Learning And Mining from DatA
http://lamda.nju.edu.cn

An exploration to non-NN deep models based on non-differentiable modules

Zhi-Hua Zhou

http://cs.nju.edu.cn/zhouzh/
Email: zhouzh@nju.edu.cn

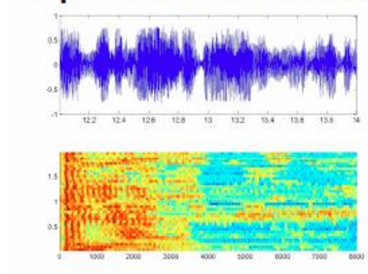
LAMDA Group
Nanjing University, China

Deep learning

Nowadays, deep learning achieves great success

Images & Video

Speech & Audio



Text & Language

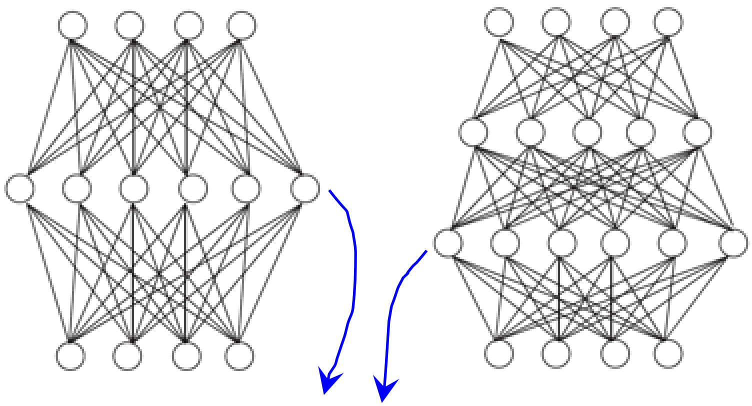
What's "Deep Learning"?

nowadays,

= Deep neural networks (DNNs)

Neural networks to deep

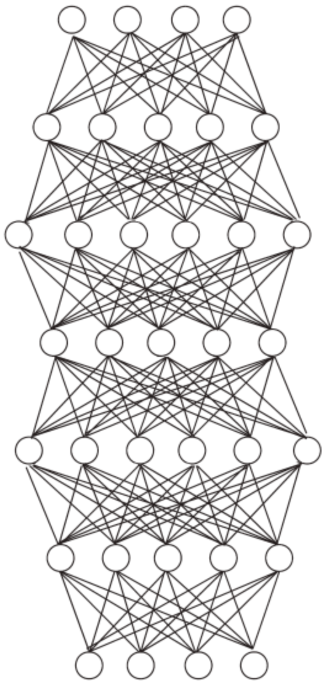
Traditional, single or double hidden layers

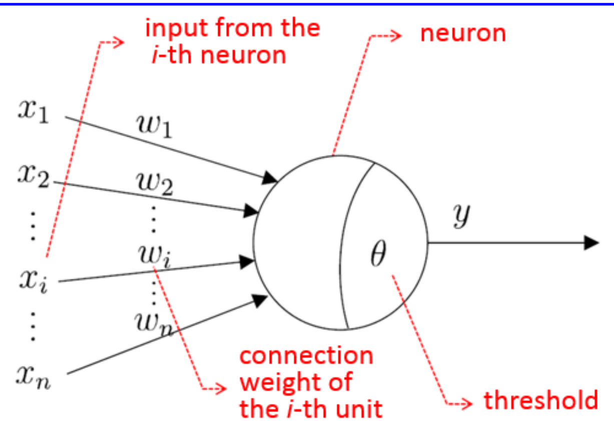


e.g., ImageNet winners:
2012: 8 layer
2015: 152 layer
2016: 1207 layer

deep

Many layers



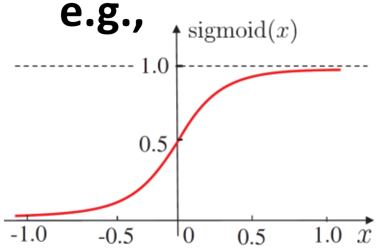


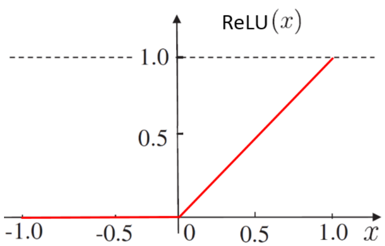
M-P model (1943) $y = f\left(\sum_{i=1}^{n} w_i x_i - \theta\right)$

Trained by
Backpropagation (BP)
or variant

***f*: continuous, differentiable**

e.g.,





Why deep? … One explanation

Learning And Mining from DatA
http://lamda.nju.edu.cn

Increase model complexity → increase learning ability

- Add hidden units (model width)
- Add hidden layers (model depth)

Adding layers is more effective than adding units

increasing not only the number of units with activation functions, but also the embedding depths of the functions

Increase model complexity → increase risk of overfitting; difficulty in training

- For overfitting: Big training data
- For training: Powerful comp facilities

Error gradient will diverge when propagated in many layers, difficult to converge to stable state, and thus difficult to use classical BP algorithm

Lots of tricks

One explanation: High complexity matters

- **BIG training data**

  The most simple yet effective way to reduce the risk of overfitting

- **Powerful computational facilities**

  Big models: Without GPU acceleration, DNNs could not be so successful

- **Training tricks**

  Heuristics, even mysteries

Error gradient will diverge when propagated in many layers, difficult to converge to stable state, thus difficult to use classical BP algo

⇒ **Enable to use high-complexity models**

⇒ **DNNs**

Why deep? … One explanation

Increase model complexity → improve learning ability

- Add hidden units (model width)
- Add hidden layers (model depth)

Adding layers is more effective than adding units

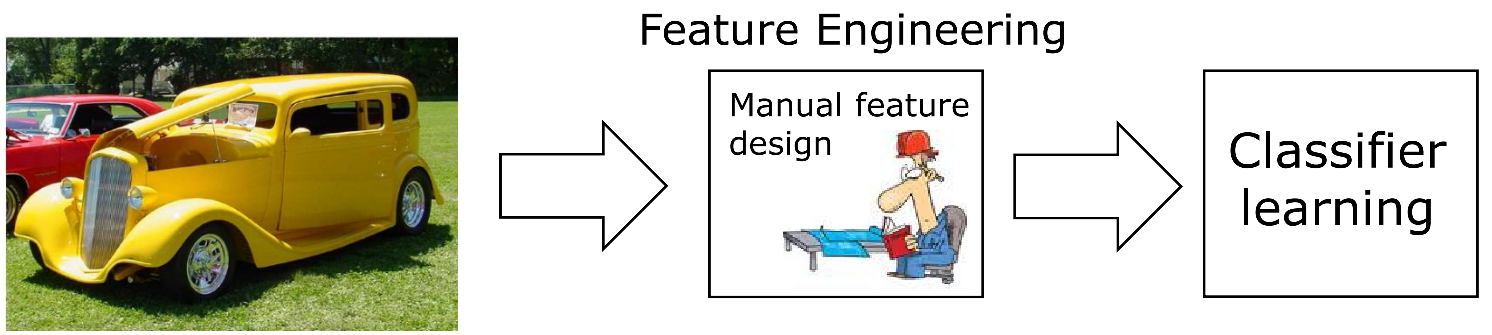
increasing not only the number of units with activation functions, but also the embedding depths of the functions

Why "shallow" not good?

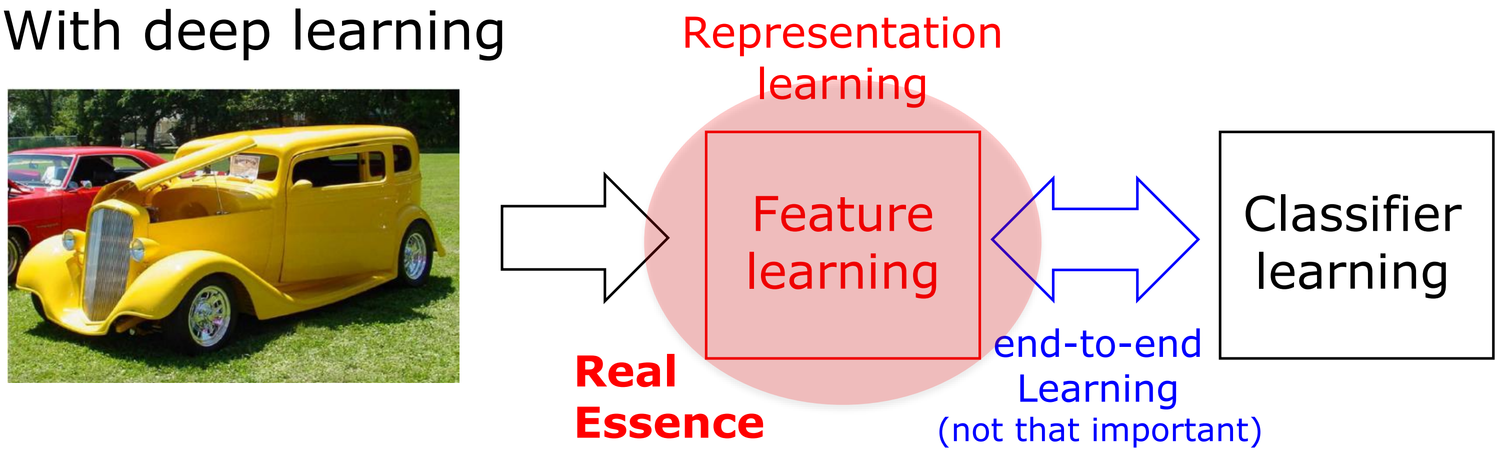
- *one-hidden-layer proved to be universal approximater*
- *complexity of one-hidden-layer can be arbitrarily high*

To think further/deeper:
What's essential with DNNs? -- Representation learning

Previously

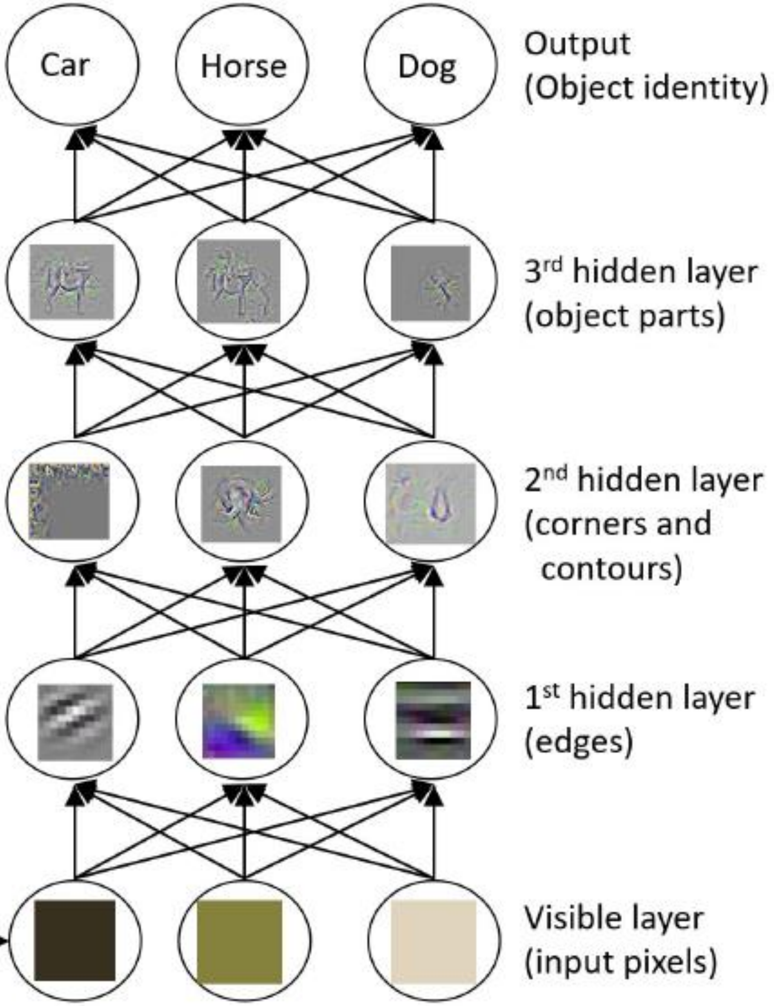


With deep learning



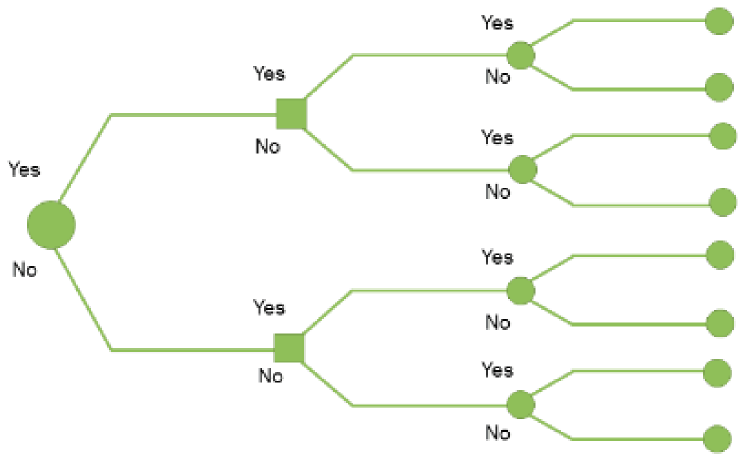
What's crucial for representation learning?

Layer-by-layer processing

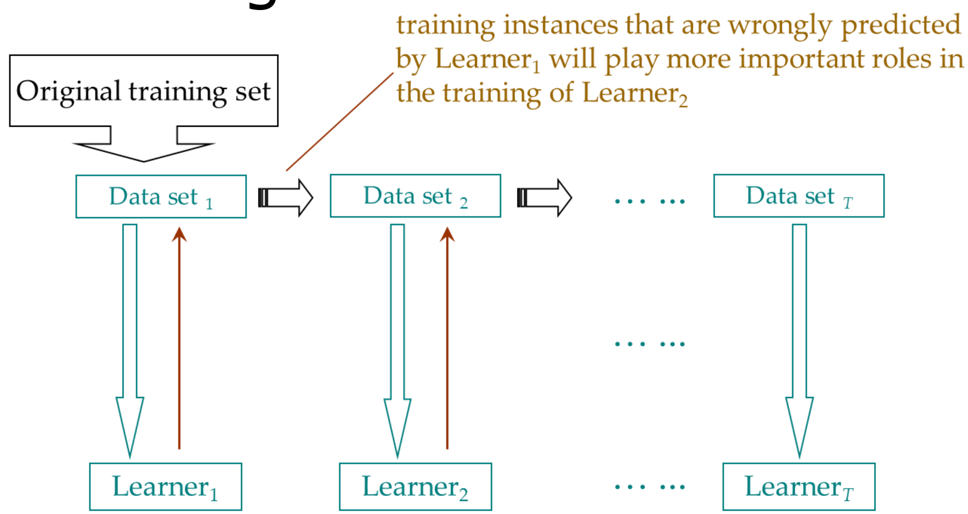


How about …

Decision trees ?



