KAIST CHEMISTRY

Constructive Transfer Learning

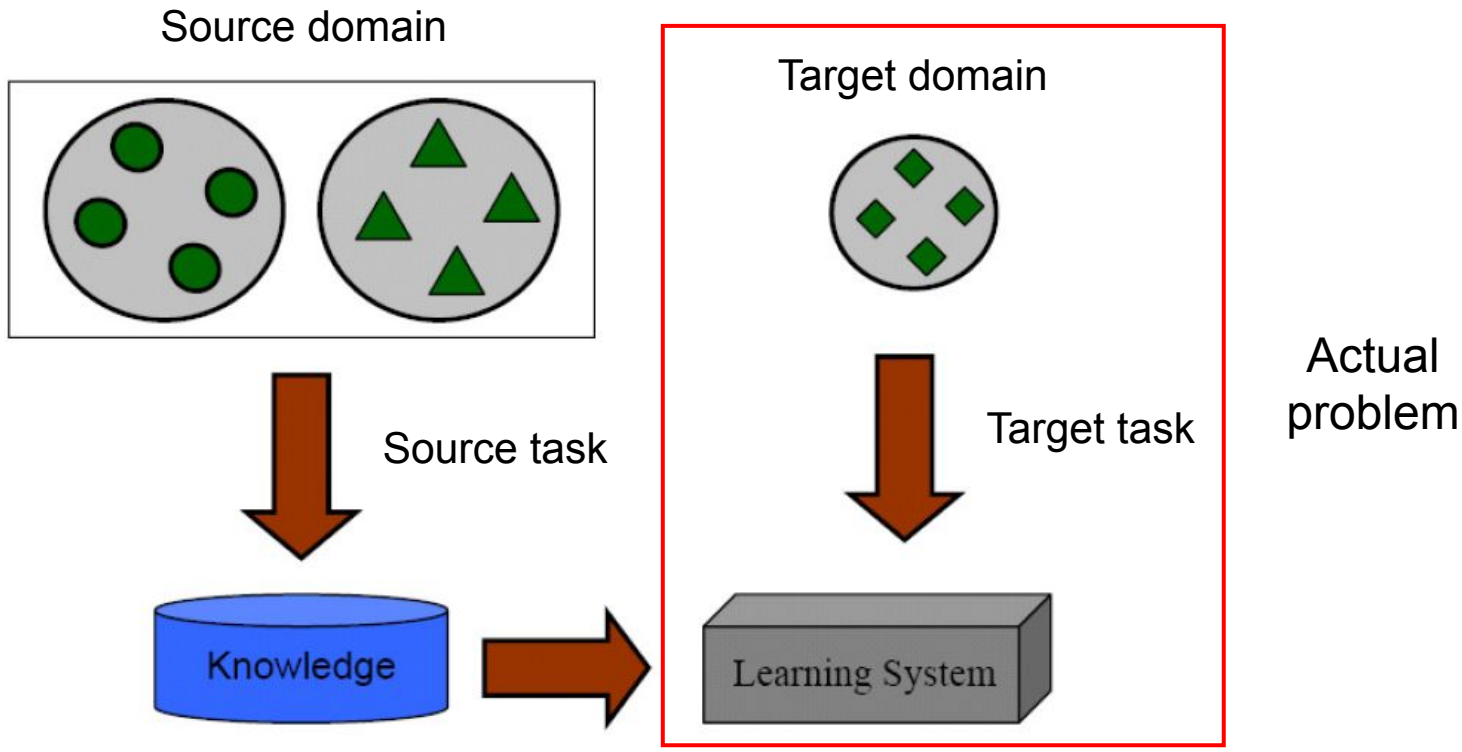
Youngju Yoo
ACE-Team
KAIST Chemistry

KAIST

카이스트 화학과 2021.08.23

Definition of Transfer Learning

- Transfer some knowledge to better solve the given **target task**
- Mainly used to prevent overfitting by introducing **large** amount of data for **source task**

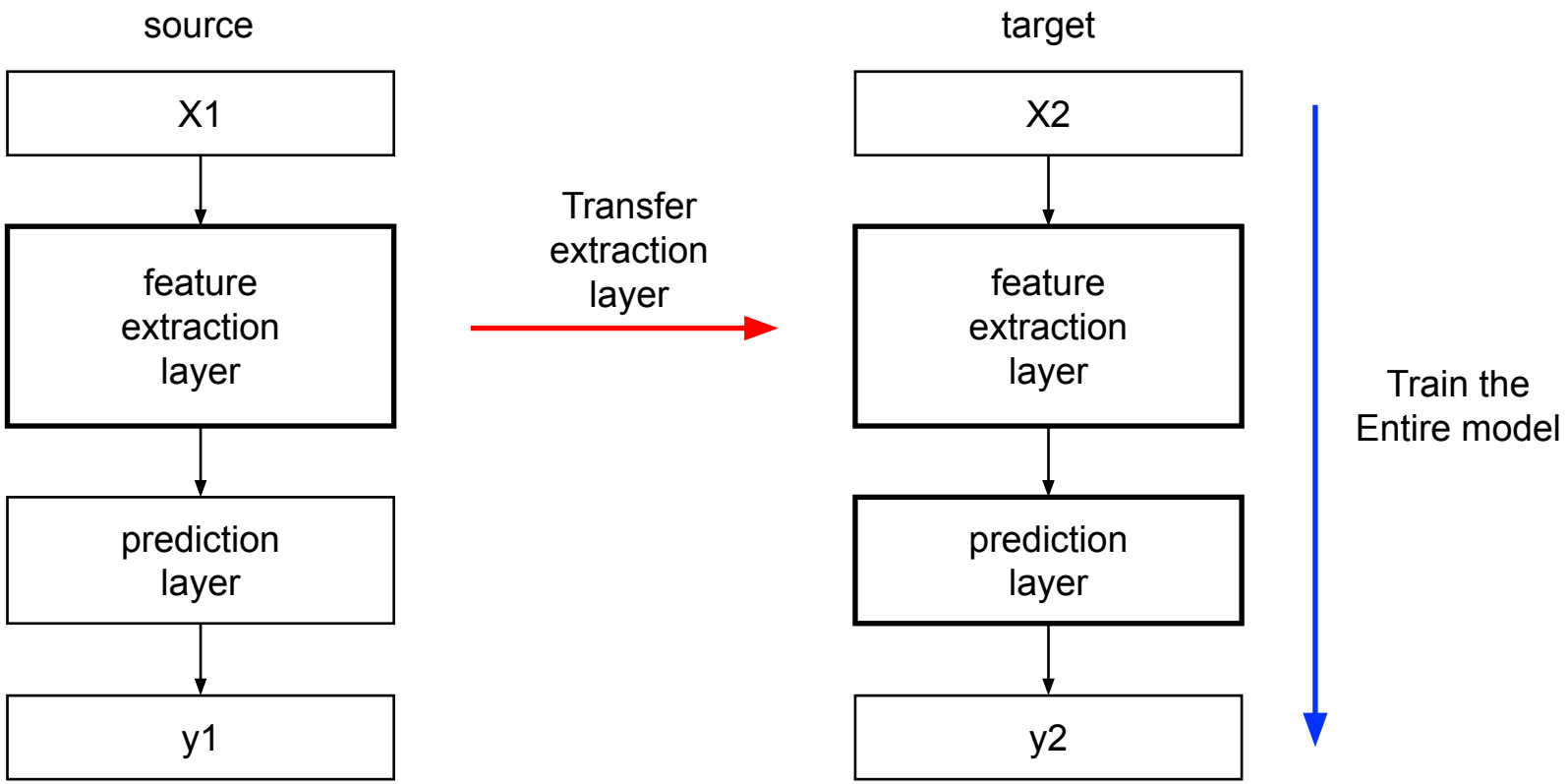


Categories of TL (knowledge)

- With respect to type of knowledge transfer, there are mainly four categories
- **Model** transfer: Reuse **model parameters** trained from source (normally used as default)
- **Feature (mapping)** transfer: Reuse **features** extracted by models that are trained from source
- **Instance** transfer: Transfer or reuse **data information**, either reweight target or remove negative data in target (Will not be discussed)
- **Relation** transfer: Very specific type of transfer, it transfers relation information within source data (Will not be discussed)

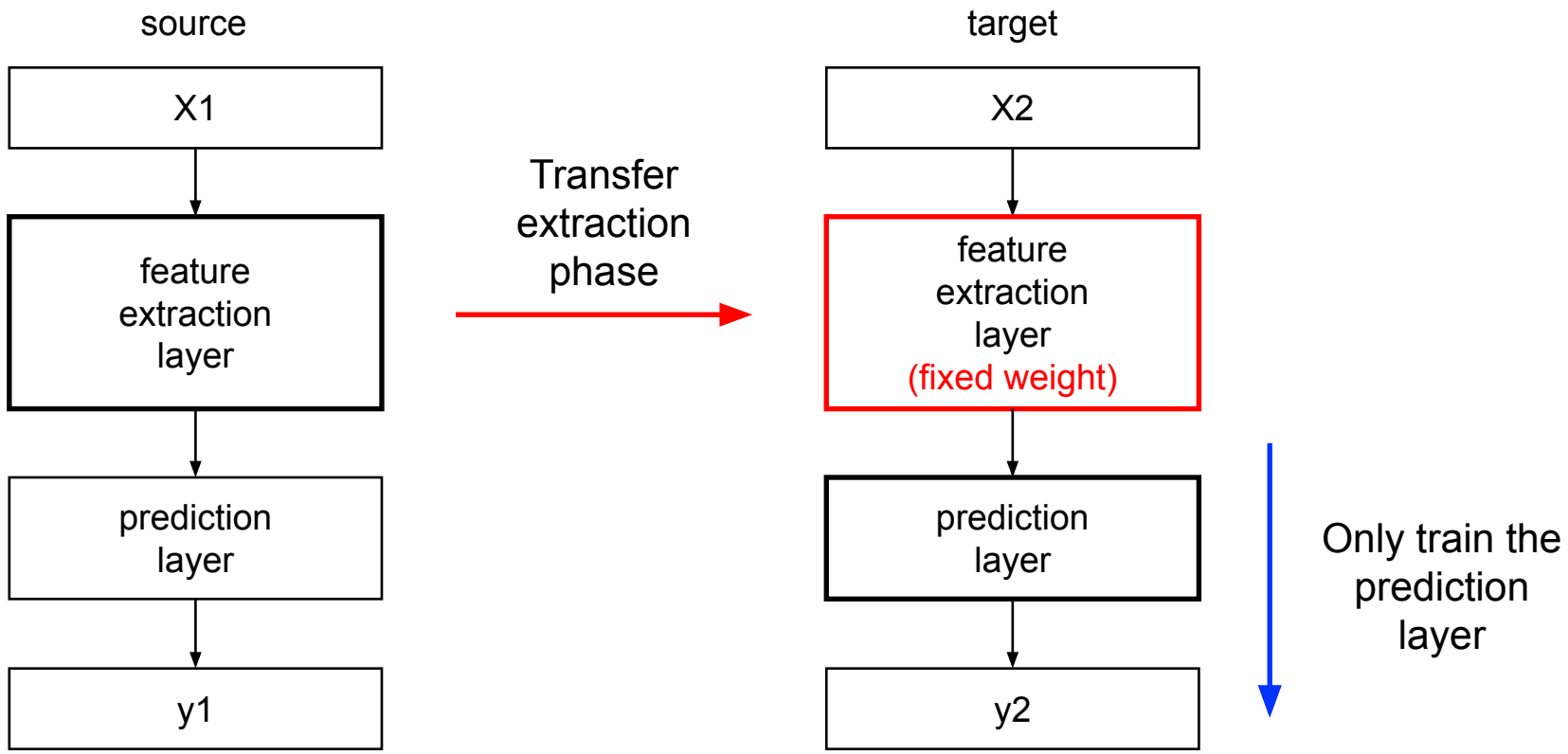
Model transfer

- Widely used transfer learning, also called pre-train + fine-tuning



Feature transfer

- Widely used transfer learning for measuring the **quality of representations**



- Simply can be considered as “model transfer with fixed weight”

Examples of TL in chemistry - NMR

Transfer Learning from Simulation to Experimental Data: NMR Chemical Shift Predictions

Herim Han and Sunghwan Choi*

database		training subset	test subset	range	mean	constituent elements
expt.	^{1}H	9657	2411	(1, 46)	16.97	H, C, N, O, F, P, S, Cl
	^{13}C	21 116	5304	(1, 40)	14.16	H, C, N, O, F, P, S, Cl
simulation		85 003	45 772	(1, 9)	8.80	H, C, N, O, F

