Learning Sparsity in Neural Networks and Robust Statistical Analysis

Yanwei Fu, Yikai Wang, Xinwei Sun
School of Data Science
Fudan University

Yuan Yao
HKUST

Wotao Yin
DAMO Academy
Alibaba

https://sparse-learning.github.io
Learning Sparsity in Neural Networks and Robust Statistical Analysis
Lecture 1:

Yanwei Fu
School of Data Science
Fudan University
http://yanweifu.github.io
Motivation
Overview of

Sparse and Low-Rank Recovery of Data

ECCV2012 tutorial:

Tutorial on Sparse and Low-rank Modeling, European Conference on Computer Vision, Firenze, Italy, October 2012

https://sparse-learning.github.io
Overview of

Sparse and Low-Rank Recovery of Data

ECCV2012 tutorial:

Tutorial on Sparse and Low-rank Modeling, European Conference on Computer Vision, Firenze, Italy, October 2012

Our CVPR2022 tutorial:

https://sparse-learning.github.io
Overview of Sparse and Low-Rank Recovery of Data

ECCV2012 tutorial:

Tutorial on Sparse and Low-rank Modeling, European Conference on Computer Vision, Firenze, Italy, October 2012

Our CVPR2022 tutorial:

Learning Sparsity in Labels/Data for Robust Statistical Analysis

Sparse data/label learning

https://sparse-learning.github.io
Overview of

Sparse and Low-Rank Recovery of Data

ECCV2012 tutorial:

Tutorial on Sparse and Low-rank Modeling, European Conference on Computer Vision, Firenze, Italy, October 2012

Our CVPR2022 tutorial:

Learning Sparsity in Labels/Data

for Robust Statistical Analysis

Sparse data/label learning

Learning Sparsity in Deep Models

for Compressive Neural Networks

Learning the sparse model

https://sparse-learning.github.io
Sparse and Low-Rank Recovery of *Data*

Part of content adapted from Prof. Yi Ma’s ECCV2012 tutorial
Sparse and Low-Rank Recovery of Data

Underdetermined Linear system

\[ y = Ax \]
Sparse and Low-Rank Recovery of Data

Underdetermined Linear system

\[ y = Ax \]

Observation \( y \in \mathbb{R}^m \)

Unknown \( x \in \mathbb{R}^n \)

\( A \in \mathbb{R}^{m \times n} \)

\( m \ll n \)

Part of content adapted from Prof. Yi Ma’s ECCV2012 tutorial
Sparse and Low-Rank Recovery of Data

Underdetermined Linear system

\[ y = Ax \]

Observation \[ y \in \mathbb{R}^m \]  
Unknown \[ x \in \mathbb{R}^n \]

Number of Observations \( m \) \(<\) Number of unknowns \( n \)

**Part of content adapted from Prof. Yi Ma’s ECCV2012 tutorial**

Sparse and Low-Rank Recovery of Data

Underdetermined Linear system

\[ y = Ax \]

Observation

\[ y \in \mathbb{R}^m \]

# Observations

\[ m \ll n \]

Unknown

\[ x \in \mathbb{R}^n \]

[A DCT basis]  \[ \approx \]

(Patches of) … input image

\[ y \]

\[ \text{Compression:} \]

\[ \text{[Wainwright, et al. IEEE TIT 2009]} \]
Sparse and Low-Rank Recovery of Data

Underdetermined Linear system
\[ y = Ax \]

Observation \( y \in \mathbb{R}^m \)
Unknown \( x \in \mathbb{R}^n \)

\( m \ll n \)

\( A \in \mathbb{R}^{m \times n} \)

Compression:
\[ y \approx Ax \]
(Patches of) input image \( A \) DCT basis \( x \) coefficients

Recognition:
\[ y = Ax \]
Test image \( A = [A_1 | A_2 | \cdots | A_k] \)
Combined training dictionary \( x \in \mathbb{R}^n \) coefficients \( e \in \mathbb{R}^m \) corruption, occlusion


Part of content adapted from Prof. Yi Ma’s ECCV2012 tutorial
Sparse and Low-Rank Recovery of *Data* (Cont.)

From recovering a *single sparse vector* to recovering *low-rank matrix* (many correlated vectors):

\[ y = [A] x + e \]

\[ Y = [X] + [E] \]
Sparse and Low-Rank Recovery of Data (Cont.)

From recovering a *single sparse vector* to recovering *low-rank matrix* (many correlated vectors):

$$y = A x + e$$

$$Y = X + E$$

Faces under varying illumination:

Part of content adapted from Prof. Yi Ma’s ECCV2012 tutorial
Sparse and Low-Rank Recovery of *Data* (Cont.)

From recovering a *single sparse vector* to recovering *low-rank matrix* (many correlated vectors):

\[ y = A x + e \]

Faces under varying illumination:

Background modeling from video:
Sparse Optimization

\[
\text{minimize} \quad \|x\|_0 \quad \text{subject to} \quad Ax = y
\]

- **\( L_1 \) norm** \( \|x\|_0 \to \|x\|_1 \)

- **Huber-Loss**: \( \|y - Ax\|_2^2 \to L_\delta(y - Ax) \)

where \( L_\delta(x) = \begin{cases} 
\frac{1}{2}(x)^2 & \text{for } |x| \leq \delta \\
\delta \cdot (|x| - \frac{1}{2}\delta), & \text{otherwise}
\end{cases} \)
Sparse Optimization

\[
\begin{align*}
\text{minimize} & \quad \|x\|_0 \quad \text{subject to} \quad Ax = y \\
\text{nonconvex} & \quad \rightarrow \quad \text{NP-hard!}
\end{align*}
\]

- \(L_1\) norm: \(\|x\|_0 \rightarrow \|x\|_1\)

- Huber-Loss:
  \[
  \|y - Ax\|_2^2 \rightarrow L_\delta(y - Ax)
  \]
  where \(L_\delta(x) = \begin{cases} 
  \frac{1}{2}(x)^2, & \text{for } |x| \leq \delta \\
  \delta \cdot (|x| - \frac{1}{2} \delta), & \text{otherwise}
  \end{cases}
  \]
Sparse Optimization

minimize $\|x\|_0$ subject to $Ax = y$

nonconvex $\rightarrow$ NP-hard!

Relax the problem

- $L_1$ norm $\|x\|_0 \rightarrow \|x\|_1$

- Huber-Loss: $\|y - Ax\|_2^2 \rightarrow L_\delta(y - Ax)$

where $L_\delta(x) = \begin{cases} 
\frac{1}{2}(x)^2 & \text{for } |x| \leq \delta \\
\delta \cdot (|x| - \frac{1}{2}\delta) & \text{otherwise}
\end{cases}$

Part of content adapted from Prof. Yi Ma’s ECCV2012 tutorial
Overview

- Sparse Learning in Data/Label
- Sparse Learning in Deep Models
Sparse learning for Noisy Data/Labels

Underdetermined Linear system

\[ y = Ax \]

\[ A \in \mathbb{R}^{m \times n} \]

\[ y \in \mathbb{R}^m \]

\[ x \in \mathbb{R}^n \]

# Observations \# unknowns

\( m \ll n \)

Part of content adapted from Prof. Yi Ma's ECCV2012 tutorial
Sparse learning for Noisy Data/Labels

Underdetermined Linear system

\[ y = Ax \]

Observation \( y \in \mathbb{R}^m \)

Unknown \( x \in \mathbb{R}^n \)

\( A \in \mathbb{R}^{m \times n} \)

Noisy One-hot Labels

\[ Y = X \beta \]

Deep Features

\[ X \in \mathbb{R}^{n \times d} \]

Fitted Coef.

\[ \beta \in \mathbb{R}^{d \times c} \]

Part of content adapted from Prof. Yi Ma’s ECCV2012 tutorial
Sparse learning for Noisy Data/Labels

Underdetermined Linear system: \( y = Ax \)

Observation: \( y \in \mathbb{R}^m \)
Observations: \( m \)
Unknowns: \( n \)

\( A \in \mathbb{R}^{m \times n} \)

Which Y is noisy?

Linear system with Noisy Data/Labels: \( Y = X\beta \)

Noisy One-hot Labels: \( Y \in \mathbb{R}^{n \times c} \)
Deep Features: \( X \in \mathbb{R}^{n \times d} \)
Fitted Coef.: \( \beta \in \mathbb{R}^{d \times c} \)

Part of content adapted from Prof. Yi Ma’s ECCV2012 tutorial
Sparse learning for Noisy Data/Labels

Underdetermined Linear system

\[ y = Ax \]

\[ A \in \mathbb{R}^{m \times n} \]

\[ y \in \mathbb{R}^m \]

\[ \# \text{Observations} \]

\[ \# \text{unknowns} \]

\[ x \in \mathbb{R}^n \]

Linear system with Noisy Data/Labels

\[ Y = X\beta \]

\[ Y \in \mathbb{R}^{n \times c} \]

\[ X \in \mathbb{R}^{n \times d} \]

\[ \beta \in \mathbb{R}^{d \times c} \]

Which Y is noisy?

We will introduce works of recovering sparse noisy data/labels for Robust Statistical Analysis.