Boosting?

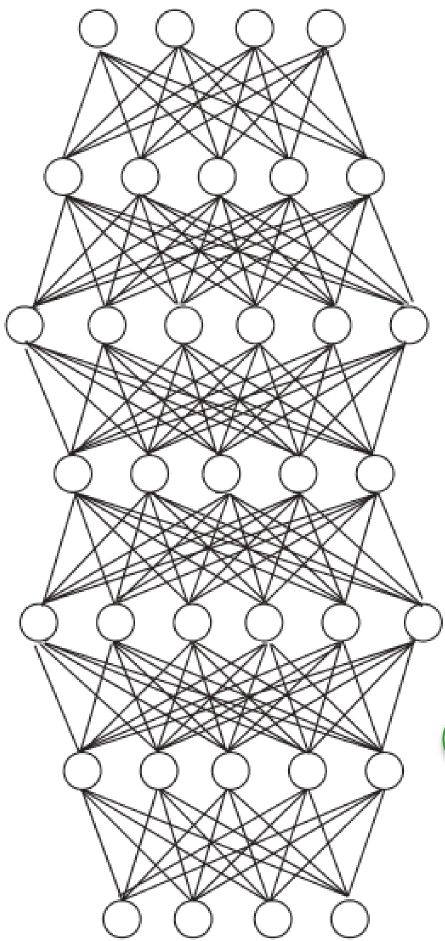


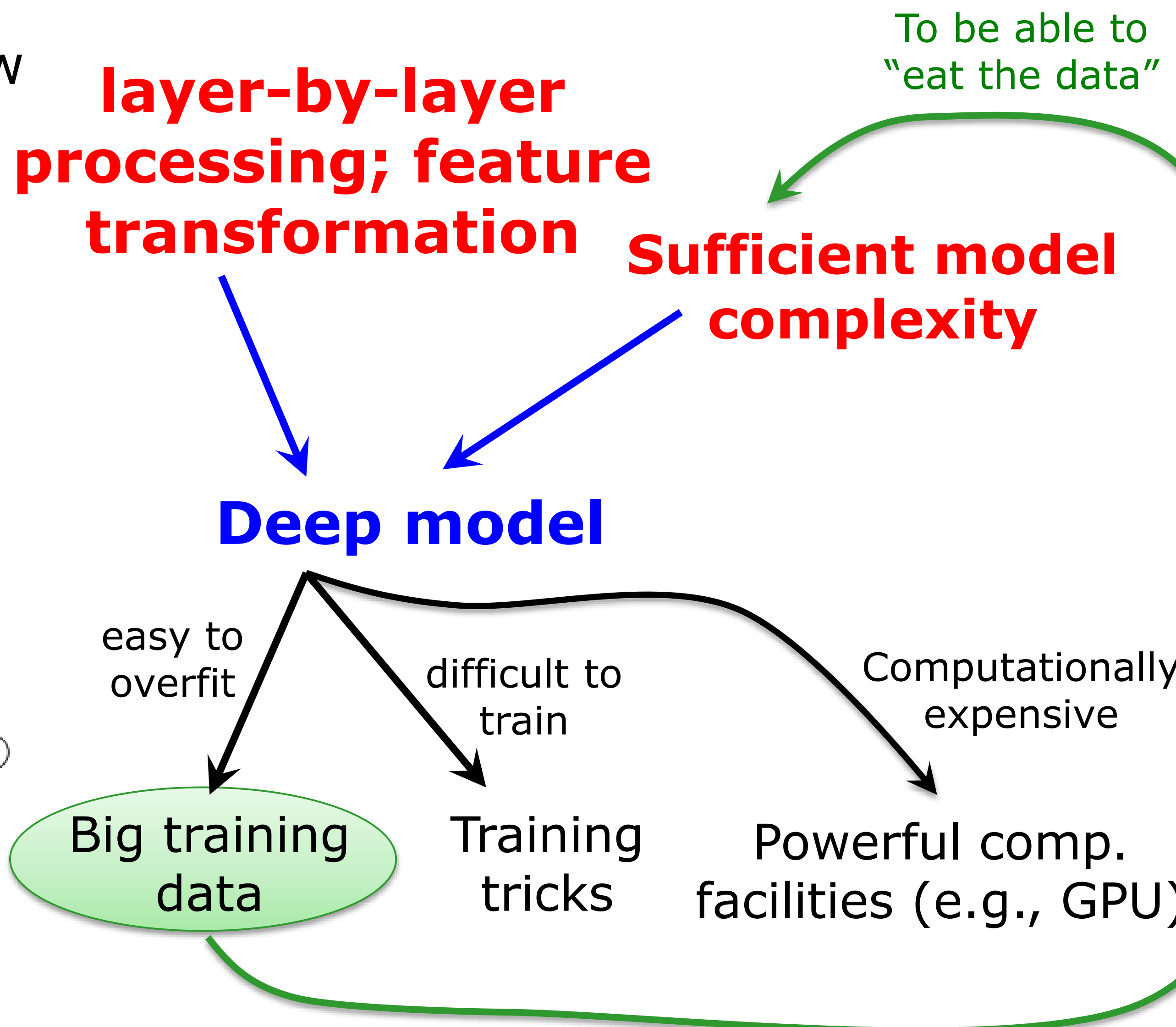
layer-by-layer processing, but …

- **insufficient complexity**
- **always on original features**

- still, **insufficient complexity**
- **always on original features**

My current view

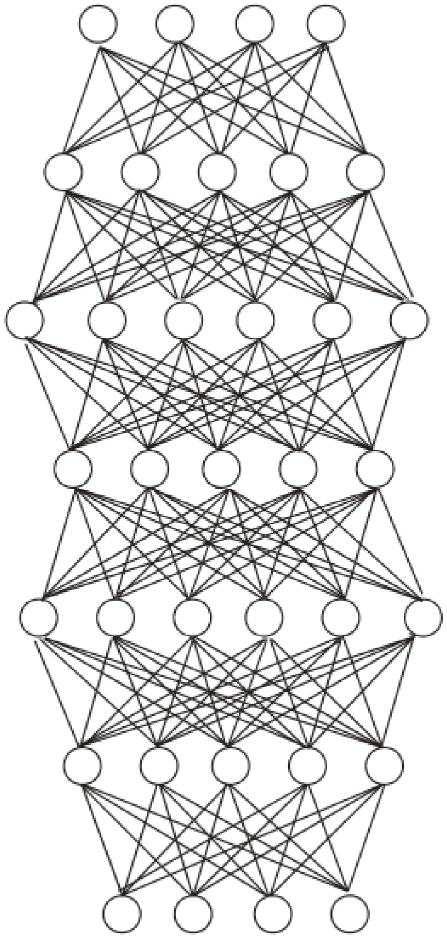




Most crucial for deep **models** :

- Layer-by-layer processing
- Feature transformation
- Sufficient model complexity

Using neural networks



- Too many hyper-parameters
  - tricky tuning, particularly when across tasks
  - Hard to repeat others' results; e.g., even when several authors all use CNNs, they are actually using different learning models due to the many different options such as convolutional layer structures
- Model complexity fixed once structure decided; usually, more than sufficient
- Big training data required
- Theoretical analysis difficult
- Blackbox
- …

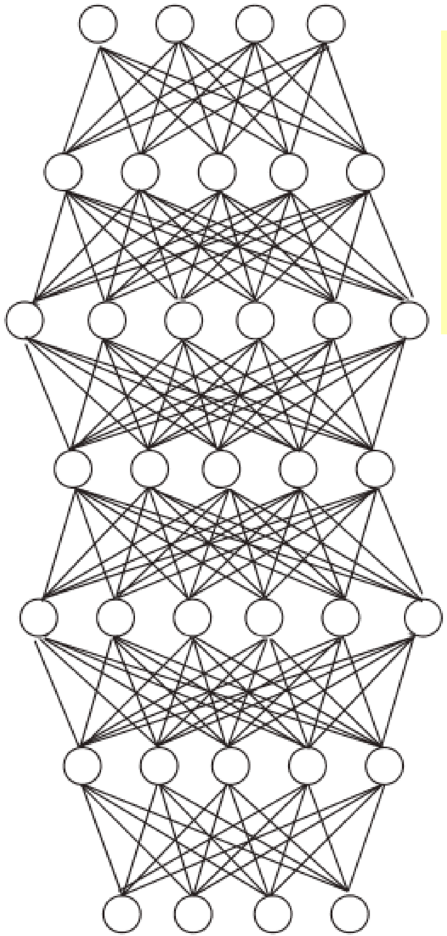
- **There are many tasks on which DNNs are not superior, sometimes even NNs inadequate**

  e.g., on many Kaggle competition tasks, Random Forest or XGBoost better

No Free Lunch !

No learning model "always" excellent

Deep models revisited



Currently, Deep Models are DNNs: multiple layers of parameterized differentiable nonlinear modules that can be trained by backpropagation

- Not all properties in the world are "differentiable", or best modelled as "differentiable"
- There are many non-differentiable learning modules (not able to be trained by backpropagation)

Can we realize deep learning with non-differentiable modules?

This is fundamental for understanding:

- Deep models ?= DNNs
- Can do DEEP with non-differentiable modules? (without backpropagation?)
- Can enable Deep model to win more tasks?
- … …

Inspiration

- To have
  - Layer-by-layer processing, and
  - Feature transformation, and
  - Sufficient model complexity

The gcForest approach

gcForest (**multi-Grained Cascade Forest**)

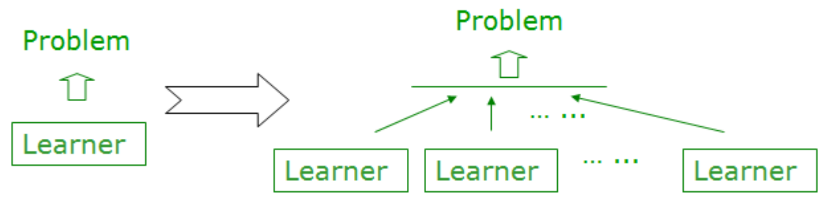
Sounds like “geek forest”

- A decision tree forest (**ensemble**) approach
- **Performance highly competitive** to DNNs across a broad range of tasks
- **Much less hyper-parameters**
  - Easier to set
  - Default setting works well across a broad range of tasks
- **Adaptive model complexity**
  - Automatically decided upon data
  - Small data applicable
- . . .

Ensemble learning

Ensemble Learning (集成学习):

Using multiple learners to solve the problem



Demonstrated great performance in real practice

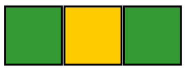
- KDDCup'07: 1st place for "... Decision Forests and ..."
- KDDCup'08: 1st place of Challenge1 for a method using Bagging; 1st place of Challenge2 for "... Using an Ensemble Method "
- KDDCup'09: 1st place of Fast Track for "Ensemble ... "; 2nd place of Fast Track for "... bagging ... boosting tree models ...", 1st place of Slow Track for "Boosting ... "; 2nd place of Slow Track for "Stochastic Gradient Boosting"
- KDDCup'10: 1st place for "... Classifier ensembling"; 2nd place for "... Gradient Boosting machines ... "
- KDDCup'11: 1st place of Track 1 for "A Linear Ensemble ... "; 2nd place of Track 1 for "Collaborative filtering Ensemble", 1st place of Track 2 for "Ensemble ..."; 2nd place of Track 2 for "Linear combination of ..."
- KDDCup'12: 1st place of Track 1 for "Combining... Additive Forest..."; 1st place of Track 2 for "A Two-stage Ensemble of..."
- KDDCup'13: 1st place of Track 1 for "Weighted Average Ensemble" ; 2nd place of Track 1 for "Gradient Boosting Machine"; 1st place of Track 2 for "Ensemble the Predictions"
- KDDCup'14: 1st place for "ensemble of GBM, ExtraTrees, Random Forest..." and "the weighted average" ; 2nd place for "use both R and Python GBMs"; 3rd place for "gradient boosting machines... random forests" and "the weighted average of..."
- KDDCup'15: 1st place for "Three-Stage Ensemble and Feature Engineering for MOOC Dropout Prediction"
- KDDCup'16: 1st place for "Gradient Boosting Decision Tree"; 2nd place for "Ensemble of Different Models for Final Prediction"
- KDDCup'17: 1st and 2nd place of Task 1 for "XGBoost"; 1st place of Task 2 for "XGBoost", 2nd place of Task 2 for "Weighted Average of Multiple Models"
- KDDCup'18: 1st place for "Gradient Boosting"; 2nd place for "Two-stage stacking"; 3rd place for "Weighted Average of Multiple Models"

During the past decade, almost all winners of KDDCup, Netflix competition, Kaggle competitions, etc., utilized ensemble techniques in their solutions

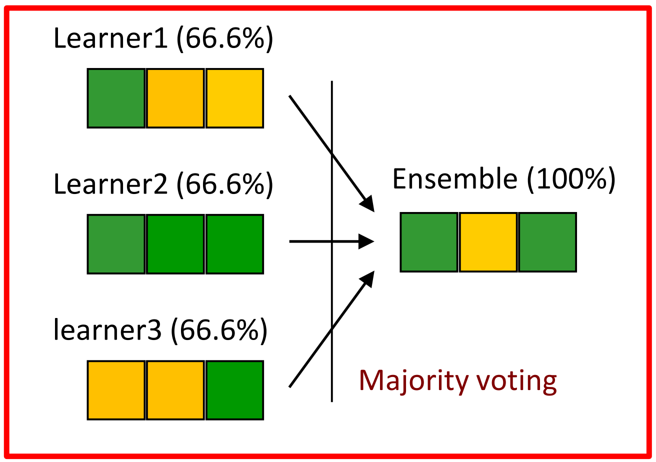
To win? Ensemble !

How to obtain a good ensemble?