Target data: NMRShiftDB2

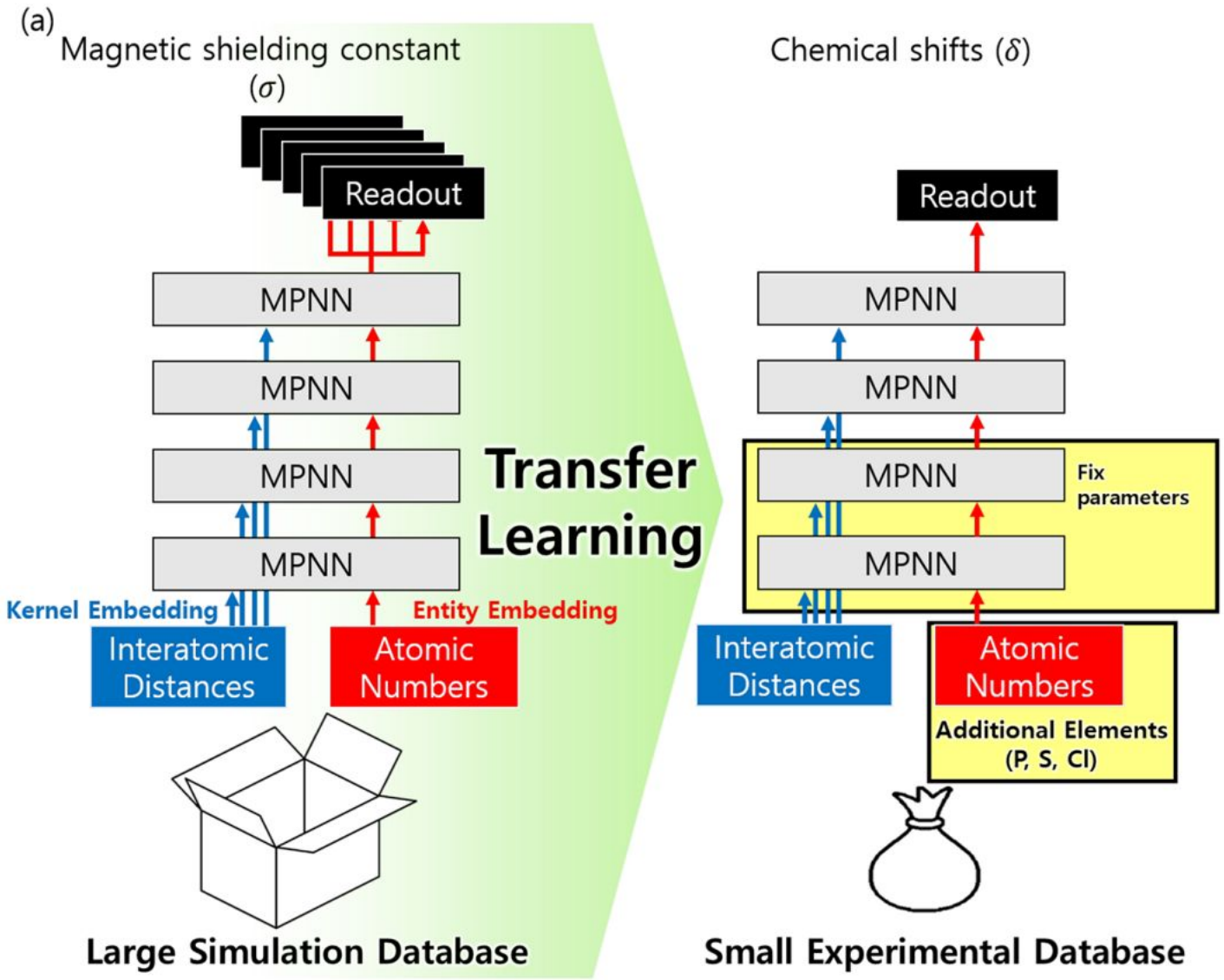
Source data: Obtained by QM calculations from QM9

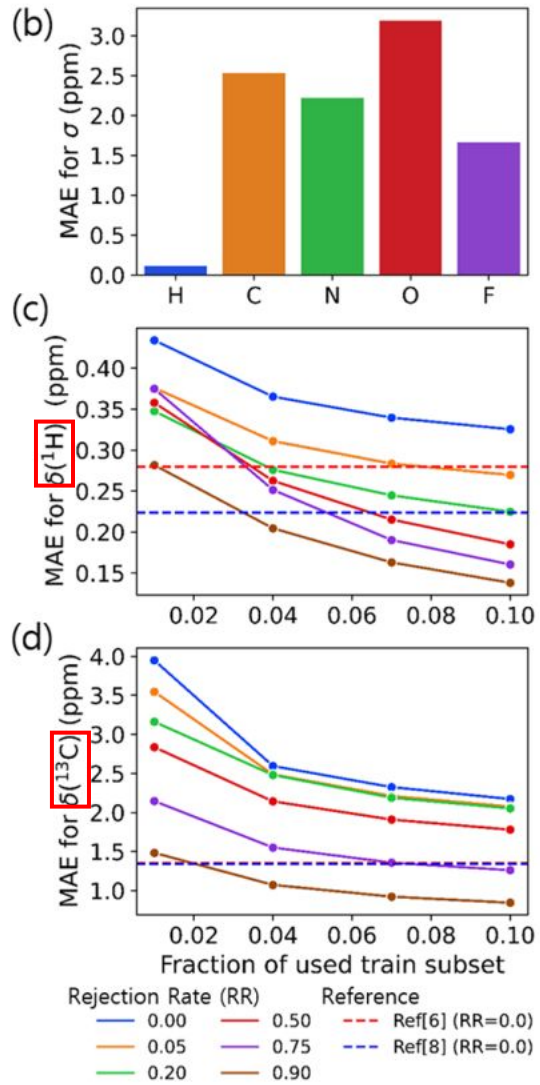
experimental database.[30] This method does not require any explicit mappings between prior knowledge and the target problem. Network-based deep transfer learning typically consists of two steps: a pretraining and fine-tuning. In the pretraining process, the model parameters are fitted to represent the knowledge for not the target but the related problem. The fine-tuning procedure reoptimizes the pretrained

Model transfer

Herim Han, and Sunghwan Choi. "Transfer Learning from Simulation to Experimental Data: NMR Chemical Shift Predictions." The Journal of Physical Chemistry Letters 12.14 (2021): 3662-3668

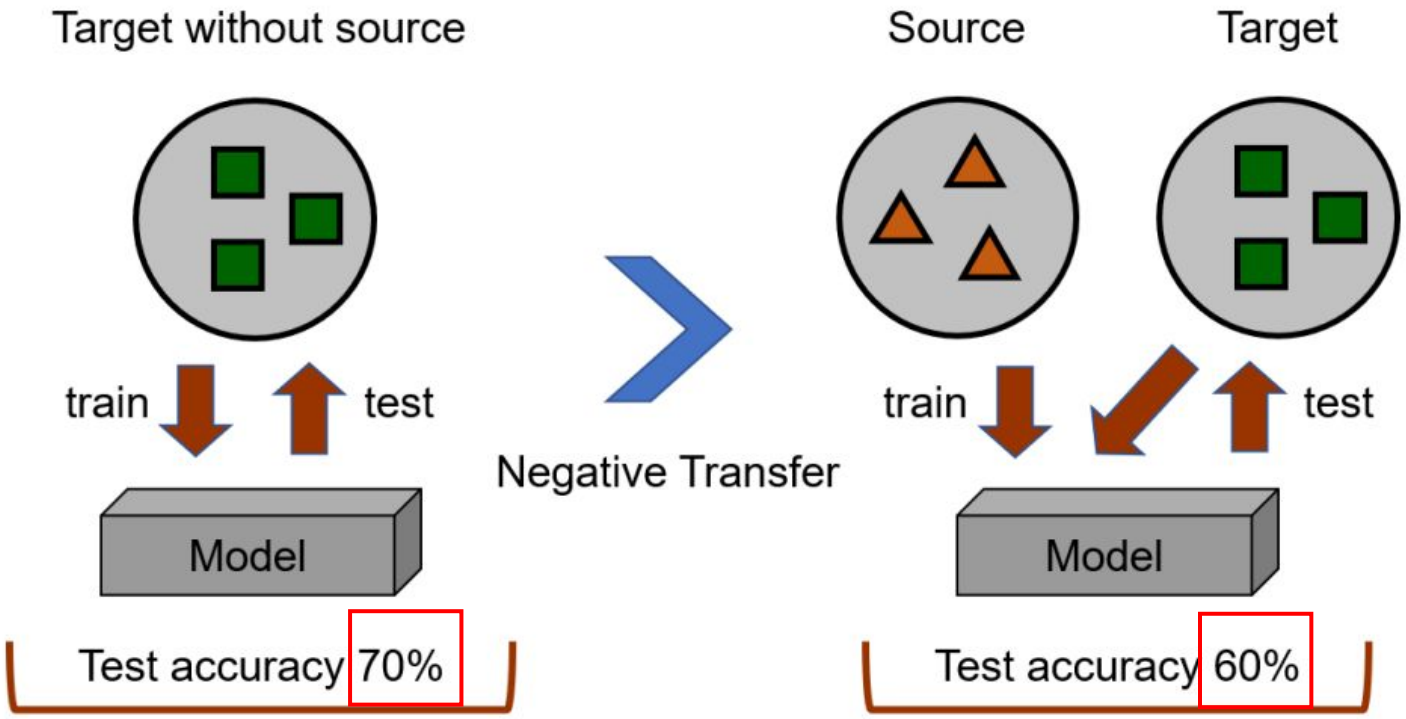
Examples of TL in chemistry - NMR





Herim Han, and Sunghwan Choi. "Transfer Learning from Simulation to Experimental Data: NMR Chemical Shift Predictions." The Journal of Physical Chemistry Letters 12.14 (2021): 3662-3668

Negative transfer

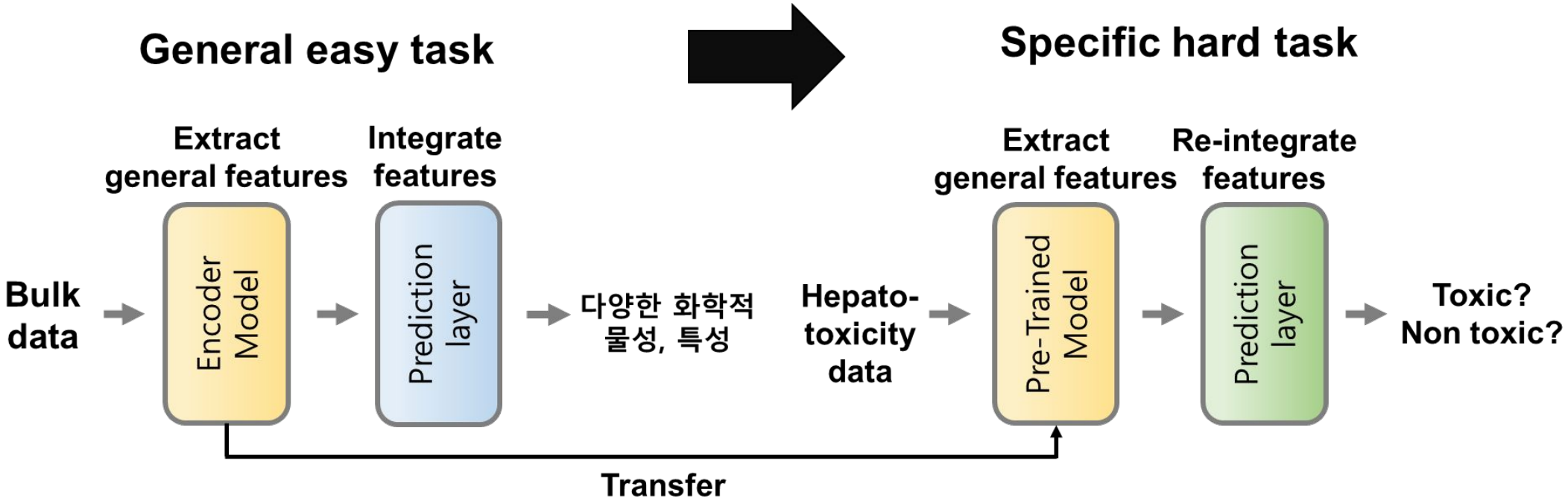


NT in their TL survey: *"When the source domain and target domain are not related to each other, brute-force transfer may be unsuccessful. In the worst case, it may even hurt the performance of learning in the target domain, a situation which is often referred to as negative transfer."*

Wen Zhang, http://arxiv.org/abs/2009.00909

Negative transfer example

- Hepatotoxicity



1. Train with **logP**, then transfer to hepatotoxicity
2. Train with **general toxicity data (TOX21)**, then transfer to hepatotoxicity
3. Train with **drug-likeness**, then transfer to hepatotoxicity