Part of content adapted from Prof. Yi Ma’s ECCV2012 tutorial
Sparse learning for Noisy Data/Labels: The Indicator

Linear system with Noisy Data/Labels:

\[ Y = X\beta + \gamma \]

- Noisy One-hot Labels: \( Y \in \mathbb{R}^{n \times c} \)
- Deep Features: \( X \in \mathbb{R}^{n \times d} \)
- Fitted Coef.: \( \beta \in \mathbb{R}^{d \times c} \)
- Noisy Data Indicator: \( \gamma \in \mathbb{R}^{n \times c} \)

Part of content adapted from Prof. Yi Ma's ECCV2012 tutorial
Sparsity in Data/Labels: Different Focus

Robust regression/Classification, [Wang et al. CVPR2020].

\[ y = x^\top \beta + \epsilon + \gamma \]

Statistical robust ranking, [Fu et al. TPAMI 16]

\[ y = x^\top \beta + \epsilon + \gamma \]

Face Recognition, [Wright et al. TPAMI 09]

\[ y = (A, I)(\begin{bmatrix} x \\ \gamma \end{bmatrix}) + \epsilon \]

\[ y \in \mathbb{R}^m \]
\[ A = [A_1 | A_2 | \cdots | A_k] \]
\[ x \in \mathbb{R}^n \]
\[ e \in \mathbb{R}^m \]

Test image

Combined training dictionary

Corruption, occlusion

[Zhao et al. ICML 2018][Fu et al. ECCV 2014/TPAMI2016], [Wang et al. CVPR2020/TPAMI2021/CVPR2022] [Huang et al. ECCV2014]

Figure of RANSAC is by Xavi.borras - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=37017886
Understanding $\gamma$ in Statistics

\[ y = x^T \beta + \epsilon + \gamma \]
Understanding $\gamma$ in Statistics

$y = x^T \beta + \epsilon + \gamma$

$\gamma_i$ equals to the residual predict error $\gamma_i = y_i - x_i^T \hat{\beta}$
Understanding $\gamma$ in Statistics

\[ y = x^T \beta + \epsilon + \gamma \]

\[ \gamma_i \text{ equals to the residual predict error } \gamma_i = y_i - x_i^T \hat{\beta} \]

Row residuals fail to detect outliers at leverage points.

Understanding $\gamma$ in Statistics

$$y = x^T \beta + \epsilon + \gamma$$

$\gamma_i$ equals to the residual predict error $\gamma_i = y_i - x_i^T \hat{\beta}$

Understanding $\gamma$ in Statistics

$$y = x^T \beta + \epsilon + \gamma$$

$\gamma_i$ equals to the residual predict error $\gamma_i = y_i - x_i^T \hat{\beta}$

Leave-one-out externally studentized residual

$$t_i = \frac{y_i - x_i^T \hat{\beta}_{(i)}}{\hat{\sigma}_{(i)} (1 + x_i (X_{(i)}^T X_{(i)})^{-1} x_i)^{1/2}}$$

Understanding \( \gamma \) in Statistics

\[
y = x^T \beta + \epsilon + \gamma
\]

\( \gamma_i \) equals to the residual predict error \( \gamma_i = y_i - x_i^T \hat{\beta} \)

Leave-one-out externally studentized residual

\[
t_i = \frac{y_i - x_i^T \hat{\beta}_{(i)}}{\hat{\sigma}_{(i)} (1 + x_i (X_{(i)}^T X_{(i)})^{-1} x_i)^{1/2}}
\]

\( \Leftrightarrow \) test whether \( \gamma = 0 \) in \( y = X \beta + \gamma 1_i + \epsilon \)

Understanding $\gamma$ in Statistics

$$y = x^T \beta + \epsilon + \gamma$$

$\gamma_i$ equals to the residual predict error $\gamma_i = y_i - x_i^T \hat{\beta}$

Leave-one-out externally studentized residual

$$t_i = \frac{y_i - x_i^T \hat{\beta}_{(i)}}{\hat{\sigma}_{(i)}(1 + x_i (X_{(i)}^T X_{(i)})^{-1} x_i)^{1/2}}$$

$\Leftrightarrow$ test whether $\gamma = 0$ in $y = X\beta + \gamma 1_i + \epsilon$

When there are multiple outliers:

**Understanding $\gamma$ in Statistics**

$$y = x^T \beta + \epsilon + \gamma$$

$\gamma_i$ equals to the residual predict error $\gamma_i = y_i - x_i^T \hat{\beta}$

Leave-one-out externally studentized residual

$$t_i = \frac{y_i - x_i^T \hat{\beta}_{(i)}}{\hat{\sigma}(i)(1 + x_i (X_{(i)}^T X_{(i)})^{-1} x_i)^{-1/2}}$$

$\Leftrightarrow$ test whether $\gamma = 0$ in $y = X \beta + \gamma 1_i + \epsilon$

When there are multiple outliers:

1. **masking**: multiple outliers may mask each other and being **undetected**;

Understanding $\gamma$ in Statistics

$$y = x^T \beta + \epsilon + \gamma$$

\(\gamma_i\) equals to the residual predict error $\gamma_i = y_i - x_i^T \hat{\beta}$

Leave-one-out externally studentized residual

$$t_i = \frac{y_i - x_i^T \hat{\beta}_{(i)}}{\hat{\sigma}_{(i)}(1 + x_i (X_{(i)}^T X_{(i)})^{-1} x_i)^{1/2}}$$

$\Leftrightarrow$ test whether $\gamma = 0$ in $y = X\beta + \gamma 1 + \epsilon$

When there are multiple outliers:

1. **masking**: multiple outliers may mask each other and being **undetected**;

2. **swamping**: multiple outliers may lead the **large** $t_i$ for clean data.

Understanding $\gamma$ in Statistics

\[ y = x^T \beta + \epsilon + \gamma \]

$\gamma_i$ equals to the residual predict error $\gamma_i = y_i - x_i^T \hat{\beta}$

Leave-one-out externally studentized residual

\[ t_i = \frac{y_i - x_i^T \hat{\beta}_{(i)}}{\hat{\sigma}_{(i)}(1 + x_i (X^T_{(i)} X_{(i)})^{-1} x_i)^{1/2}} \]

$\Leftrightarrow$ test whether $\gamma = 0$ in $y = X \beta + \gamma 1_i + \varepsilon$

Understanding $\gamma$ in Statistics

\[ y = x^T \beta + \epsilon + \gamma \]

$\gamma_i$ equals to the residual predict error $\gamma_i = y_i - x_i^T \hat{\beta}$

Leave-one-out externally studentized residual

\[ t_i = \frac{y_i - x_i^T \hat{\beta}_{(i)}}{\hat{\sigma}_{(i)} (1 + x_i (X_{(i)}^T X_{(i)})^{-1} x_i)^{1/2}} \]

$\iff$ test whether $\gamma = 0$ in $y = X\beta + \gamma 1_i + \epsilon$

\[ y = X\beta + \epsilon + \gamma \]

Overview

• Spare Learning in Data/Label
• Sparse Learning in Deep Models
Learning Sparsity in Neural Networks

Random initialized weights in convolutional layers

Trained CNN weights

Densely trained model

[Fu et al. ICML 2020/TPAMI2022]
Learning Sparsity in Neural Networks

Densely trained model

Sparsified model

Random initialized weights in convolutional layers

Trained CNN weights

Random initialized weights in convolutional layers

Trained CNN weights

[Fu et al. ICML 2020/TPAMI2022]
Tradeoff between Overparameterized and Compressive models

Over-parameterized Models

Pros
• Great Expressive Power
• Simplify Loss Landscape

Cons
• Too much parameters
• Even hard to inference on single machine

Compressive/Sparse Models

Pros
• Less memory & Running cost
• Easy to deploy on limited resource machines

Cons
• Might loss of accuracy

[Fu et al. ICML 2020/TPAMI2022]
Left figures from Li et al. Visualizing the loss landscape of neural nets. NeurIPS 2018
Potential Connection to Foundation model
Potential Connection to Foundation model

Training foundation model:
• A routine of compressing and growing network may be beneficial.
Potential Connection to Foundation model

Training foundation model:
• A routine of compressing and growing network may be beneficial.

Deploying to downstream task:
• Desirable reduced model size for the task of limited resources
The foundation model is *emergence* and *homogenization*, and should be adapted to different tasks deployed on various platforms. Figure modified from [Bommasani et al 2021].

Bommasani et al. On the Opportunities and Risks of Foundation Models. arx:108.07258, 2021
The foundation model is *emergence* and *homogenization*, *and* should be adapted to different task deployed on various platforms. Figure modified from [Bommasani et al 2021].

Bommasani et al. On the Opportunities and Risks of Foundation Models. arx:108.07258, 2021
The foundation model is emergence and homogenization, and should be adapted to different task deployed on various platforms. Figure modified from [Bommasani et al 2021].

Bommasani et al. On the Opportunities and Risks of Foundation Models. arx:108.07258, 2021
The foundation model is *emergence* and *homogenization*, and should be adapted to different task deployed on various platforms. Figure modified from [Bommasani et al 2021].

Bommasani et al. On the Opportunities and Risks of Foundation Models. arx:108.07258, 2021
The foundation model is *emergence* and *homogenization*, and should be adapted to different task deployed on various platforms. Figure modified from [Bommasani et al 2021].