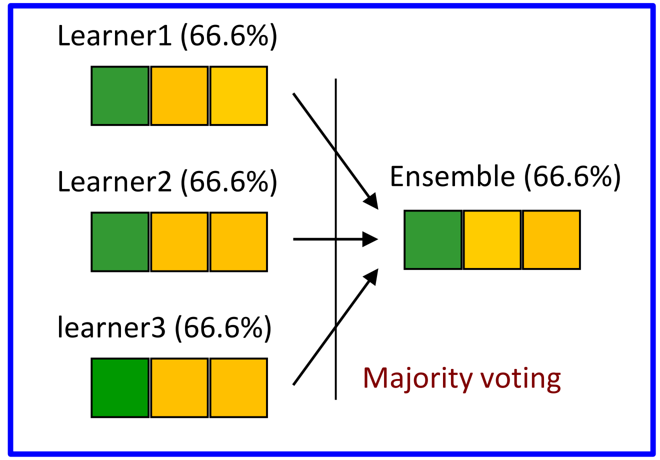
Some intuitions:



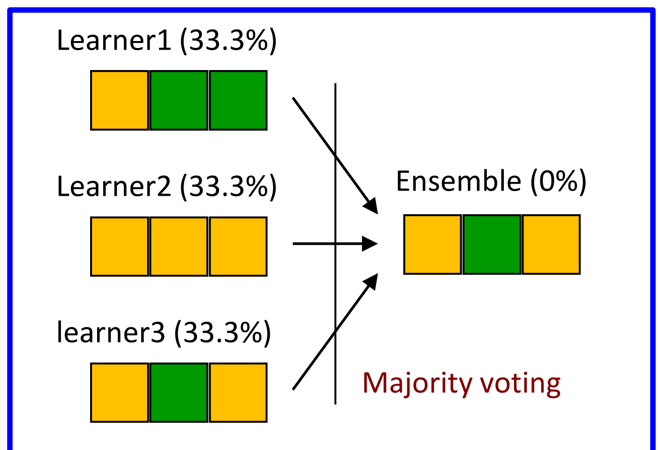
Ground-truth



Ensemble really helps



Individuals must be different



Individuals must be not-bad

Diversity is crucial

According to Error-ambiguity decomposition [Krogh & Vedelsby, NIPS'95]:

$$\boxed{E} = \bar{E} - \bar{A}$$

Ensemble error — *Ave. error of individuals* — *Ave. "ambiguity" of individuals* — *("ambiguity" later called "diversity")*

The more **accurate** and **diverse** the individual learners, the better the ensemble

However,

- the "ambiguity" does not have an operable definition
- The error-ambiguity decomposition is derivable only for regression setting with squared loss

Basic idea: To inject some randomness

Major strategies:

- **Data sample manipulation**
  e.g.,
  - bootstrap sampling in Bagging
  - importance sampling in Boosting

Strategies not always effective, e.g.,
Data sample manipulation does not work for "**stable learners**" such as linear classifiers, SVMs, etc.

- **Input feature manipulation**
  e.g.,
  - feature sampling in Random Subspace

- **Learning parameter manipulation**
  e.g.,
  - Random initialization of NN [Kolen & Pollack, NIPS'91]
  - Negative Correlation [Liu & Yao, NNJ 1999]

- **Output representation manipulation**
  e.g.,
  - ECOC [Dietterich & Bakiri, JAIR 1995]
  - Flipping Output [Breiman, MLJ 2000]

Adopt multiple strategies,
e.g.,:
- Random Forest
- FASBIR [Zhou & Yu, TSMCB 2005]

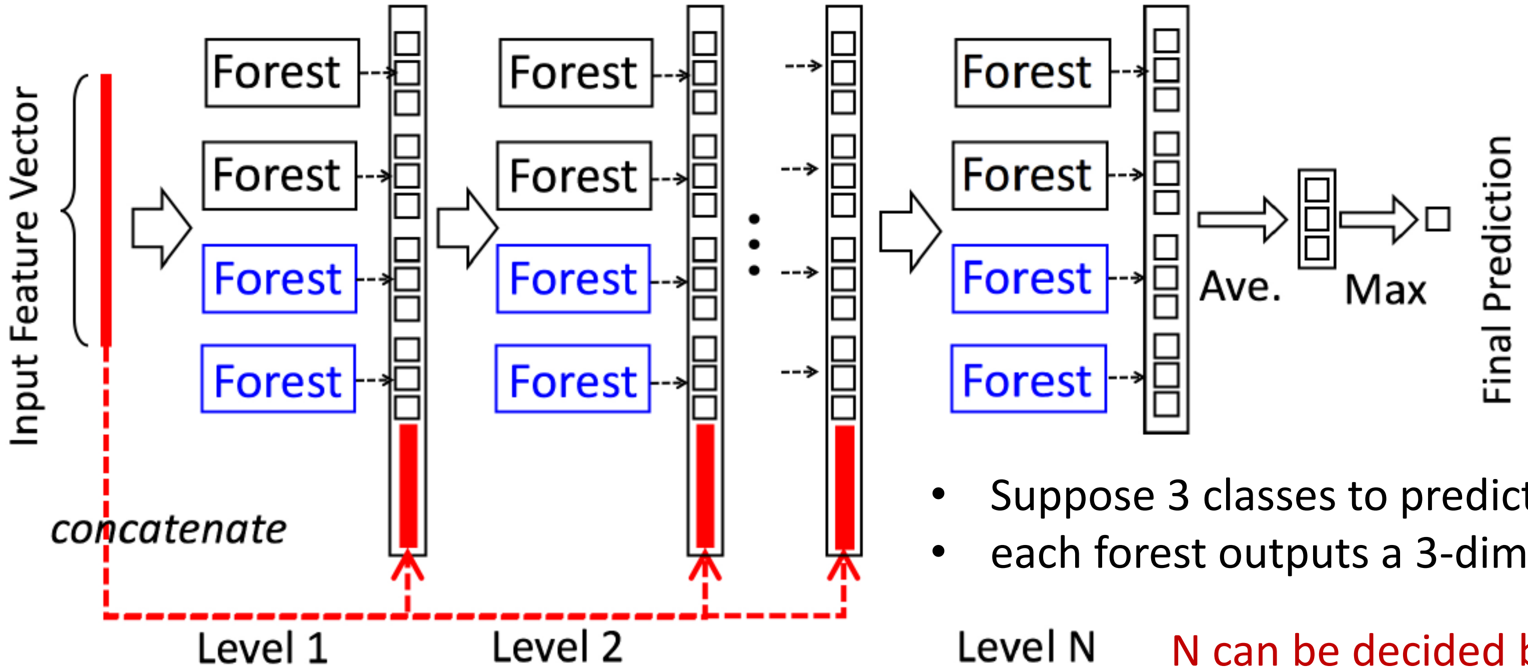
[Zhou, Ensemble book 2012, Chp5: Diversity]

the gcForest

(multi-Grained Cascade Forest)

- Cascade Forest
- Multi-grained

Cascade Forest structure

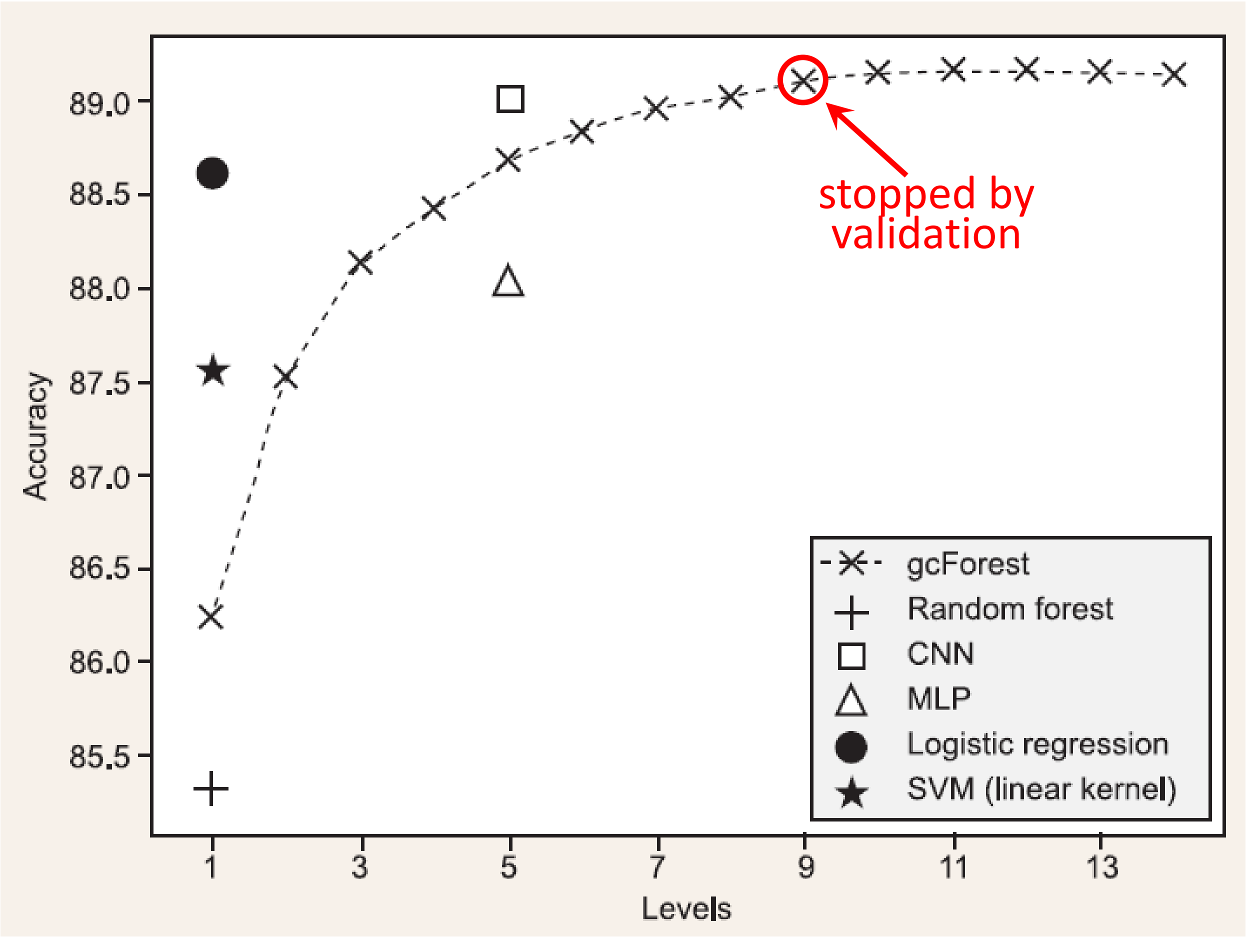


- Suppose 3 classes to predict
- each forest outputs a 3-dim **class vector**

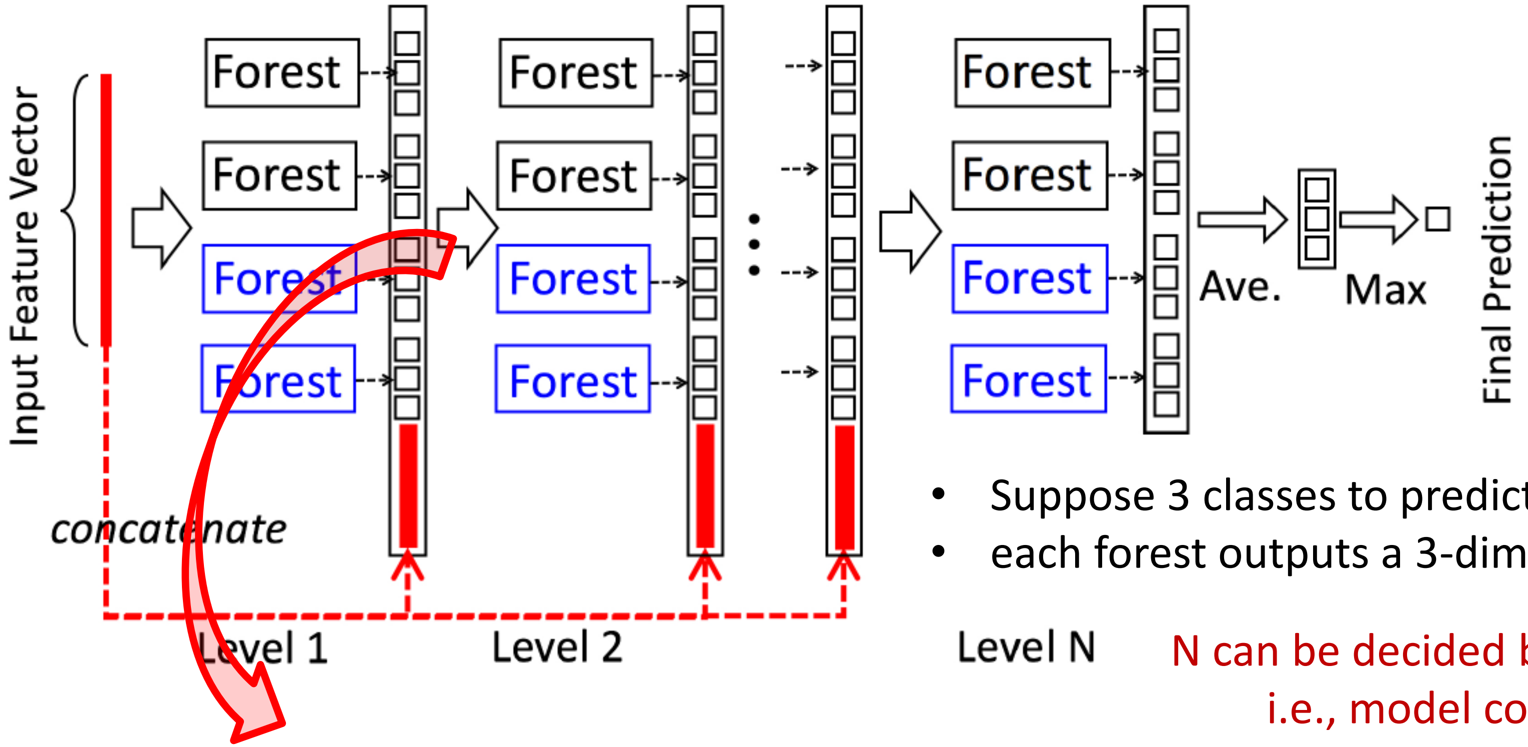
N can be decided by CV
i.e., model complexity
adaptively decided

Further experiments

Performance tendency (IMDB)