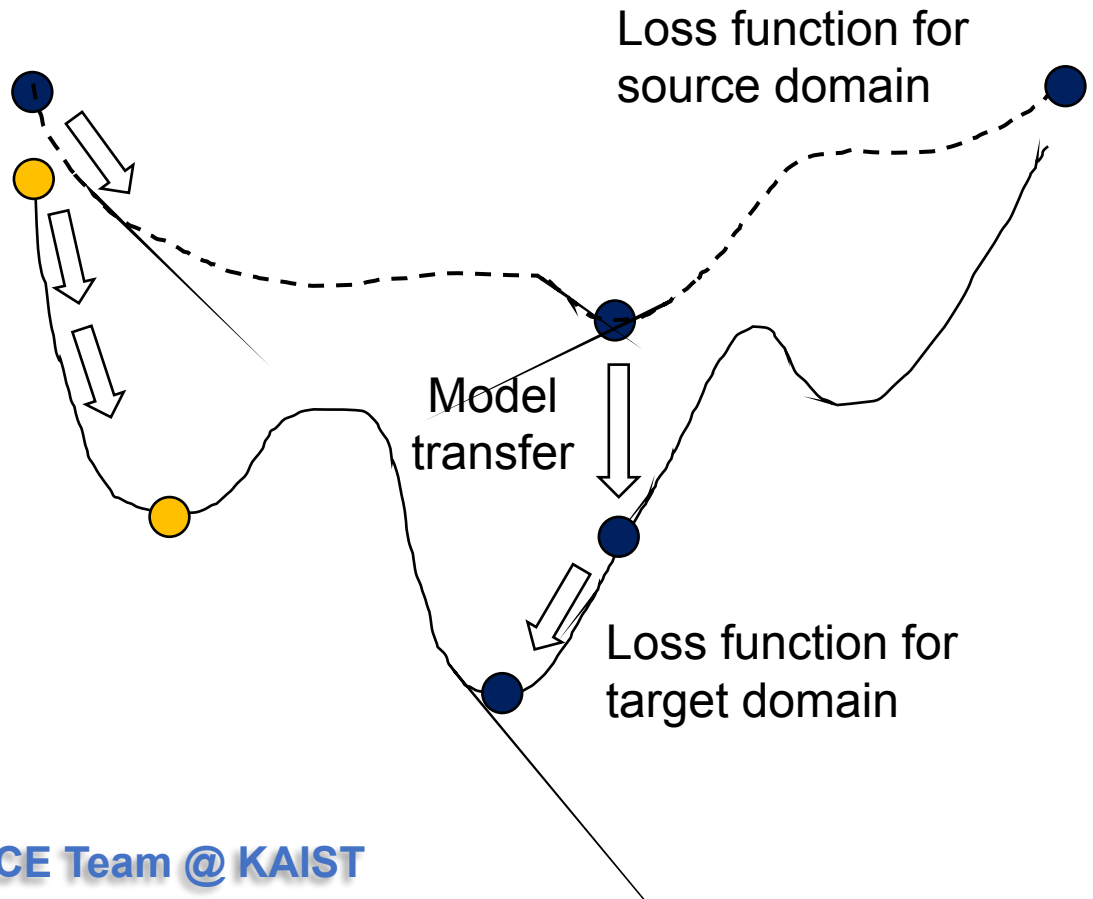
Negative transfer example

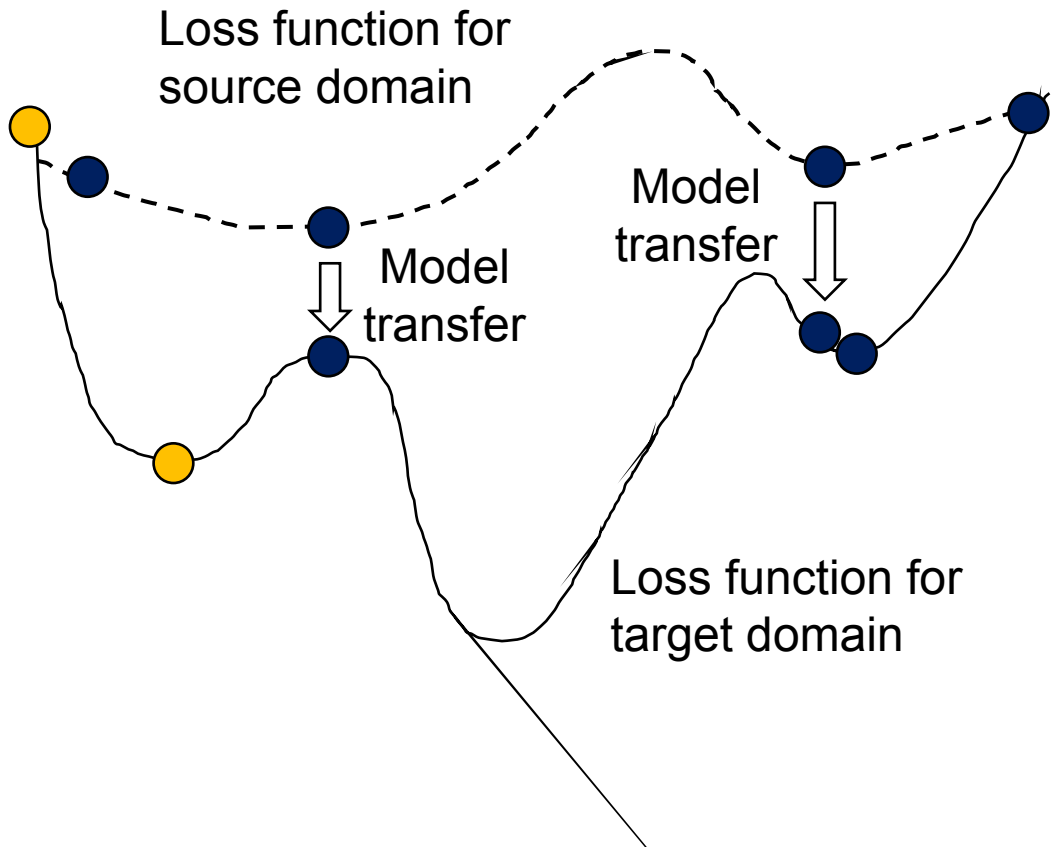
	Raw	logP	Druglikeness	TOX21
GCN	**0.802**	0.724 ▼	0.619 ▼	0.682 ▼
GAT	0.782	0.710 ▼	0.760 ▼	**0.793** ▲
JK-GAT	**0.817**	0.699 ▼	0.748 ▼	0.736 ▼

"**Negative transfer** happens when the source domain data and task contribute to the reduced performance of learning in the target domain"

Why negative transfer occurs?

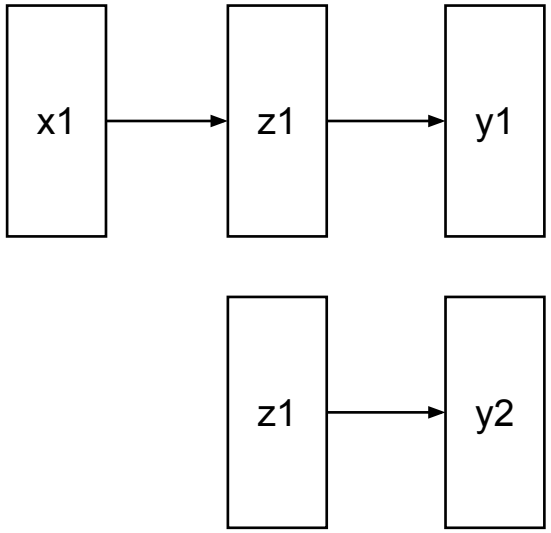
- Key for model transfer: Fitting function for source and target domains should be somehow similar



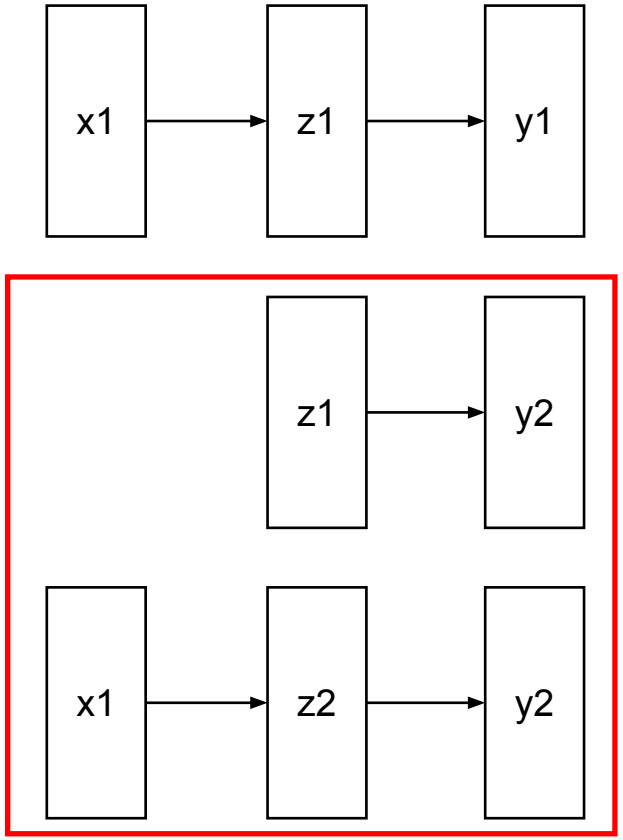


Why negative transfer occurs?

- Key success for feature transfer: Source and task should share common feature
- Ineffective representations (do not share common feature) + Loss of feature



If useful z1, it could work

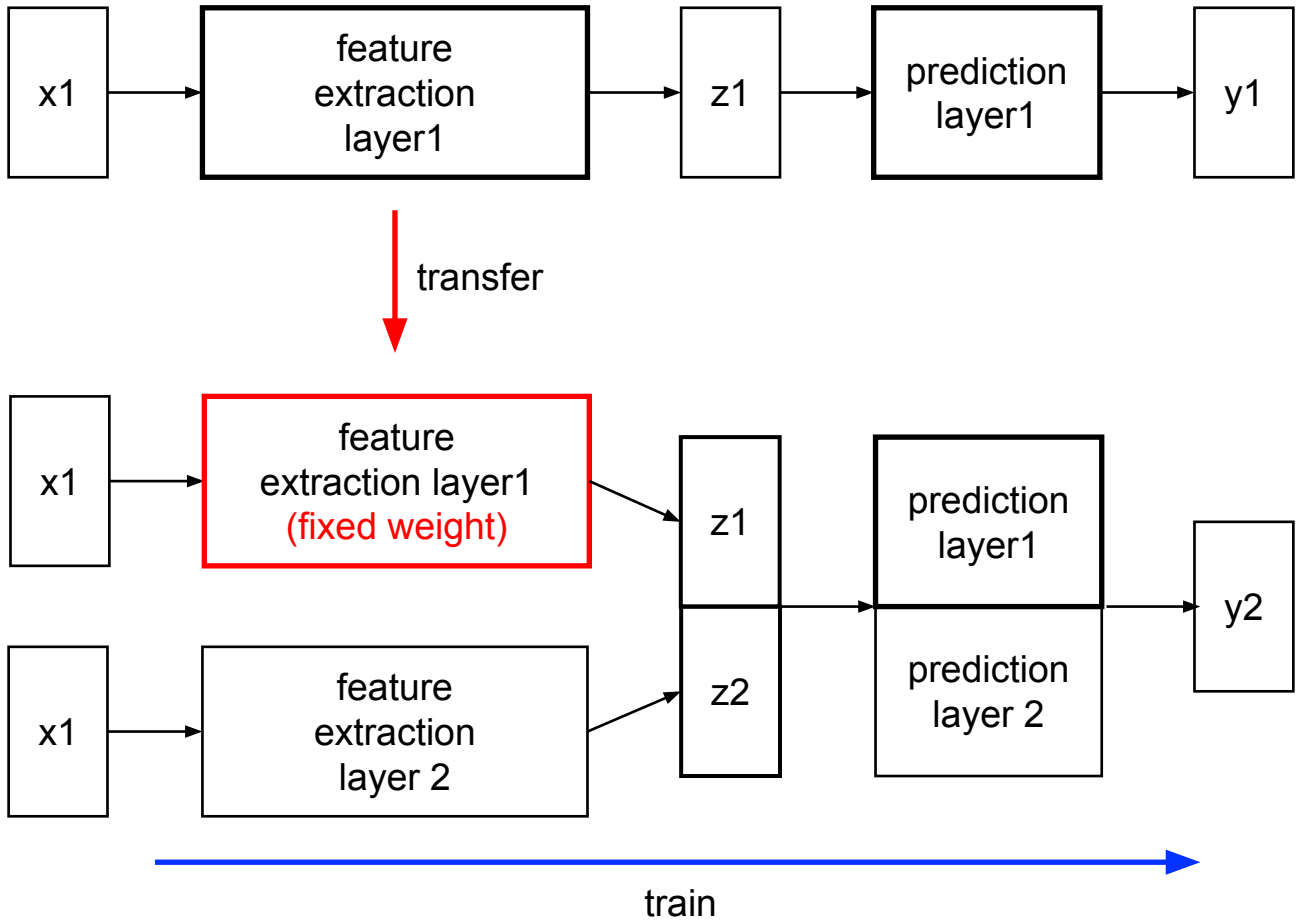


Info(x1) ≥ Info(z1) ≥ Info(y1)
For scalar y1,
Info (x1)>Info(z1)>Info(y1)

