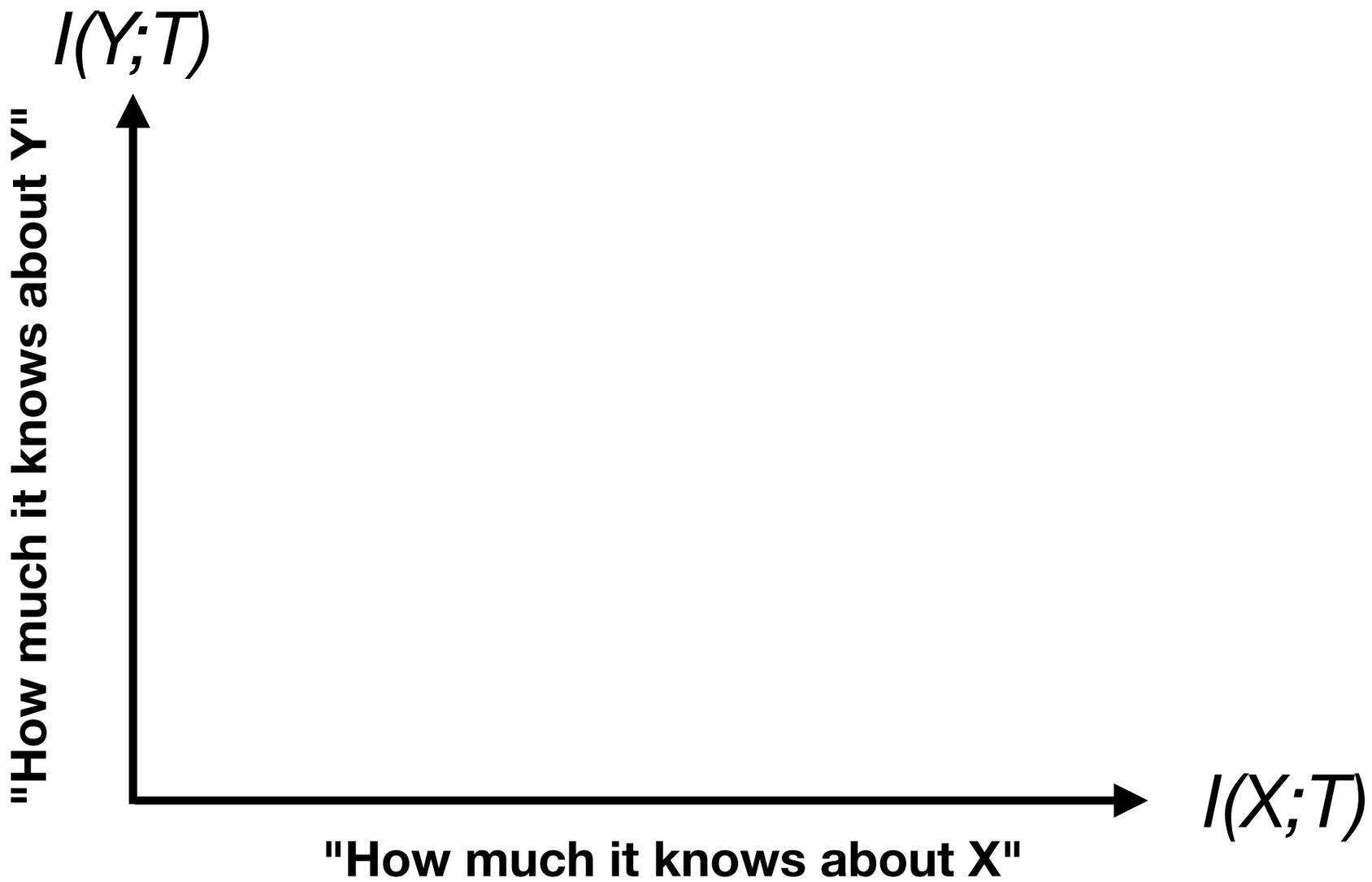
We train a bit more...



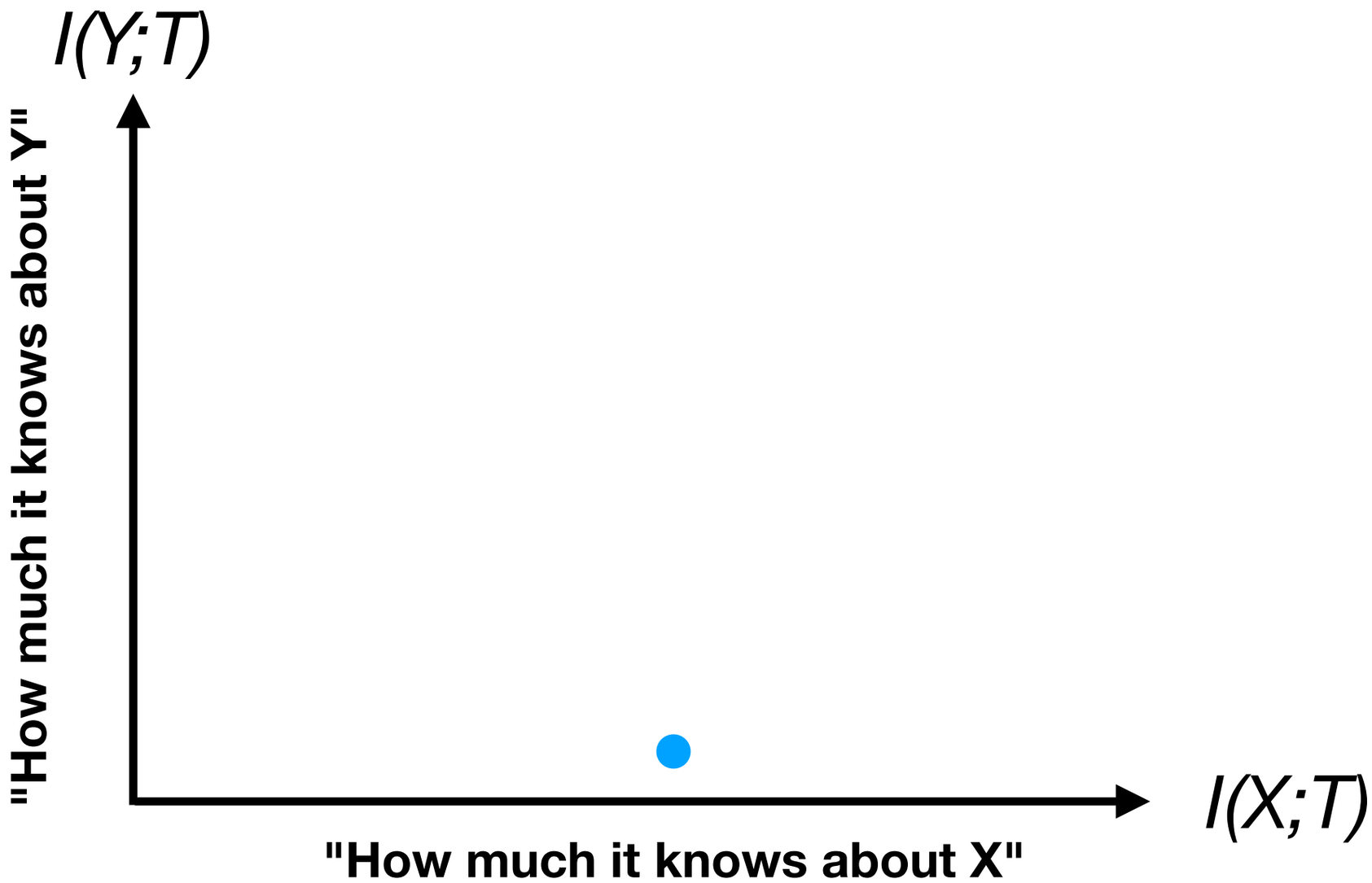
And as we train, we trace the layer's trajectory in the information plane.