Cascade Forest structure



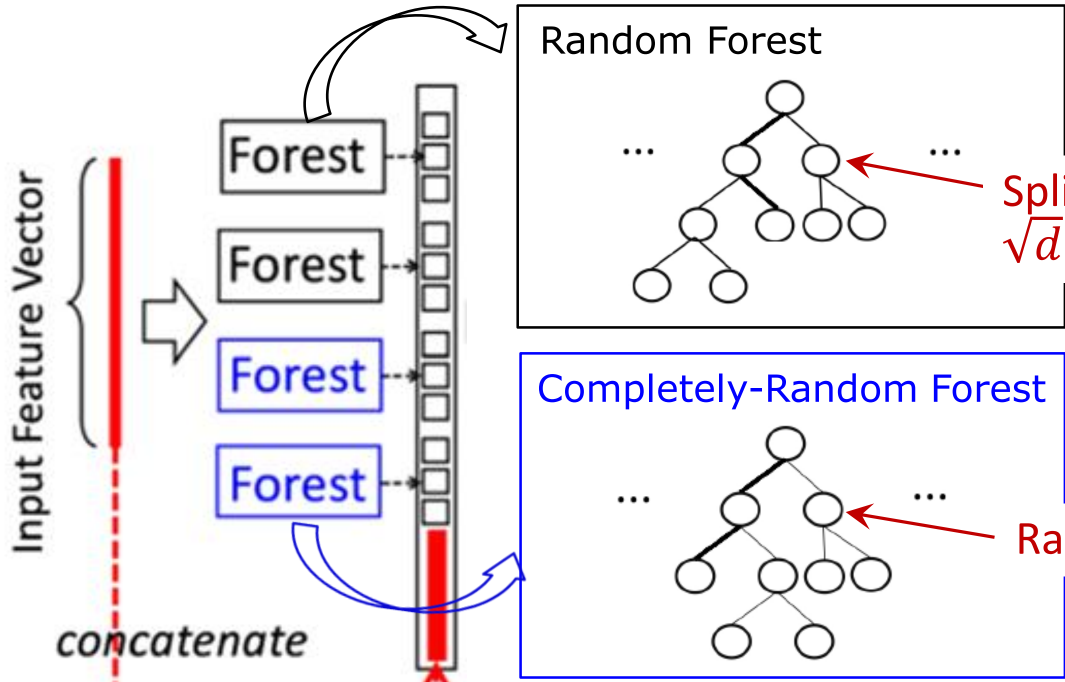
- Suppose 3 classes to predict
- each forest outputs a 3-dim **class vector**

N can be decided by CV
i.e., model complexity
adaptively decided

Passing the output of one level as input to another level:

- Related to *Stacking* [Wolpert, NNJ 1992; Breiman, MLJ 1996], a famous ensemble method
- Stacking usually one or two levels, as it is easy to overfit with more than two levels; could not enable a deep model by itself

Ensemble of ensembles



An ensemble of randomized trees

Split selection: Randomly select $\sqrt{d}$ features, then select the best

An ensemble of completely-random trees

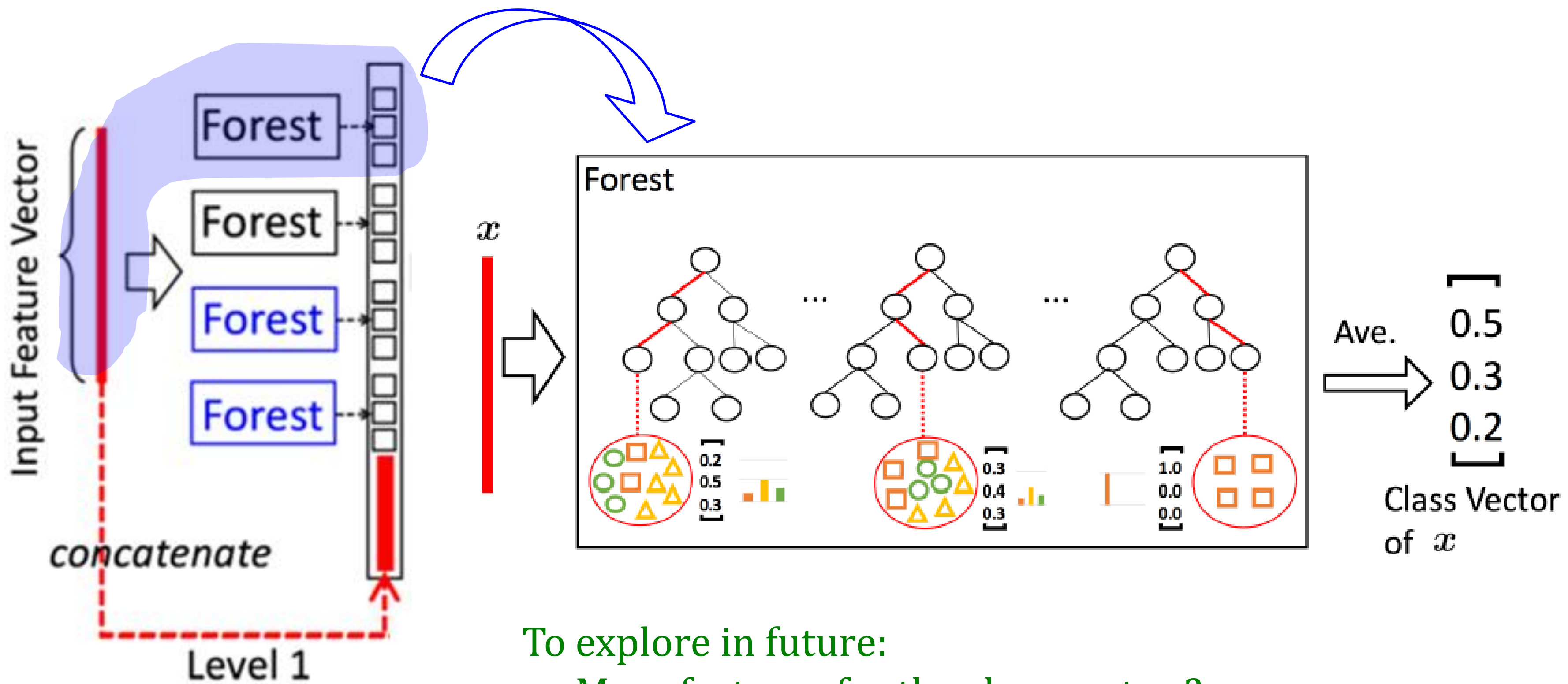
Randomly select a feature

Adopting different types of forests:
To encourage diversity in the **ensemble** of ensembles

To explore in future:

- Completely-random trees also offer the possibility of using unlabeled data

Generation of class vectors



To explore in future:

- More features for the class vector ?
  … such as parents nodes (prior distribution), sibling nodes (complement distribution), decision patch encoding, …

the gcForest

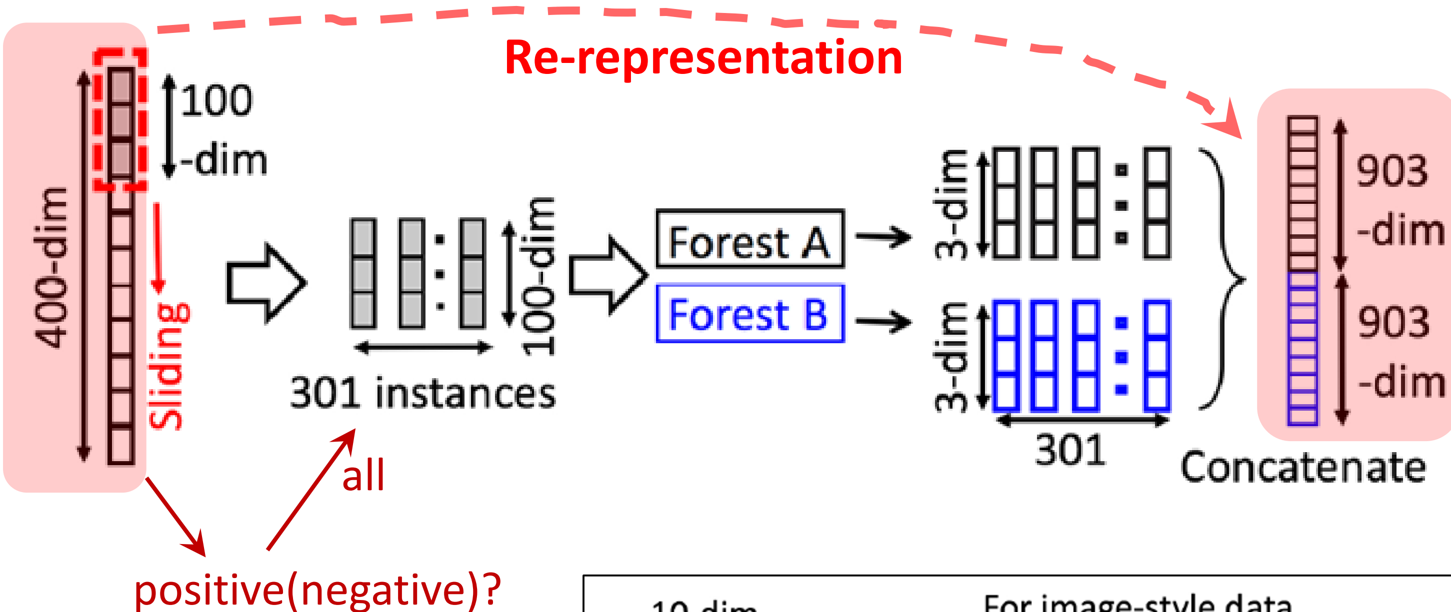
(multi-Grained Cascade Forest)

- Cascade Forest
- Multi-grained

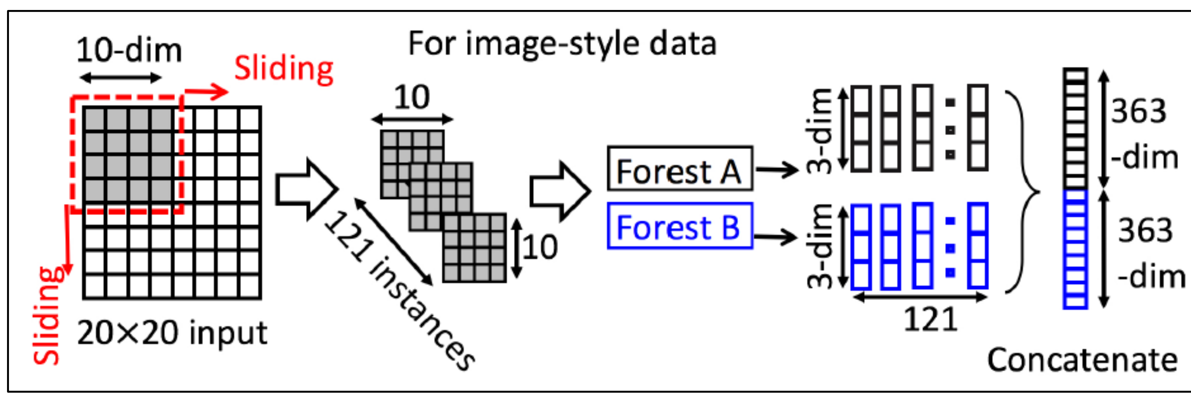
LAMDA
Learning And Mining from DatA
http://lamda.nju.edu.cn

Sliding window scanning

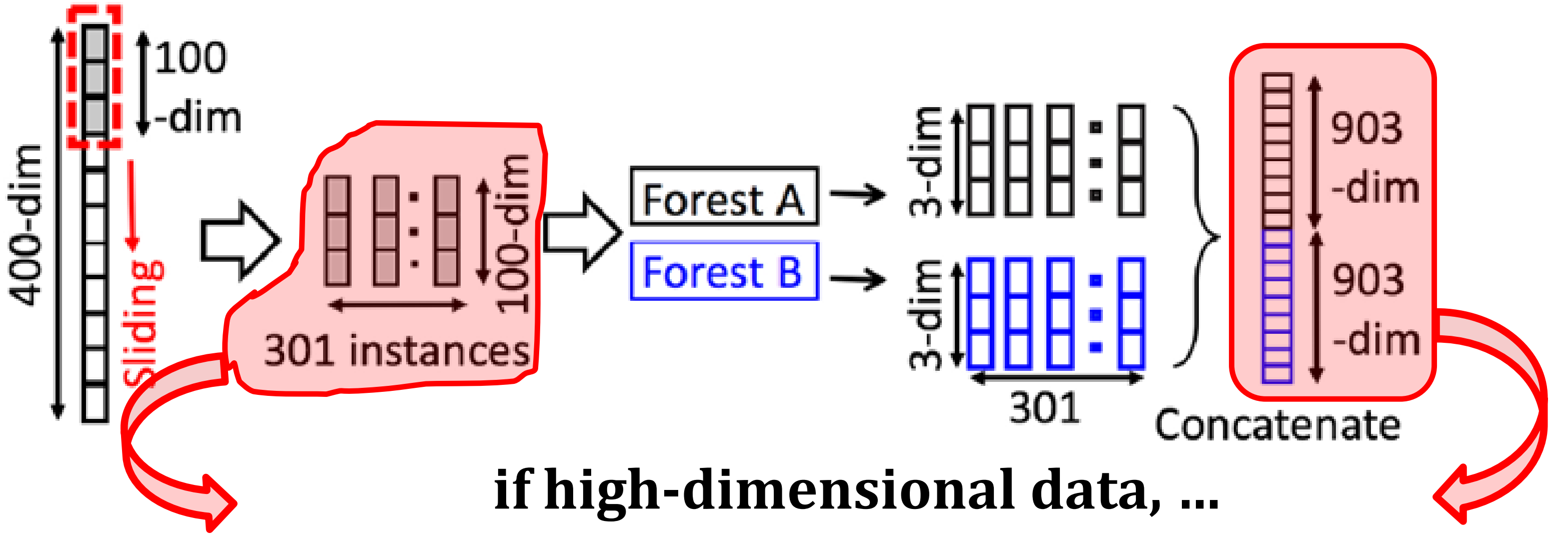
Inspired by: CNNs/RNNs exploit spatial/sequential relationships



Related to *Flipping Output* [Breiman, MLJ 2000], an output manipulation approach to encourage ensemble diversity



Sliding window scanning (con't)



if high-dimensional data, ...

Too many instances, too long vector to hold?