Bommasani et al. On the Opportunities and Risks of Foundation Models. arx:108.07258, 2021
The foundation model is emergence and homogenization, and should be adapted to different task deployed on various platforms. Figure modified from [Bommasani et al 2021].
The foundation model is *emergence* and *homogenization*, and should be adapted to different task deployed on various platforms. Figure modified from [Bommasani et al 2021].
Foundation Models in Computer Vision

❖ **CLIP**: Learning Transferable Visual Models From Natural Language Supervision, arXiv Feb. 24, ICML2021, OpenAI
   • Code/model: https://github.com/openai/CLIP

❖ **ALIGN**: Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision, ICML2021, Google Research
   • Code/model: N/A

❖ **ALBEF**: Align before Fuse: Vision and Language Representation Learning with Momentum Distillation, NeurIPS 2021, Salesforce Research
   • Code/model: https://github.com/salesforce/ALBEF

❖ **Florence**: A New Foundation Model for Computer Vision, arXiv, Nov. 22, 2021, Microsoft Cloud and AI, Microsoft Research Redmond
   • Code/model: N/A

   • Code/model: https://github.com/microsoft/NUWA

❖ **INTERN**: A New Learning Paradigm Towards General Vision, arXiv Nov. 16, 2021 *Shanghai AI Laboratory, SenseTime, CUKH, SJTU*
   • Code/model: N/A

❖ **Gopher**: Scaling Language Models: Methods, Analysis & Insights from Training Gopher, arXiv Dec. 8, 2021, DeepMind
   • Code/model: N/A

❖ **FLAVA**: A Foundational Language And Vision Alignment Model, CVPR 2022, FAIR
   • Code/model: https://flava-model.github.io

❖ **OPT**: Open Pre-trained Transformer Language Models. Meta AI 2022
Foundation Models in Computer Vision

- **CLIP**: Learning Transferable Visual Models From Natural Language Supervision, arXiv Feb. 24, ICML2021, OpenAI
  - Code/model: https://github.com/openai/CLIP

- **ALIGN**: Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision, ICML2021, Google Research
  - Code/model: N/A

- **ALBEF**: Align before Fuse: Vision and Language Representation Learning with Momentum Distillation, NeurIPS 2021, Salesforce Research
  - Code/model: https://github.com/salesforce/ALBEF

- **Florence**: A New Foundation Model for Computer Vision, arXiv, Nov. 22, 2021, Microsoft Cloud and AI, Microsoft Research Redmond
  - Code/model: N/A

  - Code/model: https://github.com/microsoft/NUWA

- **INTERN**: A New Learning Paradigm Towards General Vision, arXiv Nov. 16, 2021 Shanghai AI Laboratory, SenseTime, CUKH, SJTU
  - Code/model: N/A

- **Gopher**: Scaling Language Models: Methods, Analysis & Insights from Training Gopher, arXiv Dec. 8, 2021, DeepMind
  - Code/model: N/A

- **FLAVA**: A Foundational Language And Vision Alignment Model, CVPR 2022, FAIR
  - Code/model: https://flava-model.github.io

- **OPT**: Open Pre-trained Transformer Language Models. Meta AI 2022

It urges us to study *learning sparsity* in deep foundation models.
Learning Sparsity in Data/Labels
Few-shot Learning by Unlabeled Data

Wang et al. Instance Credibility Inference for Few-Shot Learning. CVPR 2020
Wang et al. How to Trust Unlabeled Data? Instance Credibility Inference for Few-Shot Learning. IEEE TPAMI 2021
Few-shot Learning by Unlabeled Data

Labeled Image

Train

Few-Shot Models

Labels

A
B
C
D

Wang et al. Instance Credibility Inference for Few-Shot Learning. CVPR 2020
Wang et al. How to Trust Unlabeled Data? Instance Credibility Inference for Few-Shot Learning. IEEE TPAMI 2021
Few-shot Learning by Unlabeled Data

Wang et al. Instance Credibility Inference for Few-Shot Learning. CVPR 2020
Wang et al. How to Trust Unlabeled Data? Instance Credibility Inference for Few-Shot Learning. IEEE TPAMI 2021
Few-shot Learning by Unlabeled Data

Labeled Image

Unlabeled Image

Train

Few-Shot Models

Inference

Labels

Pseudo-Labels

Wang et al. Instance Credibility Inference for Few-Shot Learning. CVPR 2020
Wang et al. How to Trust Unlabeled Data? Instance Credibility Inference for Few-Shot Learning. IEEE TPAMI 2021
Few-shot Learning by Unlabeled Data

Wang et al. Instance Credibility Inference for Few-Shot Learning. CVPR 2020
Wang et al. How to Trust Unlabeled Data? Instance Credibility Inference for Few-Shot Learning. IEEE TPAMI 2021
Sparse Labels in Semi-supervised Few-Shot Learning

We will introduce the details in the next talk.

[Wang et al. CVPR2020/TPAMI2021]
Sparse Labels in Semi-supervised Few-Shot Learning

We will introduce the details in the next talk.

[Wang et al. CVPR2020/TPAMI2021]
Sparse Labels in Semi-supervised Few-Shot Learning

We will introduce the details in the next talk.

\[ y_i = x_i^T \beta + \gamma_i + \epsilon \]

\[
\arg\min_{\beta, \gamma} \| Y - X\beta - \gamma \|_F^2 + \lambda R(\gamma)
\]

\[
\arg\min_{\gamma} \| \tilde{Y} - \tilde{X}\gamma \|_F^2 + \lambda R(\gamma)
\]

[Wang et al. CVPR2020/TPAMI2021]
Sparse Labels in Semi-supervised Few-Shot Learning

We will introduce the details in the next talk.

\[ y_i = x_i^T \beta + \gamma_i + \varepsilon \]
\[
\arg\min_{\beta, \gamma} : \| Y - X \beta - \gamma \|_F^2 + \lambda R(\gamma)
\]
\[
\arg\min_{\gamma} \| \tilde{Y} - \tilde{X} \gamma \|_F^2 + \lambda R(\gamma)
\]

A Linear regression problem!

[Wang et al. CVPR2020/TPAMI2021]
Sparse Labels in Semi-supervised Few-Shot Learning

A Linear regression problem!

\[ y_i = x_i \beta + \gamma_i + \varepsilon \]

\[ \text{argmin}_{\beta, \gamma} \| Y - X \beta - \gamma \|_F^2 + \lambda R(\gamma) \]

How to select \( \lambda \)?

\[ \text{argmin}_{\gamma} \| \tilde{Y} - \tilde{X} \gamma \|_F^2 + \lambda R(\gamma) \]

We will introduce the details in the next talk.

[Wang et al. CVPR2020/TPAMI2021]
Sparse Labels in Semi-supervised Few-Shot Learning

A Linear regression problem!

\[ y_i = x_i^T \beta + \gamma_i + \varepsilon \]

\[
\argmin_{\beta, \gamma} \|Y - X\beta - \gamma\|^2_F + \lambda R(\gamma)
\]

How to select \( \lambda \)?
- heuristics rules \( \lambda = 2.5\hat{\sigma} \)?

We will introduce the details in the next talk.

[Wang et al. CVPR2020/TPAMI2021]
Sparse Labels in Semi-supervised Few-Shot Learning

We will introduce the details in the next talk.

A Linear regression problem!

\[ y_i = x_i^T \beta + \gamma_i + \epsilon \]

\[ \argmin_{\beta, \gamma} := \| Y - X \beta - \gamma \|_F^2 + \lambda R(\gamma) \]

\[ \argmin_{\gamma} \| \tilde{Y} - \tilde{X} \gamma \|_F^2 + \lambda R(\gamma) \]

How to select \( \lambda \)?

- heuristics rules \( \lambda = 2.5\hat{\sigma} \)?
- Cross-validation?

[Wang et al. CVPR2020/TPAMI2021]
Sparse Labels in Semi-supervised Few-Shot Learning

We will introduce the details in the next talk.

A Linear regression problem!

\[ y_i = x_i^T \beta + \gamma_i + \varepsilon \]

\[ \arg\min_{\beta, \gamma} \| Y - X \beta - \gamma \|_F^2 + \lambda R(\gamma) \]

How to select \( \lambda \)?

- heuristics rules \( \lambda = 2.5\hat{\sigma} \)?
- Cross-validation?
- Data adaptive techniques?

We will introduce the details in the next talk.

[Wang et al. CVPR2020/TPAMI2021]
Sparse Labels in Semi-supervised Few-Shot Learning

We will introduce the details in the next talk.

A Linear regression problem!

\[ y_i = x_i^T \beta + \gamma_i + \varepsilon \]

\[ \text{argmin}_{\beta, \gamma} := \| Y - X \beta - \gamma \|_F^2 + \lambda R(\gamma) \]

\[ \text{argmin}_{\gamma} \| \tilde{Y} - \tilde{X} \gamma \|_F^2 + \lambda R(\gamma) \]

How to select \( \lambda \)?

- heuristics rules \( \lambda = 2.5 \hat{\sigma} \)?
- Cross-validation?
- Data adaptive techniques?
- AIC, BIC?