Let's see what it looks like for a fixed data point : ( , "bull")

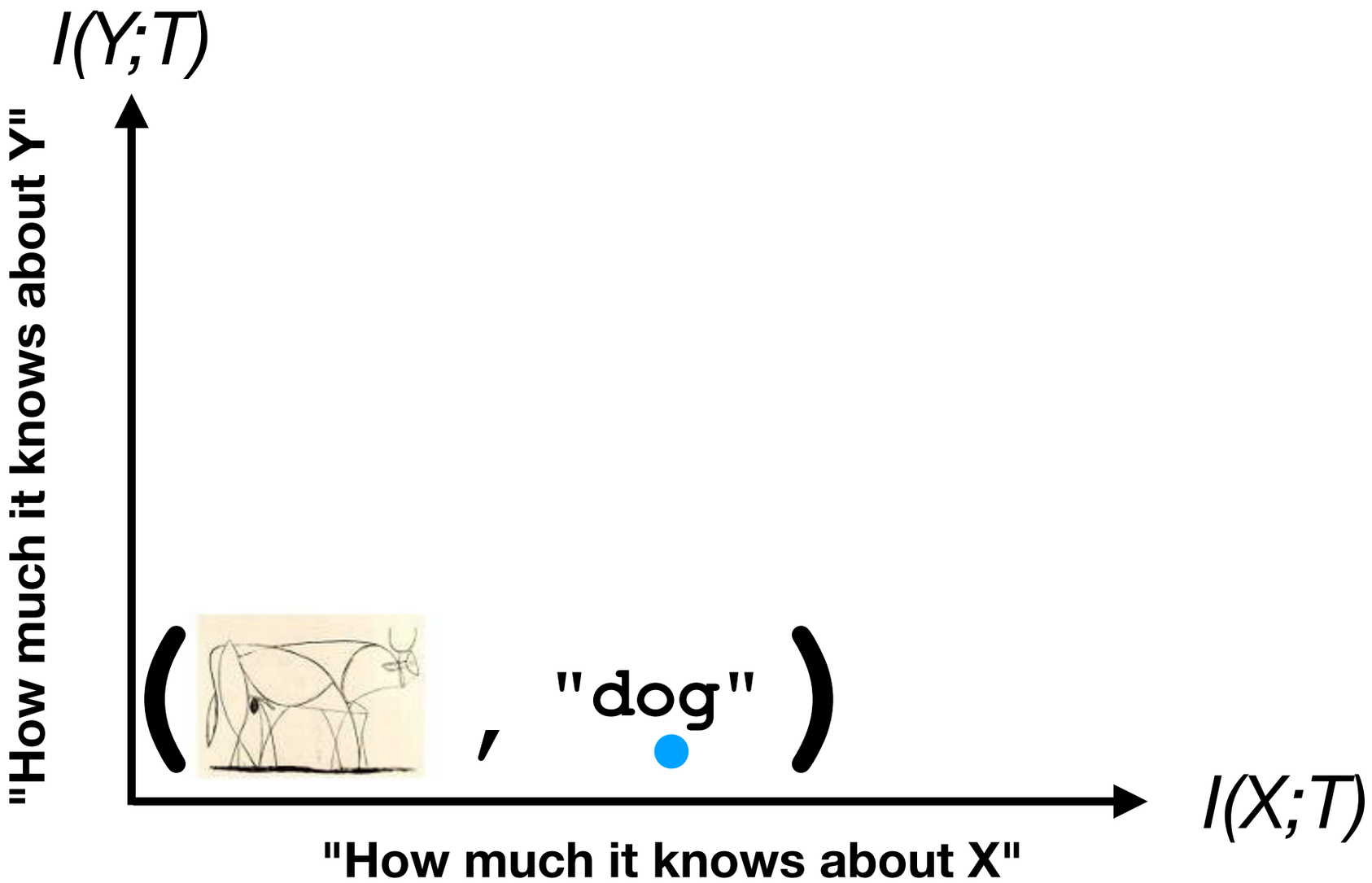


Let's see what it looks like for a fixed data point : ( , "bull")



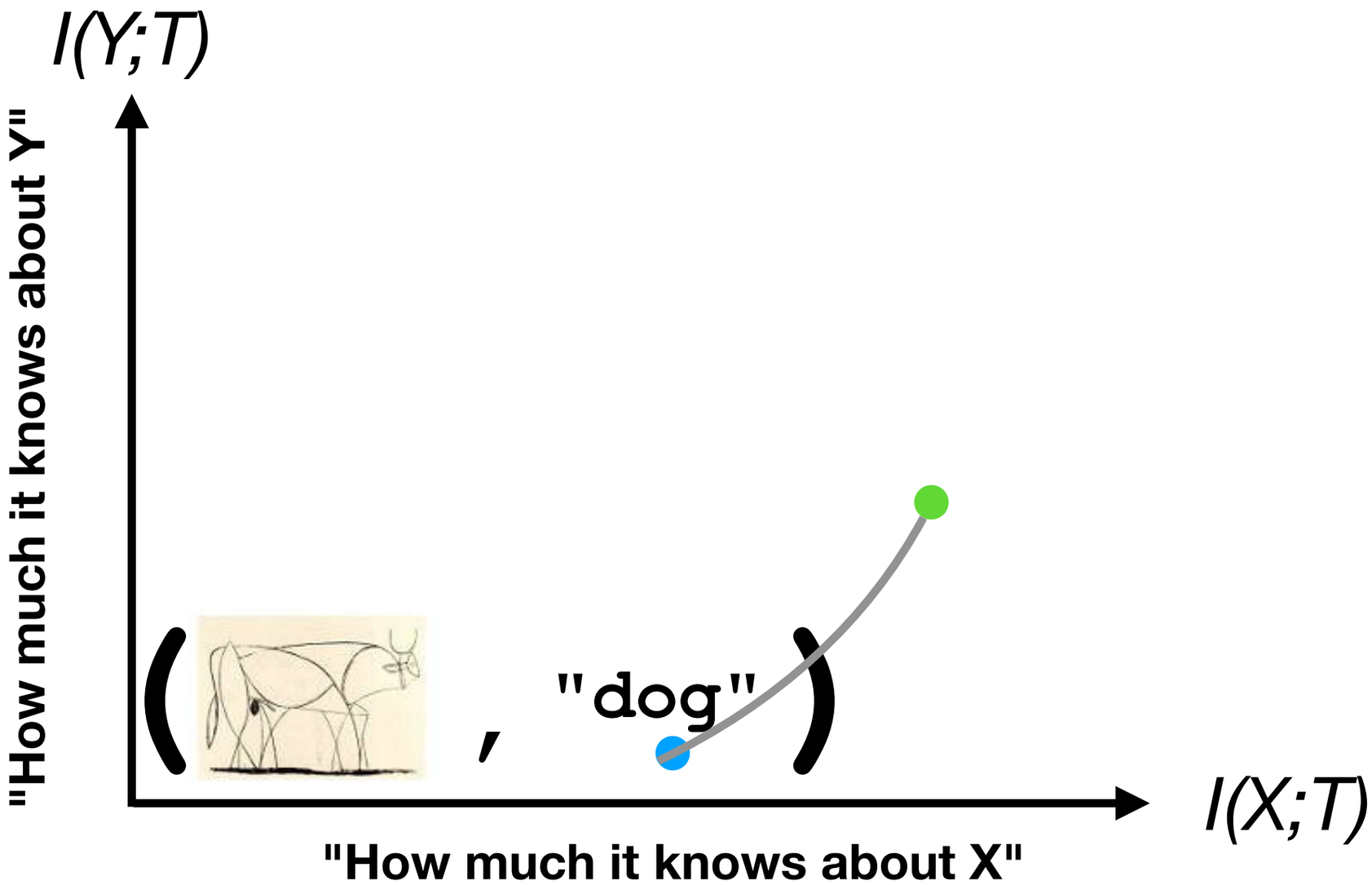
Let's see what it looks like for a fixed data point :

