Mirror Integration

with an example on functional regularization.

Jialin Lu, Oct 14th 2020
Meeting of Ester Lab

SIMON FRASER
UNIVERSITY
ENGAGING THE WORLD

The original plan.

To introduce an interesting paper that I recently read.

Accepted as oral in NeurIPS 20

Continual Deep Learning by Functional Regularisation of Memorable Past

Pingbo Pan,[1,*,†] **Siddharth Swaroop,**[2,*,‡] **Alexander Immer,**[3,†] **Runa Eschenhagen,**[4,†]
Richard E. Turner,[2] **Mohammad Emtiyaz Khan**[5,‡].

[1] University of Technology Sydney, Australia
[2] University of Cambridge, Cambridge, UK
[3] École Polytechnique Fédérale de Lausanne, Switzerland
[4] University of Tübingen, Tübingen, Germany
[5] RIKEN Center for AI Project, Tokyo, Japan

gave a tutorial on bayesian deep learning last year NeurIPS 19.
I remember many lab members attended

But then I realize that

1. the exact technical solution of paper is quite hard to understand and
2. is not necessary the techniques we have to follow. Yeah so I will not cover the math-heavy details :)
3. But the underlying logic of the approach should be quite interesting and useful.

Outline

PART 1: Introducing a categorization of relevant works

I will give descriptions and example to explain the categorization.

(Somewhere in this part, I also introduce my previous work and ask for your help.)

But the main thing I want to introduce is the mirror integration.

PART 2: Introduce the paper in the context of continual learning.

Introduce this thing called functional regularization

Introduce the paper, why it works and what we can learn from it.

PART 3: Introduce further implications and possible future works.

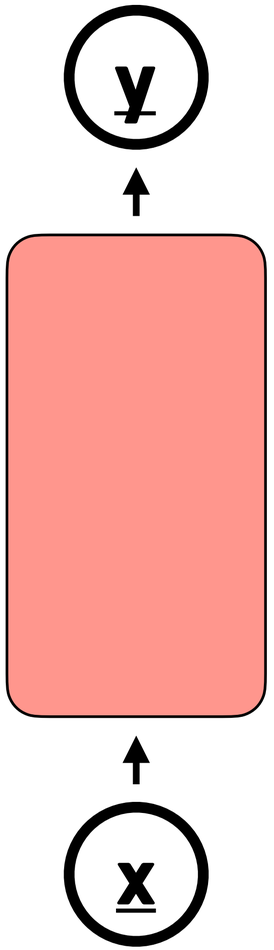
Usually, like my previous presentation, there is a link to web post that describes this presentation in text (and with math and references), this time I am sorry I fail to make it in time.

PART 1

A categorization

of hybrid models of deep learning.

Deep Learning, the usual way: A big model approach



Using a big model, massive parameters

Efficiently train the net by end-to-end backprop.

Why a big monolithic deep neural network is not so good?

1. generalization ability: in some datasets there is often quite significant distribution shift between test/train set and among different datasets.

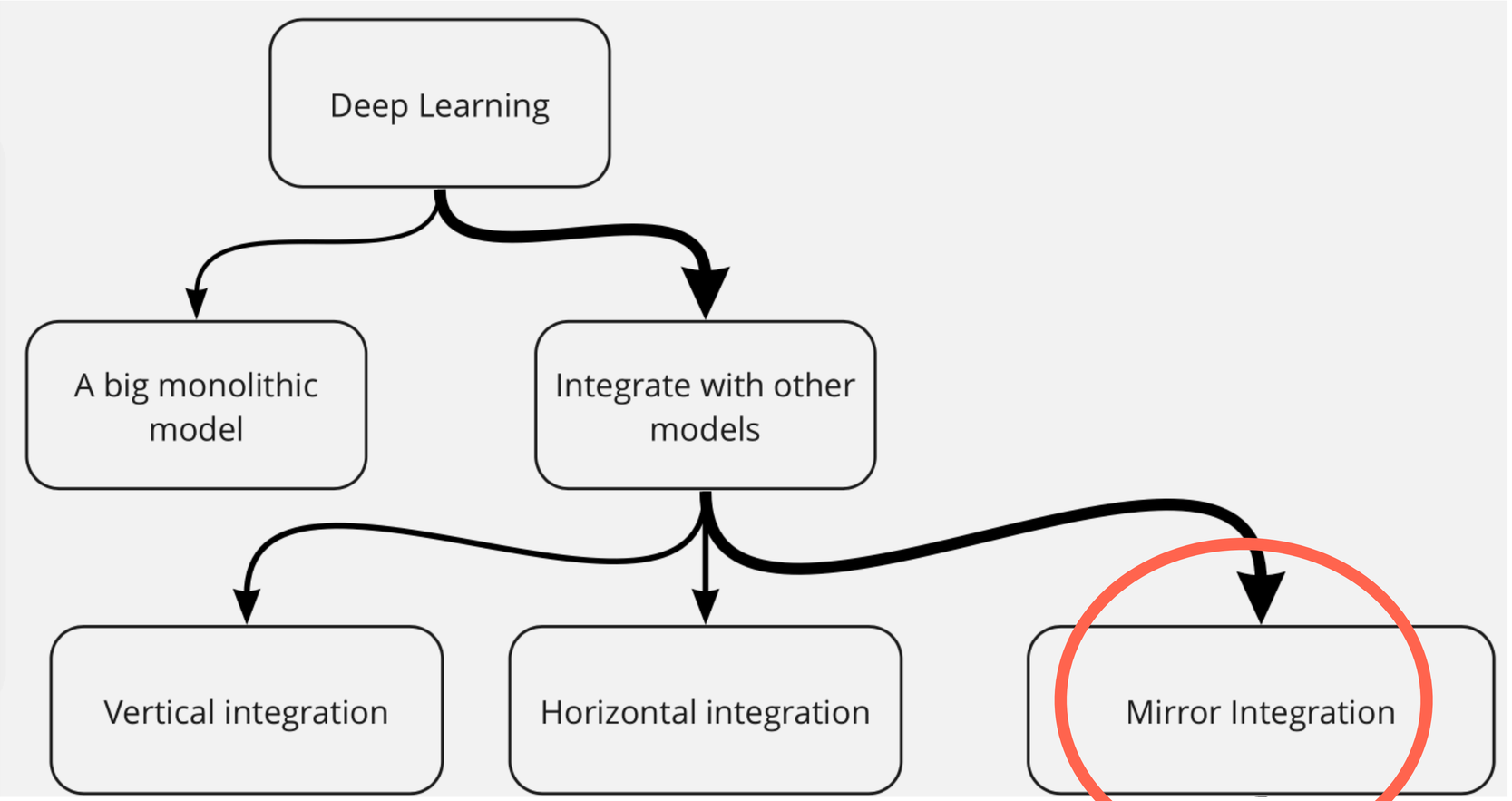
2. small dataset size and high-dimension input: a typical in some datasets, make DL hard to work

3. predictive uncertainty: well-calibrated uncertainty is certainly important. In particular, it would be ideal to somehow know how 'out of distribution' a particular test sample is.

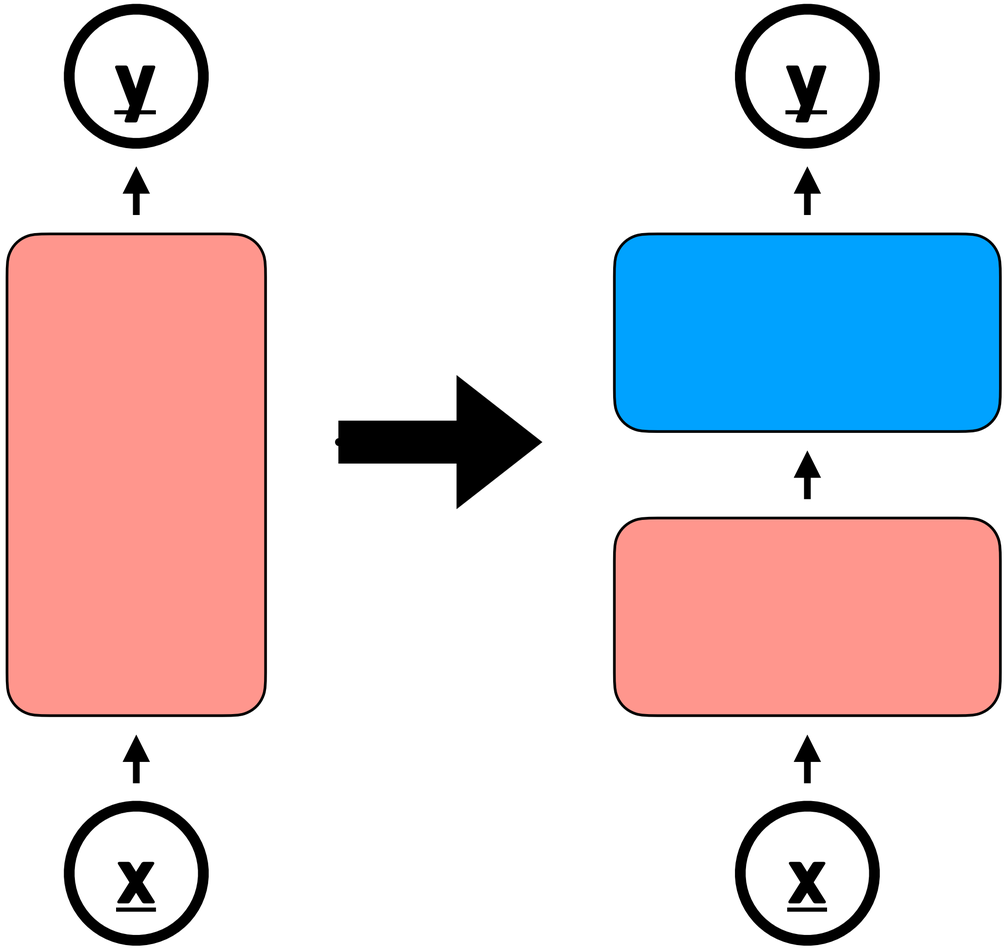
4. interpretability: we are essentially more interested in discovering the underlying mechanisms rather than a good predicting machine.

5. when we have small datasets, we want easy control for active learning, transfer learning, continual learning, etc: we do not know what is learned and what is not.