Feature sampling

(e.g., by subsampling the instances)

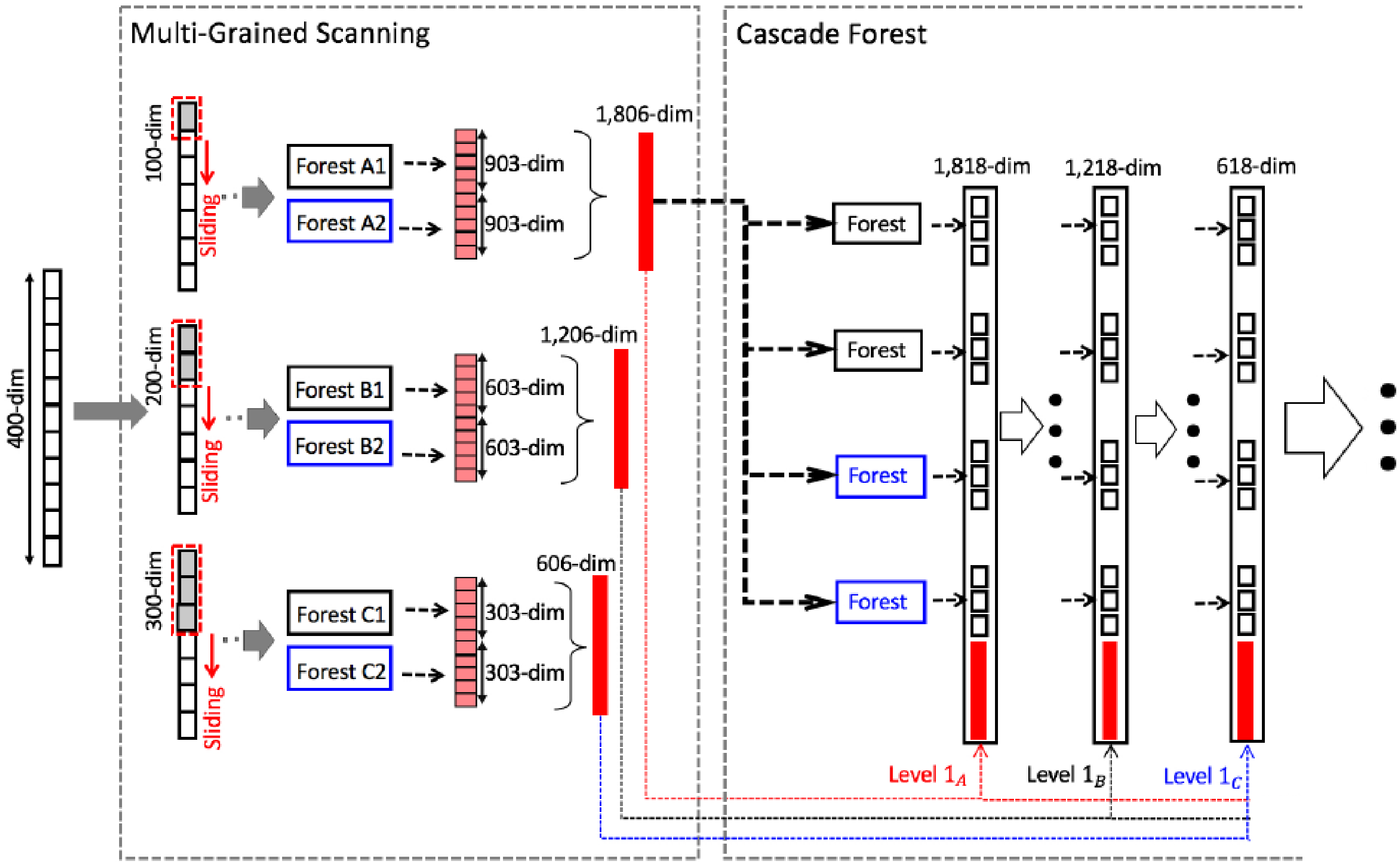
Related to *Random Subspace* [Ho, TPAMI 1998], a feature manipulation approach to encourage ensemble diversity

- **completely-random trees** not rely on feature split selection
- **random forest** quite insensitive to split feature perturbation

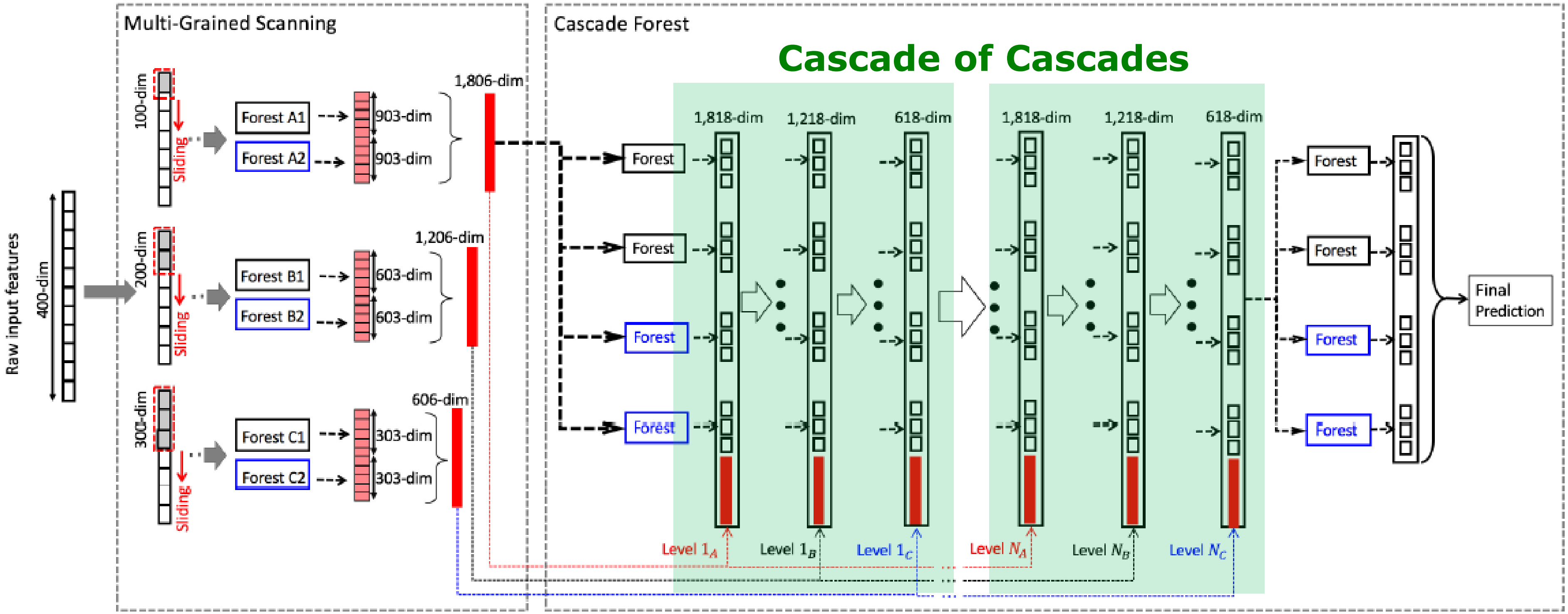
To explore in future:

- Smart sampling, feature hashing, etc.

Multi-grains → Multiple grades per level



Overall architecture



For grained scanning:

- 500 trees per forest
- Tree growth: till pure leaf, or depth =100
- Sliding window size $\lfloor d/16 \rfloor$, $\lfloor d/8 \rfloor$, $\lfloor d/4 \rfloor$

For cascade:

- 500 trees per forest
- Tree growth: till pure leaf

Deep forest results

- Non-differentiable building blocks, not rely on BP
- Much less hyper-parameters than DNNs → easier to train
- Model complexity decided upon data → applicable to small data
- Performance competitive to DNNs on a broad range of tasks

Hyper-parameters

Learning And Mining from DatA
http://lamda.nju.edu.cn

Table 1: Summary of hyper-parameters and default settings. Boldfont highlights hyper-parameters with relatively larger influence; "?" indicates default value unknown, or generally requiring different settings for different tasks.

Deep neural networks (e.g., convolutional neural networks)	gcForest
Type of activation functions: Sigmoid, ReLU, tanh, linear, etc. Architecture configurations: **No. Hidden layers**: ? **No. Nodes in hidden layer**: ? **No. Feature maps**: ? **Kernel size**: ? Optimization configurations: **Learning rate**: ? Dropout: {0.25/0.50} **Momentum**: ? **L1/L2 weight regularization penalty**: ? Weight initialization: Uniform, glorot_normal, glorot_uniform, etc. Batch size: {32/64/128}	Type of forests: Completely-random tree forest, random forest, etc. Forest in multi-grained scanning: **No. Forests**: {2} **No. Trees in each forest**: {500} Tree growth: till pure leaf, or reach depth 100 **Sliding window size**: $\{\lfloor d/16 \rfloor, \lfloor d/8 \rfloor, \lfloor d/4 \rfloor\}$ Forest in cascade: **No. Forests**: {8} **No. Trees in each forest**: {500} Tree growth: till pure leaf

In Experiments:

- **gcForest uses the same hyper-parameters for all data**
- **DNNs carefully tune per dataset**

http://cs.nju.edu.cn/zhouzh/

Experimental results

Learning And Mining from DatA
http://lamda.nju.edu.cn

Image categorization (MNIST)

gcForest	**99.26%**
LeNet-5	99.05%
Deep Belief Net	98.75% [Hinton *et al.*, 2006]
SVM (rbf kernel)	98.60%
Random Forest	96.80%

Face recognition (ORL)

	5 image	7 images	9 images
gcForest	**91.00%**	**96.67%**	**97.50%**
Random Forest	91.00%	93.33%	95.00%
CNN	86.50%	91.67%	95.00%
SVM (rbf kernel)	80.50%	82.50%	85.00%
kNN	76.00%	83.33%	92.50%

http://cs.nju.edu.cn/zhouzh/

Experimental results

Learning And Mining from DatA
http://lamda.nju.edu.cn

Music classification (GTZAN)

gcForest	**65.67%**
CNN	59.20%
MLP	58.00%
Random Forest	50.33%
Logistic Regression	50.00%
SVM (rbf kernel)	18.33%

Sentiment classification (IMDB)

gcForest	**89.16%**
CNN	89.02% [Kim, 2014]
MLP	88.04%
Logistic Regression	88.62%
SVM (linear kernel)	87.56%
Random Forest	85.32%

Hand movement recognition (sEMG)

gcForest	**71.30%**
LSTM	45.37%
MLP	38.52%
Random Forest	29.62%
SVM (rbf kernel)	29.62%
Logistic Regression	23.33%

Low-dimensional data (features: 16, 14, 8)

	LETTER	ADULT	YEAST
gcForest	**97.40%**	**86.40%**	**63.45%**
Random Forest	96.50%	85.49%	61.66%
MLP	95.70%	85.25%	55.60%

http://cs.nju.edu.cn/zhouzh/

Deep forest results

- Non-differentiable building blocks, not

Hyper-parameters

Learning And Mining from DatA
http://lamda.nju.edu.cn

Table 1: Summary of hyper-parameters and default settings. Boldfont highlights hyper-parameters with relatively larger influence; "?" indicates default value unknown, or generally requiring different settings for different tasks.

Deep neural networks (e.g., convolutional neural networks)	gcForest
Type of activation functions:	Type of forests:
Sigmoid, ReLU, tanh, linear, etc.	Completely-random tree forest, random forest, etc.
Architecture configurations:	Forest in multi-grained scanning:
No. Hidden layers: ?	**No. Forests**: {2}
No. Nodes in hidden layer: ?	**No. Trees in each forest**: {500}

This is the first deep learning model which is NOT based on NNs and which does NOT rely on BP

	5 image	7 images	9 images
gcForest	**91.00%**	**96.67%**	**97.50%**
Random Forest	91.00%	93.33%	95.00%
CNN	86.50%	91.67%	95.00%
SVM (rbf kernel)	80.50%	82.50%	85.00%
kNN	76.00%	83.33%	92.50%

gcForest	**71.30%**
LSTM	45.37%
MLP	38.52%
Random Forest	29.62%
SVM (rbf kernel)	29.62%
Logistic Regression	23.33%

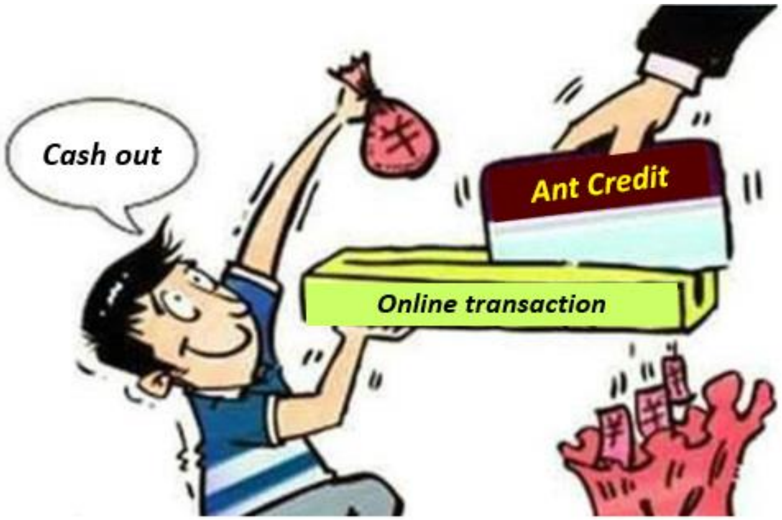
Low-dimensional data (features: 16, 14, 8)

	LETTER	ADULT	YEAST
gcForest	**97.40%**	**86.40%**	**63.45%**
Random Forest	96.50%	85.49%	61.66%
MLP	95.70%	85.25%	55.60%

http://cs.nju.edu.cn/zhouzh/

An industrial application to illegal cash-out detection

Very serious, particularly when considering the big amount of online transactions per day



For example, in 11.11 2016, more than 100 millions of transactions paid by *Ant Credit Pay*

Big loss even if only a very small portions were fraud

Results

Table 1: The number of the training and test samples.