[Wang et al. CVPR2020/TPAMI2021]
Learning Sparsity in Learning with Noisy Labels

Stage 1: Feature Learning

Stage 2: Sample Selection

[Wang et al. CVPR2022]
Learning Sparsity in Learning with Noisy Labels

Stage 1: Feature Learning

Stage 2: Sample Selection

[Wang et al. CVPR2022]
Learning Sparsity in Learning with Noisy Labels

Stage 1: Feature Learning

Stage 2: Sample Selection

[Wang et al. CVPR2022]
Learning Sparsity in Learning with Noisy Labels

Stage 1: Feature Learning

Stage 2: Sample Selection

\[ y_i = x_i^T \beta + \gamma_i + \epsilon \]
Learning Sparsity in Learning with Noisy Labels

Stage 1: Feature Learning

Stage 2: Sample Selection

[Wang et al. CVPR2022]
Learning Sparsity in Learning with Noisy Labels

Stage 1: Feature Learning

Stage 2: Sample Selection

\[
y_i = \mathbf{x}_i^\top \beta + \gamma_i + \varepsilon
\]

\[
\left\| \tilde{Y} - \tilde{X} \gamma \right\|_F^2 + \sum_{i=1}^{n} P(\gamma_i; \lambda_i)
\]

[Wang et al. CVPR2022]
Learning Sparsity in Learning with Noisy Labels

Stage 1: Feature Learning

Stage 2: Sample Selection

[Wang et al. CVPR2022]
Learning Sparsity in Learning with Noisy Labels

Stage 1: Feature Learning

Stage 2: Sample Selection

\[ y_i = \mathbf{x}_i^T \beta + \gamma_i + \epsilon \]

\[ \arg\min_{\gamma} \| \tilde{Y} - \tilde{X} \gamma \|^2_F + \sum_{i=1}^{n} P(\gamma_i; \lambda_i) \]

[Wang et al. CVPR2022]
Make it scalable to large datasets

Wang et al. Scalable Penalized Regression for Noise Detection in Learning with Noisy Labels. CVPR 2022
Make it scalable to large datasets

Wang et al. Scalable Penalized Regression for Noise Detection in Learning with Noisy Labels. CVPR 2022
Make it scalable to large datasets

Wang et al. Scalable Penalized Regression for Noise Detection in Learning with Noisy Labels. CVPR 2022
Make it scalable to large datasets

Wang et al. Scalable Penalized Regression for Noise Detection in Learning with Noisy Labels. CVPR 2022
Make it scalable to large datasets

Wang et al. Scalable Penalized Regression for Noise Detection in Learning with Noisy Labels. CVPR 2022
Make it scalable to large datasets

Wang et al. Scalable Penalized Regression for Noise Detection in Learning with Noisy Labels. CVPR 2022
Make it scalable to large datasets

\[ y_i = x_i^T \beta + \gamma_i + \varepsilon \]

Wang et al. Scalable Penalized Regression for Noise Detection in Learning with Noisy Labels. CVPR 2022
Sparse Learning in Robust Ranking

Fu et al. Interestingness Prediction by Robust Learning to Rank. ECCV 2014
Fu et al. Robust Subjective Visual Property Prediction from Crowdsourced Pairwise Labels. IEEE TPAMI 2016
Who is smiling more?
Sparse Learning in Robust Ranking

Who is smiling more?

(a) Smiling  
(b) ?  
(c) Not smiling

(d) Natural  
(e) ?  
(f) Manmade

Parikh et al. Relative Attributes, ICCV 2011, Marr Prize Paper.

Fu et al. Interestingness Prediction by Robust Learning to Rank. ECCV 2014
Fu et al. Robust Subjective Visual Property Prediction from Crowdsourced Pairwise Labels. IEEE TPAMI 2016
Sparse Learning in Robust Ranking

Who is smiling more?

1. Cultural factors
2. Psychological factors: Halo Effects
3. Ambiguous comparisons
4. Malicious/Lazy annotators

Parikh et al. Relative Attributes, ICCV 2011, Marr Prize Paper.
Who is smiling more?

1. Cultural factors
2. Psychological factors: Halo Effects
3. Ambiguous comparisons
4. Malicious/Lazy annotators

Fu et al. Interestingness Prediction by Robust Learning to Rank. ECCV 2014
Fu et al. Robust Subjective Visual Property Prediction from Crowdsourced Pairwise Labels. IEEE TPAMI 2016

Parikh et al. Relative Attributes, ICCV 2011, Marr Prize Paper.
Sparse Learning in Robust Ranking

Fu et al. Interestingness Prediction by Robust Learning to Rank. ECCV 2014
Fu et al. Robust Subjective Visual Property Prediction from Crowdsourced Pairwise Labels. IEEE TPAMI 2016
Sparse Learning in Robust Ranking

Ranking age of images by learning from crowdsourced pairs as the directed graph $G = (V, E)$

Fu et al. Interestingness Prediction by Robust Learning to Rank. ECCV 2014
Fu et al. Robust Subjective Visual Property Prediction from Crowdsourced Pairwise Labels. IEEE TPAMI 2016
Sparse Learning in Robust Ranking

Ranking age of images by learning from crowdsourced pairs as the directed graph $G = (V, E)$

Each edge is modeled as

$$Y_{ij} = \beta^T \phi_i - \beta^T \phi_j + \gamma_{ij}$$

Fu et al. Interestingness Prediction by Robust Learning to Rank. ECCV 2014
Fu et al. Robust Subjective Visual Property Prediction from Crowdsourced Pairwise Labels. IEEE TPAMI 2016
Sparse Learning in Robust Ranking

Ranking age of images by learning from crowdsourced pairs as the directed graph $G = (V, E)$

Each edge is modeled as $Y_{ij} = \beta^T \phi_i - \beta^T \phi_j + \gamma_{ij}$

The edge of whole dataset is $Y = C\Phi \beta + \epsilon + \Gamma$

Fu et al. Interestingness Prediction by Robust Learning to Rank. ECCV 2014
Fu et al. Robust Subjective Visual Property Prediction from Crowdsourced Pairwise Labels. IEEE TPAMI 2016
Sparse Learning in Robust Ranking

Ranking age of images by learning from crowdsourced pairs as the directed graph \( G = (V, E) \)

Each edge is modeled as \( Y_{ij} = \beta^T \phi_i - \beta^T \phi_j + \gamma_{ij} \)

The edge of whole dataset is \( Y = C \Phi \beta + \epsilon + \Gamma \)

\[
\min_{\beta, \Gamma} \frac{1}{2} \| Y - C \Phi \beta - \Gamma \|_2^2 + \lambda \| \Gamma \|_1
\]

Fu et al. Interestingness Prediction by Robust Learning to Rank. ECCV 2014
Fu et al. Robust Subjective Visual Property Prediction from Crowdsourced Pairwise Labels. IEEE TPAMI 2016
Sparse Learning in Robust Ranking

Ranking age of images by learning from crowdsourced pairs as the directed graph $G = (V, E)$

Each edge is modeled as $Y_{ij} = \beta^T \phi_i - \beta^T \phi_j + \gamma_{ij}$

The edge of whole dataset is $Y = C\Phi\beta + \epsilon + \Gamma$

$$\min_{\beta, \Gamma} \frac{1}{2} \| Y - C\Phi\beta - \Gamma \|_2^2 + \lambda \| \Gamma \|_1 \quad \Rightarrow \quad \hat{\Gamma} = \arg \min_{\Gamma} \| U_2^T Y - U_2^T \Gamma \| + \lambda \| \Gamma \|_1$$

Fu et al. Interestingness Prediction by Robust Learning to Rank. ECCV 2014
Fu et al. Robust Subjective Visual Property Prediction from Crowdsourced Pairwise Labels. IEEE TPAMI 2016
Checking Regularisation Path

Checking Regularisation Path

\[ \hat{\Gamma} = \arg\min_{\Gamma} \left\| U_2^T Y - U_2^T \Gamma \right\|_1 + \lambda \left\| \Gamma \right\|_1 \]

Checking Regularisation Path

\[ \hat{\Gamma} = \arg \min_{\Gamma} \| U_2^T Y - U_2^T \Gamma \| + \lambda \| \Gamma \|_1 \]

It’s one type of Preconditioned Lasso! [2]. We solve \( \Gamma \) checking regularisation path,

Checking Regularisation Path

\[ \hat{\Gamma} = \arg \min_{\Gamma} \| U^T_2 Y - U^T_2 \Gamma \| + \lambda \| \Gamma \|_1 \]

It’s one type of Preconditioned Lasso![^2]. We solve \( \Gamma \) checking regularisation path,

Checking Regularisation Path

\[ \hat{\Gamma} = \arg \min_{\Gamma} \left\| U_2^T Y - U_2^T \Gamma \right\| + \lambda \left\| \Gamma \right\|_1 \]

It's one type of Preconditioned Lasso[^2]. We solve \( \Gamma \) checking regularisation path,


Red lines&points indicate outliers; Blue lines&points are inliers.
Learning Sparsity in Neural Network
“OLD SCHOOL” For A Compromise

Over-parameterized Networks $W$

Compressive Networks $W_s$

Fu et al. Exploring Structural Sparsity of Deep Networks via Inverse Scale Spaces. IEEE TPAMI accepted (2022)
“OLD SCHOOL” For A Compromise

Over-parameterized Networks $W$

? \rightarrow

SGD + Regularization
Network Pruning
Knowledge Distill
Lottery Hypothesis

Compressive Networks $W_s$

Fu et al. Exploring Structural Sparsity of Deep Networks via Inverse Scale Spaces. IEEE TPAMI accepted (2022)
“OLD SCHOOL” For A Compromise

Over-parameterized Networks $W$

Compressive Networks $W_s$

Existing 2-Stage Approaches:
- Fully training dense network,

Fu et al. Exploring Structural Sparsity of Deep Networks via Inverse Scale Spaces. IEEE TPAMI accepted (2022)
"OLD SCHOOL" For A Compromise

Existing 2-Stage Approaches:
- Fully training dense network,
- Finding good sparse subnets.