Move from a big model to a integrated hybrid model



Vertical Integration

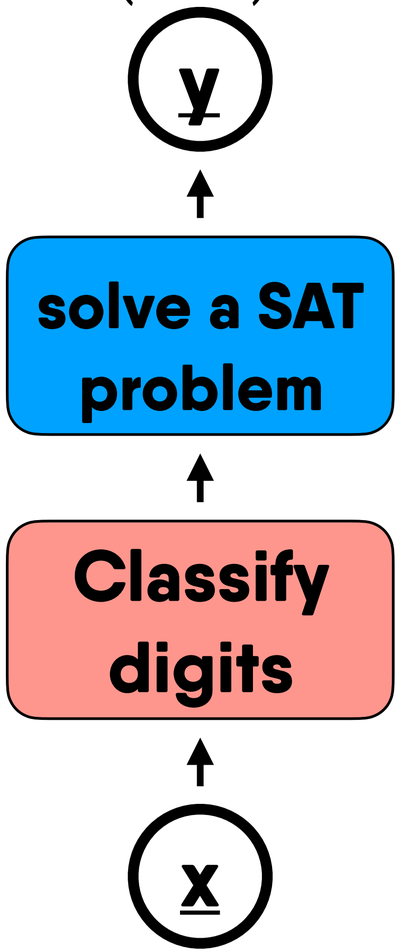


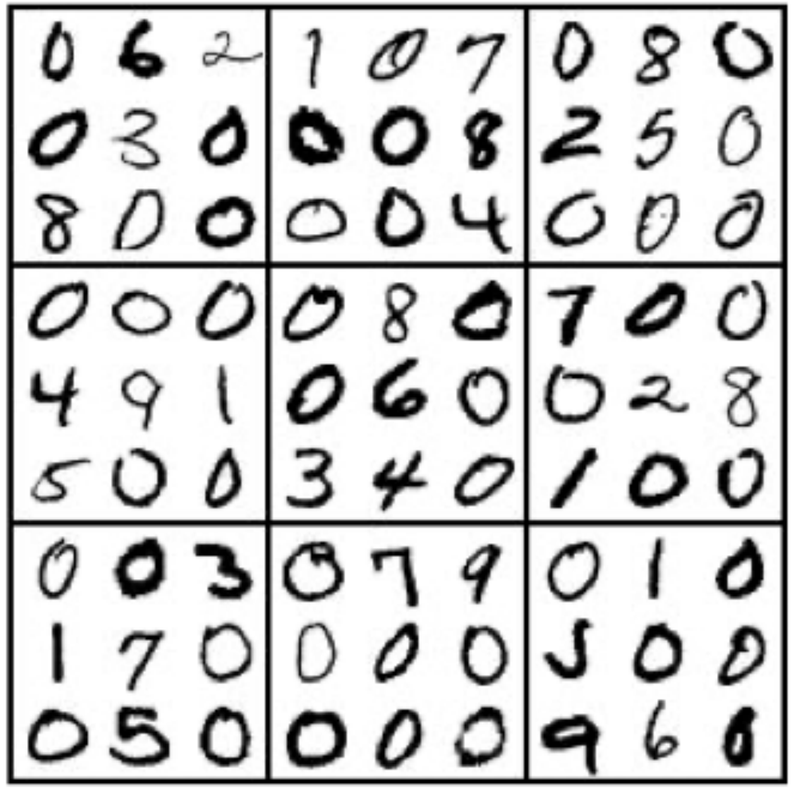
We split a task into multiple stages (when it is possible and makes sense)

There is an easy version and a hard version.
easy version: we just separately train the stages.
hard version: we expect these multiple stage can be trained jointly (end-to-end backprop)

Example

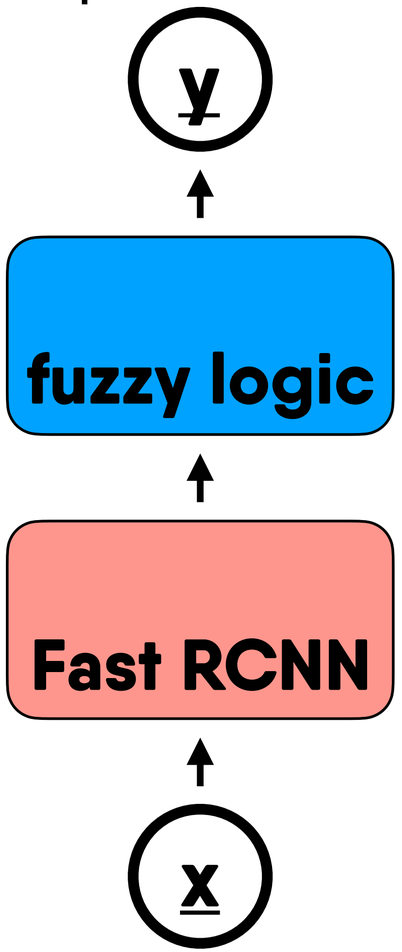
SAT-Net (Wang et al., 2019a) utilizes a symbolic SAT solver layer on top of a convolutional neural network (CNN) for solving a satisfiability-based task of visual Sudoku

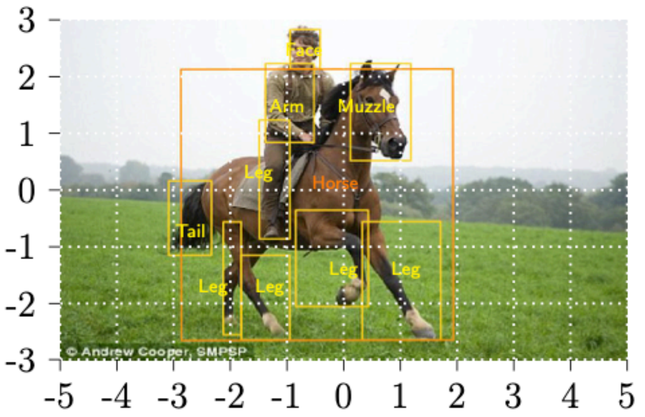




Example

Donadello et al. (2017) utilizes first-order fuzzy logic on top of a Fast-RCNN to extract structured semantic descriptions from images

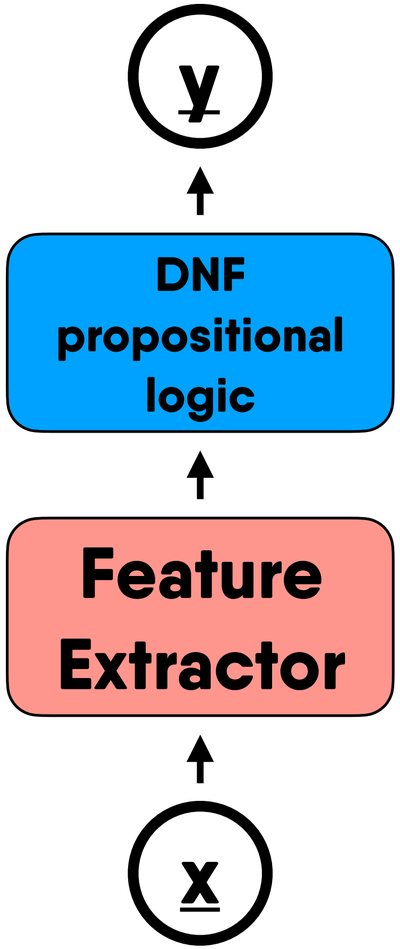


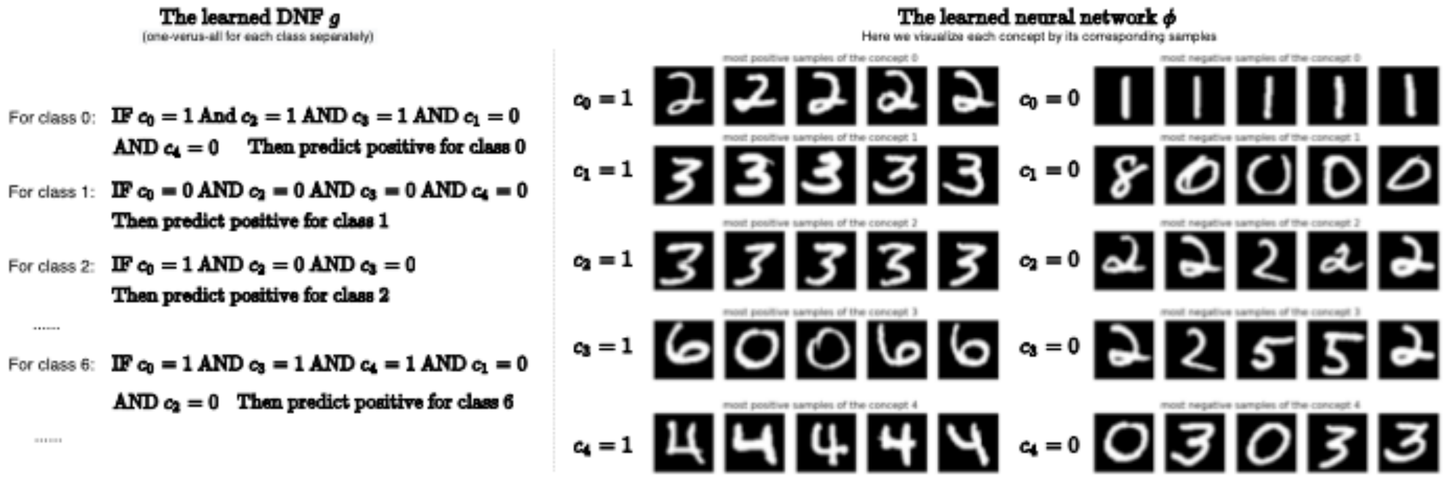


Person	$\sqsubseteq$	$(= 1)$hasParts.Face
		$\sqcap\ (= 2)$hasParts.Leg
		$\sqcap (= 2)$hasParts.Arm
Horse	$\sqsubseteq$	$(= 1)$hasParts.Muzzle
		$\sqcap\ (= 4)$hasParts.Leg
		$\sqcap\ (= 1)$hasParts.Tail
Horse	$\sqsubseteq$	$\forall$hasPart.$(\neg$Face$)$
		$\sqcap\ \forall$hasPart.$(\neg$Arm$)$
Person	$\sqsubseteq$	$\forall$hasPart.$(\neg$Muzzle$)$
		$\sqcap\ \forall$hasPart.$(\neg$Tail$)$
Horse	$\sqsubseteq$	$\forall$ride.$\bot$
Person	$\sqsubseteq$	$\forall$ride^{-}.$\bot$
Leg	$\sqsubseteq$	Limb
Arm	$\sqsubseteq$	Limb

Example

Neural Disjunctive Normal Form: use a feature extractor network to produce binary features and use propositional logic to do classification.

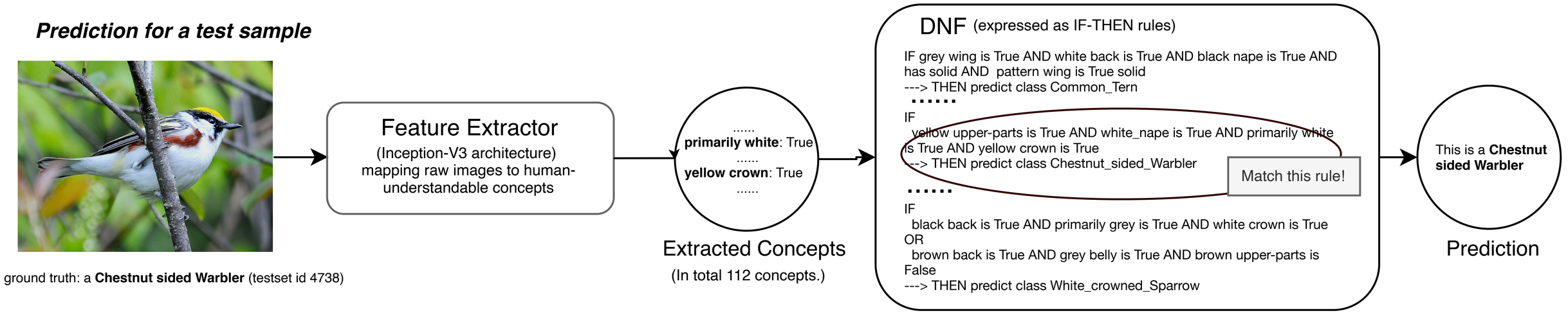




an end-to-end example on MNIST, but the features are not necessarily aligned with human perception.