	# Pos. Ins.	# Neg. Ins.	# All Ins.
Train	171,784	131,235,963	131,407,704
Test	66,221	52,423,308	52,489,529

More than 5,000 features per transaction, categorical/numeric (details are business confidential)

Evaluation with common metrics

	AUC	F1	KS
LR	0.9887	0.4334	0.8956
DNN	0.9722	0.3861	0.8551
MART	0.9957	0.5201	0.9424
gcForest	**0.9970**	**0.5440**	**0.9480**

Evaluation with specified metrics

	1/10000	1/1000	1/100
LR	0.3708	0.5603	0.8762
DNN	0.3165	0.4991	0.8471
MART	0.4661	0.6716	0.9358
gcForest	**0.4880**	**0.6950**	**0.9470**

1/100 means that 1/100 of all transactions are interrupted

Deep forest performs much better than others

However, not to expect too much immediately

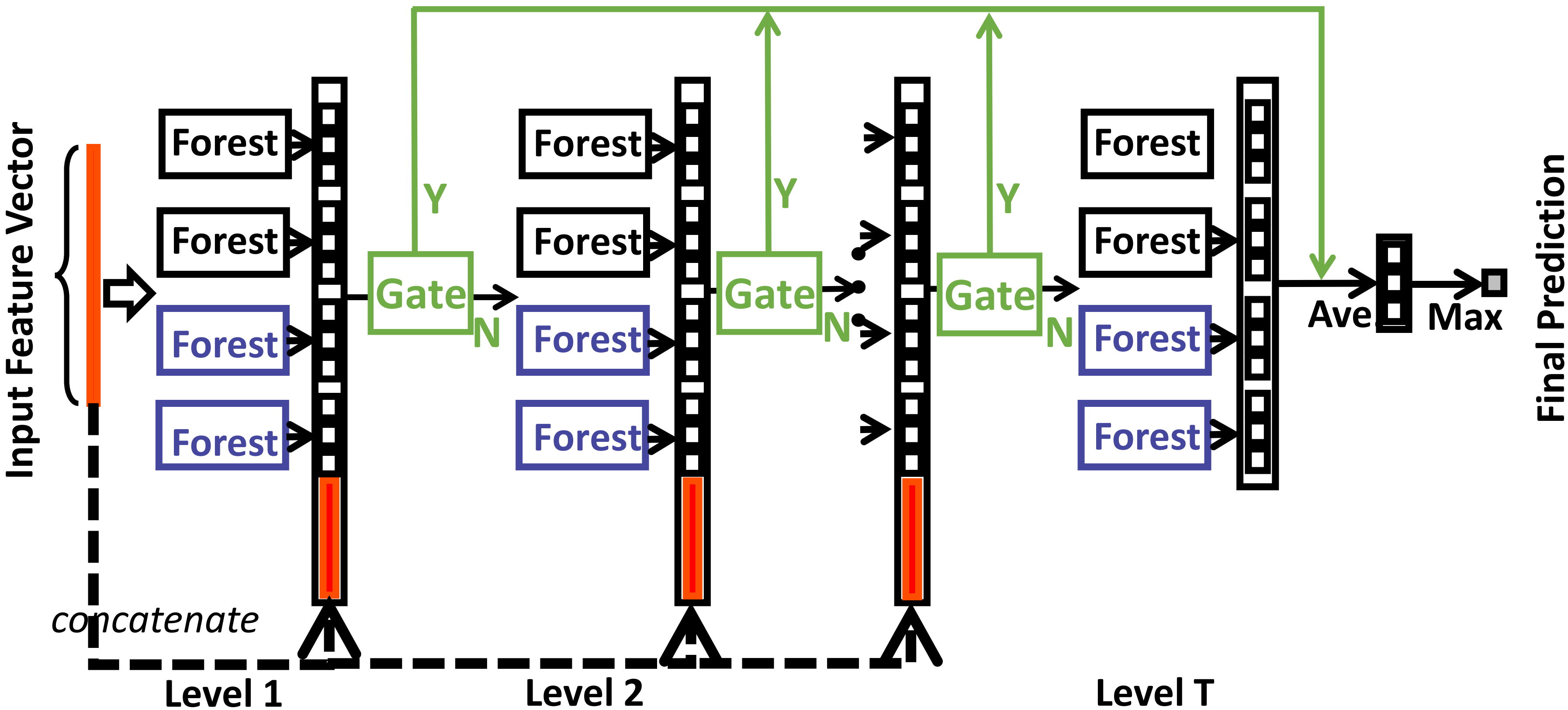
New tech usually has a long way to go

Recent improvements/variants

-- Confidence Screening

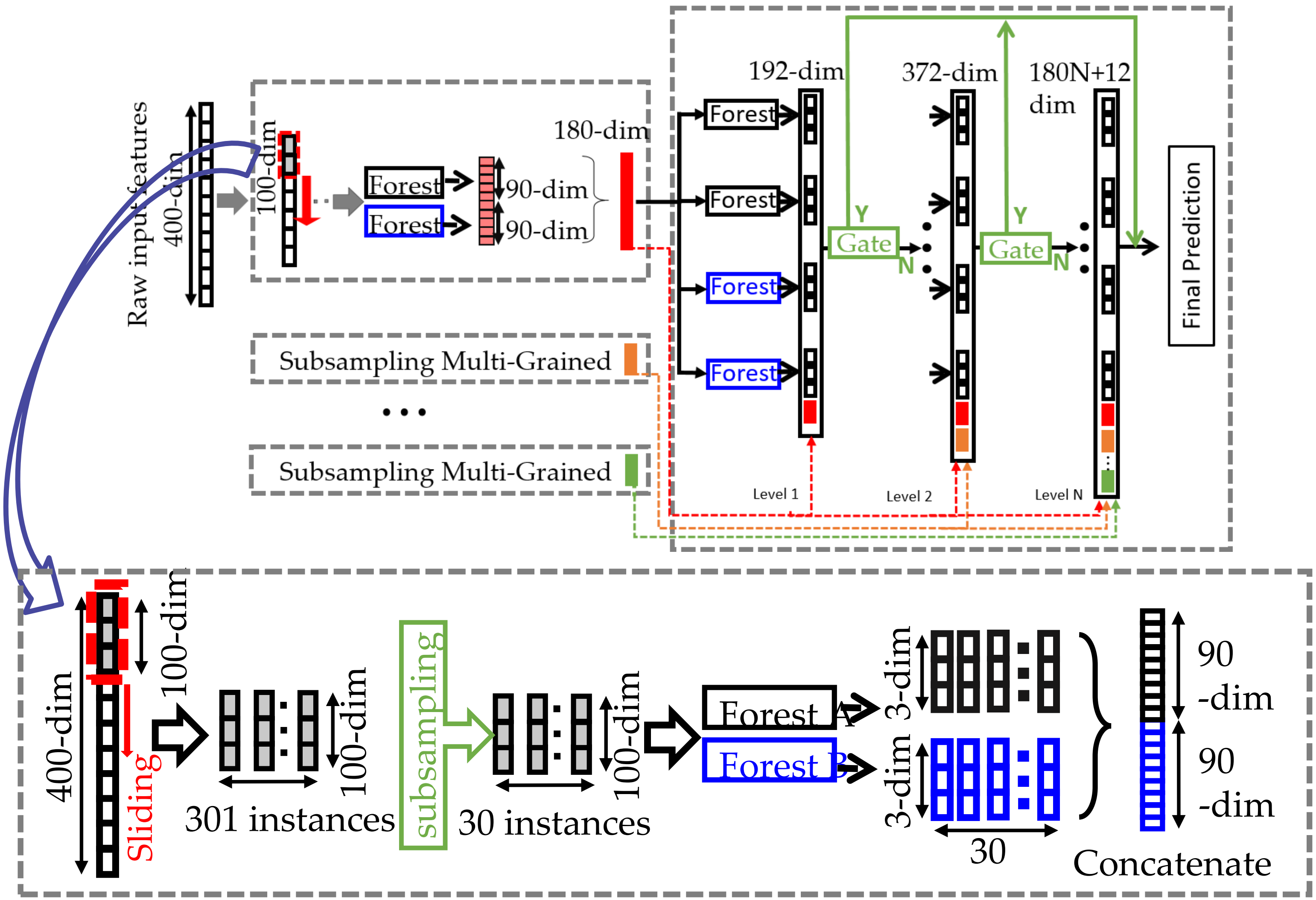
Confidence Screening

Confidence screening: passing the instances with high confidence directly to the final stage



[Pang, Ting, Zhao & Zhou, ICDM 2018]

gcForest$_{CS}$: Overall architecture



Results with multi-grained scanning

Datasets	Method	Accuracy (%)	Training time (s)	Test time (s)	Memory (M)
sEMG	gcForest$_{cs}$	72.59	1548	77	4348
	gcForest	71.30	34324	2288	41789
MNIST	gcForest$_{cs}$	99.26	1061	10	4997
	gcForest	99.26	27840	464	50518
CIFAR-10	gcForest$_{cs}$	62.62	13342	667	6875
	gcForest	61.78	63068	2102	73826

gcForest$_{cs}$ achieves comparable or even better results with an order of magnitude less cost

[Pang, Ting, Zhao & Zhou, ICDM 2018]

Results without multi-grained scanning

Datasets	Method	Accuracy (%)	Training time (s)	Test time (s)	Memory (M)
LETTER	gcForest_{cs}	97.08	75	2	915
	gcForest	97.08	86	3	4526
ADULT	gcForest_{cs}	86.11	95	7	648
	gcForest	86.06	199	12	3002
IMDB	gcForest_{cs}	89.57	1623	32	1992
	gcForest	89.20	11633	152	3750

gcForest_{cs} achieves comparable or even better results with less memory requirement and time cost

Recent improvements/variants

-- mdDF (*optimal Margin Distribution* Deep Forest)

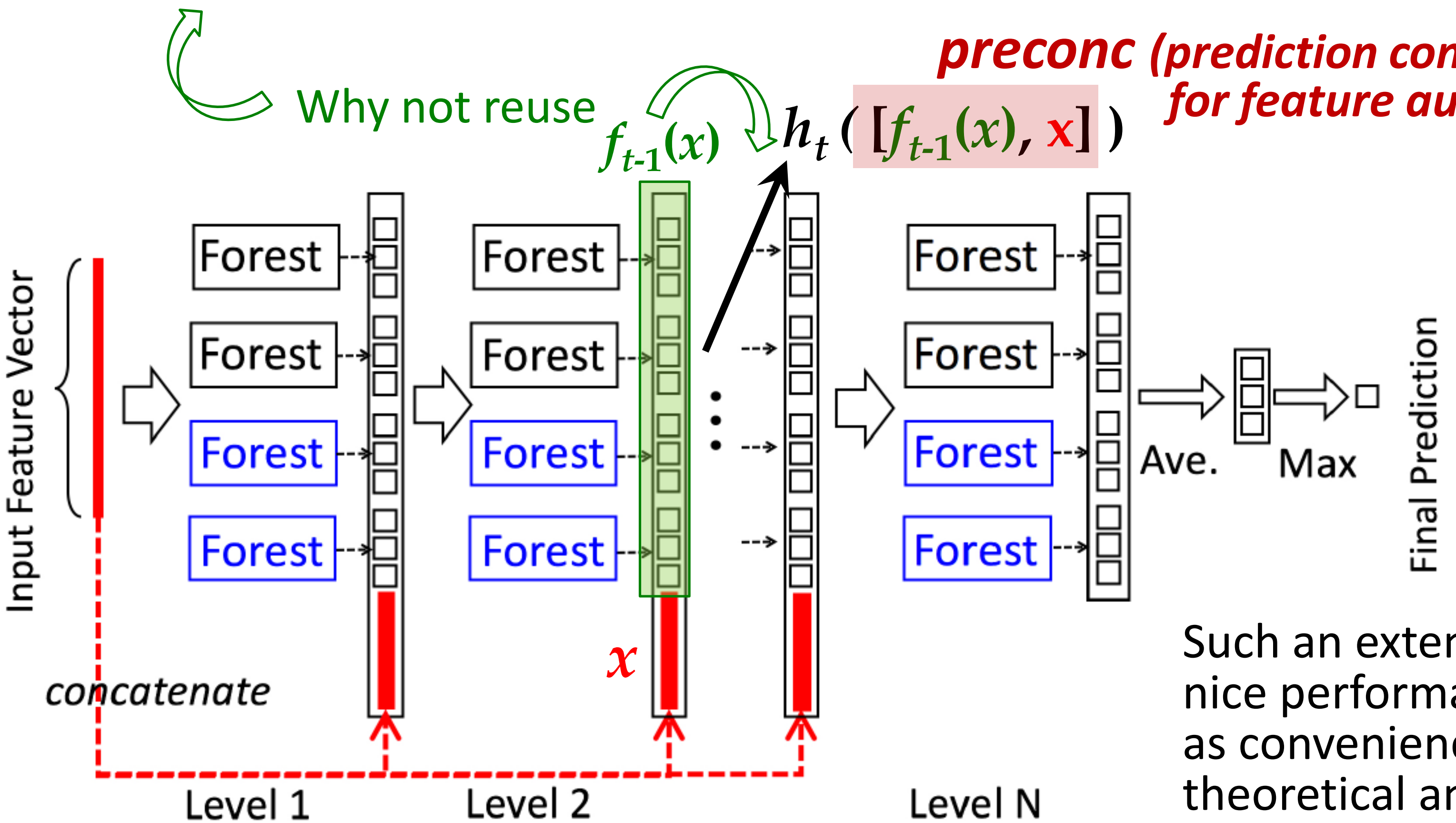
LAMDA Learning And Mining from DatA
http://lamda.nju.edu.cn

Reform each layer as additive model

$$f_t(x) = \begin{cases} h_1(x, w_1) & t = 1, \\ \alpha_t h_t([x, f_{t-1}(x)], w_t) + f_{t-1}(x) & t > 1. \end{cases}$$

- w_t is sample weight
- $w_1 = [1/m, \ldots, 1/m]$

***preconc** (prediction concatenation)*
for feature augmentation



Such an extension brings nice performance as well as convenience for theoretical analysis

Experiments

Learning And Mining from DatA
http://lamda.nju.edu.cn

Dataset	Attribute	Instance	Feature	Class
ADULT	Categorical	48842	14	2
YEAST	Categorical	1484	8	10
LETTER	Categorical	20000	16	26
PROTEIN	Categorical	24387	357	3
HAR	Mixed	10299	561	6
SENSIT	Mixed	78823	50	3
SATIMAGE	Numerical	6435	36	6
MNIST	Numerical	70000	784	10

Dataset	MLP	R.F.	XGBoost	gcForest	mdDF
ADULT	80.597	85.566	85.591	•86.276	**86.560**
YEAST	59.641	61.833	58.969	•63.004	**64.120**
LETTER	96.025	96.575	95.850	•97.375	**97.500**
PROTEIN	68.660	67.996	•71.696	71.590	**71.757**
HAR	•94.231	92.569	93.112	94.224	**94.600**
SENSIT	78.957	80.133	81.849	•82.334	**82.534**
SATIMAGE	91.125	91.200	90.450	•91.700	**91.750**
MNIST	**98.621**	96.831	97.730	98.252	•98.440
Avg. Rank	3.750	4.000	3.750	2.375	1.125

the best

2nd best

Theoretical result

Theorem 1. *Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$ and S be a sample of m examples chosen independently at random according to $\mathcal{D}$. With probability at least $1-\delta$, for $\theta > 0$, the strong classifier $F(x)$ (depth-T mdDF) satisfies that*

$$\Pr_D[yF(x) < 0] \leq \frac{1}{m^{50}} +$$

$$\inf_{\theta \in (0,1]} \left[\hat{R} + \frac{1}{m^d} + \frac{3\sqrt{\mu}}{m^{3/2}} + \frac{7\mu}{3m} + \sqrt{\frac{3\mu}{m} \left(\frac{\hat{V}_m[yF(x)]}{\mathbb{E}_S^2[yF(x)]} \right)} \right]$$