If not useful, info(z1) < Info(x1), using x1 is better for training

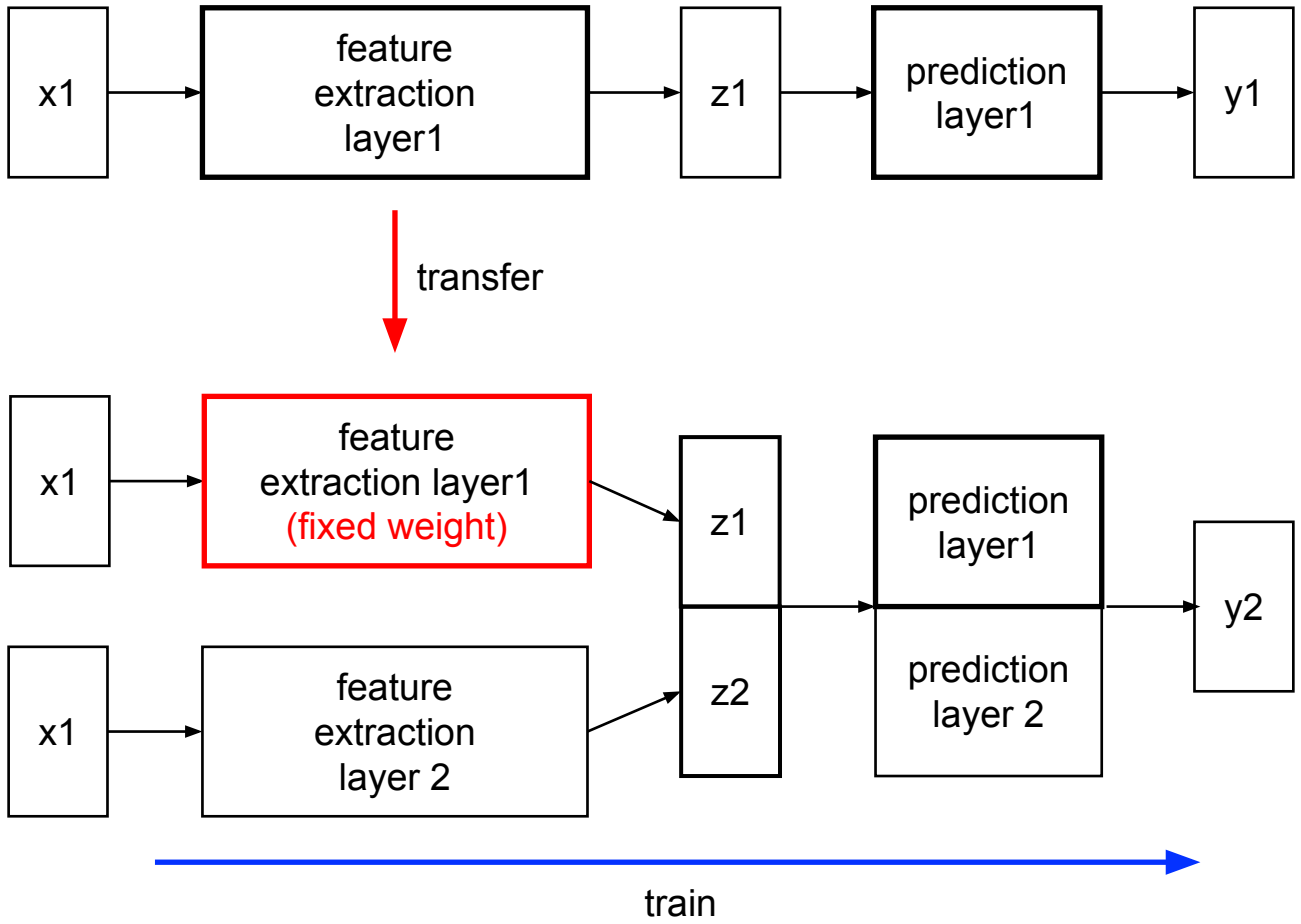
Idea of constructive transfer

- No information loss, try to overcome the previous model



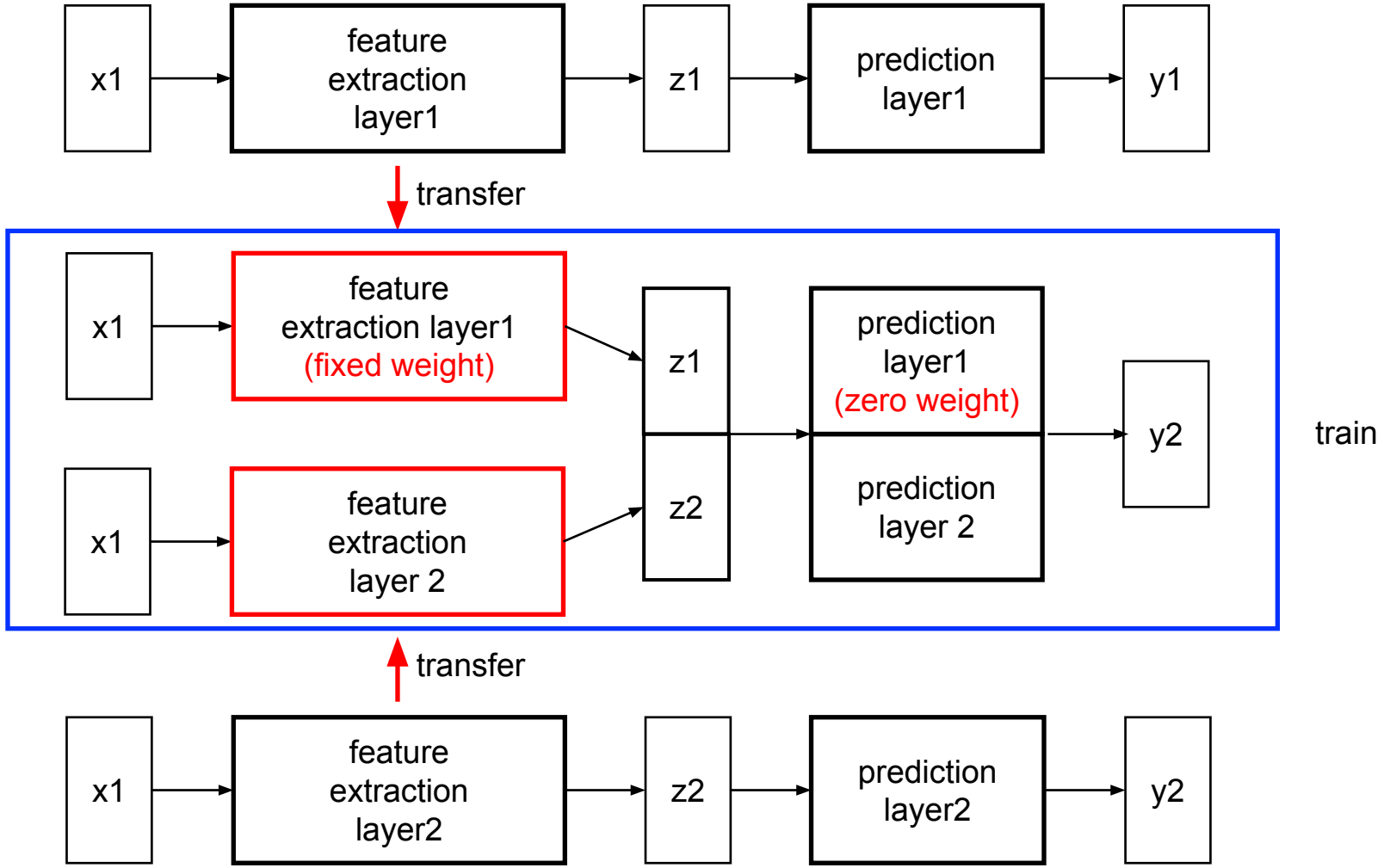
Constructive transfer learning strategy (1)

- non zero weight



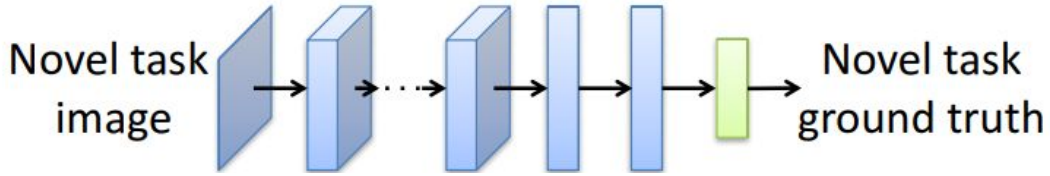
Constructive transfer learning strategy (2)

- zero weight

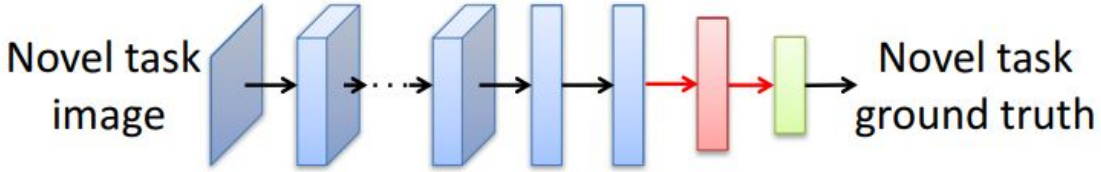


Similar work: Growing a brain

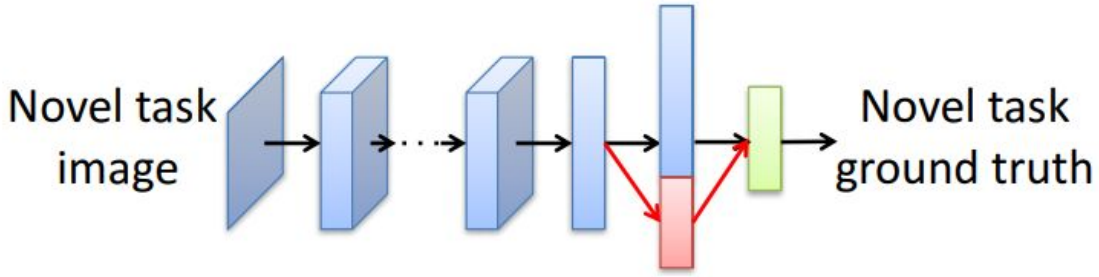
- Expand model capacity (make more neurons), our method also concats features



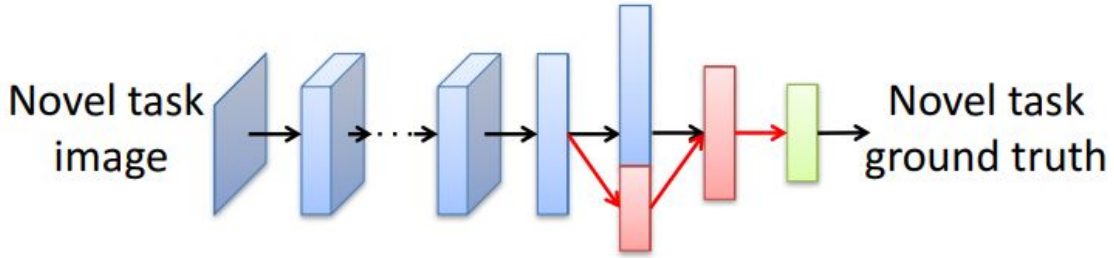
Classic fine-tuning



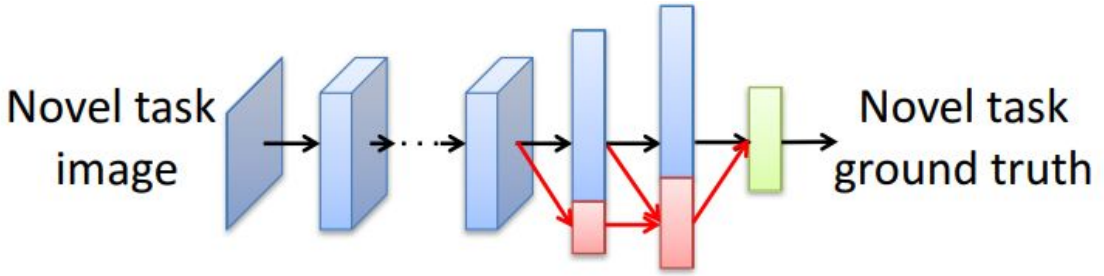
Depth augmentation



Width augmentation



Jointly depth/width augmentation

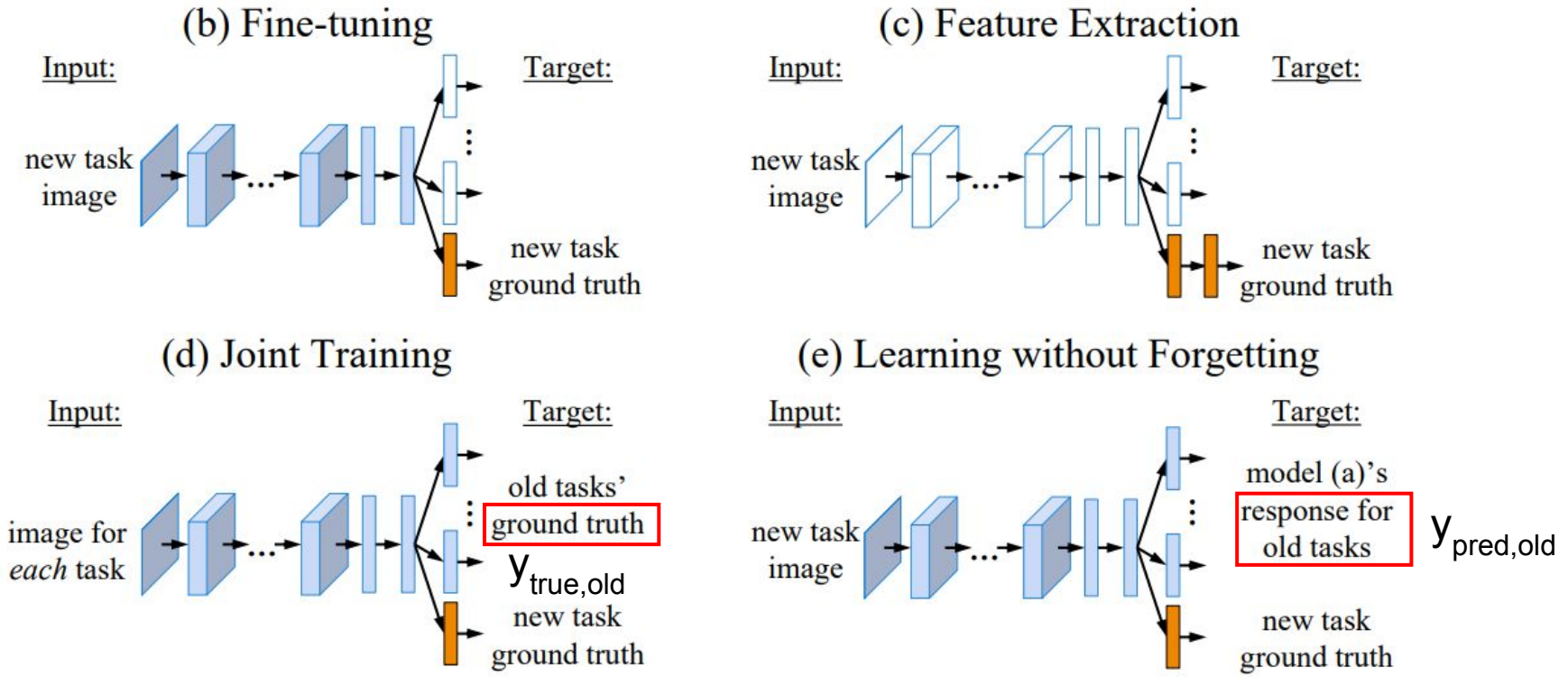


Recursive width augmentation

Yu-Xiong Wang, Deva Ramanan, Martial Hebert, "Growing a Brain: Fine-Tuning by Increasing Model Capacity", IEEE Conference on Computer Vision and Pattern Recognition(2017) 2471-2480

Similar work: Learning without forgetting

- Similar to our method that it tries to fit previous tasks: Sometimes it is just regarded as multitask learning (our model also uses the features of previous tasks)



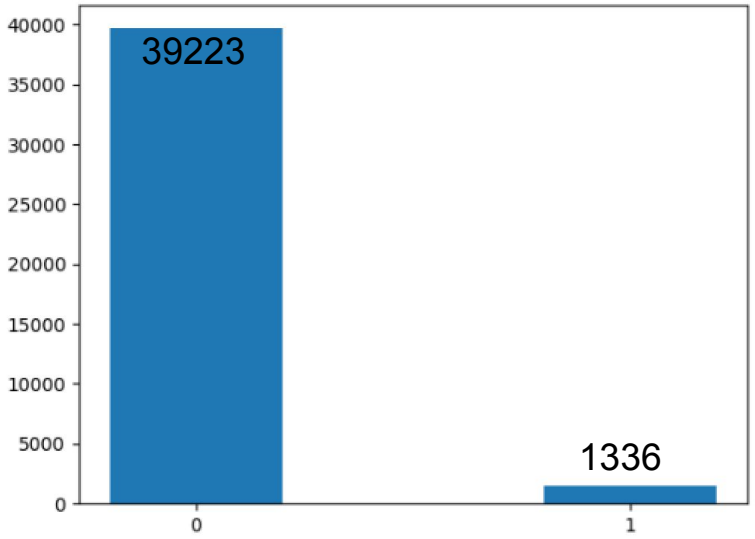
Zhizhong Li, Derek Hoiem, "Learning without Forgetting", IEEE Transactions on Pattern Analysis and Machine Intelligence 40.12 (2018) 2935 - 2947