Example

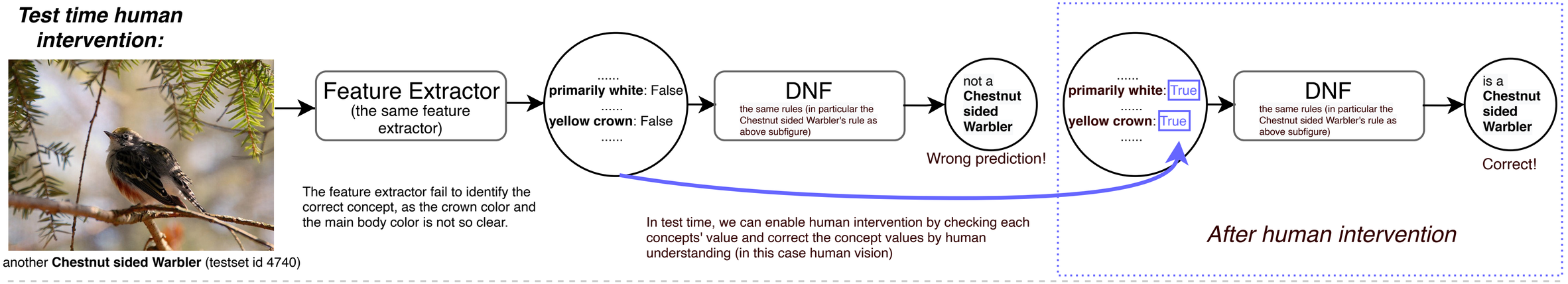
Another example of Neural DNF but this time we also have the annotated features



Now we can have a quite interpretable model for this bird species classification

Example

We can also do test time human-intervention. If the extracted is wrong, human can inspect and correct the wrong concepts and then the prediction now becomes correct



Example

We can go even fancier to classify a bird that does not exist in the dataset (or the real world), by utilizing human knowledge to tweaking the rules.

To classify an imaginary class by manipulating the model

An bird of the imaginary class
Blue-crown Chestnut sided Warbler

There is no such bird as **Blue-crown Chestnut sided Warbler**
the image is a synthetic image

However we can still be able to classify this imaginary class

The decision rule for **Chestnut sided Warbler** is as below. Note that it has a yellow crown.

IF yellow upper-parts is True AND white_nape is True AND primarily white is True AND yellow crown is True
---> THEN predict class Chestnut_sided_Warbler

In this highly interpretable Neural DNF model, we can tweak the rules to classify **Blue-crown Chestnut sided Warbler**, even if it does not exist in the training data (or even real world)

(the concept of blue crown, though, is in the training set)

IF yellow upper-parts is True AND white_nape is True AND primarily white is True AND blue crown is True
---> THEN predict class Chestnut_sided_Warbler

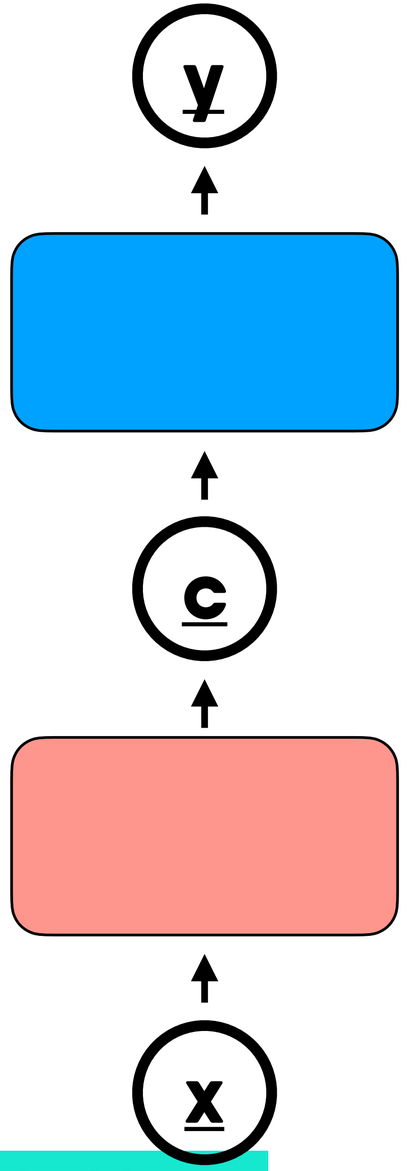
Ask for help

We are still looking for some good datasets that the Neural DNF can apply.

Either: the datasets have annotated features **C** such that we can predict X -> C -> Y

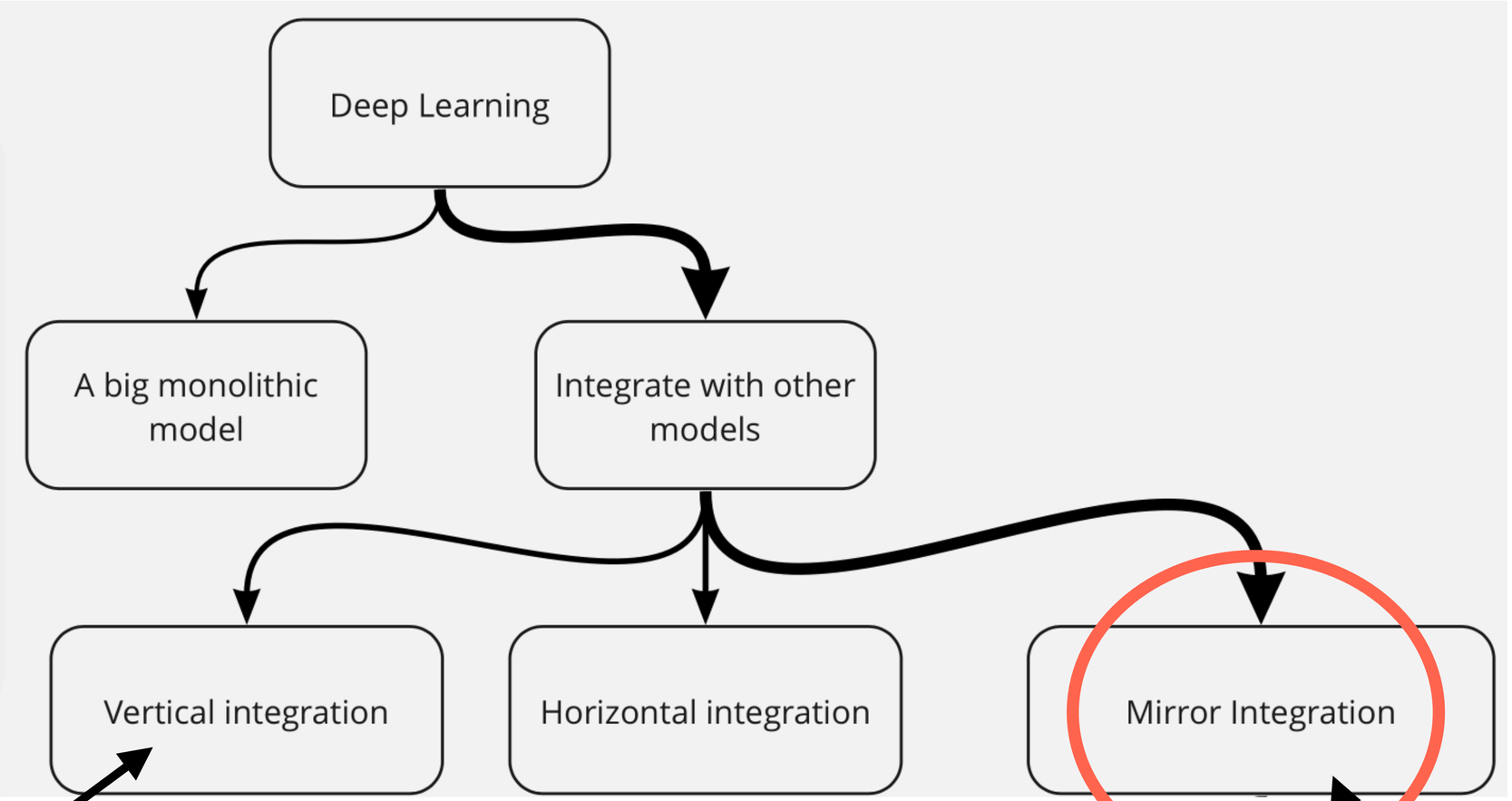
Or: the underlying classification mechanism of C->Y has a propositional logic nature. It can be described in proportional logic.

Or: based on a current supervised dataset and improvise a classification function: for example, give two MNIST digit image as input, and classify the two-digit number is a prime number or not. (Then the rules should encompass all the prime number <100.



If you know a good dataset or some relevant interesting things you think this Neural DNF model can do, then please tell me!

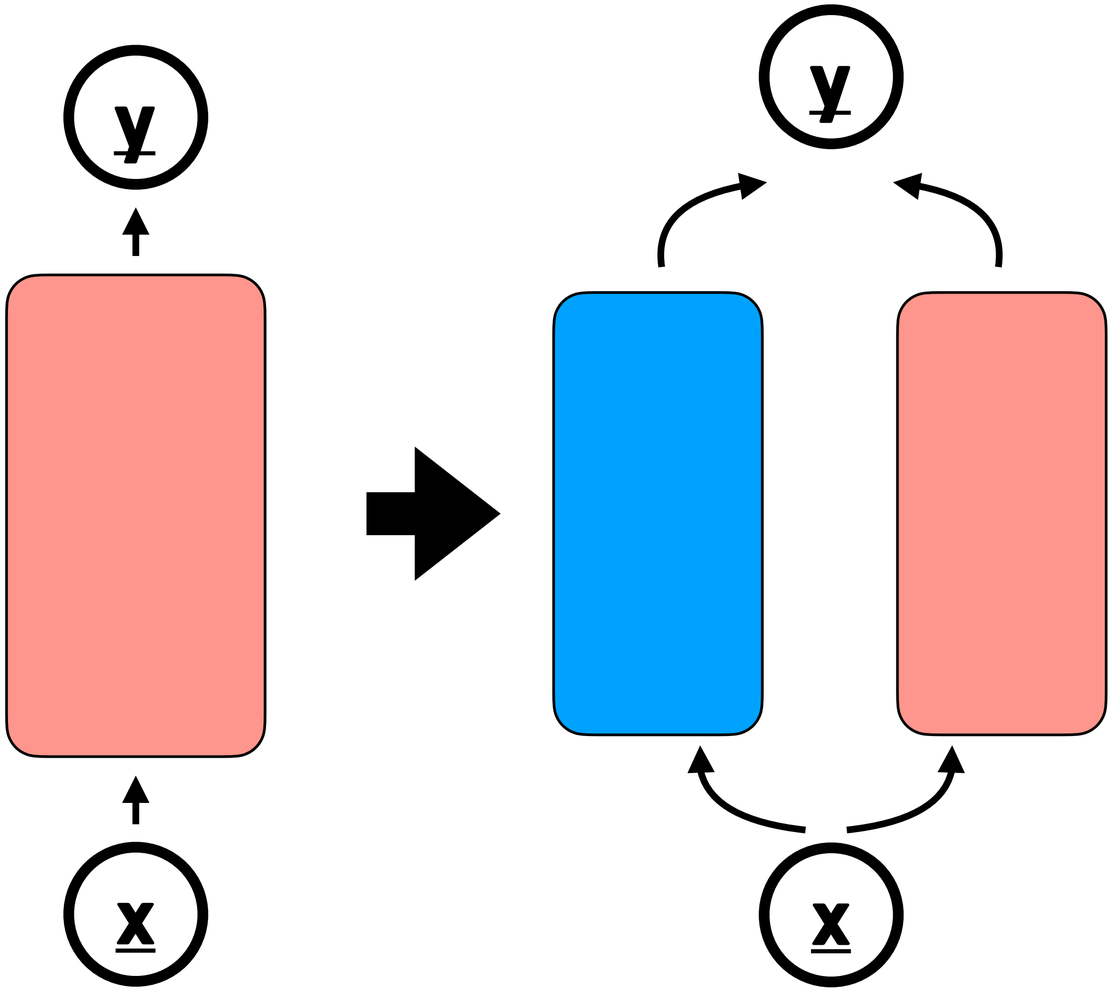
Move from a big model to a integrated hybrid model