Over-parameterized Networks $W$

Compressive Networks $W_s$

Fu et al. Exploring Structural Sparsity of Deep Networks via Inverse Scale Spaces. IEEE TPAMI accepted (2022)
“OLD SCHOOL” For A Compromise

Existing 2-Stage Approaches:
- Fully training dense network,
- Finding good sparse subnets.

Our method: 1-Stage Approach (end-to-end)
Without fully training a dense model.

Over-parameterized Networks $W$

Compressive Networks $W_s$

Fu et al. Exploring Structural Sparsity of Deep Networks via Inverse Scale Spaces. IEEE TPAMI accepted (2022)
Regularization and Overparameterization
Regularization and Overparameterization

- Regularization:
  - $\ell_1$ regularization [Collins et al. 2014]: Spurious Correlation due to highly correlated neurons
Regularization and Overparameterization

- Regularization:
  - $\ell_1$ regularization [Collins et al. 2014]: Spurious Correlation due to highly correlated neurons

Accuracy Loss due to trapping into local minima, without aid of over-parameterize model!
Regularization and Overparameterization

• Regularization:
  • $\ell_1$ regularization [Collins et al. 2014]: Spurious Correlation due to highly correlated neurons

Accuracy Loss due to trapping into local minima, without aid of over-parameterize model!

Overparameterized Model:


• Better Generalization *(weight-size dependent complexities are controlled)* (Neyshabur, 2019)
Regularization and Overparameterization

- Regularization:
  - $\ell_1$ regularization [Collins et al. 2014]: Spurious Correlation due to highly correlated neurons

Accuracy Loss due to trapping into local minima, without aid of over-parameterize model!

Overparameterized Model:

- **Better Generalization** (*weight-size dependent complexities are controlled*) (Neyshabur, 2019)

However, *Larger Inference Time and Memory Cost!*
Network Pruning:

- Weight Pruning
- Filter Pruning
Network Pruning (1)

Three-Step Training Pipeline

Network Pruning:
- Weight Pruning
- Filter Pruning

[Han, NeurIPS15]
Network Pruning (1)

Three-Step Training Pipeline

Network Pruning:
- Weight Pruning
- Filter Pruning

Pruning

Quantization

Han et al. Learning both Weights and Connections for Efficient Neural Networks. NeurIPS 2015
Han et al. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. ICLR16 (Best Paper Award)
Song Han’s talk: Hardware Efficiency Aware Neural Architecture Search. The 3rd Workshop on Energy Efficient Machine Learning and Cognitive Computing in CVPR 2019
Network Pruning (1)

Three-Step Training Pipeline

Network Pruning:
- Weight Pruning
- Filter Pruning

Pruning [Han, NeurIPS15]

Questions: Can we do sparsity in weight level, filter level, and even layer level with a unified ‘algorithm’?

Hardware Acceleration

Quantization [Han, ICLR16]

Han et al. Learning both Weights and Connections for Efficient Neural Networks. NeurIPS 2015
Han et al. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. ICLR16 (Best Paper Award)
Song Han’s talk: Hardware Efficiency Aware Neural Architecture Search. The 3rd Workshop on Energy Efficient Machine Learning and Cognitive Computing in CVPR 2019
Learning Sparsity on Models

Child et al. Generating Long Sequences with Sparse Transformers. arxiv 2019
Learning Sparsity on Models

- Designing sparse transformer architectures (Child, 2019)

Child et al. Generating Long Sequences with Sparse Transformers. arxiv 2019
Learning Sparsity on Models

- Designing sparse transformer architectures (Child, 2019)

Child et al. Generating Long Sequences with Sparse Transformers. arxiv 2019
Learning Sparsity on Models

- Designing sparse transformer architectures (Child, 2019)

- Pruning over-parameterized structure with regrowing (Dettmers, 2019)
Learning Sparsity on Models

- Designing sparse transformer architectures (Child, 2019)
- Pruning over-parameterized structure with regrowing (Dettmers, 2019)

Child et al. Generating Long Sequences with Sparse Transformers, arxiv 2019
Learning Sparsity on Models

- Designing sparse transformer architectures (Child, 2019)

- Pruning over-parameterized structure with regrowing (Dettmers, 2019)
Learning Sparsity on Models

- Designing sparse transformer architectures (Child, 2019)

- Pruning over-parameterized structure with regrowing (Dettmers, 2019)
Learning Sparsity on Models

- Designing sparse transformer architectures (Child, 2019)

- Pruning over-parameterized structure with regrowing (Dettmers, 2019)

Child et al. Generating Long Sequences with Sparse Transformers. arxiv 2019
The Concept of Knowledge Distill

- Introducing the concept of "softmax temperature".
- Utilizing the “dark knowledge” of teacher model: which classes is more similar to the predicted class.

Hinton et al. Distilling the Knowledge in a Neural Network. NIPS 2014 Deep Learning Workshop

Figure from https://intellabs.github.io/distiller/knowledge_distillation.html
Lottery Hypothesis

Over-parameterized Networks $W$

Compressive Networks $W_s$
Lottery Hypothesis

Over-parameterized Networks $W$

Compressive Networks $W_s$

Lottery Ticket Hypothesis

• Dense, randomly-initialized, feed-forward networks contain subnetworks (winning tickets) that – when trained in isolation – reach test accuracy comparable to the original network in a similar number of iterations. (Frankle & Carbin, 2019)
Lottery Hypothesis

Over-parameterized Networks $W$

Lottery Ticket Hypothesis

- Dense, randomly-initialized, feed-forward networks contain subnetworks (winning tickets) that – when trained in isolation – reach test accuracy comparable to the original network in a similar number of iterations. (Frankle & Carbin, 2019)

Rewinding the network from the initialization, and find “winning ticket” subnet
The Key Idea of the Optimizer-DessiLBI

DessiLBI: Deep structurally splitting Linearized Bregman Iteration

1. Lifting parameter space $W$ to $(W, \Gamma)$ coupling the inverse scale space.
2. $\Gamma$ learns structural sparsity in inverse scale space.
3. Network’s solution path in $(W, \Gamma)$ as the discretization of dynamics, and solved by LBI.
4. Our optimizer enjoys a provable global convergence guarantee.

Over-parameterized Network in Space $W$

We will give the mathematical and statistics introduction of Linearized Bregman Iteration (LBI) in the next talks.
The Key Idea of the Optimizer-DessiLBI

DessiLBI: Deep structurally splitting Linearized Bregman Iteration

1. Lifting parameter space $W$ to $(W, \Gamma)$ coupling the inverse scale space.
2. $\Gamma$ learns structural sparsity in inverse scale space.
3. Network’s solution path in $(W, \Gamma)$ as the discretization of dynamics, and solved by LBI.
4. Our optimizer enjoys a provable global convergence guarantee.

We will give the mathematical and statistics introduction of Linearized Bregman Iteration (LBI) in the next talks.
The Key Idea of the Optimizer-DessiLBI

DessiLBI: Deep structurally splitting Linearized Bregman Iteration

1. Lifting parameter space $W$ to $(W, \Gamma)$ coupling the inverse scale space.
2. $\Gamma$ learns structural sparsity in inverse scale space.
3. Network’s solution path in $(W, \Gamma)$ as the discretization of dynamics, and solved by LBI.
4. Our optimizer enjoys a provable global convergence guarantee.

Over-parameterized Network in Space $W$  

Space$(W, \Gamma)$

We will give the mathematical and statistics introduction of Linearized Bregman Iteration (LBI) in the next talks.
The Key Idea of the Optimizer-DessiLBI

DessiLBI: Deep structurally splitting Linearized Bregman Iteration

1. Lifting parameter space $W$ to $(W, \Gamma)$, coupling the inverse scale space.
2. $\Gamma$ learns structural sparsity in inverse scale space.
3. Network’s solution path in $(W, \Gamma)$ as the discretization of dynamics, and solved by LBI.
4. Our optimizer enjoys a provable global convergence guarantee.

We will give the mathematical and statistics introduction of Linearized Bregman Iteration (LBI) in the next talks.
The Key Idea of the Optimizer-DessiLBI

DessiLBI: Deep structurally splitting Linearized Bregman Iteration

1. Lifting parameter space $W$ to $(W, \Gamma)$, coupling the inverse scale space.
2. $\Gamma$ learns structural sparsity in inverse scale space.
3. Network’s solution path in $(W, \Gamma)$ as the discretization of dynamics, and solved by LBI.
4. Our optimizer enjoys a provable global convergence guarantee.

Over-parameterized Network in Space $W$

Space $(W, \Gamma)$

Compressive Networks $W_s$

We will give the mathematical and statistics introduction of Linearized Bregman Iteration (LBI) in the next talks.
The Key Idea of the Optimizer-DessiLBI

DessiLBI: Deep structurally splitting Linearized Bregman Iteration

1. Lifting parameter space $W$ to $(W, \Gamma)$, coupling the inverse scale space.
2. $\Gamma$ learns structural sparsity in inverse scale space.
3. Network’s solution path in $(W, \Gamma)$ as the discretization of dynamics, and solved by LBI.
4. Our optimizer enjoys a provable global convergence guarantee.