Main difference with our method

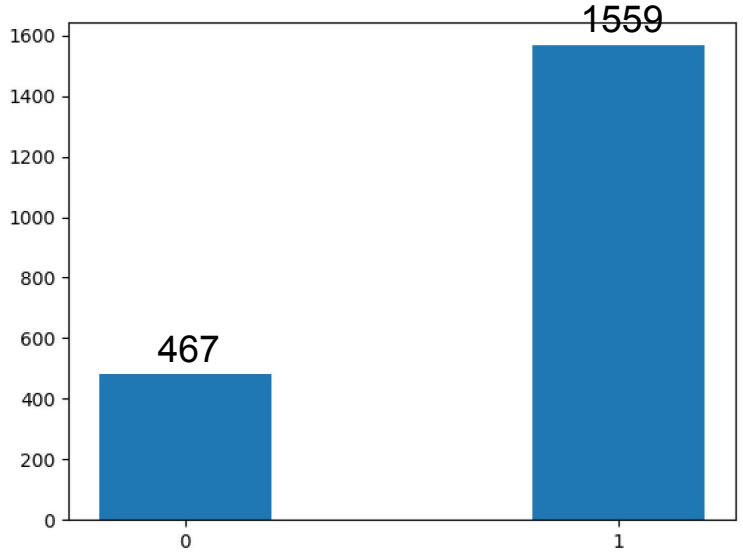
- Growing brain pros: Transfer learning must work, since larger capacity should guarantee better model
- Growing brain cons: Number of trainable parameters explode, also some possibility for extraction new features do not work
- Without forgetting pros: Definitely try to reuse the previous tasks
- Without forgetting cons: If the tasks are not related, it could occur negative transfer
- Constructive transfer learning: Somehow combination of these two methods which can work efficiently

Dataset

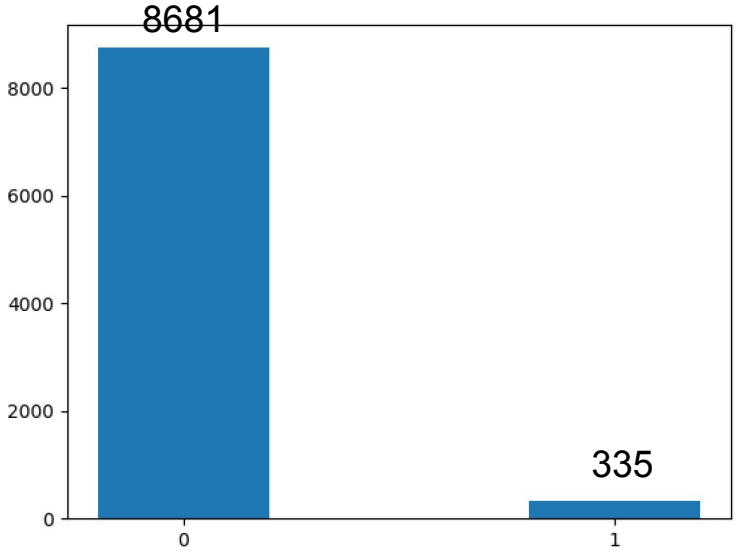
- Classification data(BBBP, HIV, TOX21 and etc) : **Highly imbalanced**



HIV



BBBP

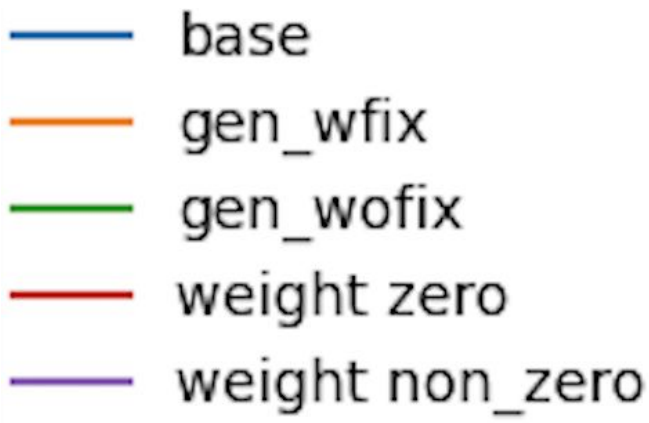


TOX21

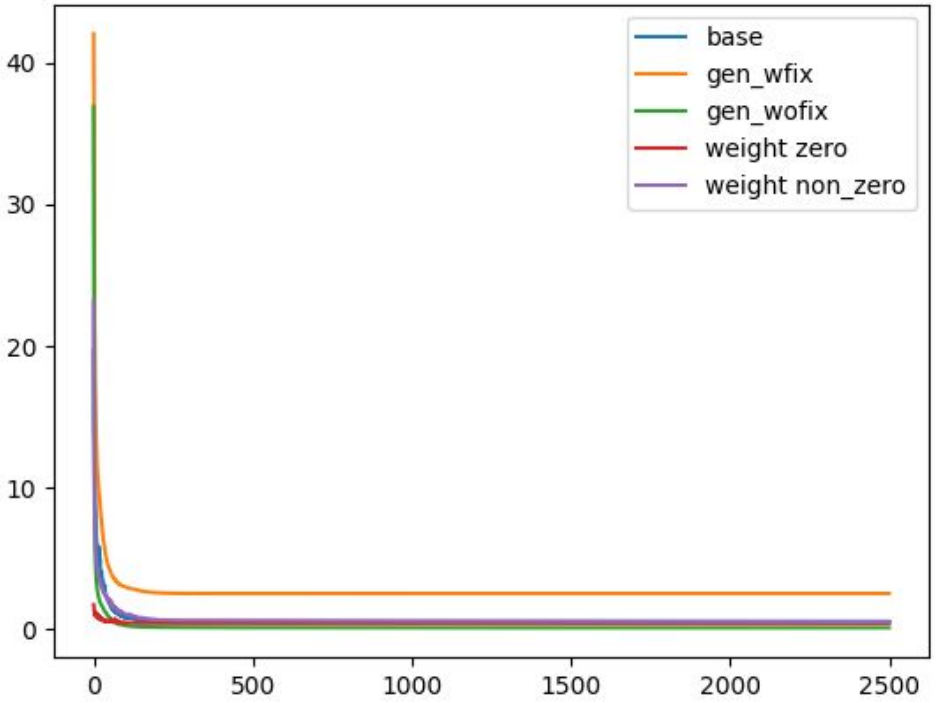
- Regression data(ESOL, freesolv, Lipophilicity, logP, qm9-homo, qm9-lumo) : more fair for comparing the model performance

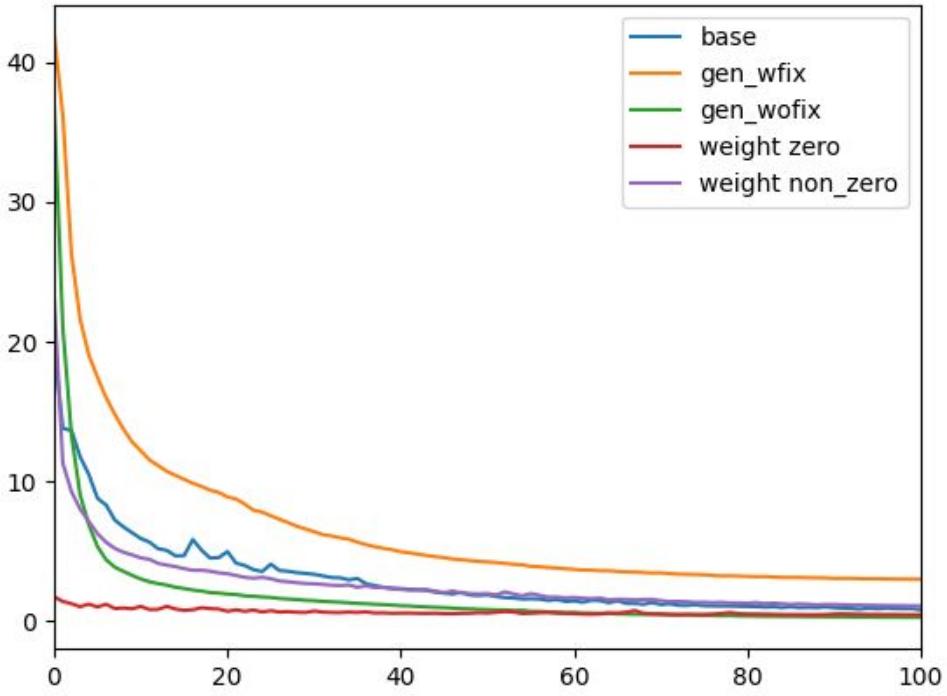
Model Evaluation

- General GCN model (Baseline model)
- Feature transfer learning **(general transfer with fixed weight)**
- Model transfer learning **(general transfer without fixed weight)**
- Constructive transfer learning **with non zero weight**
- Constructive transfer learning **with zero weight**
- Source task : logP, qm9-homo
- Target task : freesolv, Lipophilicity



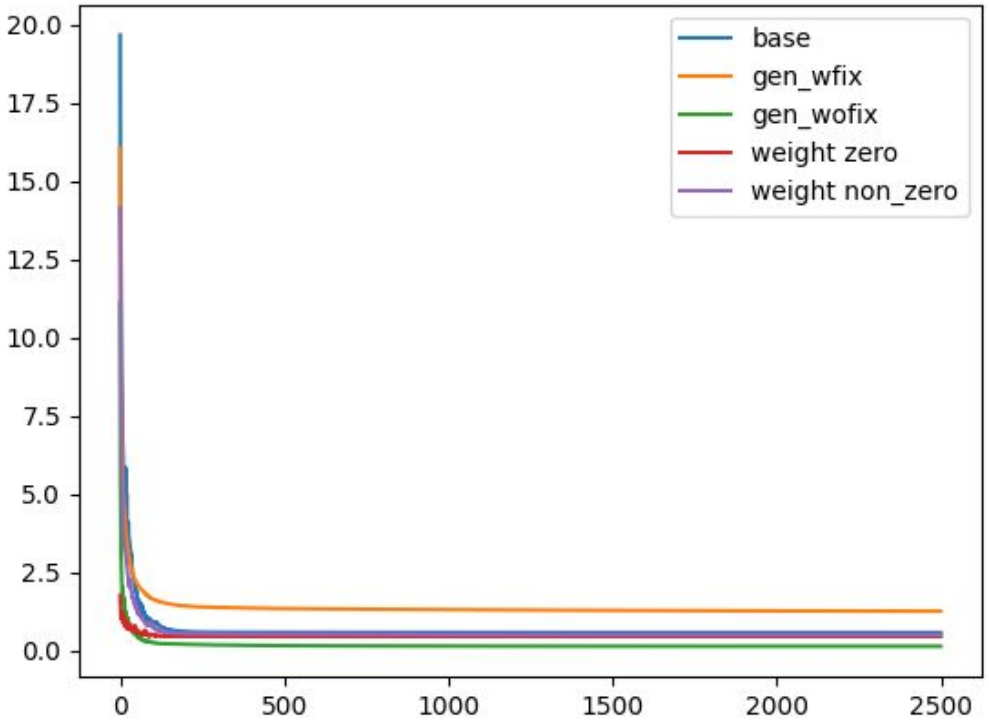
Task (1) : logP → freesolv

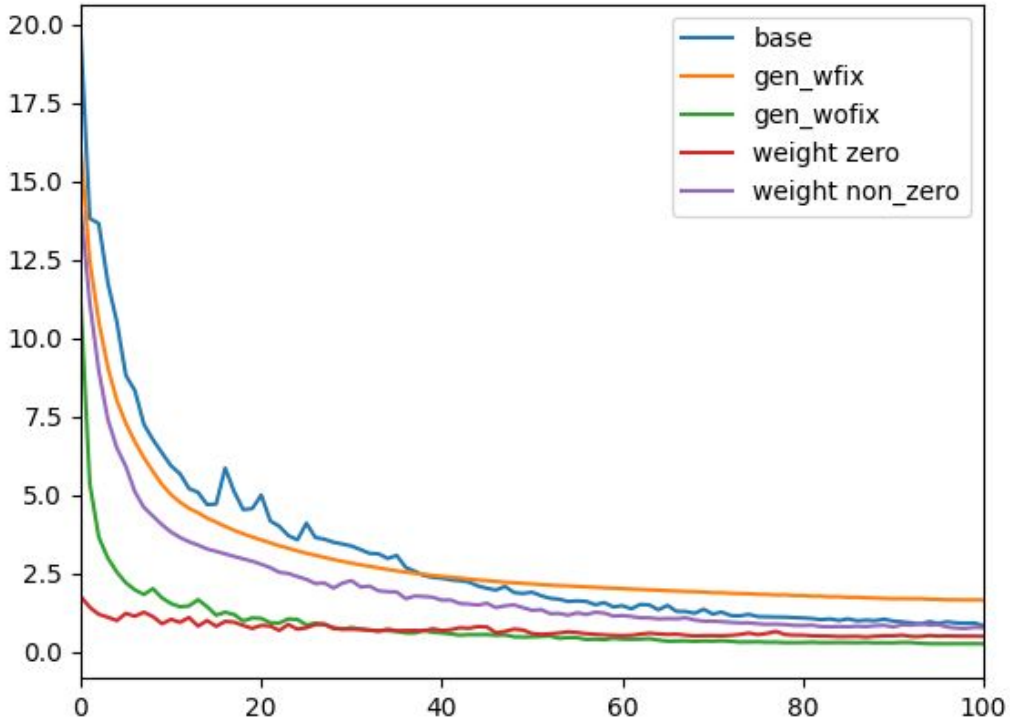




	base	Feature transfer	Model transfer	Constructive weight non zero	Constructive weight zero
Train loss	0.6489	2.5471 ▲	0.1327 ▼	0.5583 ▼	0.4445 ▼
Test loss	1.8694	2.7406 ▲	1.9477 ▲	1.2959 ▼	1.7449 ▼