("bull")



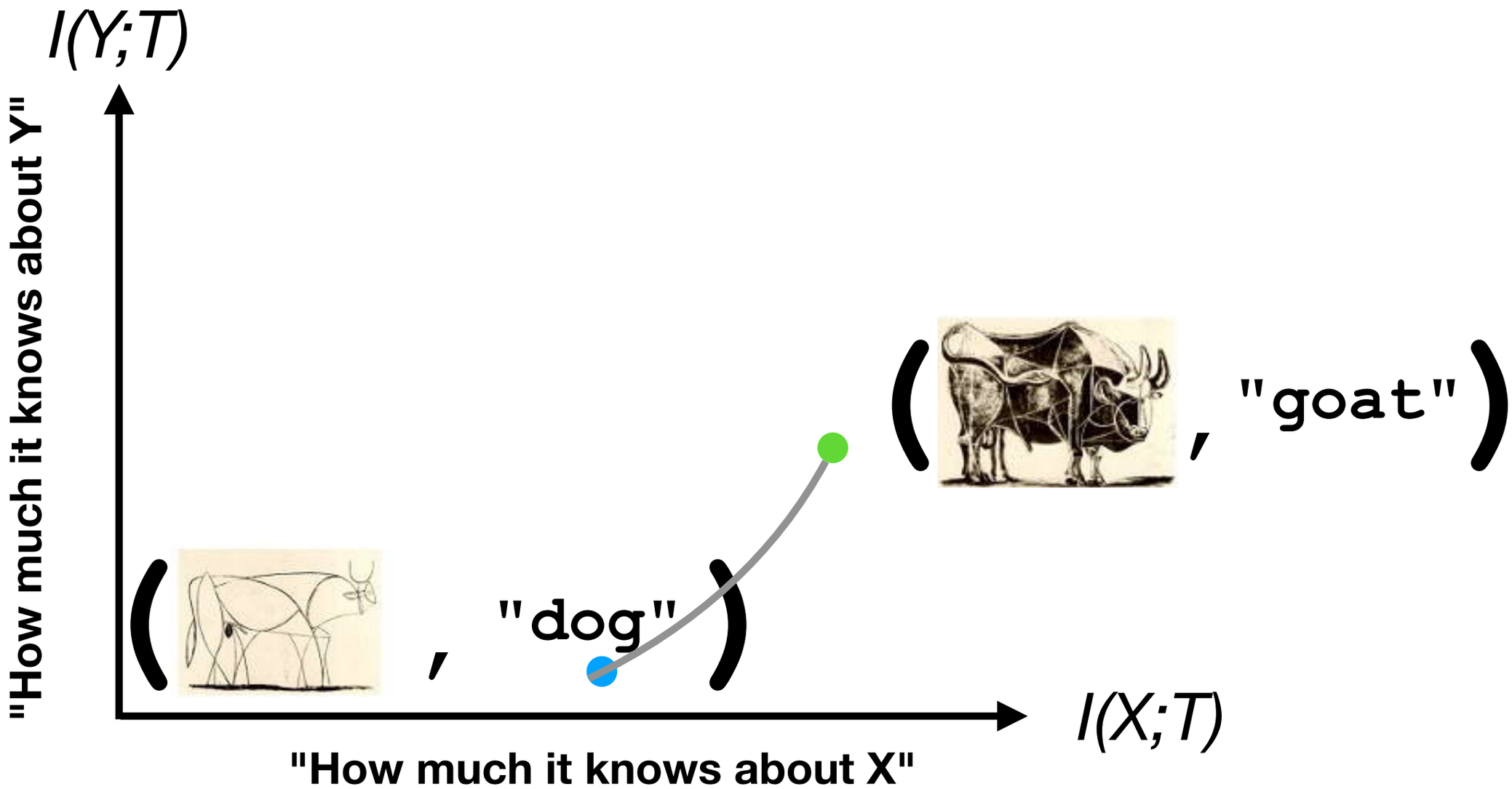
Let's see what it looks like for a fixed data point :

, "bull")



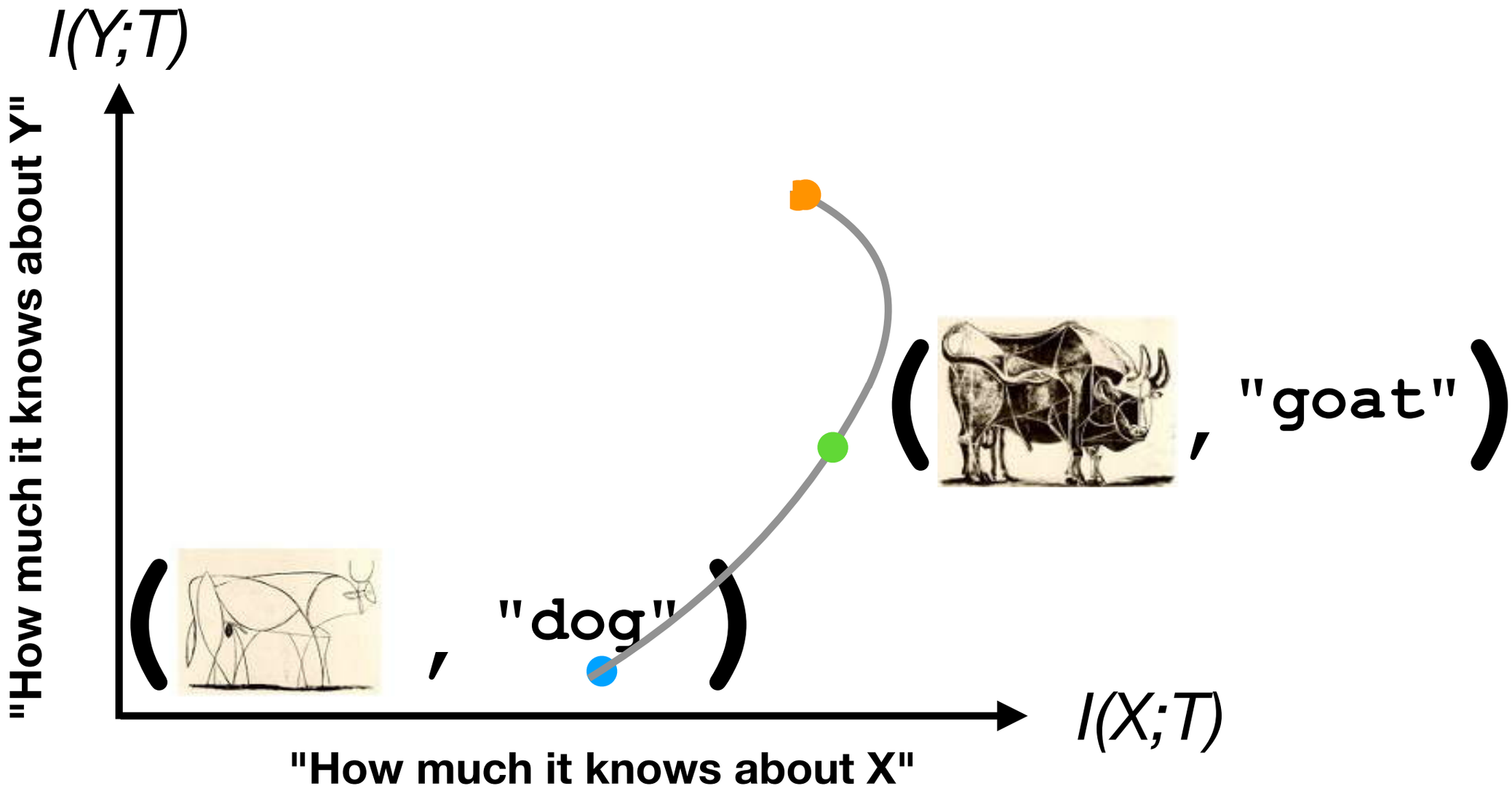
Let's see what it looks like for a fixed data point :