We will give the mathematical and statistics introduction of Linearized Bregman Iteration (LBI) in the next talks.
Any Better Solutions?

Two Stage Method, Training Dense Network $\rightarrow$ Produce sparse subnet

One Stage Method, An end-to-end method, directly producing structurally sparse Subnet.

Charles et al. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. CVPR 2017
Any Better Solutions?

Two Stage Method, Training Dense Network $\rightarrow$ Produce sparse subnet

One Stage Method, An end-to-end method, directly producing structurally sparse Subnet.

Inspired by PointNet

Charles et al. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. CVPR 2017
Any Better Solutions?

Two Stage Method, Training Dense Network → Produce sparse subnet

One Stage Method, An end-to-end method, directly producing structurally sparse Subnet.

Inspired by PointNet

Charles et al. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. CVPR 2017
Any Better Solutions?

Two Stage Method, Training Dense Network → Produce sparse subnet

One Stage Method, An end-to-end method, directly producing structurally sparse Subnet.

(1,2,3)
(1,1,1)  g = max
(2,3,2)
\vdots
(2,3,4)

Discover naïve/extreme property of geometry

Inspired by PointNet

Charles et al. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. CVPR 2017
Any Better Solutions?

Two Stage Method, Training Dense Network $\rightarrow$ Produce sparse subnet

One Stage Method, An end-to-end method, directly producing structurally sparse Subnet.

- Discover naïve/extreme property of geometry
- Aggregation in high-dim space preserves interesting properties of the geometry

Inspired by PointNet

Charles et al. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. CVPR 2017
Insights for the Inverse Scale Space

\[ W_s: \text{wining ticket} \]
\[ W: \text{dense parameters} \]
\[ W_s \cup W_{Sc} = W \]
\[ W_s \text{ and } W_{Sc} \text{ correlated} \]

[Fu et al. ICML 2020/TPAMI2022]
Insights for the Inverse Scale Space

\[ W_s: \text{wining ticket} \]
\[ W: \text{dense parameters} \]
\[ W_s \cup W_{Sc} = W \]
\[ W_s \text{ and } W_{Sc} \text{ correlated} \]

GD of Weight Space

\[ \dot{W}_t = -\nabla_W \hat{L}_n (W) \]

[Fu et al. ICML 2020/TPAMI2022]
Insights for the Inverse Scale Space

$W_s$: winning ticket
$W$: dense parameters
$W_s \cup W_{Sc} = W$
$W_s$ and $W_{Sc}$ correlated

GD of Weight Space

$$\dot{W}_t = -\nabla_W \hat{L}_n (W)$$

[Fu et al. ICML 2020/TPAMI2022]
Insights for the Inverse Scale Space

$W_s$: winning ticket

$W$: dense parameters

$W_s \cup W_{Sc} = W$

$W_s$ and $W_{Sc}$ correlated

GD of Weight Space

$\dot{W}_t = -\nabla_W \hat{\mathcal{L}}_n (W)$

[Fu et al. ICML 2020/TPAMI2022]
Insights for the Inverse Scale Space

\[ W_s: \text{wining ticket} \]
\[ W: \text{dense parameters} \]
\[ W_s \cup W_{Sc} = W \]
\[ W_s \text{ and } W_{Sc} \text{ correlated} \]

GD of Weight Space

\[ \dot{W}_t = -\nabla_W \hat{L}_n (W) \]

Differential Inclusion of Inverse Scale Space

\[ \bar{L} (W, \Gamma) = \hat{L}_n (W) + \frac{1}{2\nu} \|W - \Gamma\|^2_F \]

[Fu et al. ICML 2020/TPAMI2022]
Insights for the Inverse Scale Space

\[ W_s: \text{wining ticket} \]
\[ W: \text{dense parameters} \]
\[ W_s \cup W_{sc} = W \]
\[ W_s \text{ and } W_{sc} \text{ correlated} \]

GD of Weight Space

\[ \dot{W}_t = -\nabla_W \hat{\mathcal{L}}_n (W) \]

[Differential Inclusion of Inverse Scale Space]

\[ \mathcal{L} (W, \Gamma) = \hat{\mathcal{L}}_n (W) + \frac{1}{2\nu} \| W - \Gamma \|_F^2 \]

[Fu et al. ICML 2020/TPAMI2022]
Formulations

\[
\frac{\dot{W}_t}{\kappa} = -\nabla_W \bar{\mathcal{L}}(W_t, \Gamma_t)
\]

\[
\dot{V}_t = -\nabla_{\Gamma} \bar{\mathcal{L}}(W_t, \Gamma_t)
\]

\[
V_t \in \partial \left( \Omega(\Gamma) + \frac{1}{2\kappa} \|\Gamma\|^2 \right)
\]

\[
\bar{\mathcal{L}}(W, \Gamma) = \hat{\mathcal{L}}_n(W) + \frac{1}{2\nu} \|W - \Gamma\|_F^2
\]

Differential Inclusion of Inverse Scale Space

The Simple Discretization - DessLBI

\[
W_{k+1} = W_k - \kappa \alpha_k \cdot \nabla_W \bar{\mathcal{L}}(W_k, \Gamma_k)
\]

\[
V_{k+1} = V_k - \alpha_k \cdot \nabla_{\Gamma} \bar{\mathcal{L}}(W_k, \Gamma_k),
\]

\[
\Gamma_{k+1} = \kappa \cdot \text{Prox}_{\Omega_\lambda}(V_{k+1})
\]

V is the sub-gradient for some sparsity-enforced, often non-differentiable regularization \(\Omega_\lambda(\Gamma) = \lambda \Omega_1(\Gamma), (\lambda \in \mathbb{R}_+)\) such as Lasso or group Lasso penalties for \(\Omega_1(\Gamma)\)

[Fu et al. ICML 2020/TPAMI2022]
Formulations

\[
\frac{\dot{W}_t}{\kappa} = -\nabla_W \tilde{\mathcal{L}}(W_t, \Gamma_t) \\
\dot{V}_t = -\nabla_\Gamma \tilde{\mathcal{L}}(W_t, \Gamma_t) \\
V_t \in \partial \left( \Omega(\Gamma) + \frac{1}{2\kappa} \|\Gamma\|^2 \right)
\]

\[
\tilde{\mathcal{L}}(W, \Gamma) = \hat{\mathcal{L}}_n(W) + \frac{1}{2\nu} \|W - \Gamma\|^2_F
\]

Differential Inclusion of Inverse Scale Space

The Simple Discretization - DessLBI

\[
W_{k+1} = W_k - \kappa \alpha_k \cdot \nabla_W \tilde{\mathcal{L}}(W_k, \Gamma_k) \\
V_{k+1} = V_k - \alpha_k \cdot \nabla_\Gamma \tilde{\mathcal{L}}(W_k, \Gamma_k), \\
\Gamma_{k+1} = \kappa \cdot \text{Prox}_{\Omega_\lambda}(V_{k+1})
\]

V is the sub-gradient for some sparsity-enforced, often non-differentiable regularization \( \Omega_\lambda(\Gamma) = \lambda \Omega_1(\Gamma), (\lambda \in \mathbb{R}_+) \) such as Lasso or group Lasso penalties for \( \Omega_1(\Gamma) \)

- \( W_t \) follows the gradient descent with \( \ell_2 \)-regularization

[Fu et al. ICML 2020/TPAMI2022]
Formulations

\[
\begin{align*}
\frac{\dot{W}_t}{\kappa} &= -\nabla_W \tilde{\mathcal{L}}(W_t, \Gamma_t) \\
\dot{V}_t &= -\nabla_{\Gamma} \tilde{\mathcal{L}}(W_t, \Gamma_t) \\
V_t &\in \partial \left( \Omega(\Gamma) + \frac{1}{2\kappa} \|\Gamma\|^2 \right)
\end{align*}
\]

\[
\tilde{\mathcal{L}}(W, \Gamma) = \hat{\mathcal{L}}_n(W) + \frac{1}{2\nu} \|W - \Gamma\|_F^2
\]

The Simple Discretization - DessLBI

\[
\begin{align*}
W_{k+1} &= W_k - \kappa \alpha_k \cdot \nabla_W \tilde{\mathcal{L}}(W_k, \Gamma_k) \\
V_{k+1} &= V_k - \alpha_k \cdot \nabla_{\Gamma} \tilde{\mathcal{L}}(W_k, \Gamma_k), \\
\Gamma_{k+1} &= \kappa \cdot \text{Prox}_{\Omega_{\lambda}}(V_{k+1})
\end{align*}
\]

V is the sub-gradient for some sparsity-enforced, often non-differentiable regularization such as Lasso or group Lasso penalties for \( \Omega_1(\Gamma) \)

- \( W_t \) follows the gradient descent with \( \ell_2 \)-regularization
- Important Features of \( \Gamma_t \) are first selected: *Inverse Scale Space*

[Fu et al. ICML 2020/TPAMI2022]
Formulations

\[
\frac{\dot{W}_t}{\kappa} = -\nabla_W \tilde{L} (W_t, \Gamma_t) \\
\dot{V}_t = -\nabla_\Gamma \tilde{L} (W_t, \Gamma_t) \\
V_t \in \partial \left( \Omega(\Gamma) + \frac{1}{2\kappa} \|\Gamma\|^2 \right)
\]

\[
\tilde{L}(W, \Gamma) = \hat{L}_n(W) + \frac{1}{2\nu} \|W - \Gamma\|_F^2
\]