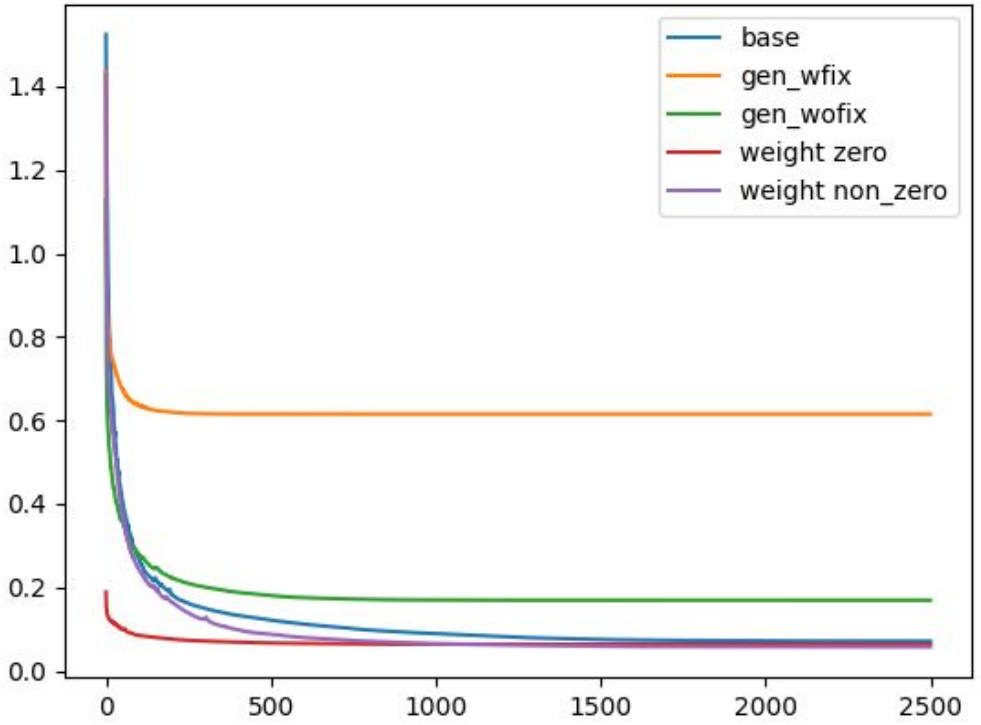
Task (2) : homo → freesolv

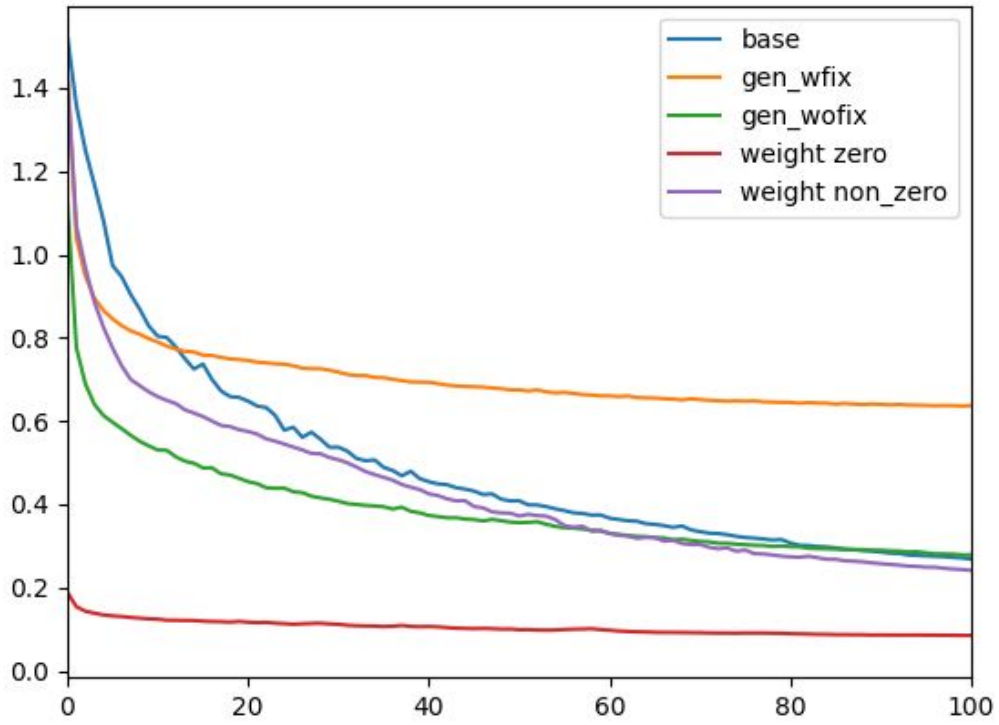




	base	Feature transfer	Model transfer	Constructive weight non zero	Constructive weight zero
Train loss	0.6489	1.1261 ▲	0.3101 ▼	0.4927 ▼	0.4511 ▼
Test loss	1.8694	2.400 ▲	1.4180 ▼	1.5477 ▼	1.7319 ▼

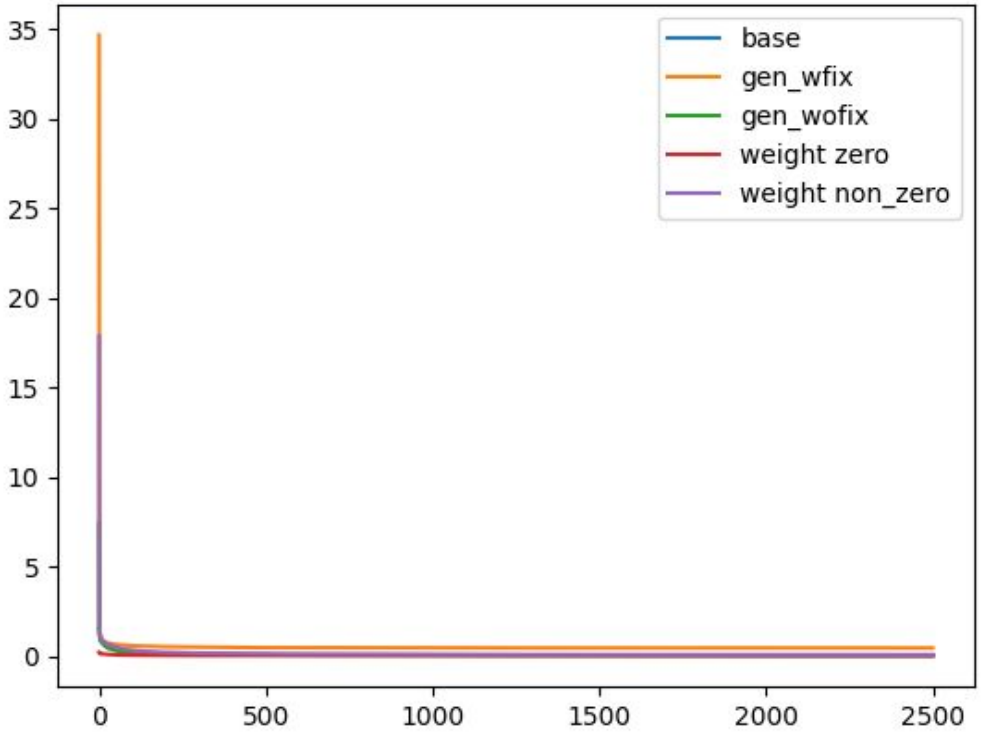
Task (3) : logP → Lipophilicity

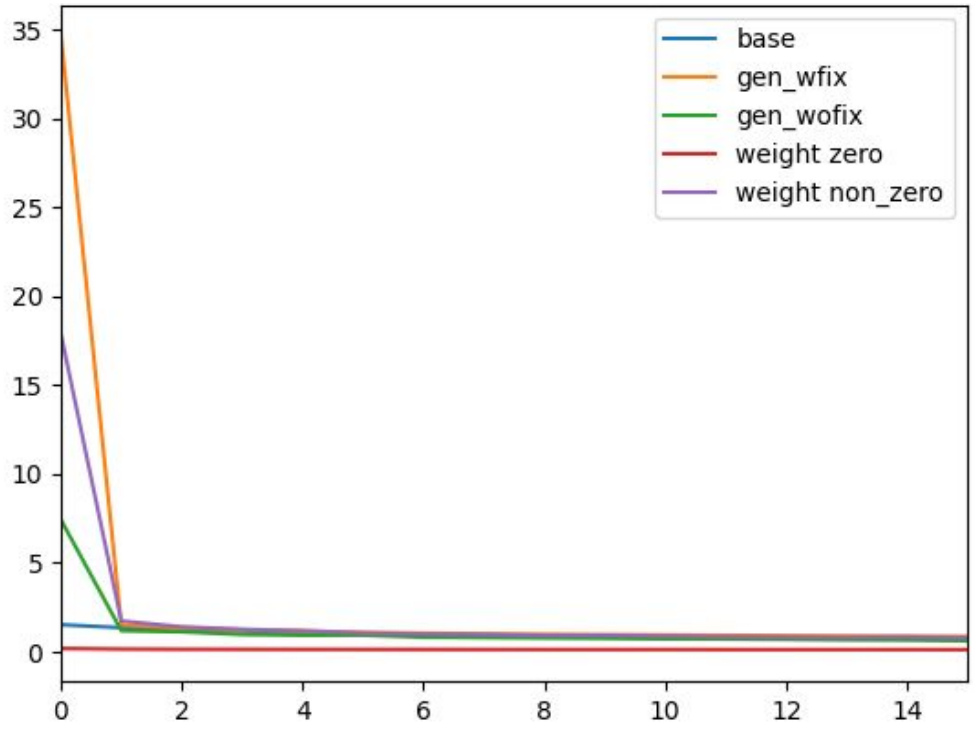




	base	Feature transfer	Model transfer	Constructive weight non zero	Constructive weight zero
Train loss	0.08602	0.6152 ▲	0.1694 ▲	0.05780 ▼	0.05877 ▼
Test loss	0.5095	0.6809 ▲	0.5281 ▲	0.4341 ▼	0.5449 ▲

Task (4) : homo → Lipophilicity





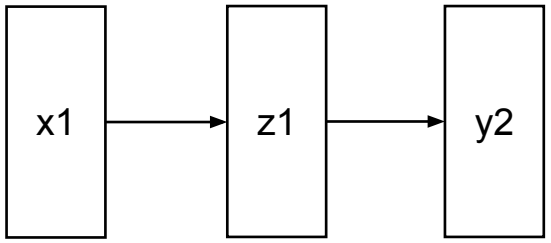
	base	Feature transfer	Model trasnfer	Constructive weight non zero	Constructive weight zero
Train loss	0.08602	0.4857 ▲	0.07161 ▼	0.1144 ▲	0.05884 ▼
Test loss	0.5095	0.5595 ▲	0.4291 ▼	0.4242 ▼	0.5569 ▲

Result of Constructive transfer learning

- For common **model transfer**, generally, training loss decreased, but not test loss (**underfit solved**, however, **overfit still not solved**, don't know why)
- For common **feature transfer**, as we expected, it showed **high negative transfer**. Given feature was not useful
- In general, our suggested transfer learning (constructive transfer learning) showed performance improvement for both training loss and test loss **regardless of tasks** (Achieved basic objective)
- However, the improvement was **marginal rather than significant**, less significant than model transfer

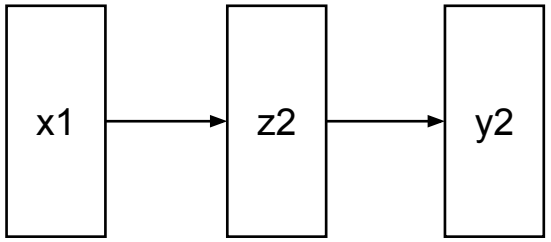
Demonstration of SSL using Fingerprint

- Negative learning for feature transfer



Info(x1) ≥ Info(z1) ≥ Info(y1)
For scalar y1,
Info (x1)>Info(z1)>Info(y1)

If we set y1 = x1, or Info(y1) = Info(x1), we can get Info(z1) = Info(x1)=Info(y1)



If not useful, info(z1) < Info(x1), using x1 is better for training

In this case, since Info(z1)=Info(x1), using z1 could be similar with using x1, or even better

Smiles to fingerprint with GCN model

- MLP model generally has a very low train loss and a very high test loss -> highly overfit
- GCN model less overfits than MLP or sometimes underfits

	freesolv	ESOL	Lipoph-ilicity
Train loss	0.001988	0.06410	8.023e-4
Test loss	2.0820	0.6142	0.6515

MLP

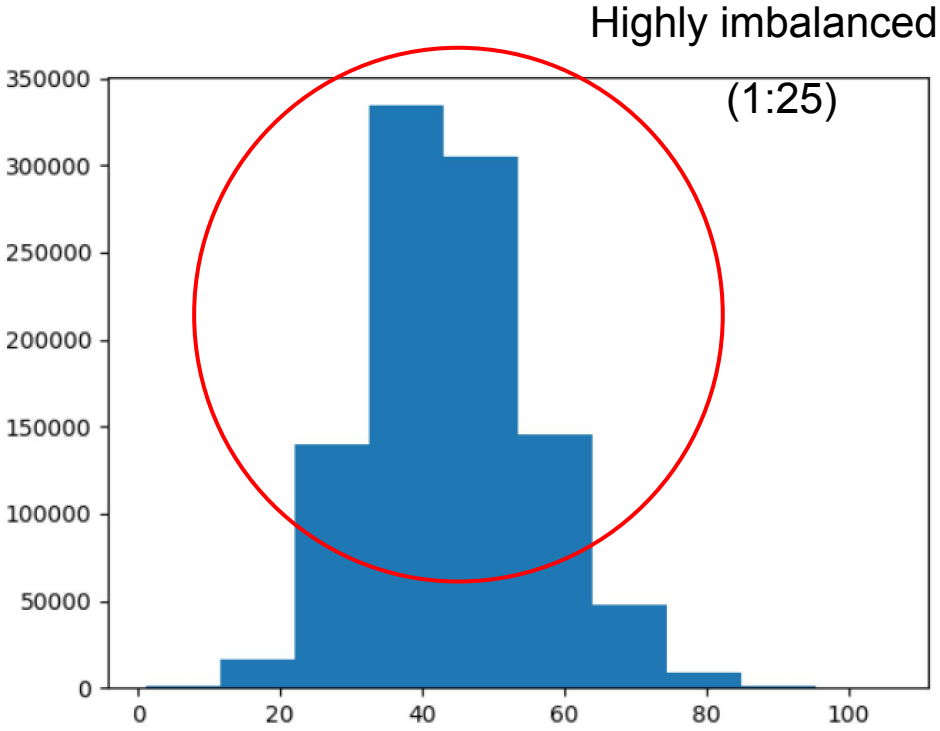
	freesolv	ESOL	Lipoph-ilicity
Train loss	0.6489	0.1431	0.08602
Test loss	1.8694	0.2218	0.5095