("bull")



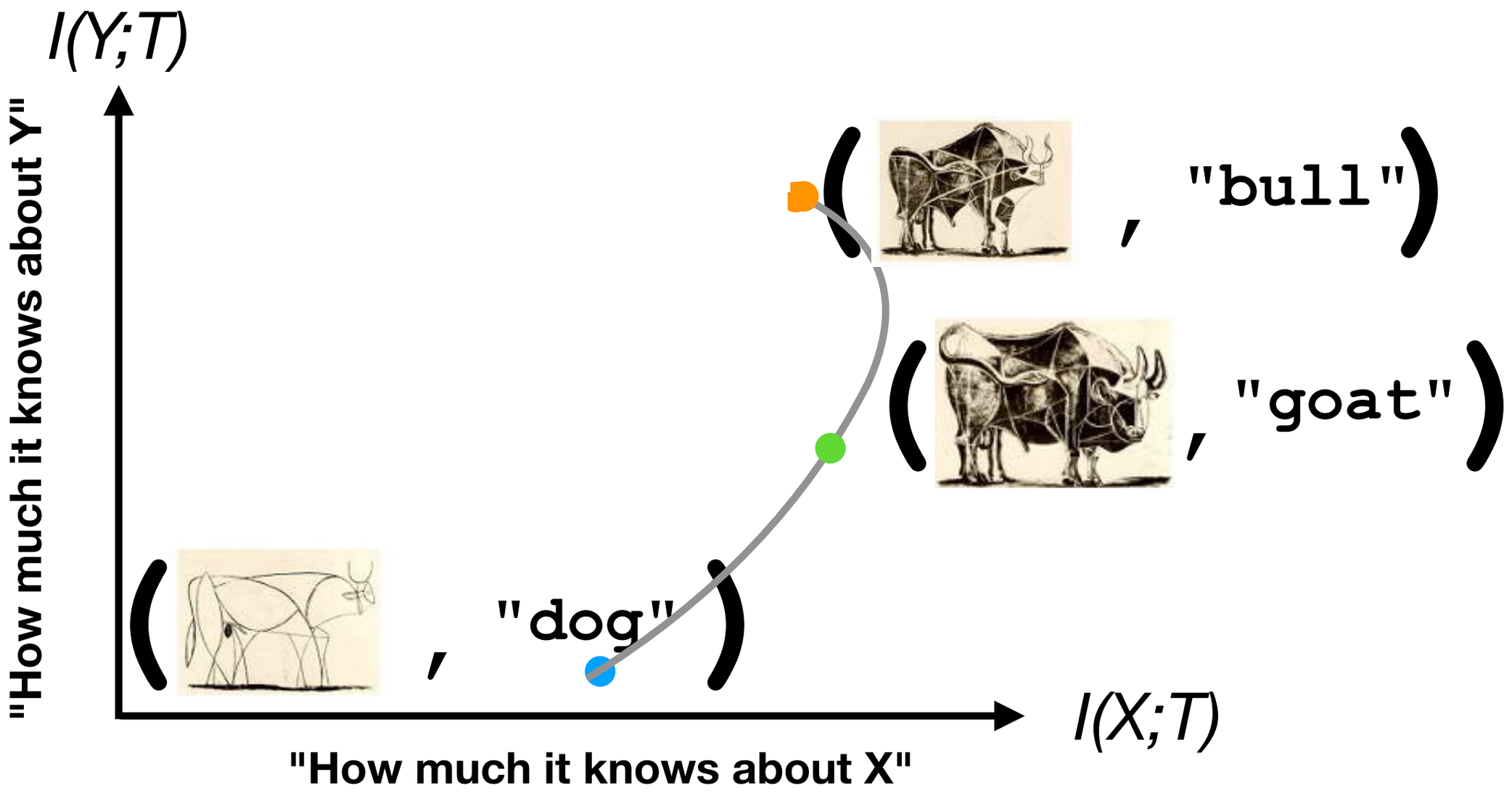
Let's see what it looks like for a fixed data point :

("bull")



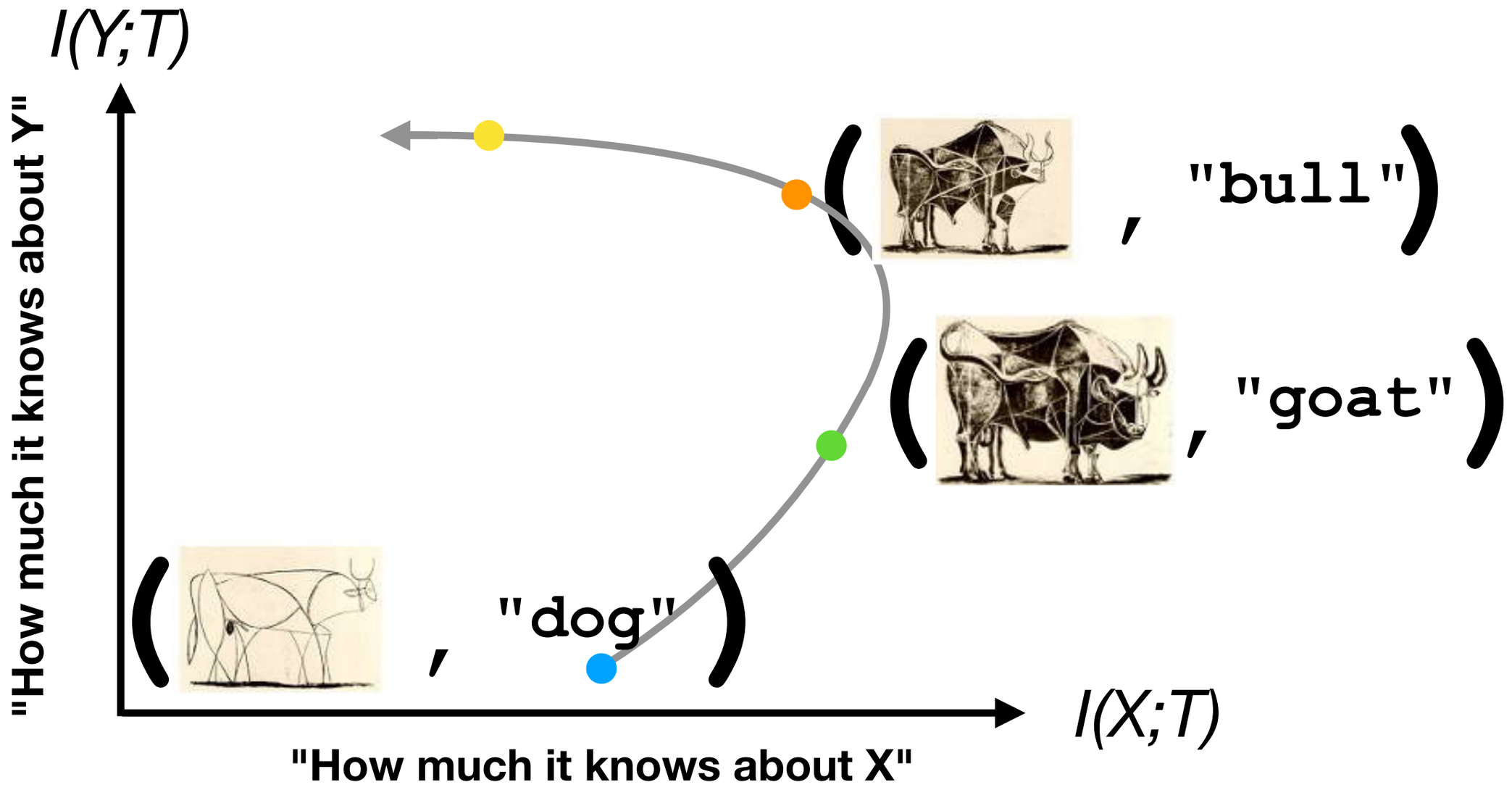
Let's see what it looks like for a fixed data point :