V is the sub-gradient for some sparsity-enforced, often non-differentiable regularization \( \Omega_\lambda(\Gamma) = \lambda \Omega_1(\Gamma) \), \( \lambda \in \mathbb{R}_+ \) such as Lasso or group Lasso penalties for \( \Omega_1(\Gamma) \)

- \( W_t \) follows the gradient descent with \( \ell_2 \)-regularization
- Important Features of \( \Gamma_t \) are first selected: Inverse Scale Space

\[ W_{k+1} = W_k - \kappa \alpha_k \cdot \nabla_W \tilde{L}(W_k, \Gamma_k) \]
\[ V_{k+1} = V_k - \alpha_k \cdot \nabla_\Gamma \tilde{L}(W_k, \Gamma_k), \]
\[ \Gamma_{k+1} = \kappa \cdot \text{Prox}_{\Omega_\lambda}(V_{k+1}) \]

[ Fu et al. ICML 2020/TPAMI 2022]
Formulations

\[
\begin{align*}
\dot{\hat{W}}_t &= -\nabla_W \tilde{\mathcal{L}}(W_t, \Gamma_t) \\
\dot{V}_t &= -\nabla_\Gamma \tilde{\mathcal{L}}(W_t, \Gamma_t) \\
V_t &\in \partial \left( \Omega(\Gamma) + \frac{1}{2\kappa} \|\Gamma\|^2 \right)
\end{align*}
\]

\[\tilde{\mathcal{L}}(W, \Gamma) = \hat{\mathcal{L}}_n(W) + \frac{1}{2\nu} \|W - \Gamma\|^2_F\]

Differential Inclusion of Inverse Scale Space

\[\begin{aligned}
W_{k+1} &= W_k - \kappa \alpha_k \cdot \nabla_W \tilde{\mathcal{L}}(W_k, \Gamma_k) \\
V_{k+1} &= V_k - \alpha_k \cdot \nabla_\Gamma \tilde{\mathcal{L}}(W_k, \Gamma_k), \\
\Gamma_{k+1} &= \kappa \cdot \text{Prox}_{\Omega_\lambda}(V_{k+1})
\end{aligned}\]

\[W_{k+1} = W_k - \kappa \alpha_k \cdot \nabla_W \tilde{\mathcal{L}}(W_k, \Gamma_k) ;
V_{k+1} = V_k - \alpha_k \cdot \nabla_\Gamma \tilde{\mathcal{L}}(W_k, \Gamma_k)\]

\[\Gamma_{k+1} = \kappa \cdot \text{Prox}_{\Omega}(V_{k+1})
\]

\[\text{Prox}_{\Omega}(V) = \arg \min_{\Gamma} \left\{ \frac{1}{2} \|\Gamma - V\|^2_F + \Omega(\Gamma) \right\}\]

Gradient Descent

Proximal Mapping

V is the sub-gradient for some sparsity-enforced, often non-differentiable regularization \(\Omega_\lambda(\Gamma) = \lambda \Omega_1(\Gamma), (\lambda \in \mathbb{R}_+)\) such as Lasso or group Lasso penalties for \(\Omega_1(\Gamma)\)

- \(W_t\) follows the gradient descent with \(\ell_2\)-regularization
- Important Features of \(\Gamma_t\) are first selected: Inverse Scale Space

[Fu et al. ICML 2020/TPAMI2022]
Proximal Mapping Controls the Sparsity

$$\text{Prox}_\Omega(V) = \arg \min_{\Gamma} \left\{ \frac{1}{2} \| \Gamma - V \|_2^2 + \Omega(\Gamma) \right\}$$
Proximal Mapping Controls the Sparsity

\[
\text{Prox}_\Omega(V) = \arg \min_{\Gamma} \left\{ \frac{1}{2} \| \Gamma - V \|_2^2 + \Omega(\Gamma) \right\}
\]

DessLBI enforce structural sparsity by Group lasso penalty,

\[
\Omega(\Gamma) = \sum_{g} \| \Gamma^g \|_2 = \sum_{g} \sqrt{\sum_{i=1}^{\frac{\| \Gamma^g \|}{\| V^g \|_2}} (\Gamma^g_i)^2}
\]

A close form solution:

\[
\Gamma^g = \kappa \cdot \max(0, 1 - 1/\| V^g \|_2) \cdot V^g
\]

[Fu et al. ICML 2020/TPAMI2022]
Proximal Mapping Controls the Sparsity

\[
\text{Prox}_\Omega(V) = \arg \min_{\Gamma} \left\{ \frac{1}{2} \| \Gamma - V \|^2_2 + \Omega(\Gamma) \right\}
\]

DessLBI enforce structural sparsity by Group lasso penalty,

\[
\Omega(\Gamma) = \sum_g \| \Gamma^g \|_2 = \sum_g \sqrt{\sum_{i=1}^{\| \Gamma^g \|} (\Gamma^g_i)^2}
\]

Convolutional layer,
\[\Gamma^g = \Gamma^g(c_{in}, c_{out}, \text{size})\]

Fully connected layer
\[\Gamma = \Gamma(c_{in}, c_{out})\]

A close form solution:
\[
\Gamma^g = \kappa \cdot \max(0, 1 - 1/\| V^g \|_2) V^g
\]

\[c_{in} \text{ : No. of input channel} \]

\[c_{out} \text{ : No. of output channel} \]

\[\text{size: kernel size} \]

[Fu et al. ICML 2020/TPAMI2022]
Proximal Mapping Controls the Sparsity

\[ \text{Prox}_\Omega (V) = \arg \min \Gamma \left\{ \frac{1}{2} \| \Gamma - V \|_2^2 + \Omega (\Gamma) \right\} \]

DessLBI enforce structural sparsity by Group lasso penalty,

\[ \Omega (\Gamma) = \sum_g \| \Gamma^g \|_2 = \sum_g \sqrt{\sum_{i=1}^{\| \Gamma^g \|_2^2}^2} \]

Convolutional layer,

\[ \Gamma^g = \Gamma^g (c_{in}, c_{out}, \text{size}) \]

Fully connected layer

\[ \Gamma = \Gamma (c_{in}, c_{out}) \]

• (Batch) DessLBI w./w.o. Momentum and Weight-decay (Mom-Wd)
• We have a theorem that guarantees the global convergence of DessiLBI: from any initialization, DessiLBI sequence converges to a critical point.

[ Fu et al. ICML 2020/TPAMI2022 ]
Visualization of Sparse Filters

Solution path and filter patterns of the 3rd conv. layer of LetNet-5 on MNIST

[Fu et al. ICML 2020/TPAMI 2022]
Winning Tickets by DessiLBI

(c) VGG-16 (Lasso)  
(d) ResNet-50 (Lasso)

[Fu et al. ICML 2020/TPAMI2022]
Winning Tickets by DessiLBI

(c) VGG-16 (Lasso)
(d) ResNet-50 (Lasso)

Train DessiLBI with Early Stopping

[Fu et al. ICML 2020/TPAMI2022]
Winning Tickets by DessiLBI

(c) VGG-16 (Lasso)  
(d) ResNet-50 (Lasso)
Winning Tickets by DessiLBI

(c) VGG-16 (Lasso)

(d) ResNet-50 (Lasso)

Extract $\Gamma$ as subnetwork structure

[Fu et al. ICML 2020/TPAMI2022]
Winning Tickets by DessiLBI

(c) VGG-16 (Lasso)  
(d) ResNet-50 (Lasso)

[Fu et al. ICML 2020/TPAMI2022]
Winning Tickets by DessiLBI

(c) VGG-16 (Lasso)  
ResNet-50 (Lasso) Pruning

Retrain subnetwork

[Fu et al. ICML 2020/TPAMI2022]
Winning Tickets by DessiLBI

(c) VGG-16 (Lasso)

(d) ResNet-50 (Lasso)

[Fu et al. ICML 2020/TPAMI2022]
Winning Tickets by DessiLBI

(c) VGG-16 (Lasso)  
(d) ResNet-50 (Lasso)

[Fu et al. ICML 2020/TPAMI2022]
Winning Tickets by DessiLBI

(c) VGG-16 (Lasso)

(d) ResNet-50 (Lasso)

[Fu et al. ICML 2020/TPAMI2022]
Toolbox: Very Easy to Use

It is install-free, put slbi_opt.py and slbi_toolbox.py into the project folder and import them.

Quick Example to Start with,

```python
python ./example/train/train_lenet.py
```

To initialize the toolbox, the following codes are needed.

```python
from slbi_toolbox import SLBI_ToolBox
import torch
optimizer = SLBI_ToolBox(model.parameters(), lr=lr, kappa=kappa, mu=mu, weight_decay=0)
optimizer.assign_name(name_list)
optimizer.initialize_slbi(layer_list)
```

For training a neural network, the process is similar to one that uses built-in optimizer

```python
optimizer.zero_grad()
loss.backward()
optimizer.step()
```
Training Neural Network

The training process is the same as original Pytorch Optimizer.