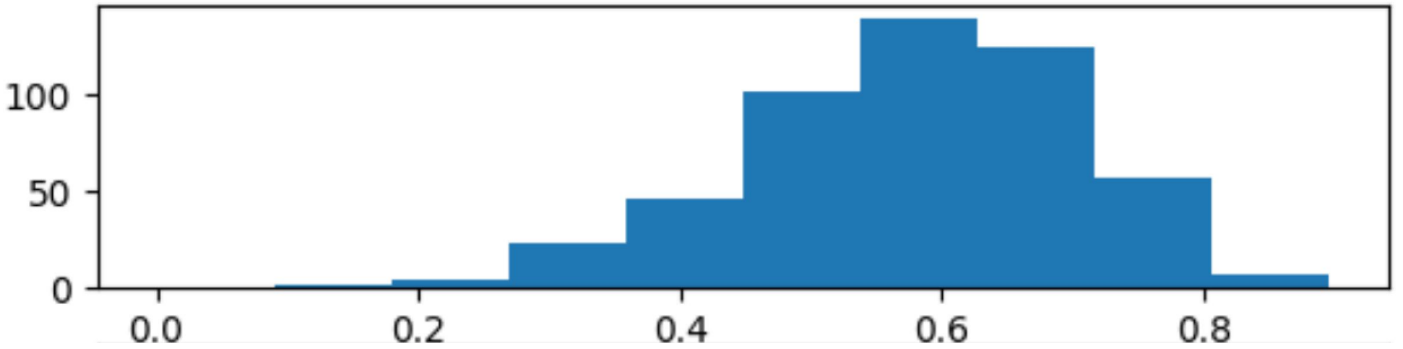
GCN

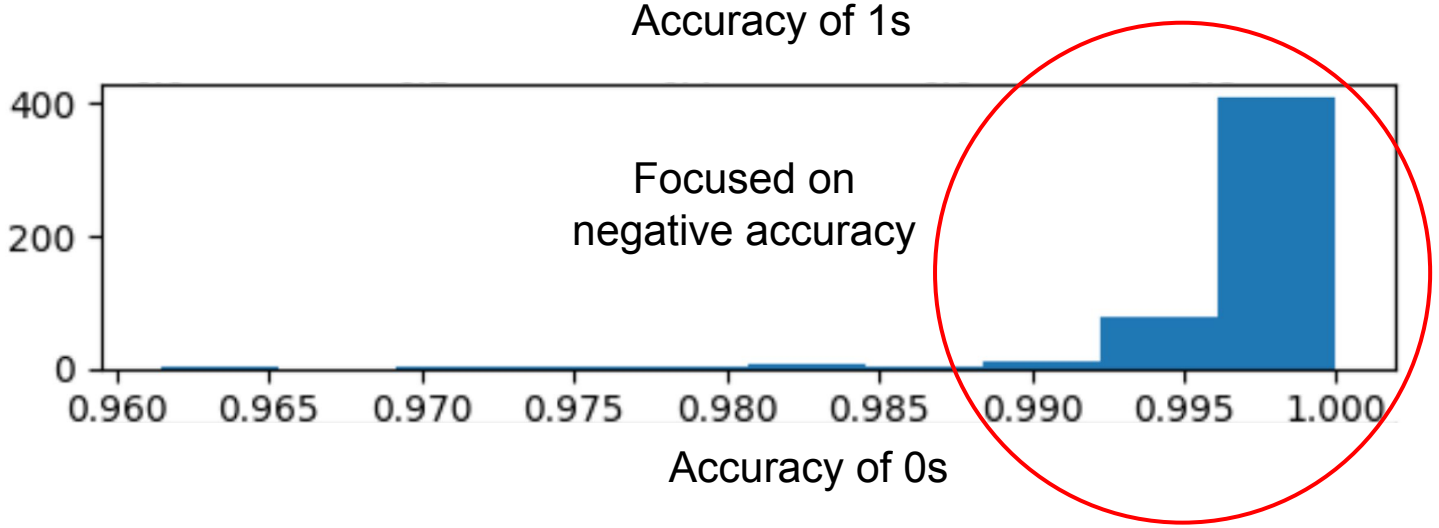
- Therefore, we can expect the **feature of fingerprint** to be useful for improving GCN model
- By transferring fp feature with transfer learning, we can expect improvement of GCN model
- Also, since fingerprint is another representation of molecule, the information to extracted features is equal to the original molecule information (In general, Info(fp) < Info(mol)

Learn fingerprint with GCN

- Train GCN model to predict every element of fingerprint using sigmoid function
- Test loss = 0.002651







Class balanced loss for fp

- Redefine loss function by reducing weight to the loss of major element(‘0’)

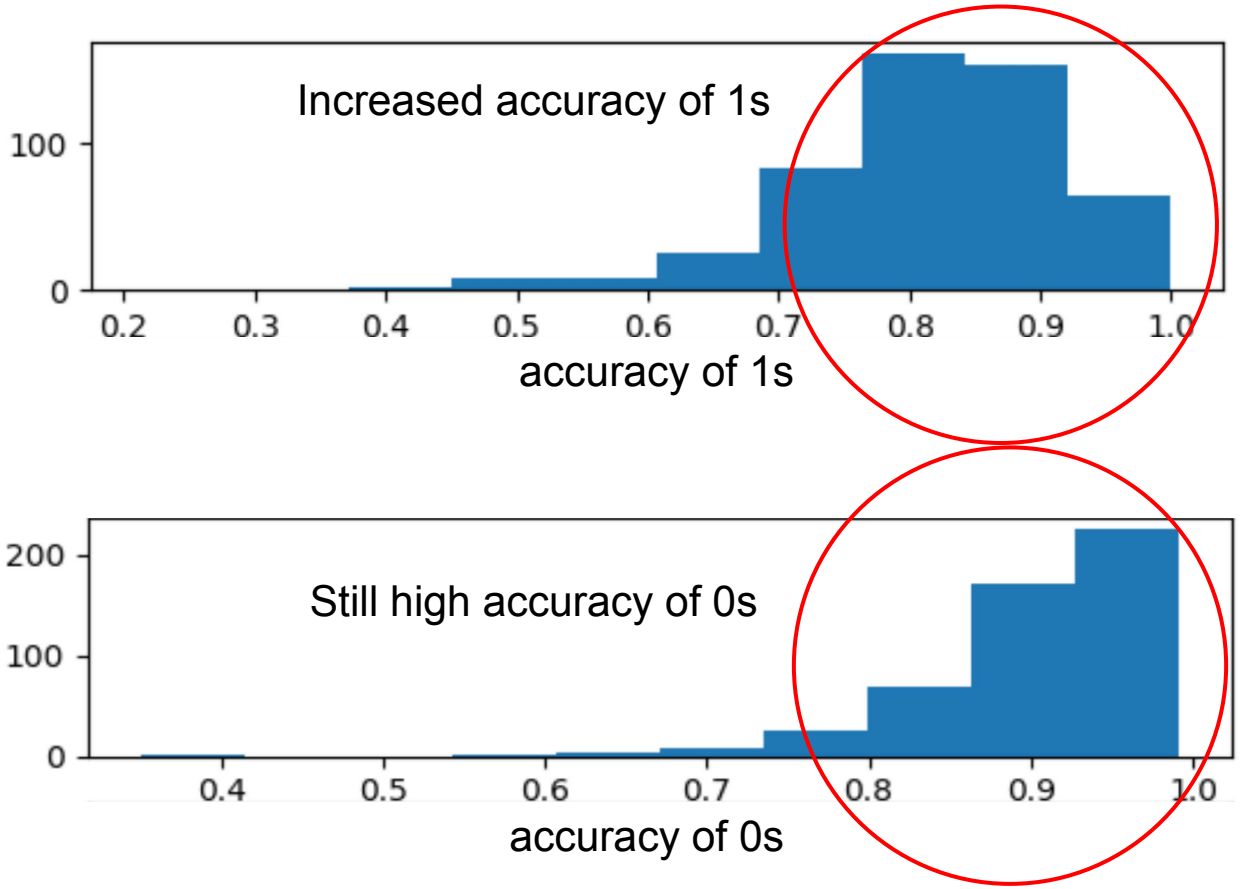
```
def loss(pred, y):
    c = 0.04
    pred = torch.where(pred < 1e-5, pred + 1e-5, pred)
    pred = torch.where(pred > (1-1e-5), pred - 1e-5, pred)
    loss = -(y.mul(torch.log(pred)) + (1 - y).mul(torch.log(1-pred)))
    positive_loss = loss.mul(y)
    negative_loss = loss.mul(1-y)
    total_loss = positive_loss + negative_loss * c
    return total_loss.mean()
```