( , "bull")



Let's see what it looks like for a fixed data point :

("bull")

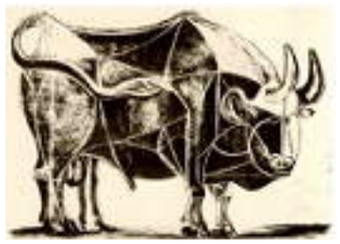


Let's see what it looks like for a fixed data point :

("bull")

("bull")

("bull")



("goat")

("dog")

I(Y;T)

"How much it knows about Y"

I(X;T)

"How much it knows about X"

Numerical Experiments and Results

Examining the dynamics of SGD in the mutual information plane

Experimental Setup

- Explored fully connected feed-forward neural nets, with no other architecture constraints: 7 fully connected hidden layers with widths 12-10-7-5-4-3-2

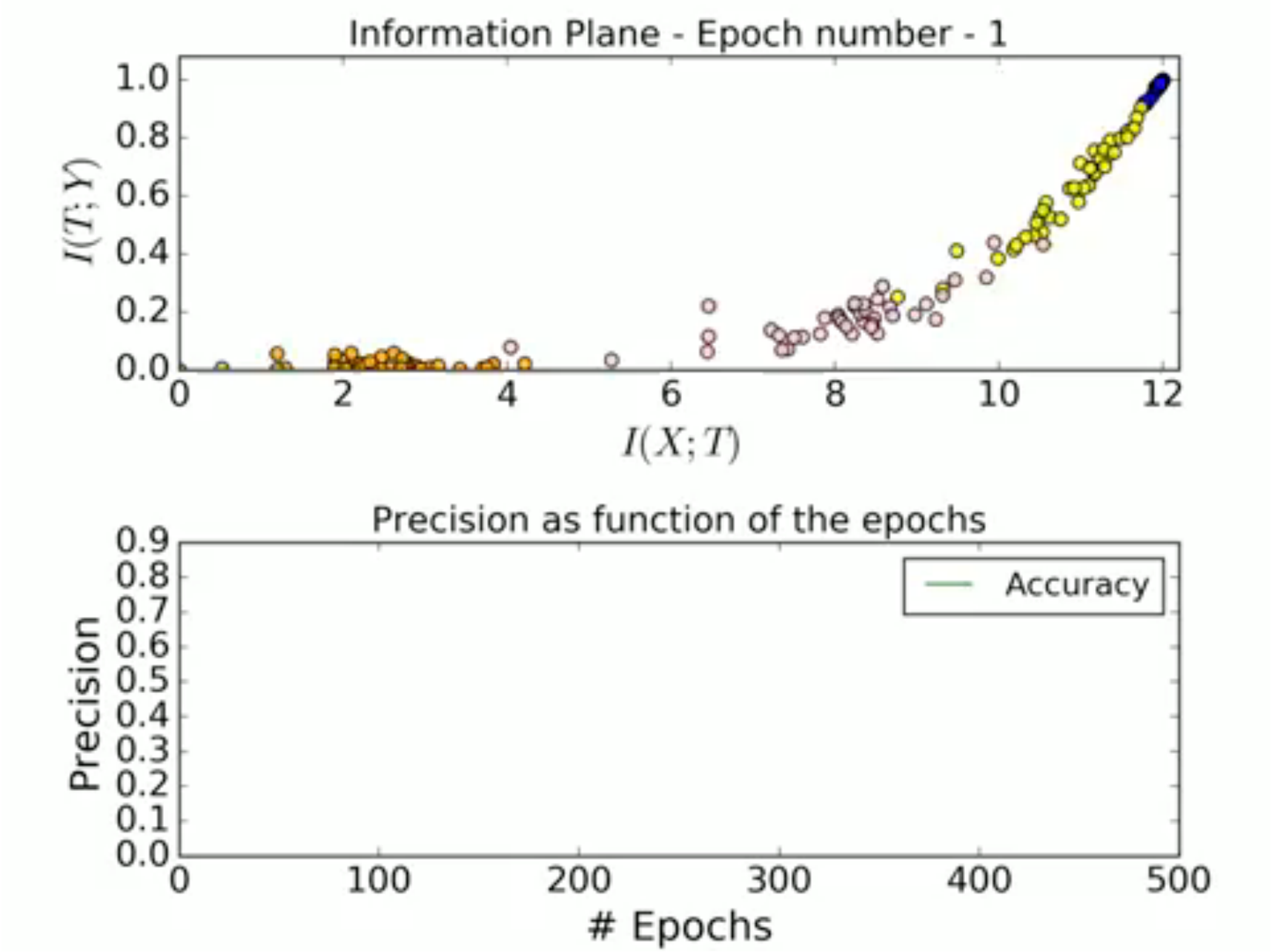
- sigmoid activation on final layer, tanh activation on all other layers

- Binary decision task, synthetic data is used $D \sim P(X, Y)$

- Experiment with 50 different randomized weight initializations and 50 different datasets generated from the same distribution

- trained with SGD to minimize the cross-entropy loss function, and with no regularization

Dynamics in the Information Plane



Dynamics in the Information Plane

- All 50 test runs follow similar paths in the information plane

Dynamics in the Information Plane

- All 50 test runs follow similar paths in the information plane
- Two different phases of training: a fast ‘ER reduction’ phase and a slower ‘representation compression’ phase