ImageNet Training Example

This part of code is included in example/imagenet. To do this demo, run

```
python train_imagenet_slbi.py
```

```
for ep in range(args.epoch):
    model.train()
    descent_lr(args.lr, ep, optimizer, args.interval)
    loss_val = 0
    correct = num = 0
    for iter, pack in enumerate(train_loader):
        data, target = pack[0].to(device), pack[1].to(device)
        logits = model(data)
        loss = F.nll_loss(logits, target)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        _, pred = logits.max(1)
        loss_val += loss.item()
        correct += pred.eq(target).sum().item()
        num += data.shape[0]

if 'z_buffer' in param_state:
    new_grad = d_p * lr_kappa + {p.data - param_state['gamma_buffer']} * lr_kappa / nu
    last_p = copy.deepcopy(p.data)
    p.data.add_(-new_grad)
    param_state['z_buffer'].add_(-lr_gamma, param_state['gamma_buffer'] - last_p)
    if len(p.data.size()) == 2:
        param_state['gamma_buffer'] = kappa + self.shrink(param_state['gamma_buffer'], 1)
    elif len(p.data.size()) == 4:
        param_state['gamma_buffer'] = kappa + self.shrink_group(param_state['z_buffer'])
    else:
        pass
else:
    p.data.add_(-lr_kappa, d_p) 
# For bias update as vanilla sgd

We record the path via two buffer during training.

https://github.com/DessiLBI2020/DessiLBI
Training Neural Network

The training process is the same as original Pytorch Optimizer

ImageNet Training Example

This part of code is included in example/imagenet. To do this demo, run

```python
python train_imagenet_slbi.py
```

It is a network optimizer with

- finding important structural sparsity in model learning,
- Shorter training time,
- Exploring regularization path,
- Nice theoretical properties,
- Good interpretation of important parameters

https://github.com/DessiLBI2020/DessiLBI

```python
for ep in range(args.epoch):
    model.train()
    descent_lr(args.lr, ep, optimizer, args.interval)
    loss_val = 0
    correct = num = 0
    for iter, (data, target) in enumerate(train_loader):
        data, target = pack[0].to(device), pack[1].to(device)
        logits = model(data)
        loss = F.nll_loss(logits, target)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        _, pred = logits.max(1)
        loss_val += loss.item()
        correct += pred.eq(target).sum().item()
        num += data.shape[0]

        if 'z_buffer' in param_state:
            new_grad = d_p * lr_kappa + (p.data - param_state['gamma_buffer']) * lr_kappa / nu
            last_p = copy.deepcopy(p.data)
            p.data.add_(-new_grad)
            param_state['z_buffer'].add_(-lr_gama, param_state['gamma_buffer'] - last_p)
            if len(p.data.size()) == 2:
                param_state['gamma_buffer'] = kappa + self.shrink(param_state['gamma_buffer'], 1)
            elif len(p.data.size()) == 4:
                param_state['gamma_buffer'] = kappa + self.shrink_group(param_state['z_buffer'])
            else:
                pass
        else:
            p.data.add_(-lr_kappa, d_p)#for bias update as vanilla sgd
```

We record the path via two buffer during training
Pruning Neural Network

We can prune the network according to the information of augmented variable $\Gamma$

For pruning a neural network, the code is as follows.

```python
optimizer.update_prune_order(epoch)
optimizer.prune_layer_by_order_by_list(percent, layer_name)
```

Filter Pruning

```python
ts_reshape = torch.reshape(param_state['w_star'], (param_state['w_star'].shape[0], -1))
ts_norm = torch.norm(ts_reshape, 2, 1)
num_selected_filters = torch.sum(ts_norm != 0).item()
param_state['original_params'] = copy.deepcopy(p.data)
p.data = param_state['w_star']
```

Weight Pruning

```python
num_selected_units = (param_state['w_star'] > 0.0).sum().item()
param_state['original_params'] = copy.deepcopy(p.data)
p.data = param_state['w star']
```
Growing Neural Network

We add new filters according to the support set of augmented $\Gamma$, to enlarge the model capacity.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>Params.</th>
<th>Acc(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR10</td>
<td>AutoGrow</td>
<td>4.06 M</td>
<td>94.27</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>2.69 M</td>
<td>94.82</td>
</tr>
<tr>
<td>CIFAR100</td>
<td>AutoGrow</td>
<td>5.13 M</td>
<td>74.72</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>3.37 M</td>
<td>76.86</td>
</tr>
</tbody>
</table>

```python
def grow_filter(model, new_arc, NET, args, logger, topk_dict=None):
    # new_arc: [basic_block, [black_num list], [filter_num list]]
    # layer_num: the layer to be grown
    old_params = {}

    for n, p in model.named_parameters():
        if 'module' in n:
            n = n.join(n.split('!', '/')[1])
        old_params[n] = p.data

    new_net = NET(new_arc[0], new_arc[1], new_arc[2], num_classes=new_arc[3], resolution=new_arc[4])

    for n, p in new_net.named_parameters():
        if n in old_params:
            if p.data.size() == old_params[n].size():
                old_size = old_params[n].size()
                if len(old_size) == 4:
                    try:
                        filter_idx = topk_dict[n]
                        n_out, n_in, k1, k2 = old_size
                        for idx in filter_idx:
                            p.data[idx, in, in, k1, k2] = old_params[idx, i, j, k]
                    except:
                        shortcut_weight
                        n_out, n_in, k1, k2 = old_size
                        p.data[in, out, n_in, k1, k2] = old_params[n]

        elif len(old_size) == 2:
            num_out, num_in = old_size
            p.data[n.num_out, n.num_in] = old_params[n]

        elif len(old_size) == 1:
            a = old_size
            p.data[a] = old_params[n]

        else:
            this layer did not grow
            p.data = old_params[n]

    logger.info('I have succeeded parameters from last model'.format())

    else:
        pass

    return new_net
```
Different from ADMM

\[ W_{k+1} = \arg\max_W \mathcal{L}(W) + \frac{\rho}{2} \| W - \Gamma_k + U_k \|^2 \]

\[ \Gamma_{k+1} = \arg\max_W \Omega(\Gamma) + \frac{\rho}{2} \| W_{k+1} - \Gamma + U_k \|^2 \]

\[ U_{k+1} = U_k + W_{k+1} - \Gamma_{k+1} \]

• Different from ours
  • ADMM targets on convergence result, with objective function: \[ \mathcal{L}(W) + \lambda \cdot \Omega(W) \]
    • DessiLBI is discretization of Differential Inclusion
  • DessiLBI cares the regularized solution path; it returns a sequence of models from simple to complex, corresponding to different regularization parameters;

[Fu et al. ICML 2020/TPAMI2022]
References: Our Works

- Fu et al. Interestingness Prediction by Robust Learning to Rank. ECCV 2014
- Fu et al. Robust Subjective Visual Property Prediction from Crowdsourced Pairwise Labels. IEEE TPAMI 2016
- Wang et al. Instance Credibility Inference for Few-Shot Learning. CVPR 2020
- Wang et al. Scalable Penalized Regression for Noise Detection in Learning with Noisy Labels. CVPR 2022
- Zhao et al. MSplit LBI: Realizing Feature Selection and Dense Estimation Simultaneously in Few-shot and Zero-shot Learning. ICML 2018
- Fu et al. Exploring Structural Sparsity of Deep Networks via Inverse Scale Spaces. IEEE TPAMI accepted (2022)
References

• Tutorial on Sparse and Low-rank Modeling, European Conference on Computer Vision, Firenze, Italy, October 2012
• Bommasani et al. On the Opportunities and Risks of Foundation Models. arx:108.07258, 2021
• ALIGN : Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision, ICML2021, Google Research
• ALBEF : Align before Fuse: Vision and Language Representation Learning with Momentum Distillation, NeurIPS 2021, Salesforce Research
• Florence: A New Foundation Model for Computer Vision, arXiv, Nov. 22, 2021, Microsoft Cloud and AI, Microsoft Research Redmond
• INTERN: A New Learning Paradigm Towards General Vision, arXiv Nov. 16, 2021 Shanghai AI Laboratory, SenseTime, CUKH, SJTU
• Gopher: Scaling Language Models: Methods, Analysis & Insights from Training Gopher, arXiv Dec. 8, 2021, DeepMind
• FLAVA: A Foundational Language And Vision Alignment Model, CVPR 2022, FAIR
• OPT: Open Pre-trained Transformer Language Models. Meta AI 2022
References

- Wright et al. Sharp Thresholds for High-Dimensional and Noisy Sparsity Recovery Using l1-Constrained Quadratic Programming (Lasso). IEEE TIT 2009
- Fan, et al. Partial consistency with sparse incidental parameters. Statistica Sinica, 2018
- Geirhos et al. IMAGENET-TRAINED CNNS ARE BIASED TOWARDS TEXTURE; INCREASING SHAPE BIAS IMPROVES ACCURACY AND ROBUSTNESS. ICLR 2019
References

- Allen-Zhu, Z., Li, Y., and Song, Z. A convergence theory for deep learning via over-parameterization. ICML 2019
Learning for Sparse Optimization
Outlines

1. Inverse scale space method, differential inclusions, linearized bregman iterations and mirror descent
2. Structural sparsity and splitting method
3. Statistical regularization path and model selection consistency
4. Huber’s robust statistics and outlier detection
5. False discovery rate control and (split) Knockoffs
Prof. Wotao Yin: Learning to Optimize (L2O)

Outlines

1. L2O idea and typical work flow.
2. Unrolling a classic algorithm and learning its parameters
3. Generalization and convergence safeguard
4. Learning regularization and plug-and-play
5. Fixed-point network and Jacobian-free back propagation