$$L_{original} = L_0 + L_1$$

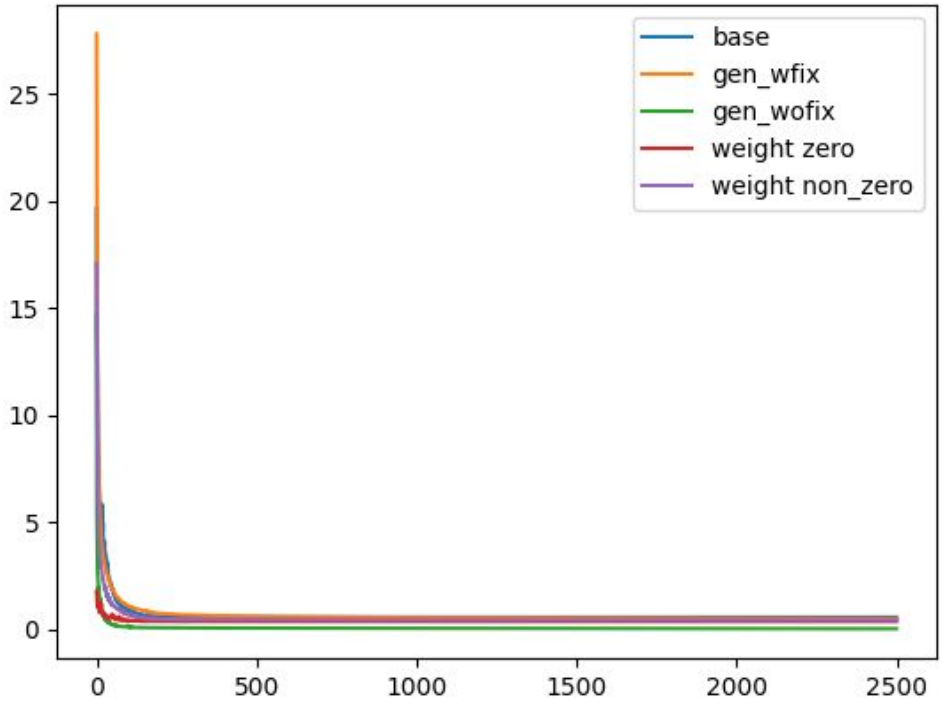
$$L_{balanced} = 0.04*L_0 + L_1$$

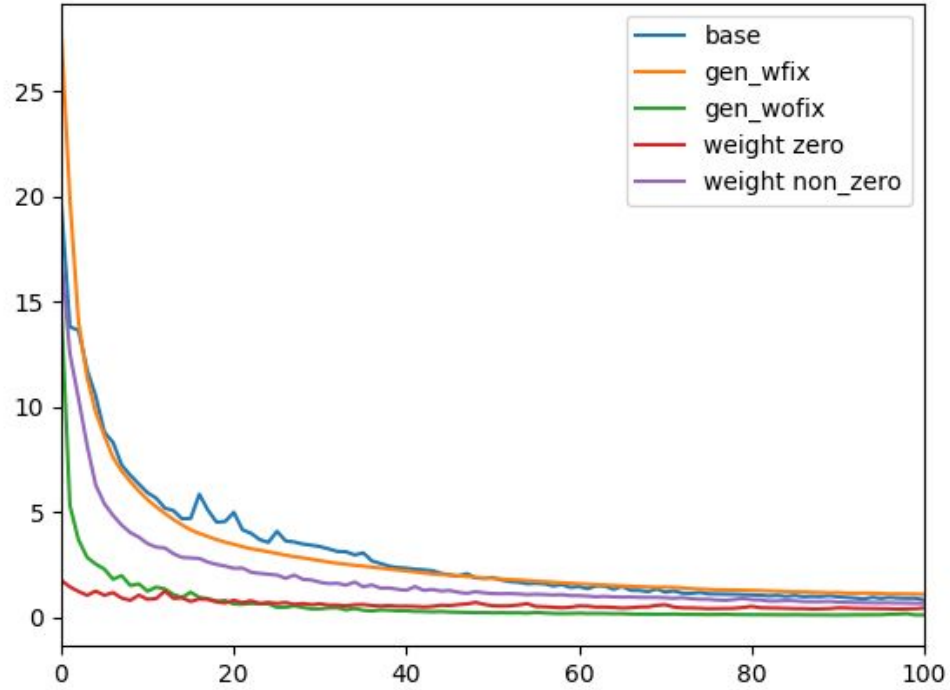
L_0: Loss of 0s

L_1: Loss of 1s



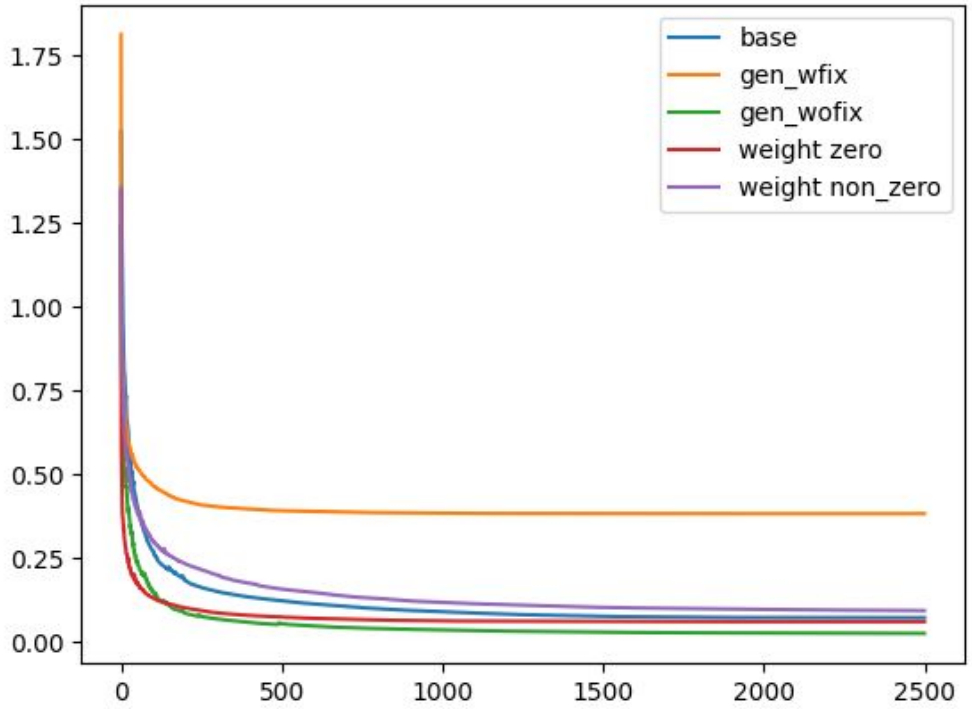
Task (1) : fingerprint → freesolv

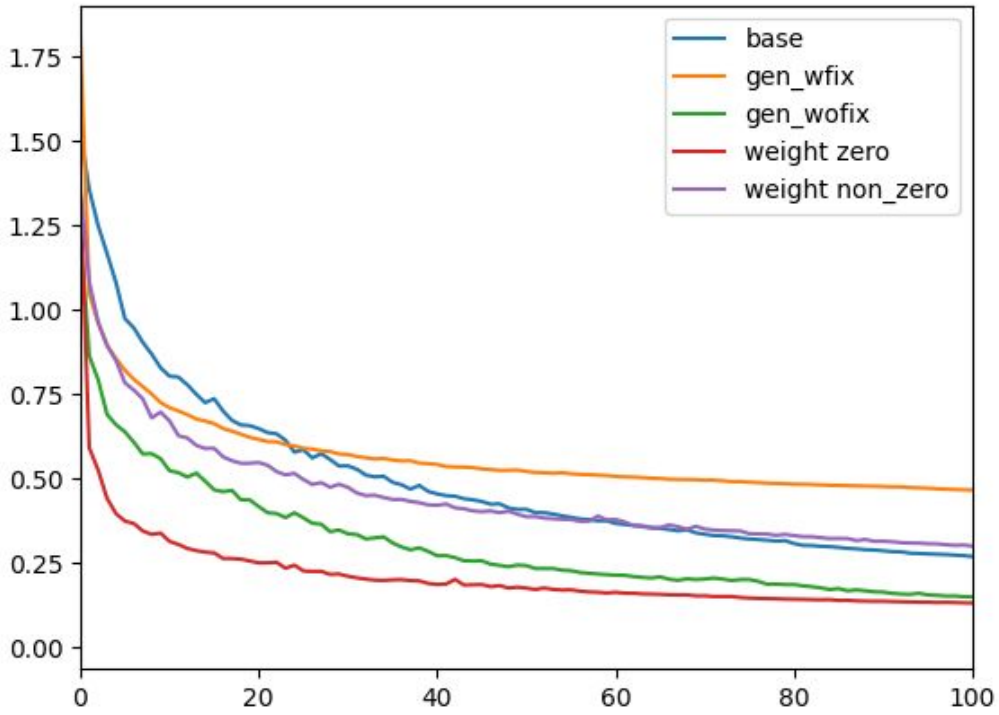




	base	Feature transfer	Model transfer	Constructive weight non zero	Constructive weight zero
Train loss	0.6489	0.5000 ▼	0.03166 ▼	0.4355 ▼	0.3727 ▼
Test loss	1.8694	2.3008 ▲	1.8333 ▼	1.3507 ▼	1.594 ▼

Task (2) : fingerprint → Lipophilicity





	base	Feature transfer	Model transfer	Constructive weight non zero	Constructive weight zero
Train loss	0.08602	0.3824 ▲	0.02628 ▼	0.09264 ▲	0.05964 ▼
Test loss	0.5095	0.4797 ▼	0.5072 ▼	0.4287 ▼	0.5073 ▼

Result of SSL

- For SSL, training loss even decreased for feature transfer. Meaning, that SSL with FP helps to make better representation for solving several problems
- However, it could not solve overfit issue. Therefore, we think SSL can solve underfit problem but cannot solve overfitting problem (for this, future discussion by L.K.H, using unsupervised learning)
- Our suggested transfer learning (constructive transfer learning) also showed similar result with the previous task, marginal improvement for both training loss and test loss

Conclusion

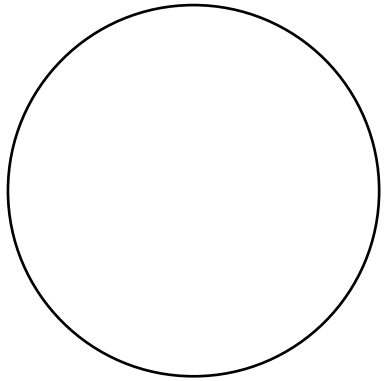
- Our suggested transfer learning (constructive transfer learning) improves performance over the base model and feature transfer but not model transfer.
- To solve negative transfer problem, we use SSL with Fingerprint and it helps to improve performance but can not solve overfitting problem.
- SSL with fingerprint also showed performance improvement in constructive transfer learning.
- Research is needed on improving the performance of the transfer learning better and solving overfitting problem.

Appendix

Detail of Transfer Learning Formalism

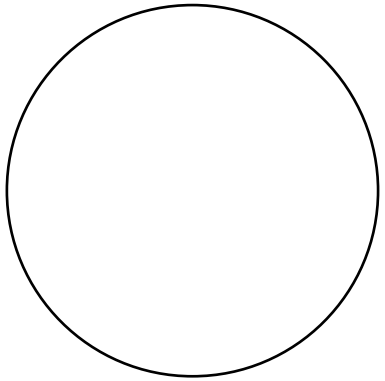
- Formalism of model transfer

Source domain
(x_S, y_S)



Fitting function
$y_S = f(x_S)$

Task domain
(x_T, y_T)



Fitting function
$y_T = g(x_T)$

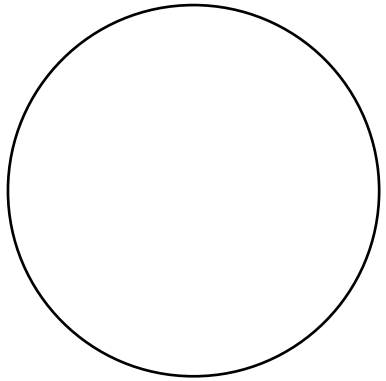
*** Key for success

1. $xT \subseteq xS$:
2. Since, it shares same model architecture, f and g should be similar (f~g: Not clearly defined, one example: **f(x)=ag(x) + b**)
3. In other words, would be easy transfer if yS and yT are correlated with each other
4. How can prevent overfitting: If fitting function f also works for general set xT, overfitting can be removed

Detail of Transfer Learning Formalism

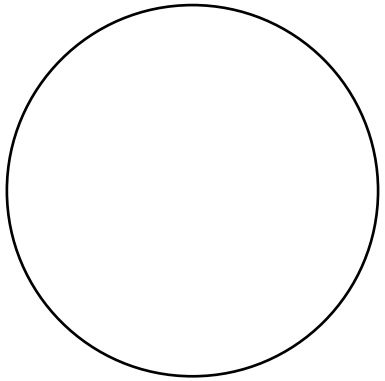
- Formalism of feature transfer

Source domain
(xS,yS)



Fitting function
yS = f(h(xS))

Task domain
(xT,yT)



Fitting function
yT = g(h(xT))

*** Key for success

1. Belief that both fitting function should share common feature extracted by h
2. Therefore, **h(xS) = f-1(yS) ≈ g-1(yT)**, thus xT ⊆ xS must hold
3. Therefore, yT can be well expressed as yT = F(yS), the problem may hold well throughout some common feature
4. How can prevent overfitting: If fitting function f also works for general set xT, overfitting can be removed

Relation transfer (Appendix)

- Transfer relation information learned from source
- Therefore, at least domain should contain information of relation, we call this "relational domain"
- Useful if the given source and target have different data type (Ex. Undefined class)
- Example 1: Physics to financial
- Example 2: Train classify dog and cat and apply to train classify airplane and human

Examples of TL in chemistry (3) – Fine-Tuning

Deep Transferable Compound Representation across Domains and Tasks for Low Data Drug Discovery

Karim Abbasi,[†] Antti Poso,[‡] Jahanbakhsh Ghasemi,[§] Massoud Amanlou,[∥] and Ali Masoudi-Nejad*[,†]

Physiology Data Sets
(Source → Target)

Tox21 → SIDER

Tox21 → SIDER	10+/10−		1+/1−	
	ROC-AUC	paired *t* test	ROC-AUC	paired *t* test
RF (100 trees)	0.55 ± 0.04	5.63×10^{-32}	0.51 ± 0.04	1.25×10^{-29}
SVM	0.49 ± 0.07	9.91×10^{-33}	0.48 ± 0.05	1.17×10^{-25}
GraphConv (GC)	0.54 ± 0.02	4.48×10^{-12}	0.52 ± 0.02	1.59×10^{-9}
Siamese	0.51 ± 0.03			
AttnLSTM	0.51 ± 0.01			
IterRefLSTM	0.51 ± 0.01			
Multitask (Source + labeled target data)	0.56 ± 0.02	1.42×10^{-5}	0.53 ± 0.03	3.13×10^{-13}
our approach (without semantic transfer)	0.58 ± 0.01	2.56×10^{-4}	0.54 ± 0.03	4.31×10^{-8}
our approach	**0.63 ± 0.02**		**0.60 ± 0.02**	

[a]In this experiment, the ROC-AUC score is reported. 10+/10− indicates that there are ten positive and ten negative samples in the labeled target data. 1+/1− indicates that there are one positive and one negative sample in the labeled target data.

SIDER → Tox21

SIDER → Tox21	ROC-AUC	paired *t* test	ROC-AUC	paired *t* test
RF (100 trees)	0.58 ± 0.05	7.23×10^{-6}	0.56 ± 0.03	5.27×10^{-8}
SVM	0.53 ± 0.03	1.2×10^{-12}	0.49 ± 0.04	2.62×10^{-12}
GC	0.56 ± 0.03	8.1×10^{-6}	0.59 ± 0.03	4.28×10^{-5}
Multitask (Source + labeled target data)	0.63 ± 0.04	3.13×10^{-4}	0.60 ± 0.01	1.66×10^{-4}
our approach (without semantic transfer)	0.65 ± 0.02	6.37×10^{-4}	0.59 ± 0.02	5.57×10^{-6}
our approach	**0.74 ± 0.02**		**0.69 ± 0.02**	

Examples of TL in chemistry (3) – Fine-Tuning

Deep Transferable Compound Representation across Domains and Tasks for Low Data Drug Discovery

Karim Abbasi,[†] Antti Poso,[‡] Jahanbakhsh Ghasemi,[§] Massoud Amanlou,[∥] and Ali Masoudi-Nejad*[,†]

Biophysics Data Sets

HIV → BACE	10+/10−	1+/1−
RF (100 trees)	0.65 ± 0.06	0.52 ± 0.03
SVM	0.46 ± 0.07	0.39 ± 0.03
GraphConv (GC)	0.55 ± 0.03	0.47 ± 0.06
Multitask (Source + labeled target data)	0.62 ± 0.03	0.54 ± 0.02
our approach (without semantic transfer)	0.61 ± 0.02	0.57 ± 0.02
our approach	**0.69 ± 0.03**	**0.59 ± 0.03**

[a]In this experiment, the ROC-AUC score is reported. Randomness is over the choice of labeled target data; experiment is repeated ten times. Numbers denote the mean and standard deviation of these experiments.

HIV → BACE

BACE → HIV	10+/10−	1+/1−
RF (100 trees)	0.52 ± 0.06	0.56 ± 0.07
SVM	0.43 ± 0.09	0.41 ± 0.05
GraphConv (GC)	0.50 ± 0.06	0.45 ± 0.03
Multitask (Source + labeled target data)	0.58 ± 0.02	0.55 ± 0.02
our approach (without semantic transfer)	0.61 ± 0.04	0.55 ± 0.02
our approach	**0.66 ± 0.04**	**0.61 ± 0.04**

BACE → HIV

Examples of TL in chemistry (3) – Fine-Tuning

Deep Transferable Compound Representation across Domains and Tasks for Low Data Drug Discovery

Karim Abbasi,[†] Antti Poso,[‡] Jahanbakhsh Ghasemi,[§] Massoud Amanlou,[∥] and Ali Masoudi-Nejad*[,†]

Physiology:
Tox21
SIDER
ToxCast

Biophysics:
HIV
BACE

Best

Worst

Transfer Across Biophysics and Physiology Data Sets

	SIDER → Tox21	ToxCast → Tox21	HIV → Tox21
10+/10−	0.74 ± 0.02	0.74 ± 0.05	0.65 ± 0.04
1+/1−	0.69 ± 0.02	0.71 ± 0.03	0.62 ± 0.04
	Tox21 → HIV	**SIDER → HIV**	**BACE → HIV**
10+/10−	0.63 ± 0.02	0.60 ± 0.02	0.66 ± 0.04
1+/1−	0.56 ± 0.03	0.53 ± 0.04	0.61 ± 0.04
	Tox21 → SIDER	**ToxCast → SIDER**	**HIV → SIDER**
10+/10−	0.63 ± 0.02	0.63 ± 0.01	0.58 ± 0.03
1+/1−	0.57 ± 0.02	0.58 ± 0.05	0.54 ± 0.04