Martin tell me to also introduce my work that is in this category and ask for help. but it is not the focus of today

But anyway this Mirror integration is the focus of today

Horizontal Integration



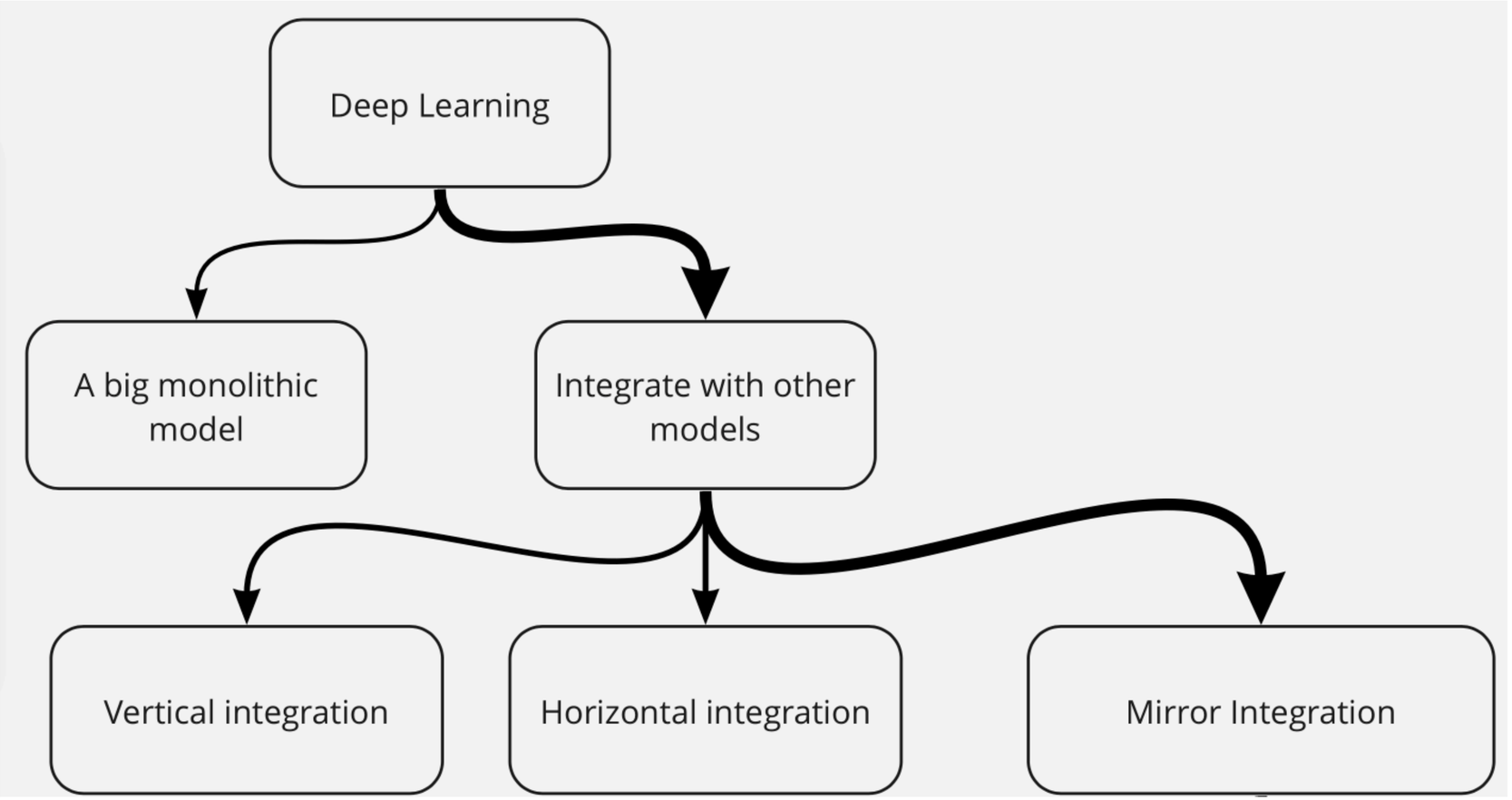
Horizontal integration means you have multiple different models that serve the same purpose, one of them might are better than others in certain test samples.

At one end of extreme, you can think it as conditional computation.

At the other end of extreme, you can think it as an ensemble.

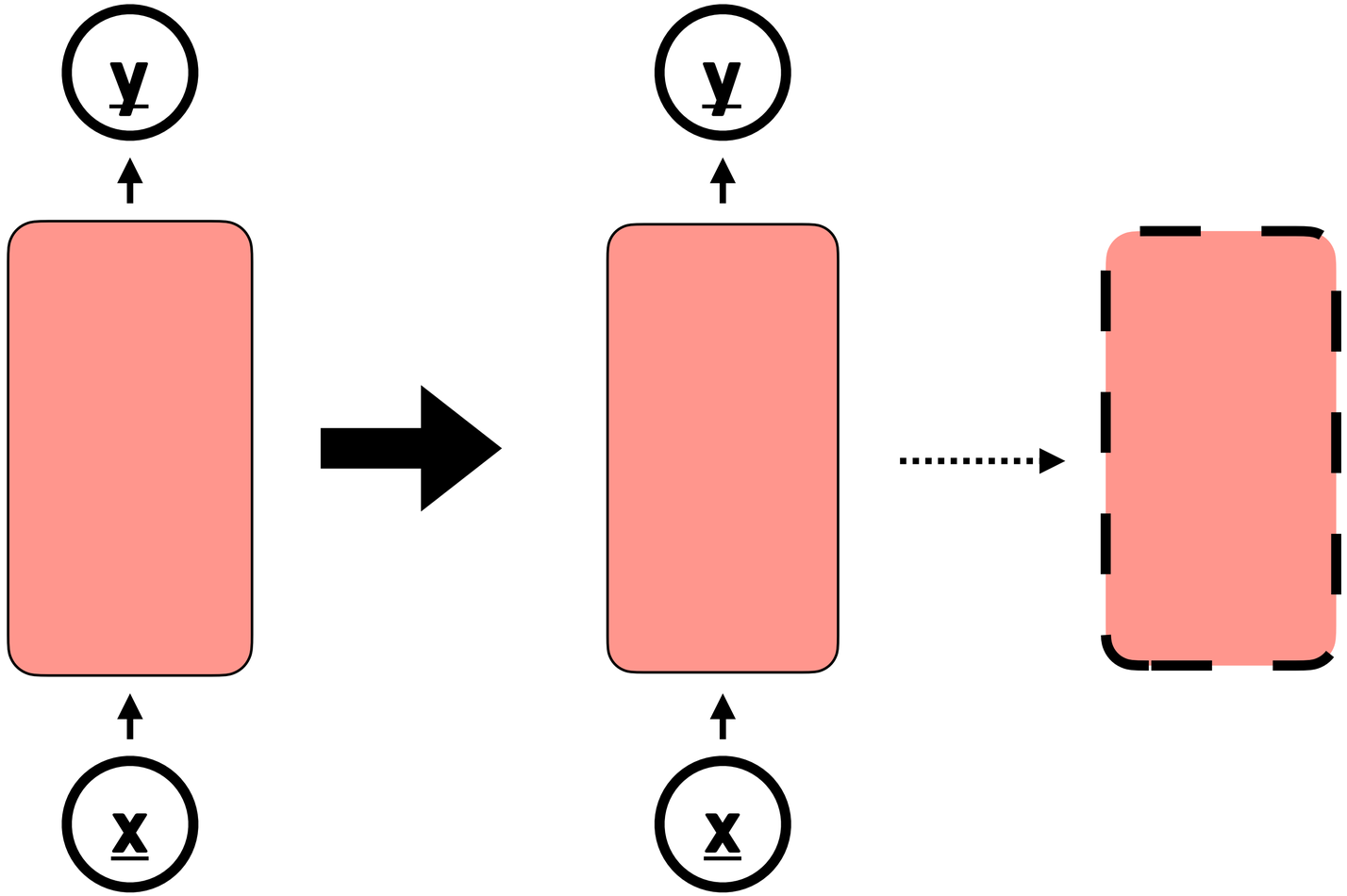
But it can also be the case that these modules can have complex interactions.

Move from a big model to a integrated hybrid model



The term of Vertical and Horizontal comes from marketing terminology

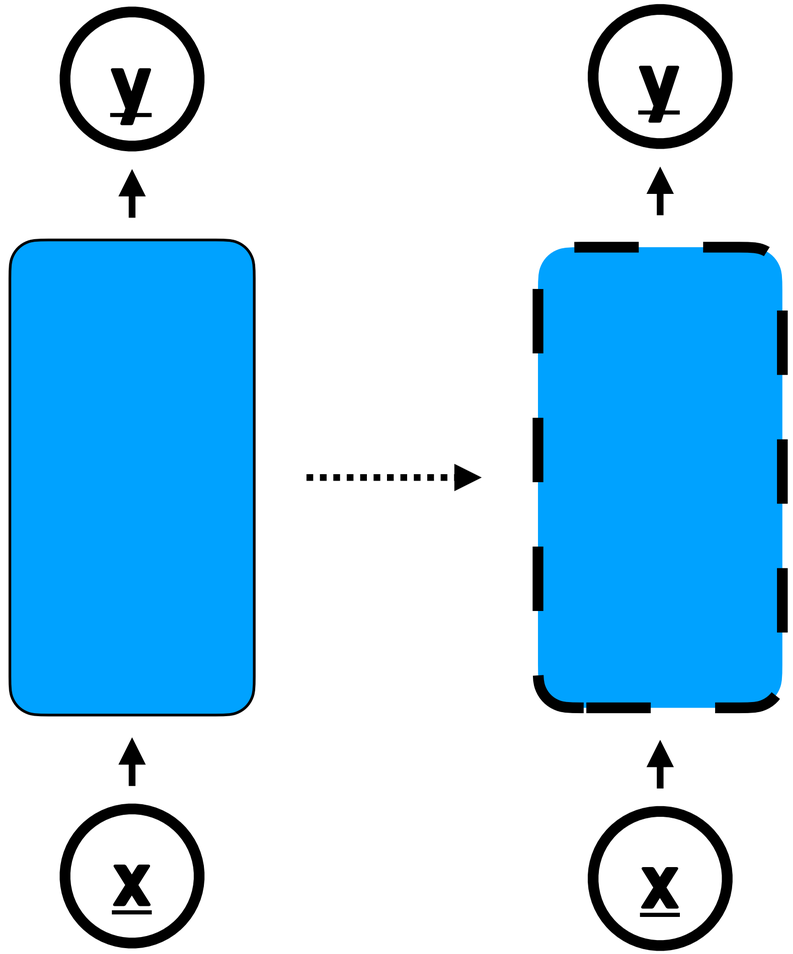
Mirror Integration



We first train a neural work, and then we use another model to approximate it.