Dynamics in the Information Plane

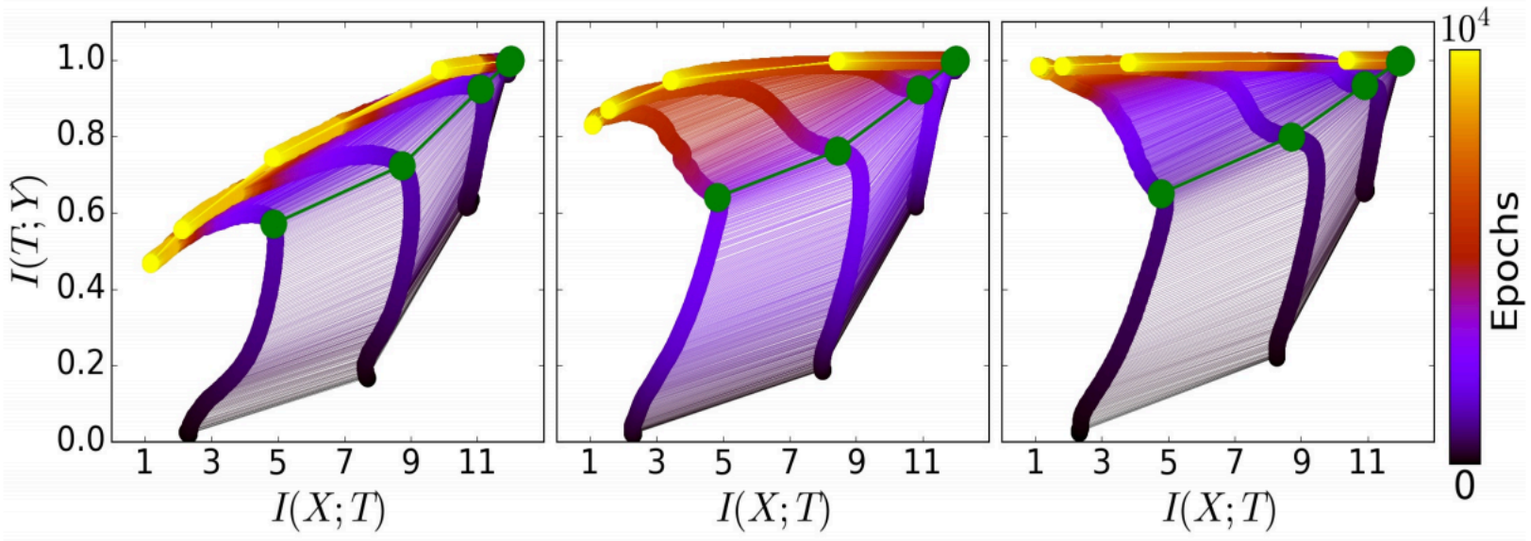
- All 50 test runs follow similar paths in the information plane
- Two different phases of training: a fast ‘ER reduction’ phase and a slower ‘representation compression’ phase
- In the first phase (~400 epochs), the test error is rapidly reduced (increase in $I(T;Y)$)

Dynamics in the Information Plane

- All 50 test runs follow similar paths in the information plane
- Two different phases of training: a fast ‘ER reduction’ phase and a slower ‘representation compression’ phase
- In the first phase (~400 epochs), the test error is rapidly reduced (increase in $I(T;Y)$)
- In the second phase (from 400-9000 epochs) the error is relatively unchanged but the layers lose input information (decrease in $I(X;T)$)

Representation Compression

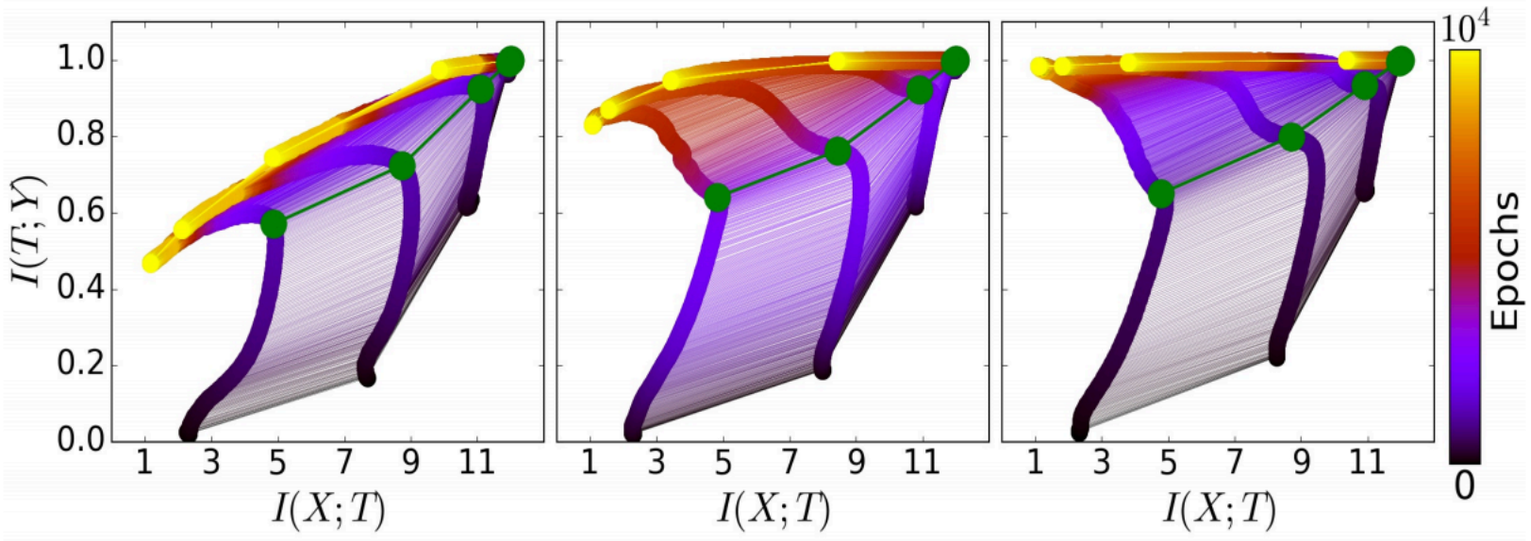
- The loss of input information occurs without any form of regularization



5%, 45%, and 85% of the data respectively

Representation Compression

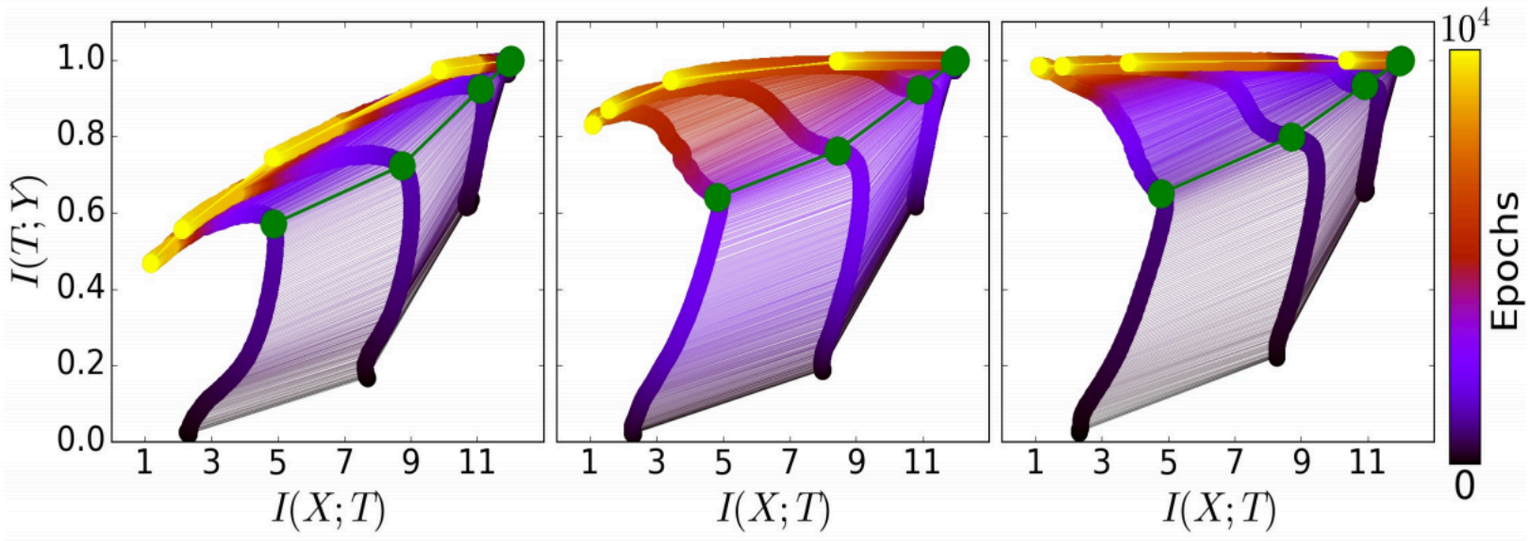
- The loss of input information occurs without any form of regularization
- This prevents overfitting since the layers lose irrelevant information (generalizing by forgetting)