Related to the rate between ***margin variance*** and ***margin mean***, implying "shaper" margin distribution (with **smaller margin variance** and **larger margin mean**) lead to better generalization

where

$$\hat{R} = \Pr_S[yF(x) < \theta],$$

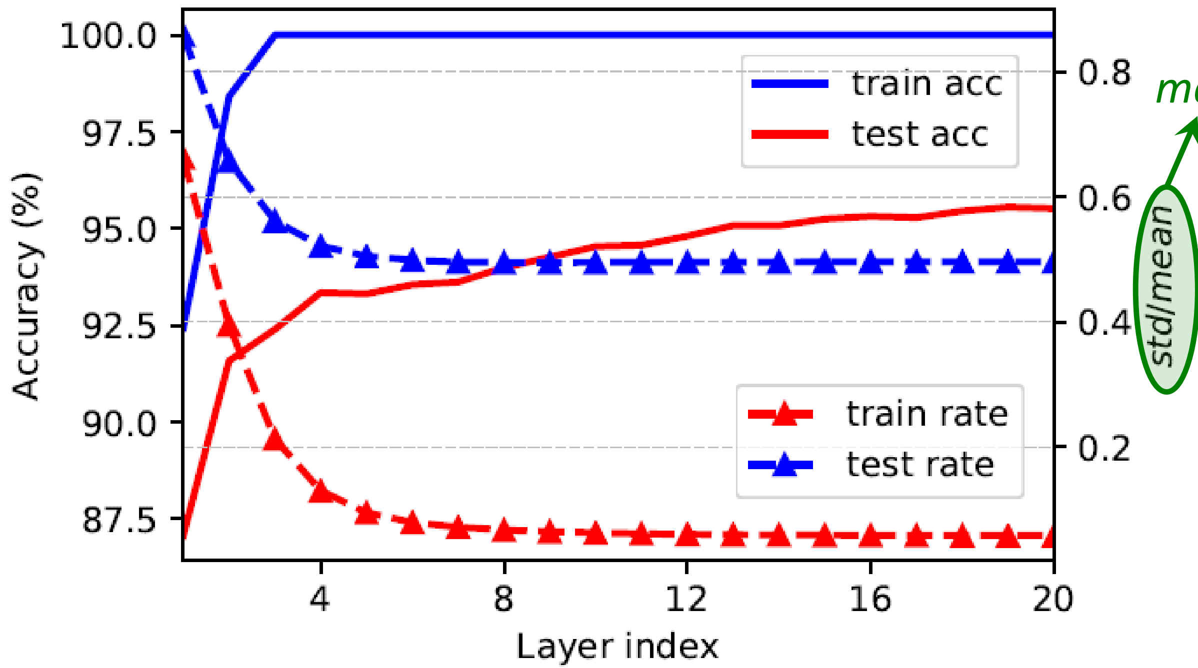
$$d = \frac{2}{1 - \mathbb{E}_S^2[yF(x)] + \theta/9},$$

$$\mu = 144 \ln m \ln(2 \sum_{t=1}^{T} \alpha_t |H_t|)/\theta^2 + \ln \left(\frac{2 \sum_{t=1}^{T} \alpha_t |H_t|}{\delta} \right),$$

$$\hat{V}_m[yF(x)] = \mathbb{E}_S[(yF(x))^2] - \mathbb{E}_S^2[yF(x)].$$

Accuracy vs. Margin rate

HAR data (10299 instances, 561 features, 6 classes)



margin rate = $\frac{\textit{margin std.}}{\textit{margin mean}}$

Experiments consistent with theoretical results:
Smaller margin rate → better generalization

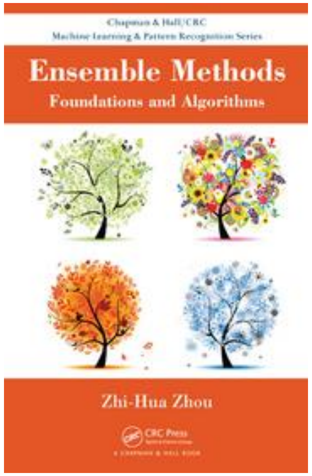
Challenges/Open Problems

Challenges/Open Problems

-- Diversity

gcForest is a success of ensemble methods

- "**Diversity**" is crucial for ensembles
- gcForest utilizes almost all kinds of strategies for diversity enhancement



Z.-H. Zhou. [Ensemble Methods: Foundations and Algorithms](), Boca Raton, FL: Chapman & Hall/CRC, Jun. 2012. (ISBN 978-1-439-830031)

Diversity

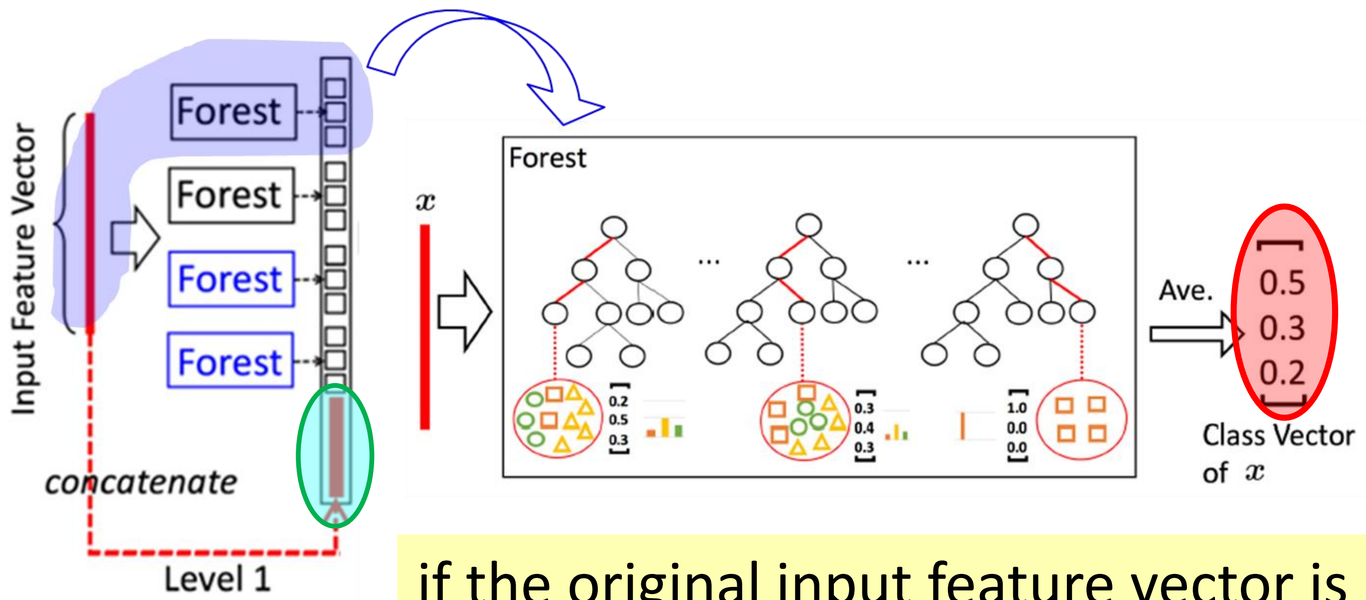
During training process,

- **Deep NN**: to avoid **Gradient vanishing**
- **Deep Forest**: to avoid **Diversity vanishing**
  - **It is a fundamental challenge to maintain sufficient *diversity* to enable DF to go deeper**
  - Tricks currently inspired by *ensemble methods; more fresh ones?*

Challenges/Open Problems

-- Feature Augmentation

Feature Augmentation



if the original input feature vector is high-dim
→ the 3-bit class vector easy to be drown out

It is fundamental to extract helpful enriched features from forests

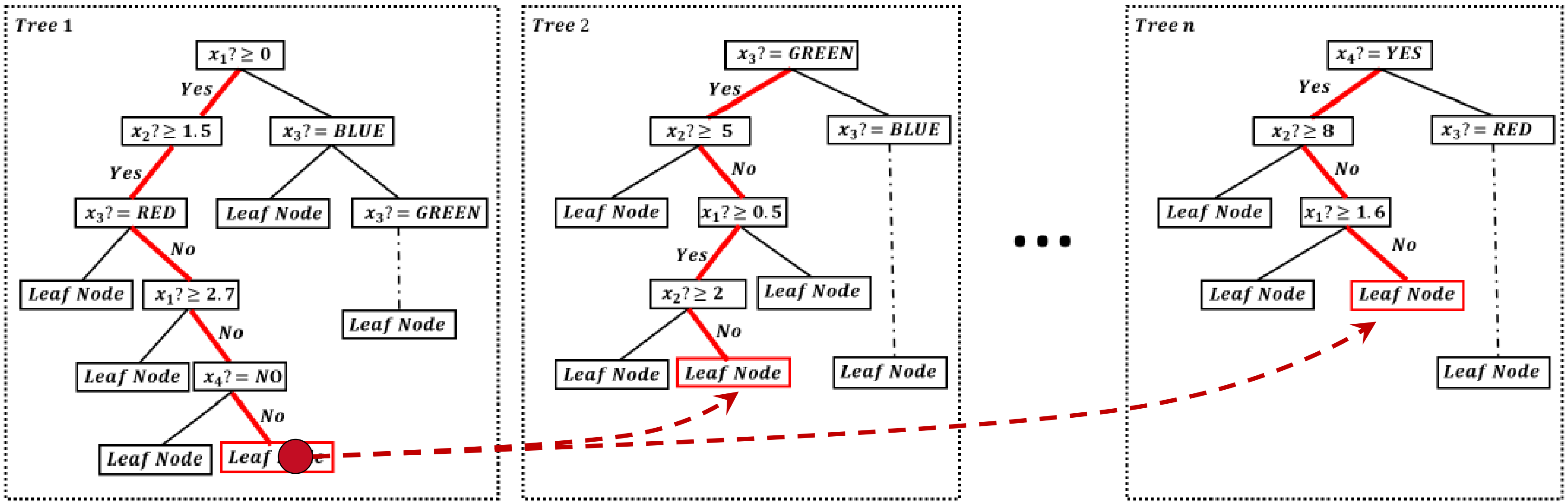
Can Forest offer sufficient information?

Learning And Mining from DatA
http://lamda.nju.edu.cn

Forest contains rich information

A trained forest can even be used as AutoEncoder

- Unknown before
- AutoEncoder was thought as special property of NNs



$$(x_1 \geq 0) \wedge (x_2 \geq 1.5) \wedge \neg(x_3 == RED) \wedge \neg(x_1 \geq 2.7) \wedge \neg(x_4 == NO)$$

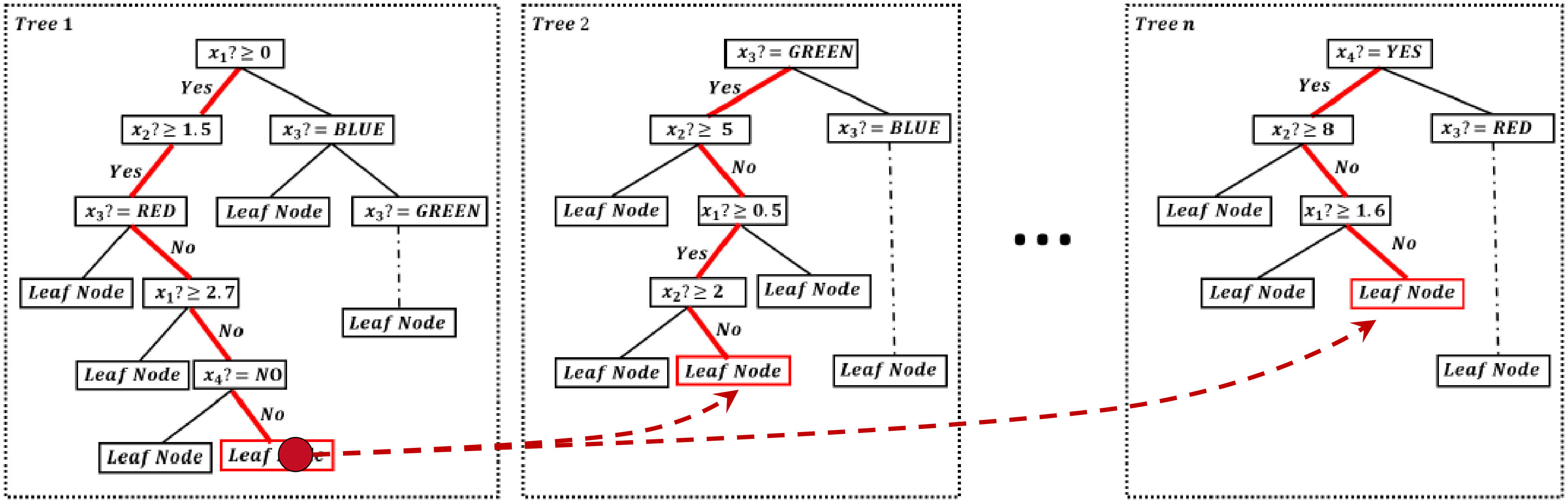
$$(x_3 == GREEN) \wedge \neg(x_2 \geq 5) \wedge (x_1 \geq 0.5) \wedge \neg(x_2 \geq 2)$$

$$\cdots$$

$$(x_4 == YES) \wedge \neg(x_2 \geq 8) \wedge \neg(x_1 \geq 1.6)$$

[Feng & Zhou, AAAI 2018]

Forest contains rich information



$(x_1 \geq 0) \wedge (x_2 \geq 1.5)$
$\wedge \neg(x_3 == RED)$
$\wedge \neg(x_1 \geq 2.7)$
$\wedge \neg(x_4 == NO)$

$(x_3 == GREEN) \wedge \neg(x_2 \geq 5)$
$\wedge (x_1 \geq 0.5) \wedge \neg(x_2 \geq 2)$

...

$(x_4 == YES) \wedge \neg(x_2 \geq 8)$
$\wedge \neg(x_1 \geq 1.6)$

...

MCR (Maximal-Compatible Rule):

$$(1.6 \geq x_1 \geq 0.5) \wedge (2 \geq x_2 \geq 1.5)$$
$$\wedge (x_3 == GREEN) \wedge (x_4 == YES)$$

[Theorem] The original sample must reside in the input region defined by the MCR.