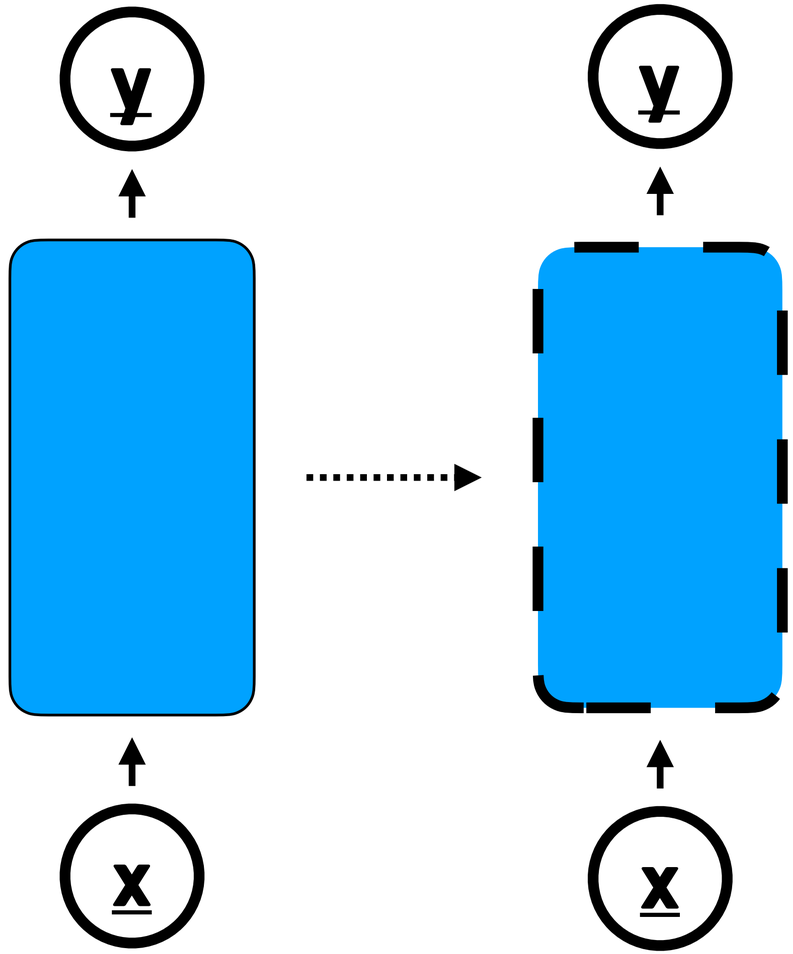
At first it seems very counterintuitive ……

Mirror Integration Type 1



The goal is to use the new model.

Mirror Integration Type 1

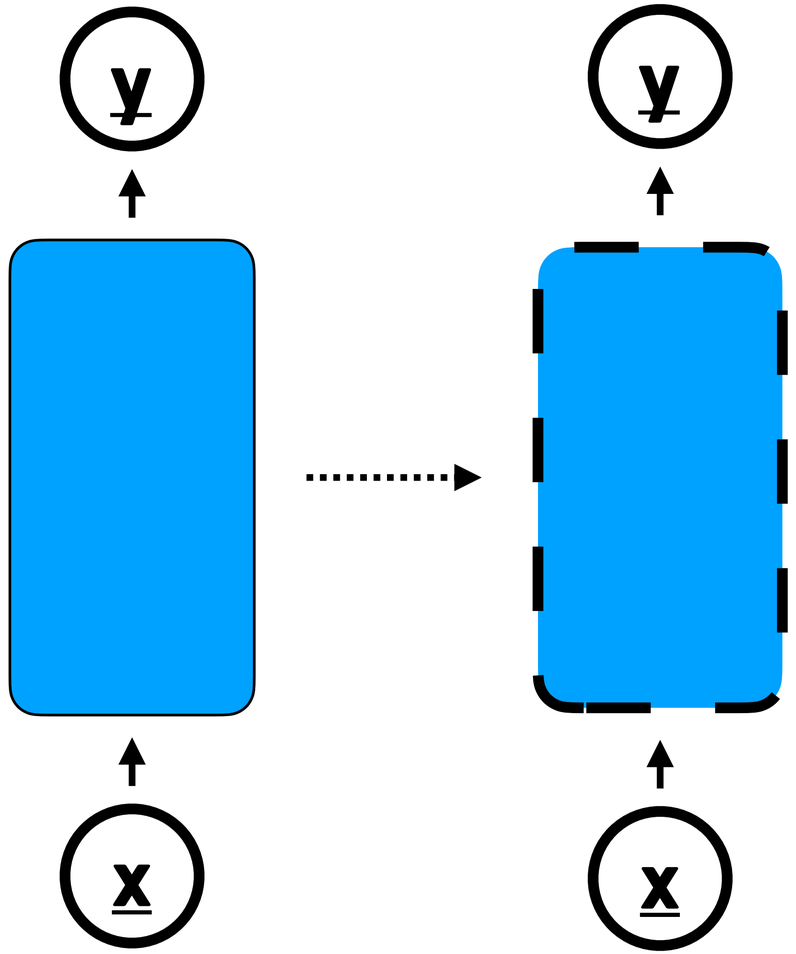


1. Use the new model to do better prediction, better predictive accuracy, better predictive uncertainty, etc.

2. Fit a simpler and interpretable model for interpretable prediction.

3. Fit a model whose structure can be better interpreted by domain experts for knowledge discovery.

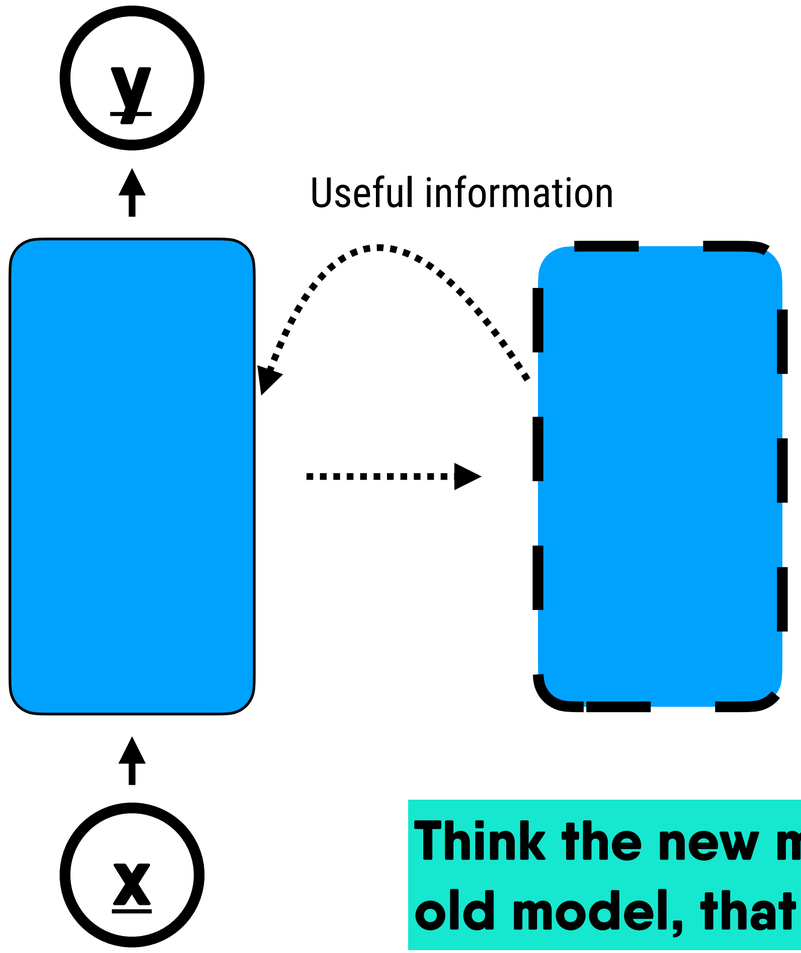
Mirror Integration Type 1



Why not learn from the ground truth data but approximate the old model?
Why this works?

1. soft label.
2. act as an oracle for active learning, in theory you can arbitrarily enlarge the dataset
3. The structure of DNN can be utilized for good approximation.

Mirror Integration Type 2

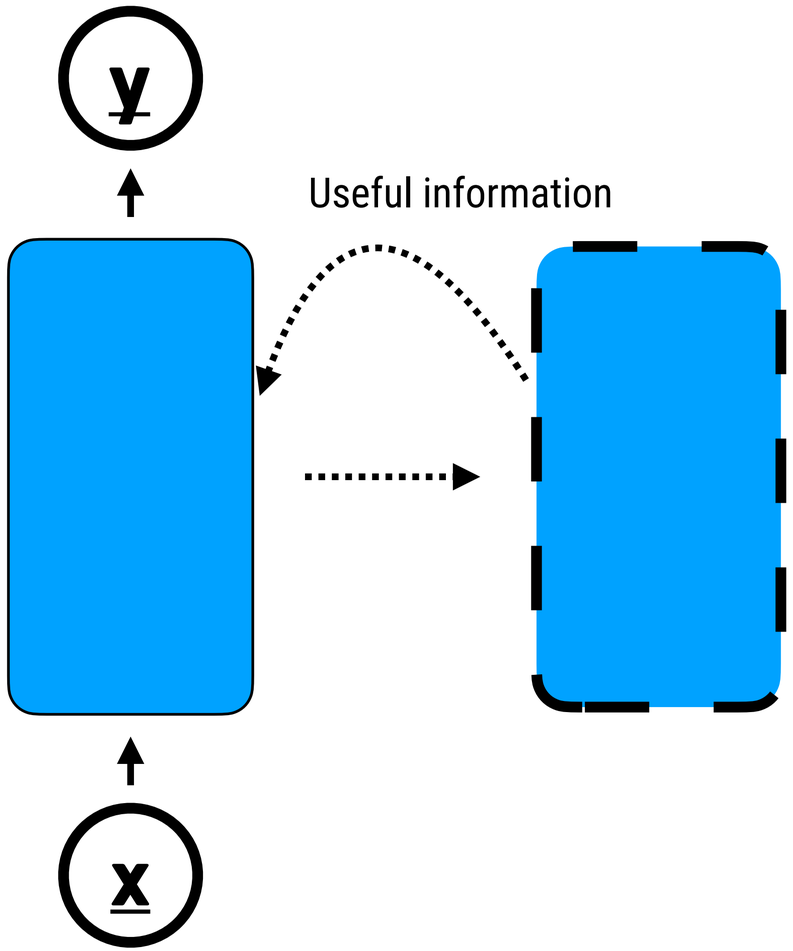


Use the new model to augment the old model.