5%, 45%, and 85% of the data respectively

Representation Compression

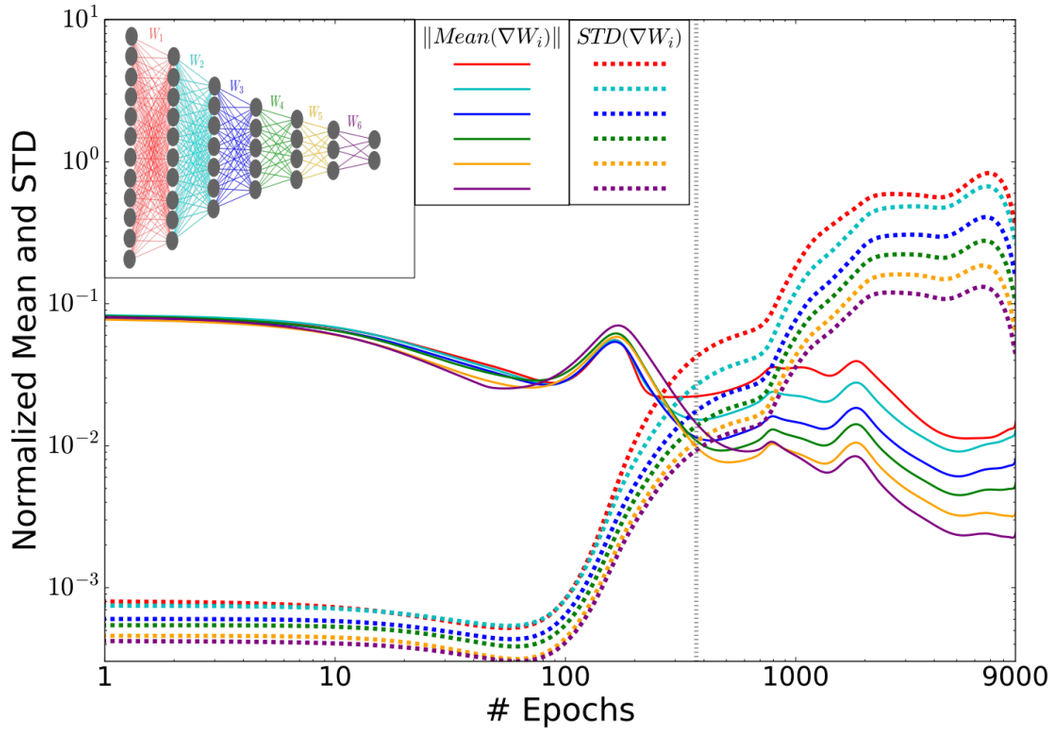
- The loss of input information occurs without any form of regularization
- This prevents overfitting since the layers lose irrelevant information (generalizing by forgetting)
- However, overfitting can still occur with less data



5%, 45%, and 85% of the data respectively

Phase transitions

- The ‘ER reduction’ phase is called a *drift* phase, where the gradients are large and the weights are changing rapidly (high signal-to-noise)

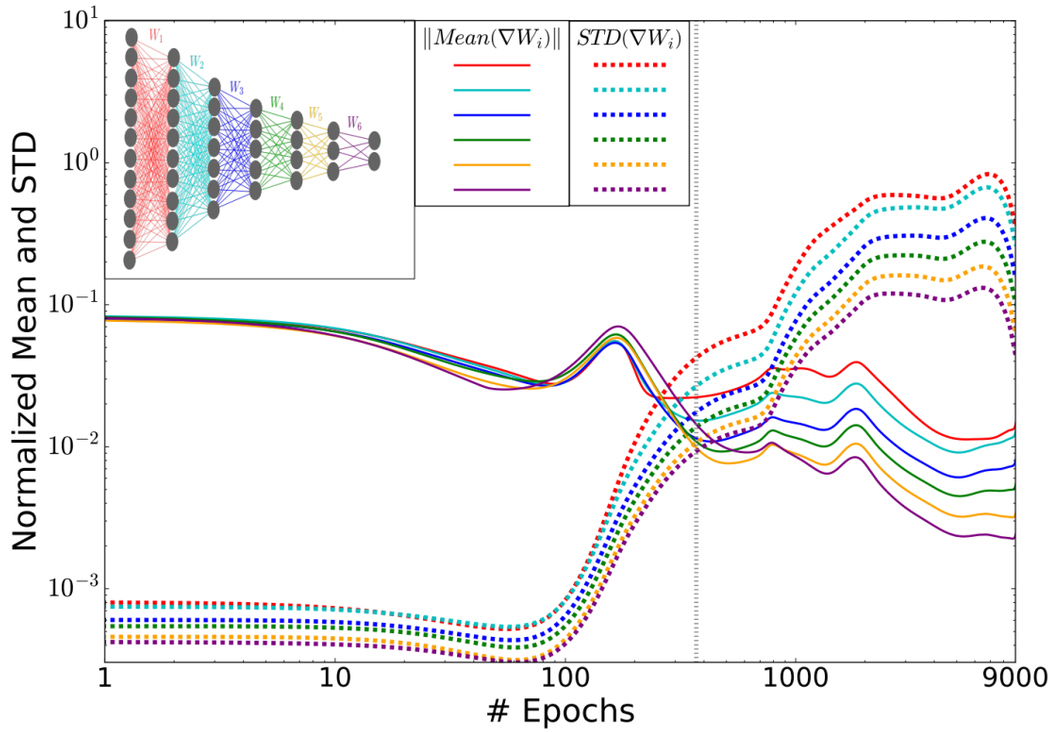


The phase transition occurs at the dotted line

Phase transitions

- The ‘ER reduction’ phase is called a *drift* phase, where the gradients are large and the weights are changing rapidly (high signal-to-noise)

- The ‘Representation compression’ phase is called a *diffusion* phase, where the gradients are small compared to their variance (low signal-to-noise)



The phase transition occurs at the dotted line

Effectiveness of Deep Nets

- Because of the low Signal-to-noise Ratio in the diffusion phase, the final weights obtained by the DNN are effectively random

Effectiveness of Deep Nets

- Because of the low Signal-to-noise Ratio in the diffusion phase, the final weights obtained by the DNN are effectively random
- Across different experiments, the correlations between the weights of different neurons in the same layer was very small