Experimental results of Forest AutoEncoder

Learning And Mining from DatA
http://lamda.nju.edu.cn

Performance comparison (MSE)

	MNIST	CIFAR-10
MLP_1	266.85	1284.98
MLP_2	163.97	1226.52
CNN-AE	768.02	865.63
SWW-AE	159.8	590.76
$eForest^s_{500}$	1386.96	1623.93
$eForest^s_{1000}$	701.99	567.64
$eForest^u_{500}$	27.39	579.337
$eForest^u_{1000}$	**6.86**	**153.68**

Directly applicable to Text data (e.g., IMDB)

	Cosine Distance
$eForest^s_{500}$	0.1132
$eForest^s_{1000}$	0.0676
$eForest^u_{500}$	0.0070
$eForest^u_{1000}$	**0.0023**

For text data, NNs AutoEncoder require the help of additional mechanism such as word2vec

Thus, there seems rich possibilities to design better feature augmentation scheme based on forests

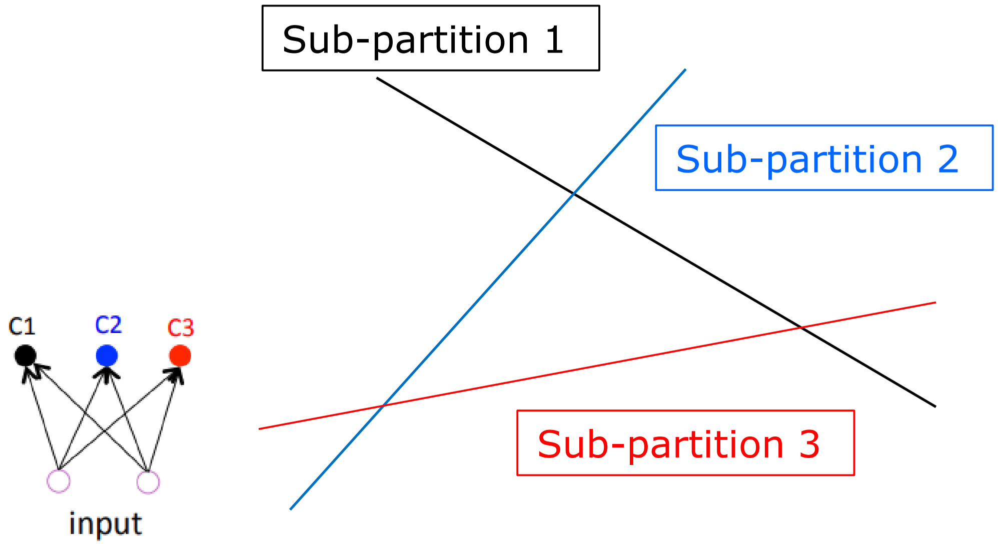
**In addition to the AutoEncoder ability,
Forest also possesses other abilities that were
believed to be special for NNs, e.g. →**

Distributed representation learning

A forest can do distributed representation learning

- Unknown before
- It was thought as special property of NNs

"distributed representation learning is critical" [Bengio et al., PAMI13]



Number of distinguishable regions grows almost exponentially with number of parameters

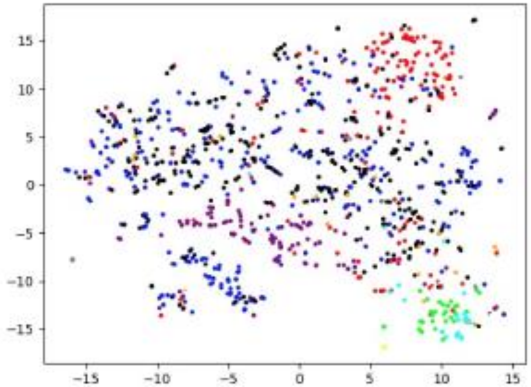
Each parameter influences many regions instead of just local neighbors

… …

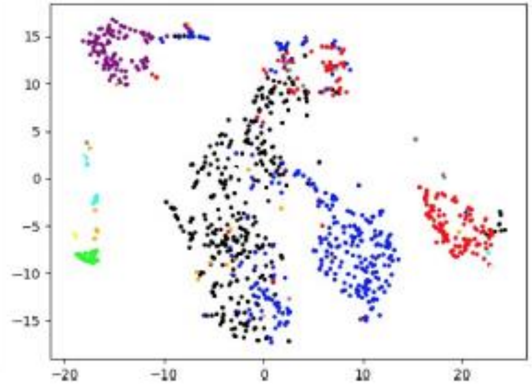
LAMDA
Learning And Mining from DatA
http://lamda.nju.edu.cn

Visualization results

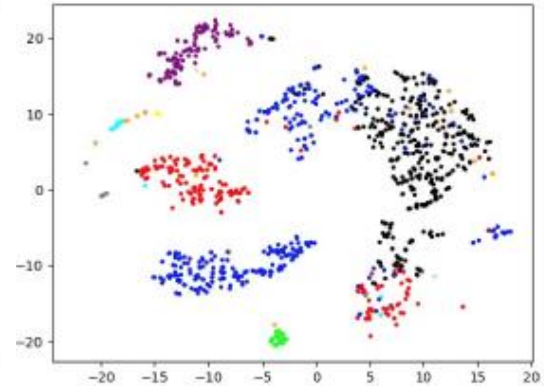
protein **dataset: original**



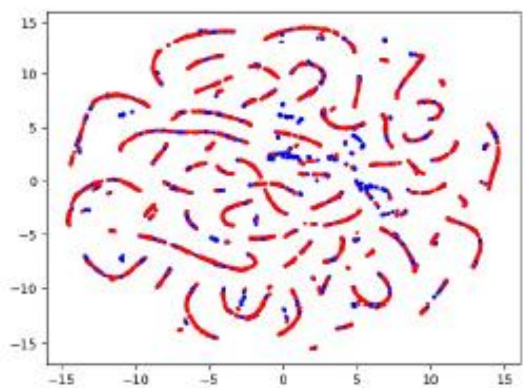
1st-layer representation



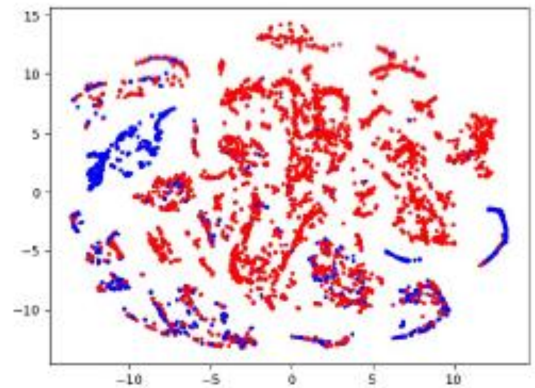
2nd-layer representation



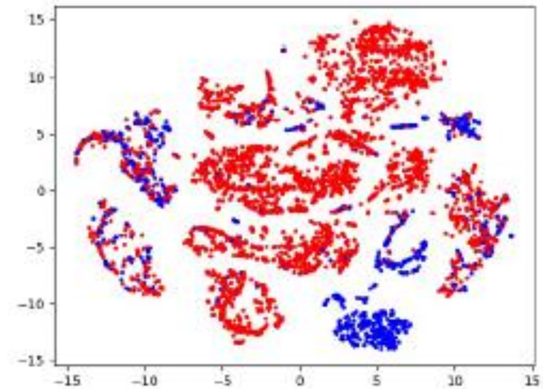
income **dataset: original**



1st-layer representation



2nd-layer representation



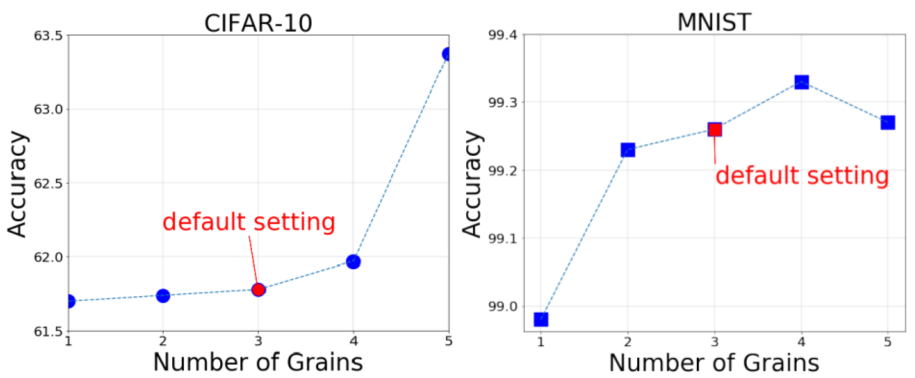
mGBDT setting: 5 trees added per GBDT per epoch; maximum tree depth 5

Challenges/Open Problems

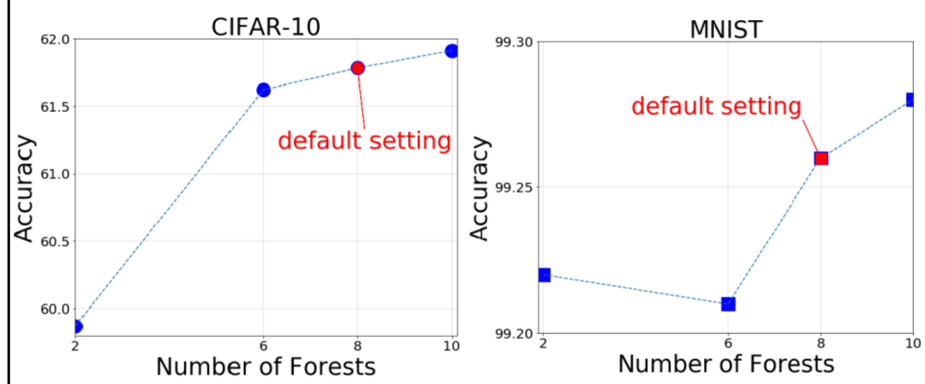
-- Hardware Speedup

Larger models tend to be better

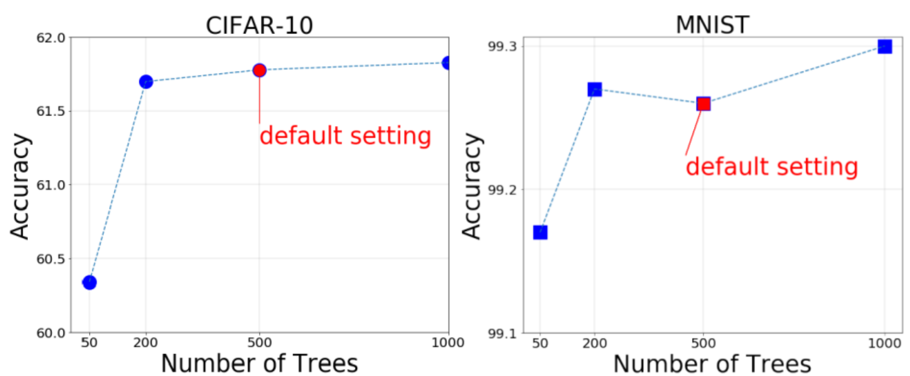
with increasing number of grains



with increasing number of forests per grade



with increasing number of trees per forest



Larger model might tend to offer better performances

But, currently we could not do larger

Computational facilities crucial for training larger models

e.g., GPUs for DNNs.

Hardware

Computational cost: DF training < DNN training

However,

- DNN gets great speedup by GPU
- DF naturally unsuited to GPU

If GPU speedup counted, DNN even more efficient

Can DF get speedup from suitable hardware?

*Can **KNL** (or some other architecture) do for DF as GPU for DNN ?*

Challenges/Open Problems

-- Algorithms

Algorithms

- DNNs have been studied for almost 30 years

  e.g., CNN/LSTM developed in 1990s

  with contributions from millions of researchers/ practitioners

- **DF still infant**

  Better algorithms to be developed

Challenges/Open Problems

-- Theory

Theory

One of the most serious deficiency with Deep Learning is the lack of theoretical foundation

- DNNs hard for theoretical analysis
- DF seems better

mdDF gets some preliminary theoretical results

Still not easy, more investigation required

Effort worthwhile

No Free Lunch !

No learning model "always" superior

Our conjecture:

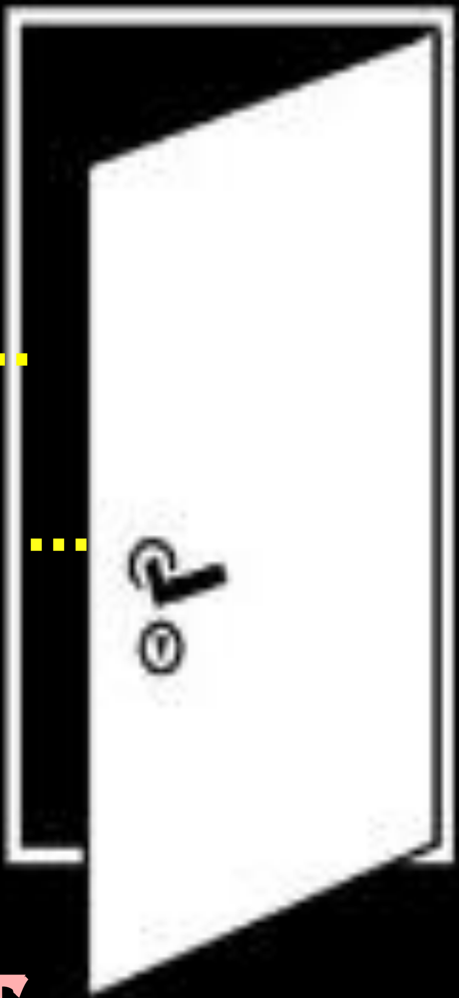
- **Numerical modeling → DNNs**
  e.g., image/vision data
- **Non-numerical modeling → DF ?**
  e.g., symbolic/discrete/tabular data

DEEP LEARNING ROOM

Deep neural networks

Deep Forest ... ...

... ...



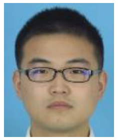
THE IS JUST A START

For details

- ✓ **Z.-H. Zhou and J. Feng. Deep Forest. National Science Review, 2019** (doi.org/10.1093/nsr/nwy108)
- ✓ Z.-H. Zhou and J. Feng. Deep forest: Towards an alternative to deep neural networks. In: IJCAI'17
  Code: http://lamda.nju.edu.cn/code_gcForest.ashx (for small- or medium-scale data)
- ✓ J. Feng and Z.-H. Zhou. AutoEncoder by forest. In: AAAI'18
  Code: http://lamda.nju.edu.cn/code_eForest.ashx
- ✓ J. Feng, Y. Yu and Z.-H. Zhou. Multi-layered gradient boosting decision tree. In: NeuIPS'18
  Code: http://lamda.nju.edu.cn/code_mGBDT.ashx
- ✓ M. Pang, K. M. Ting, P. Zhao and Z.-H. Zhou. Improving deep forest by confidence screening. In: ICDM'18
  Code: http://lamda.nju.edu.cn/code_gcForestCS.ashx
- ✓ Y.-L. Zhang, J. Zhou, W. Zheng, J. Feng, L. Li, Z. Liu, M. Li, Z. Zhang, C. Chen, X. Li, and Z.-H. Zhou. Distributed Deep Forest and its Application to Automatic Detection of Cash-out Fraud. arXiv 1805.04234.
- ✓ S.-H. Lv and Z.-H. Zhou. Forest representation learning guided by margin distribution. In preparation.

Joint work with my current students:


J. Feng
(冯霁)


S.-H. Lv
(吕沈欢)


M. Pang
(庞明)


P. Zhao
(赵鹏)

and many collaborators
and ex-students (above)

Thanks