1. choose a good form of the new model
2. Dig useful information/insights from the new model
3. Use these information to better train the old model

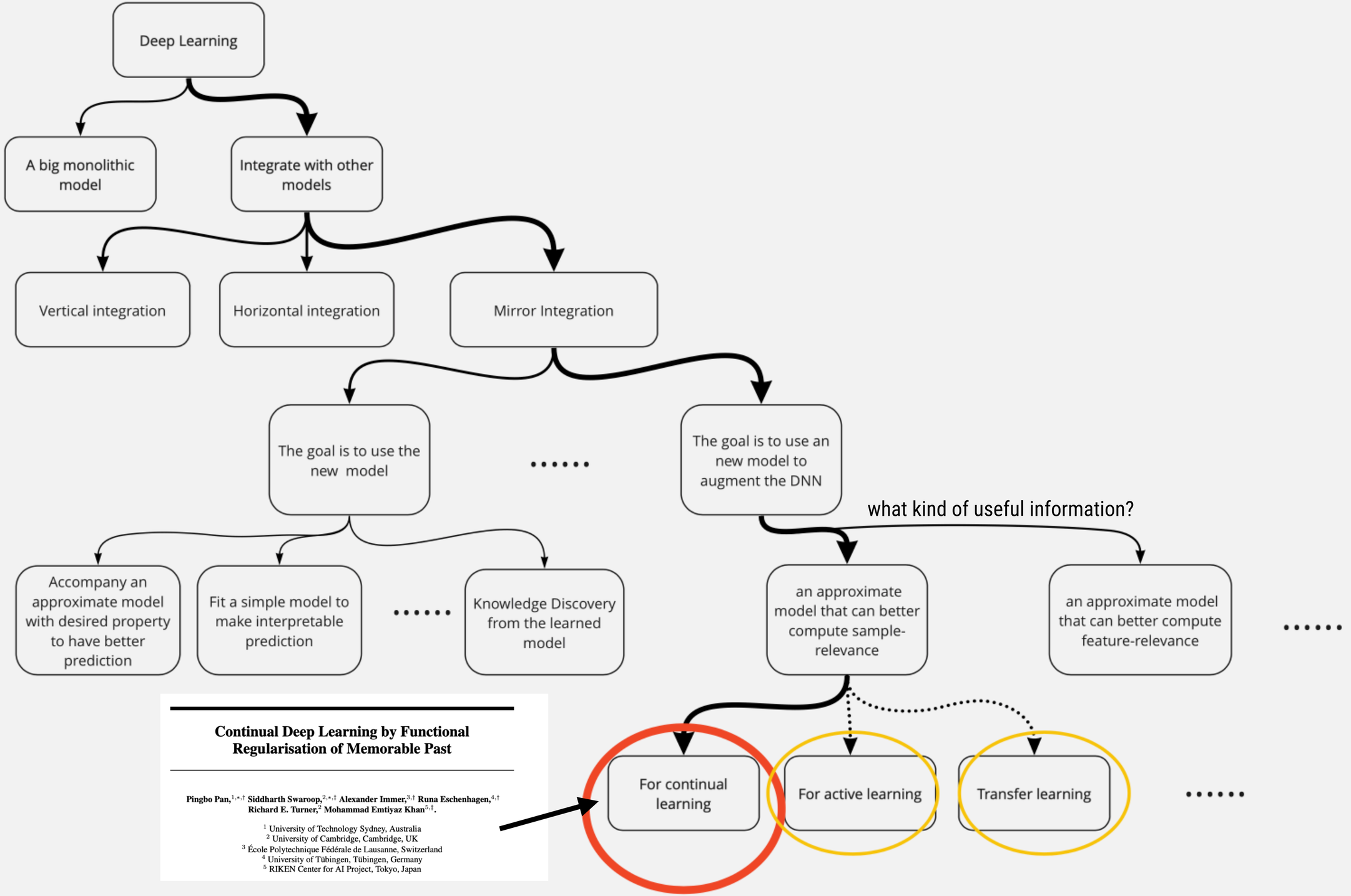
Think the new model as a kind of easier version of the old model, that can easily dig up useful informations

Mirror Integration Type 2



What makes type 2 mirror integration works is that the new model

1. approximate the old DNN very well, very faithfully
2. It can dig up useful information, easier than the DNN


Deep Learning
A big monolithic model
Integrate with other models
Vertical integration
Horizontal integration
Mirror Integration
The goal is to use the new model
The goal is to use an new model to augment the DNN
what kind of useful information?
Accompany an approximate model with desired property to have better prediction
Fit a simple model to make interpretable prediction
Knowledge Discovery from the learned model
an approximate model that can better compute sample-relevance
an approximate model that can better compute feature-relevance
For continual learning
For active learning
Transfer learning
Continual Deep Learning by Functional Regularisation of Memorable Past
Pingbo Pan,[1,*,†] Siddharth Swaroop,[2,*,‡] Alexander Immer,[3,†] Runa Eschenhagen,[4,†]
Richard E. Turner,[2] Mohammad Emtiyaz Khan[5,‡].
[1] University of Technology Sydney, Australia
[2] University of Cambridge, Cambridge, UK
[3] École Polytechnique Fédérale de Lausanne, Switzerland
[4] University of Tübingen, Tübingen, Germany
[5] RIKEN Center for AI Project, Tokyo, Japan

PART 2

Functional Regularization for continual learning

Continual learning: the forgetting problem

Suppose we have two datasets D1 and D2. Consider D1 and D2 has the same output, just the data X has some distribution shift.

We first train on D1 and then on D2.

The forgetting problem referring to the situation that the model trained on D2, cannot predict as well as it used to be on D1.

But in fact, this problem is not only in continual learning,

1. Even in the standard single dataset learning, this forgetting problem also happens.
2. In online learning setting, if the DNN is updated by the new mini-batches of data online. Then if there is distribution shift, the forgetting will also happen.

Option 1: Weight regularization

Weight regularization is the more usual solution, once we trained on D1, obtained the weight parameters, and start to train on D2, we put a constraint to the weight parameters of DNN that it must be close to the old ones.

$$L = \sum_{x,y \in D_2} L_{classification}(f(x;\theta), y) + \lambda ||\theta - \theta_{D_1}||$$

Simple idea, good performance.

But the main problem, does the distance measure in weight space really make sense?

A DNN is typically overparameterized with massive parameters……
distance in such high dimension……

Option 2: Functional regularization

Functional regularization, or **Memorization**

Functional regularization can be viewed as a prior that enforces the new model to **memorize**, to produce similar output on samples from previous data set D1.

$$L = \sum_{x,y \in D_2} L_{classification}(f(x;\theta), y) + \lambda \sum_{x,y \in D_1} ||[f(x;\theta) - f(x;\theta_{D_1})||$$

Functional regularization, cont:

$$L = \sum_{x,y \in D_2} L_{classification}(f(x;\theta), y) + \lambda \sum_{x,y \in D_1} ||f(x;\theta) - f(x;\theta_{D_1})||$$

Q1: Now this looks we are storing the entire old dataset, is it feasible to do this? No. That is why we need to only pick a small number of data from D1 to memorize

Q2: Why are we storing the previous output of the entire old dataset, but not the ground truth label?

1. The previous output is a soft label, which should give more information.
2. The data points does not have to from D1, it can be arbitrary data, the so-called pseudo-data.

$$L = \sum_{x,y \in D_2} L_{classification}(f(x;\theta), y) + \lambda \sum_{x,y \in D'} ||f(x;\theta) - f(x;\theta_{D'})||$$

But, functional regularization makes sense but do not outperform weight regularization.

Possible Reason 1:
The weight regularization is simpler and also older, already have many sophisticated works

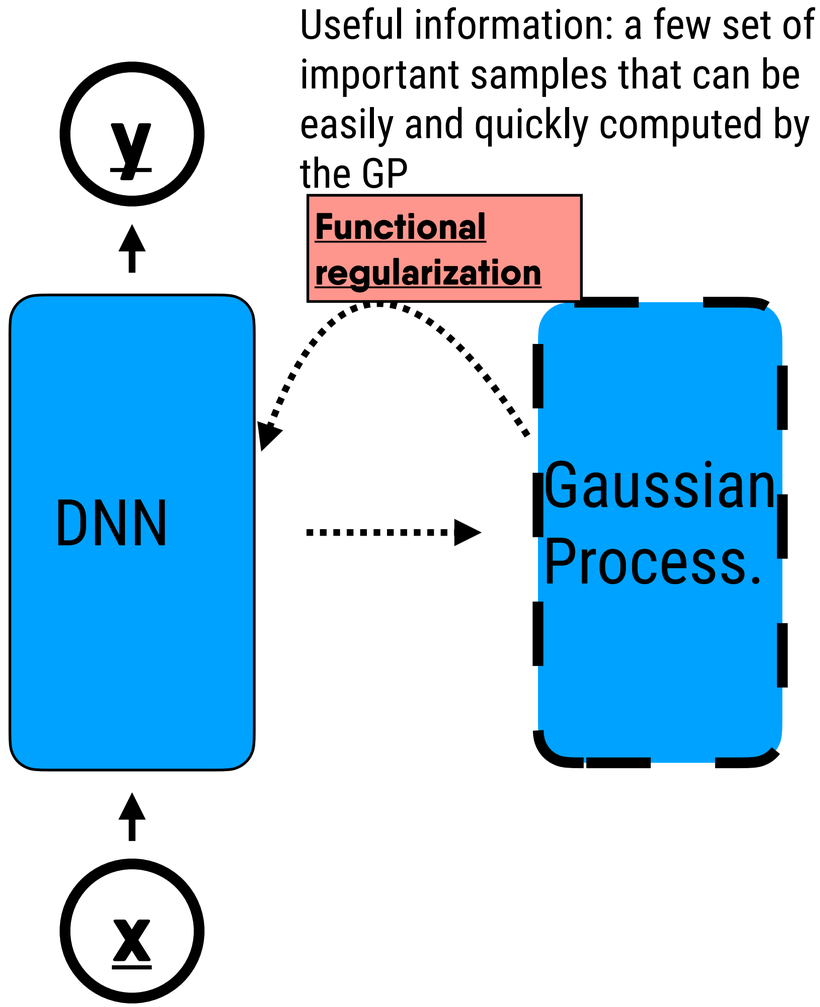
Possible Reason 2.
The performance of functional regularization actually depends on how you select the previous data points D'. Ideally these these points should summarize the old dataset well, and points that near decision boundary.

One important reason, you cannot easily compute and collect the good data samples to memorize. In DNN, this is hard.

Now comes the central part of this paper !
Use a mirror model that compute the relevant samples easily, better and faster!

The proposed approach

Continual Deep Learning by Functional Regularisation of Memorable Past



1. Transform the DNN into a Gaussian Process based on a previous work.

Approximate Inference Turns Deep Networks into Gaussian Processes

2. Identify a few important data samples to memorize, which can be easily computed from the GP

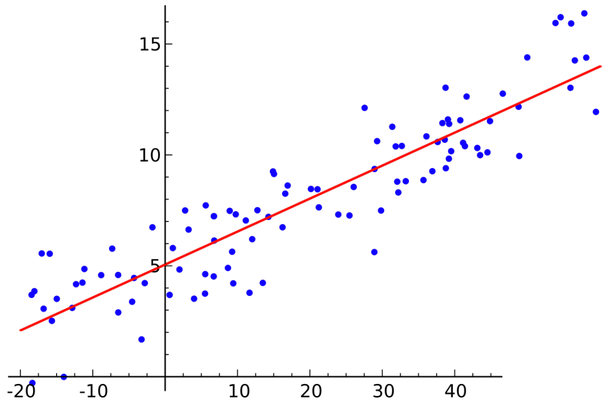
3. Using these “memorize” data samples to add a new functional regularization term. And now train the DNN in the new dataset.

Why to transform to GP
and how to identify the important data samples using GP?

The thing is that Gaussian Process can easily get you the important samples, and transform DNN to GP has some elegant math-heavy solutions.

I will not cover the math, because it is so hard to tell the story here. But I will give an easy version.

Converting DNN to GP,
now only consider converting a linear regression model to K nearest neighbour



A linear model predicts based on the learned weight.

$$y = f(x) = w * x + b$$

But a k nearest neighbour (consider k to be the size of training dataset), predicts based on the training data.

$$y = f(x) = \sum_{(x',y' \in D)} similarity(x, x')y' + \frac{1}{|D|} \sum_{(x',y' \in D)} y'$$

Now, an intuitive way of finding important samples is to check the similarity values and pick the sample that is similar to many samples (and thus important for predicting this samples.

Gaussian Process is a quite fancy version of this thing.