Effectiveness of Deep Nets

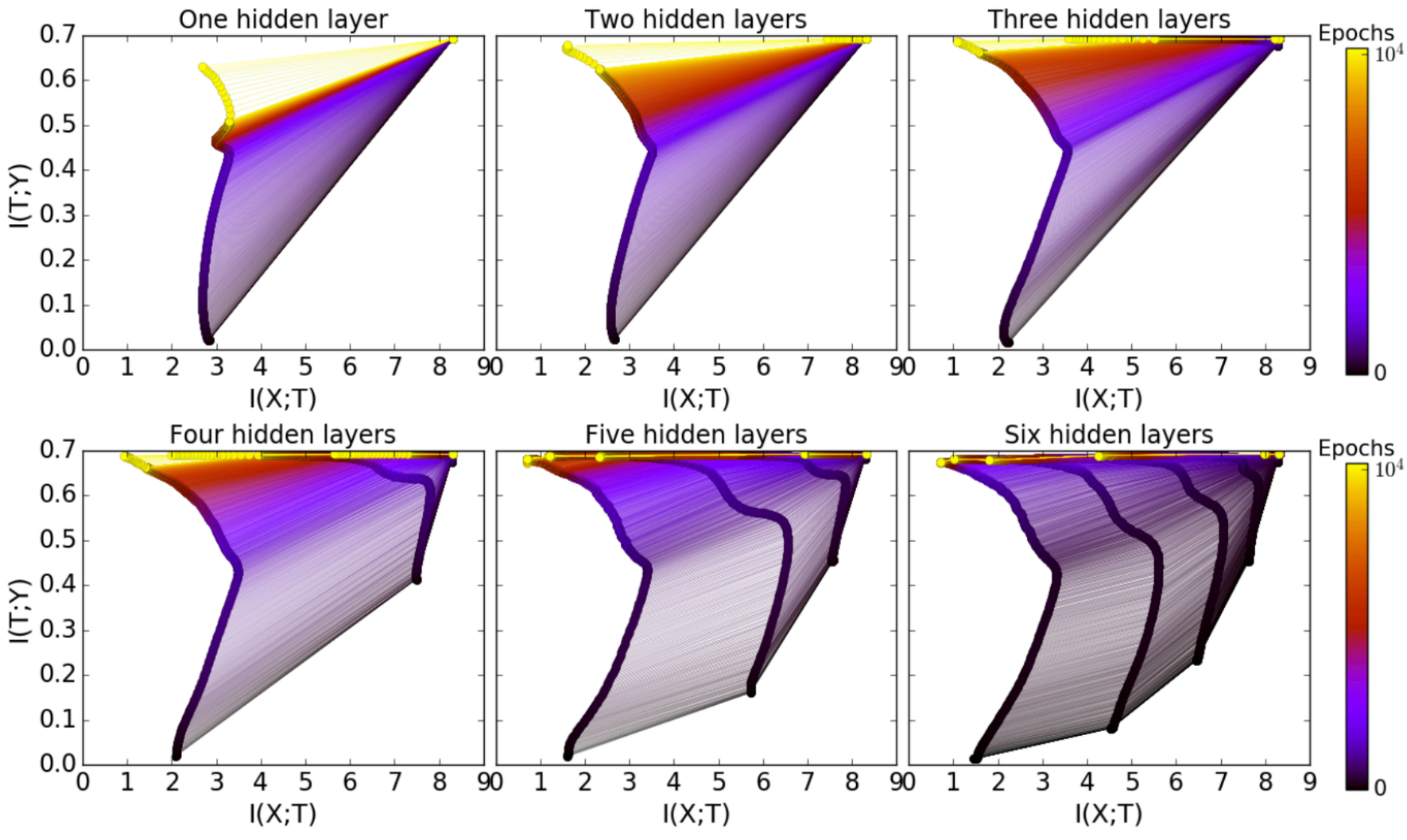
- Because of the low Signal-to-noise Ratio in the diffusion phase, the final weights obtained by the DNN are effectively random

- Across different experiments, the correlations between the weights of different neurons in the same layer was very small

- “This indicates that there is a huge number of different networks with essentially optimal performance, and attempts to interpret single weights or single neurons in such networks can be meaningless”

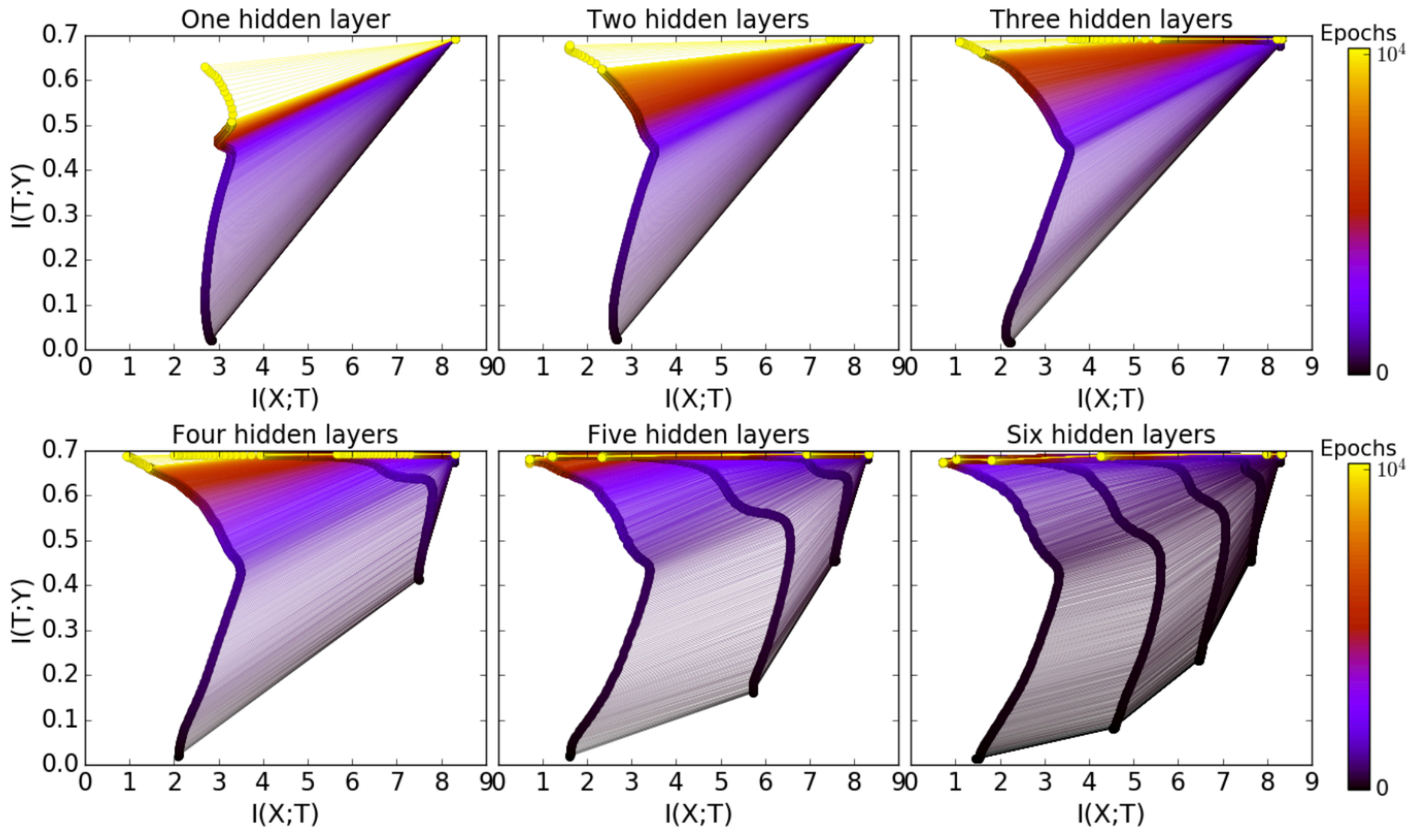
Why go deep?

- Faster ER minimization (in epochs)



Why go deep?

- Faster ER minimization (in epochs)
- Faster representation compression time (in epochs)



Discussions/Disclaimers

- The claims being made are certainly very interesting, although the scope of experiments is limited: only one specific distribution and one specific network are examined.

Discussions/Disclaimers

- The claims being made are certainly very interesting, although the scope of experiments is limited: only one specific distribution and one specific network are examined.

- The paper acknowledges that different setups need to be tested: do the results hold with different decision rules and network architectures? And is this observed in “real world” problems?

Discussions/Disclaimers

- As of now there is some controversy about whether or not these claims hold up: see “On the Information Bottleneck Theory of Deep Learning (Saxe et al., 2018)”

- “ Here we show that none of these claims hold true in the general case. […] we demonstrate that the information plane trajectory is predominantly a function of the neural nonlinearity employed […] Moreover, we find that there is no evident causal connection between compression and generalization: networks that do not compress are still capable of generalization, and vice versa. Next, we show that the compression phase, when it exists, does not arise from stochasticity in training by demonstrating that we can replicate the IB findings using full batch gradient descent rather than stochastic gradient descent.”

The End