1. Converting this linear model to gaussian process is textbook example.
2. Converting DNN to GP employs a similar but more sophisticated trick
3. The actual way to compute the data importance is not as simple as I said…

Why it works?

There are some other previous works that try to do this, identifying good data samples and put as functional regularization. Why these previous work do not outperform weight-regularization and this one approach does?

Remember that in order to makes type 2 mirror integration work, we need the new model to:

1. approximate the old DNN very well, very faithfully

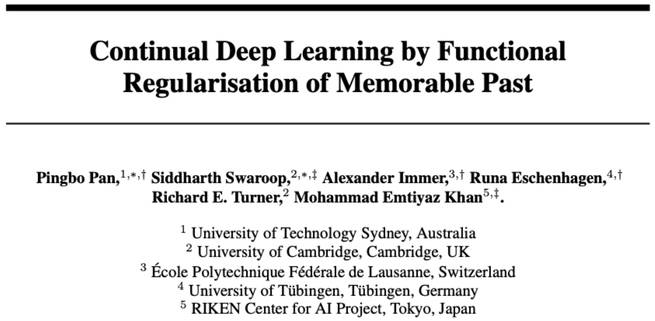
   There are many possible reasons on approximating well
   1. soft label.
   2. act as an oracle for active learning for arbitrarily large dataset
   3. The structure of DNN can be utilized for good approximation.

      transform a DNN into GP is faithful, following the 3rd reason.

2. It can dig up useful information, easier than the DNN

   GP computes the data relevance better, which truly indicates the relevance for the prediction. And the computation is very fast and scalable.

A four line summary of mirror integration



Continual Deep Learning by Functional Regularisation of Memorable Past

Pingbo Pan,[1,*,†] Siddharth Swaroop,[2,*,‡] Alexander Immer,[3,†] Runa Eschenhagen,[4,†]
Richard E. Turner,[2] Mohammad Emtiyaz Khan[5,‡].

[1] University of Technology Sydney, Australia
[2] University of Cambridge, Cambridge, UK
[3] École Polytechnique Fédérale de Lausanne, Switzerland
[4] University of Tübingen, Tübingen, Germany
[5] RIKEN Center for AI Project, Tokyo, Japan

problem $\mathbb{P}$ → Forgetting in continual learning

approach $\mathbb{A}$ → Functional regularization

information $\mathbb{I}$ → Important data samples for the old dataset

mirror model $\mathbb{M}$ → A Gaussian Process

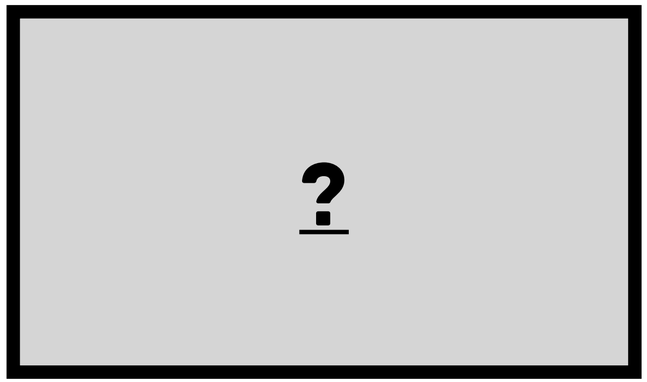
1. In this paper, the author wants to solve **problem** $\mathbb{P}$ with a DNN.
2. so they choose **approach** $\mathbb{A}$ to augment the DNN which require **information** $\mathbb{I}$
3. The usual ways do not work well, so they use mirror integration, approximate a DNN with **mirror model** $\mathbb{M}$
4. The **mirror model** $\mathbb{M}$ can acquire **information** $\mathbb{I}$ better and thus do **approach** $\mathbb{A}$ better,

   And thus solve **problem** $\mathbb{P}$

PART 3

Further implications and potential applications

Applying mirror integration



problem $\mathbb{P}$ → ???

approach $\mathbb{A}$ → ???

information $\mathbb{I}$ → ???

mirror model $\mathbb{M}$ → ???

① In this paper, the author wants to solve **problem** $\mathbb{P}$ with a DNN.

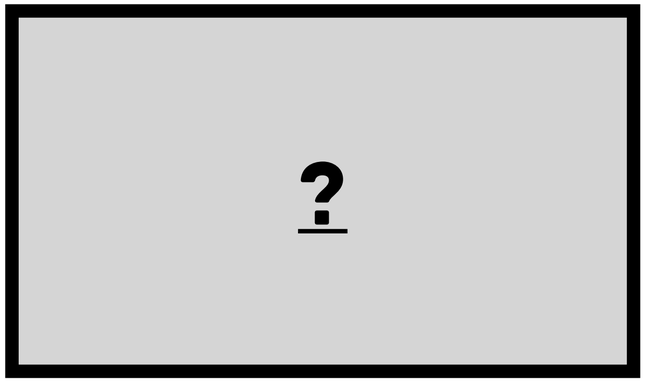
② so they choose **approach** $\mathbb{A}$ to augment the DNN which require **information** $\mathbb{I}$

③ The usual ways do not work well, so they use mirror integration, approximate a DNN with **mirror model** $\mathbb{M}$

④ The **mirror model** $\mathbb{M}$ can acquire **information** $\mathbb{I}$ better and thus do **approach** $\mathbb{A}$ better,

And thus solve **problem** $\mathbb{P}$

Active Learning?



problem P → ??? in active learning

approach A → ???

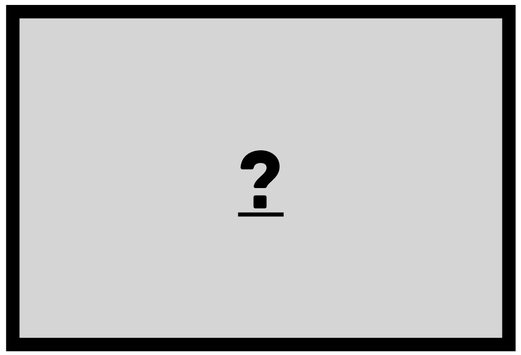
information I → ??? Maybe also certain important data samples

mirror model M → ???

From I see there are many important problems of active learning in DNN.

1. The new queried samples should really helps the learning of DNN. Otherwise it is meaningless.
2. But in the same time, a new queried sample does not guaranteed to have a substantial change of the DNN, because it is overparameterized.

Active Learning?



problem → ??? in active learning

approach → ???

information → ??? Maybe also certain important data

mirror model → ???

Perhaps functional regularization is a good thing after all

A general topic in active learning is that the new queried samples must be

1. near the decision boundary (so querying them must can help)
2. Within the data distribution.

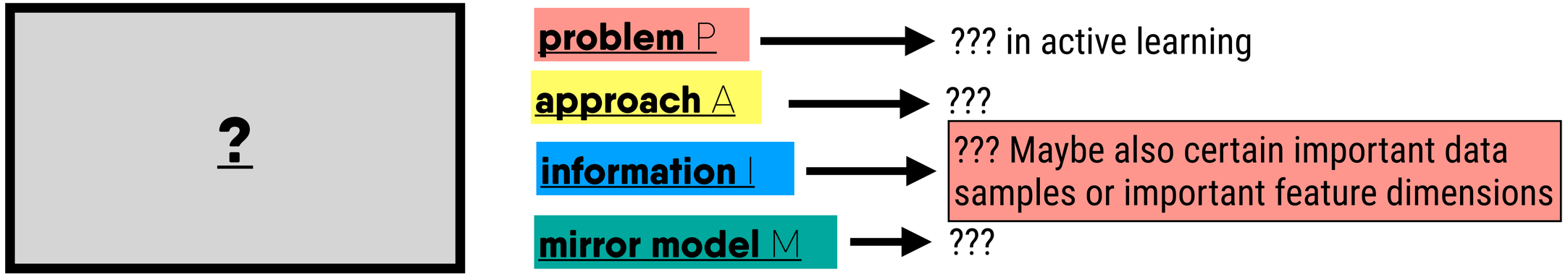
I believe this information is quite hard to compute, as DNN's uncertainty measure is so unreliable.

And how to compute of "in-distribution" and "out-of-distribution" with the DNN alone?

But more particularly, for active learning in DNN

1. overparameterization of DNN makes it hard to see the queried samples actually result in the change of DNN.
2. how to make sure that the prediction of other samples remain similar?

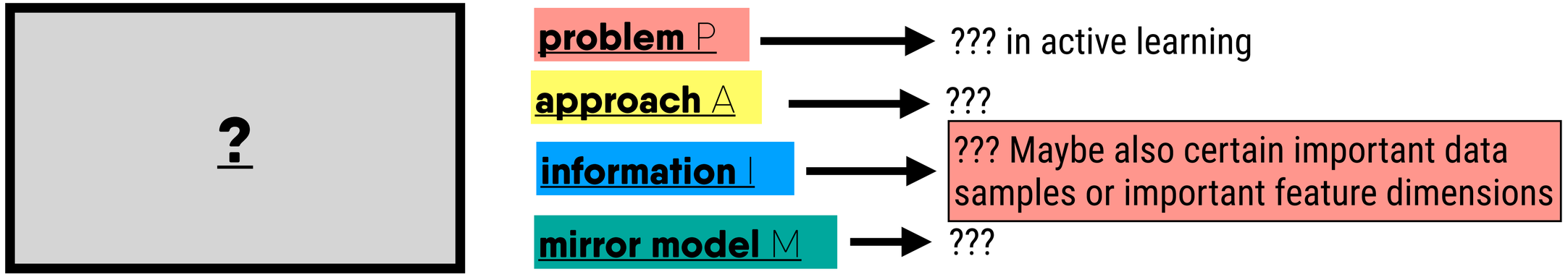
Transfer learning, or learning from multiple dataset



The problem I think is quite interesting and not addressed in Deep transfer learning is simply

1. what is actually transferred?
2. If we know what is actually transferred, then we can do better in improving the right thing to transfer

Transfer learning, or learning from multiple dataset



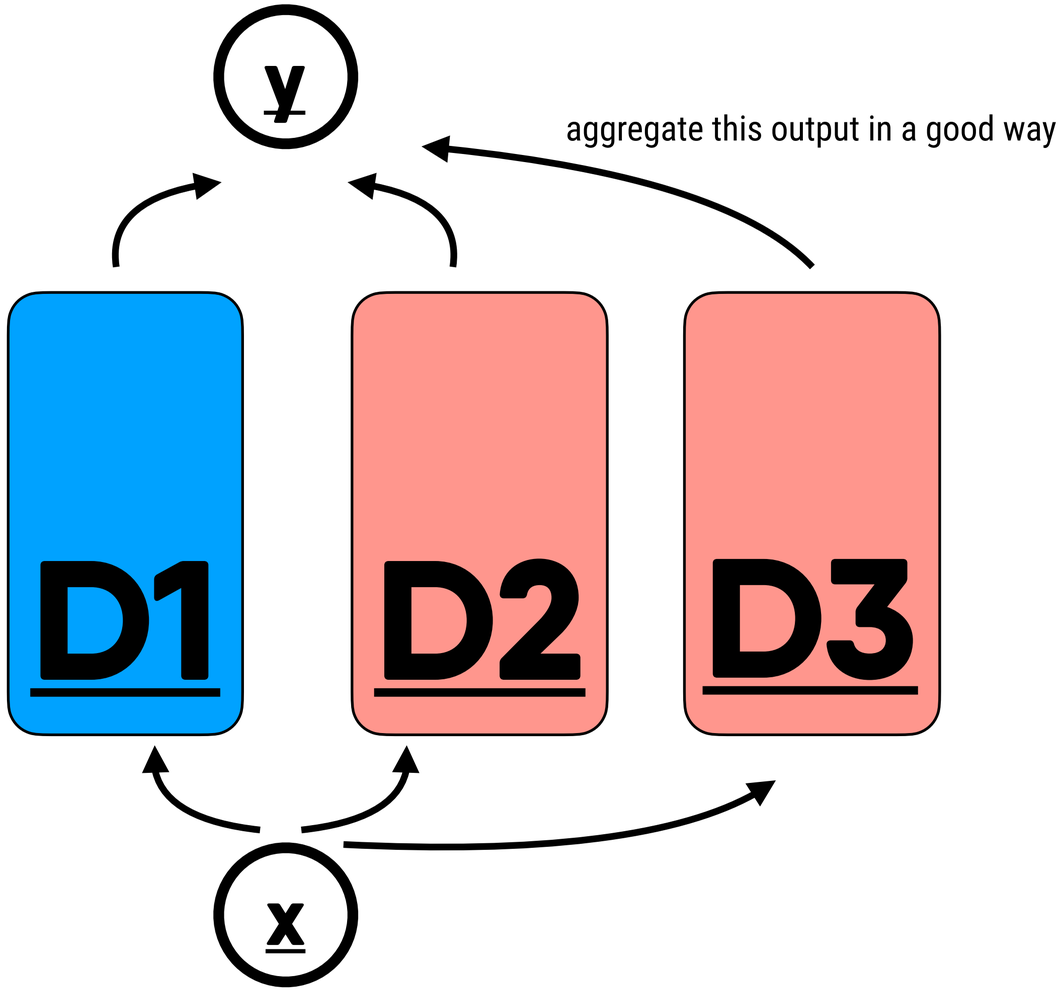
One particular approach is to also use weight-regularization, just like in continual learning (well, continual learning is a kind of transfer learning)

Can we use the data samples as useful information and do functional regularization?

We may perhaps identify the useful data samples that can help learning from the old dataset to the new dataset, then use these data samples.

Transfer learning, or learning from multiple dataset

Another option, feature adaptation of many backbone model trained in different datasets.



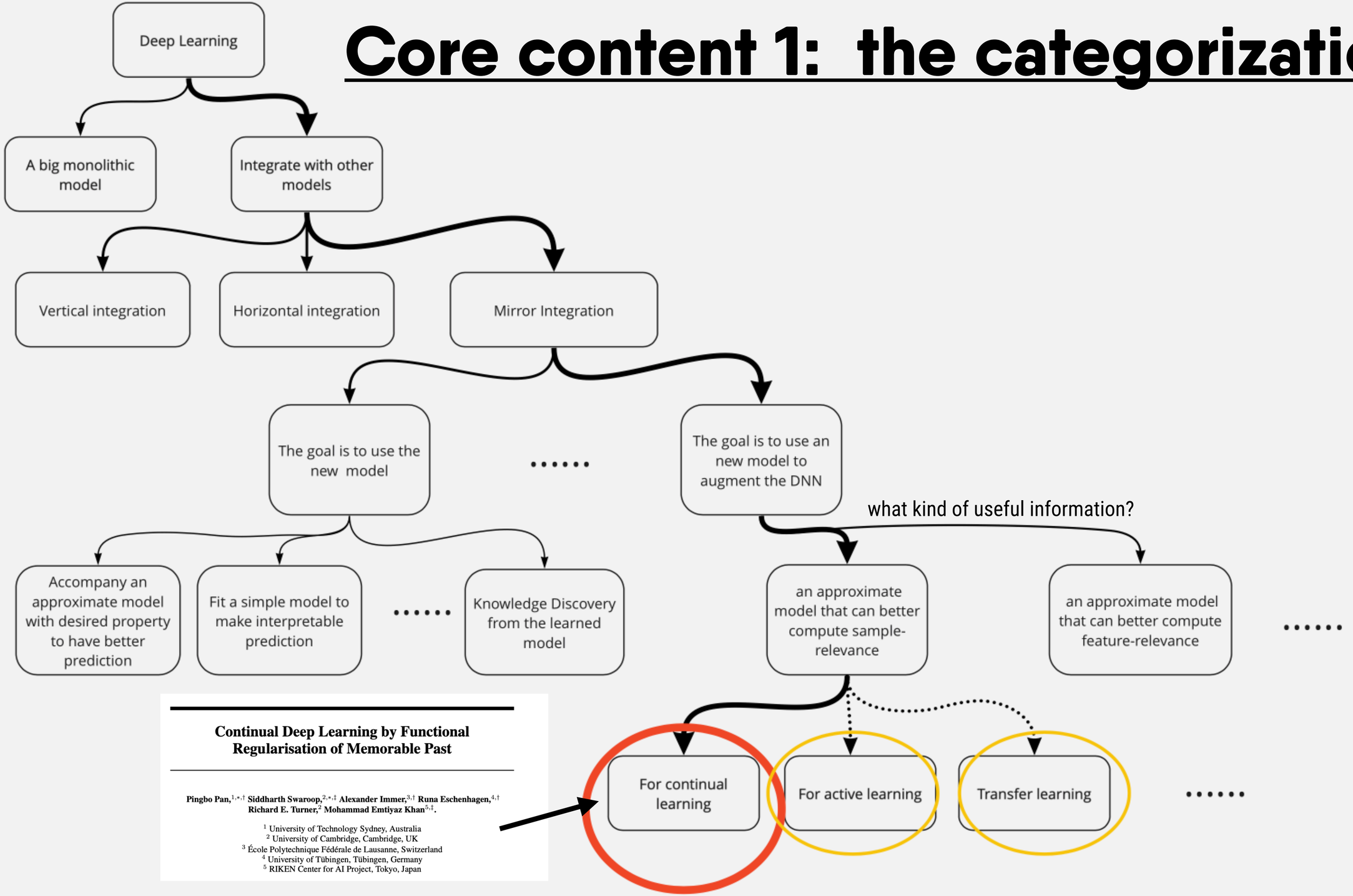
What backbone to attend and What kind of feature to emphasize is a hard problem.

Perhaps it is difficult ti compute such information from the DNN, but maybe, we can think of a mirror model that can better compute “feature-relevance”,

Instead of Gaussian Process that computes data relevance?

Conclusion

Core content 1: the categorization

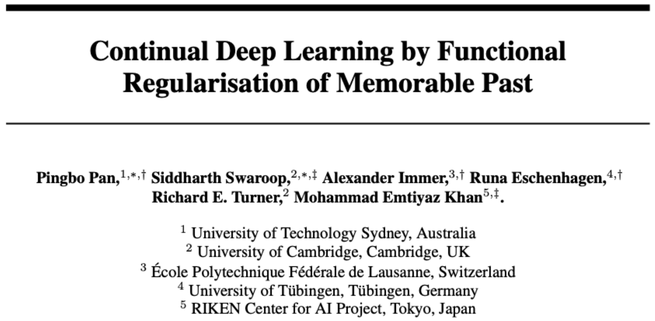


Continual Deep Learning by Functional Regularisation of Memorable Past

Pingbo Pan,[1,*,†] Siddharth Swaroop,[2,*,‡] Alexander Immer,[3,†] Runa Eschenhagen,[4,†]
Richard E. Turner,[2] Mohammad Emtiyaz Khan[5,‡].

[1] University of Technology Sydney, Australia
[2] University of Cambridge, Cambridge, UK
[3] École Polytechnique Fédérale de Lausanne, Switzerland
[4] University of Tübingen, Tübingen, Germany
[5] RIKEN Center for AI Project, Tokyo, Japan

Core content 2: A four line summary of mirror integration



Continual Deep Learning by Functional Regularisation of Memorable Past

Pingbo Pan,[1,*,†] Siddharth Swaroop,[2,*,‡] Alexander Immer,[3,†] Runa Eschenhagen,[4,†]
Richard E. Turner,[2] Mohammad Emtiyaz Khan[5,‡].

[1] University of Technology Sydney, Australia
[2] University of Cambridge, Cambridge, UK
[3] École Polytechnique Fédérale de Lausanne, Switzerland
[4] University of Tübingen, Tübingen, Germany
[5] RIKEN Center for AI Project, Tokyo, Japan

problem $\mathbb{P}$ → Forgetting in continual learning

approach $\mathbb{A}$ → Functional regularization

information $\mathbb{I}$ → Important data samples

mirror model $\mathbb{M}$ → A Gaussian Process

(1) In this paper, the author wants to solve **problem** $\mathbb{P}$ with a DNN.

(2) so they choose **approach** $\mathbb{A}$ to augment the DNN which require **information** $\mathbb{I}$

(3) The usual ways do not work well, so they use mirror integration, approximate a DNN with **mirror model** $\mathbb{M}$

(4) The **mirror model** $\mathbb{M}$ can acquire **information** $\mathbb{I}$ better and thus do **approach** $\mathbb{A}$ better,

And thus solve **problem** $\mathbb{P}$

Message 1:
Consider many ways of new hybrid model (integration) rather than a single big DNN,

And I generally do not think devising some domain-specific architectures, like specifically designed for bio-medical datasets etc, are a currently wise thing to do. It becomes quite risky, messy… and difficult.

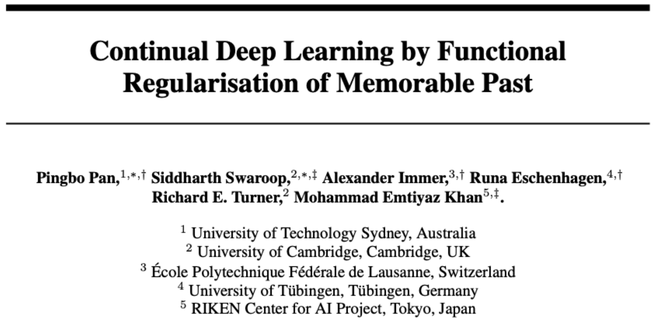
Message 2:
use the mirror integration if something important is not easy with DNN or even, cannot be done by DNN

Whether it is type 1 (using the new model to predict)
or type 2 (using the new model to augment the DNN)

If you have some information that cannot be easily obtained with the DNN, why not consider a mirror model? It sounds a little weird at first, but maybe it can work.

Thank you.

A four line summary of mirror integration



Continual Deep Learning by Functional Regularisation of Memorable Past

Pingbo Pan,[1,*,†] Siddharth Swaroop,[2,*,‡] Alexander Immer,[3,†] Runa Eschenhagen,[4,†]
Richard E. Turner,[2] Mohammad Emtiyaz Khan[5,‡].

[1] University of Technology Sydney, Australia
[2] University of Cambridge, Cambridge, UK
[3] École Polytechnique Fédérale de Lausanne, Switzerland
[4] University of Tübingen, Tübingen, Germany
[5] RIKEN Center for AI Project, Tokyo, Japan

problem $\mathbb{P}$ → Forgetting in continual learning

approach $\mathbb{A}$ → Functional regularization

information $\mathbb{I}$ → Important data samples

mirror model $\mathbb{M}$ → A Gaussian Process

1. In this paper, the author wants to solve **problem** $\mathbb{P}$ with a DNN.
2. so they choose **approach** $\mathbb{A}$ to augment the DNN which require **information** $\mathbb{I}$
3. The usual ways do not work well, so they use mirror integration, approximate a DNN with **mirror model** $\mathbb{M}$
4. The **mirror model** $\mathbb{M}$ can acquire **information** $\mathbb{I}$ better and thus do **approach** $\mathbb{A}$ better,

   And thus solve **problem** $\mathbb{P}$