

# Part3

## 3. Layer-wise Analysis and Design of Deep Neural Networks

### 3.1 Two Data Complexity measures

- Separation index (SI)
- Smoothness index(Sml)

### 3.2 Layer-wise Analysis by Separation and Smoothness indices

- Dataset evaluation, ranking and dividing (SI, Sml)
- Subset Selection (SI, Sml)
- Layer-wise Model evaluation (SI, Sml)
- Pre-train Model ranking (SI, Sml)
- Model Confidence and Guarantee (SI, Sml)

### 3.3 Layer-wise Design by Separation and Smoothness indices

- Model Compressing(SI, Sml)
- Forward learning in the first layer(SI, Sml)
- Layer-wise forward learning(SI, Sml)
- Layer-wise Forward Auto Encoder Learning(Sml)
- Layer-wise branching(SI, Sml)
- Layer-wise Fusion(SI, Sml)
- Forward Design(SI, Sml)
- Forward Multi-Task Design(SI, Sml)
- Artificial Brain Design(SI, Sml)

Indicator	Research (state)
SI, Sml	Initial studies have been done
SI,Sml	There are some prepared/under-review works
SI,Sml	New idea

### 3.4 Related works in local Layer-wise learning

# 3.1 Two Data Complexity measures

## Data Complexity measures

### 3.1.1 Separation index

- First order SI
- High order SI
- High order soft SI
- Center Based SI
- **Self supervised SI**
- **Cross SI**

### 3.1.2 Smoothness index

- First order Sml
- High order Sml
- High order soft Sml
- **Cross Sml**
- **Auto Sml**

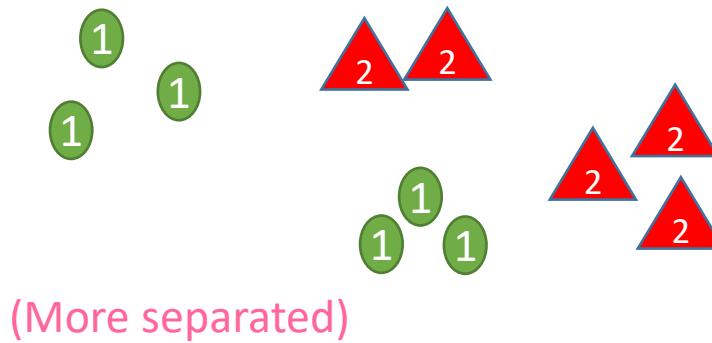
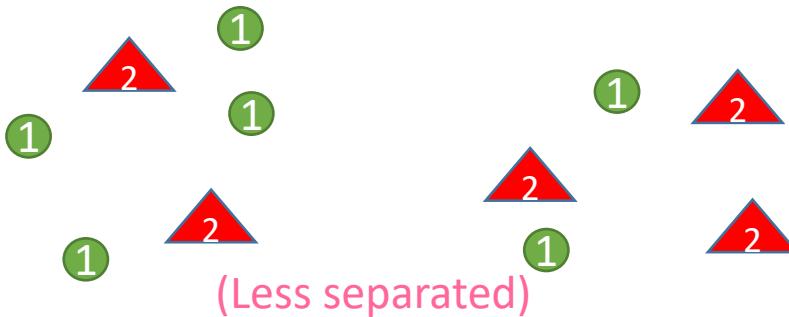
Complexity measures	Overall evaluating approach
✓ Feature-based	Discovering informative features by evaluating each feature independently (Orriols-Puig et al., 2010; Cummins, 2013))
✓ Linearity separation	Evaluating the linearly separation of different classes (Bottou & Lin, 2007)
✓ Neighborhood	Evaluating the shape of the decision boundary to distinguish different classes overlap (Lorena et al., 2012; Leyva et al., 2014)
✓ Network	Evaluating the data dataset structure and relationships by representing it as a graph (Garcia et al., 2015)
✓ Dimensionality	Evaluating the sparsity of the data and the average number of features at each dimension (Lorena et al., 2012; Basu & Ho, 2006)
✓ Class imbalanced	Evaluating the proportion of dataset number between different classes (Lorena et al., 2012)

Table 1. Some complexity measures and their evaluating approaches in a classification problem

# Two Complexity measures

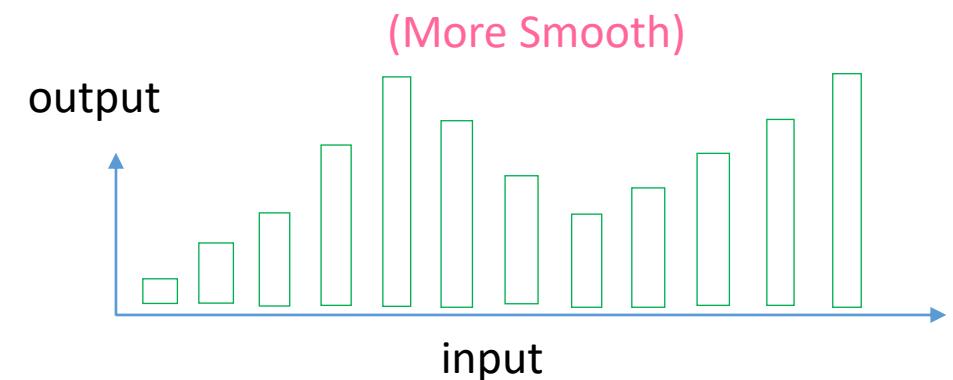
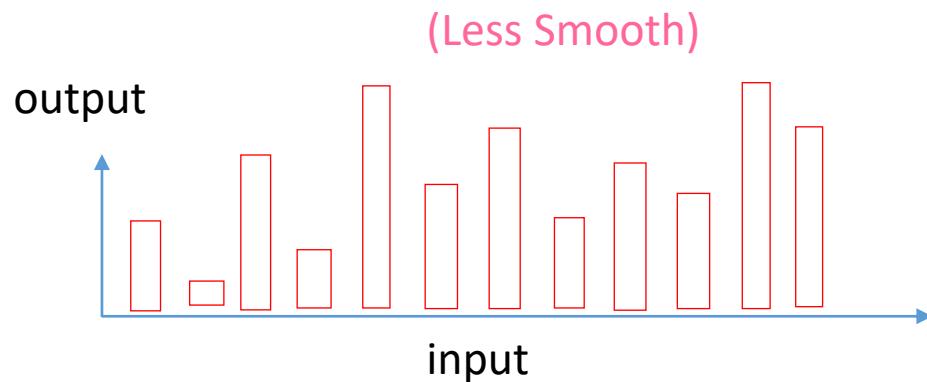
## 1. A separation measure (in classification problems)

It shows that how much input data points separate the labels from each others.



## 2. An smoothness measure (in regression problems)

It shows that how much input data points make the output targets smooth



# Separation index (SI)

“SI” measures that how much input data points separate class labels from each others.

### 3.1.1 Separation index (SI)

#### 1. First order SI

*Data* =  $\{(\mathbf{x}_i, l_i)\}_{i=1}^m \forall i: \mathbf{x}^i \in R^{n \times 1} \quad l_i \in \{1, 2, \dots, n_C\} \quad n_C$ : number of classes

\*it is assumed that “Data” is a measured sample from a domain with high enough diversity.

\* $\mathbf{x}_i$  may have any format (video, image, time series, etc.) ; however, to compute SI, it must be reshaped as a vector.

$$SI(Data) = \frac{1}{m} \sum_{i=1}^m \delta(l_i, l_{i^*})$$

$$i^* = \arg \min_{\forall q \neq i} \|\mathbf{x}_i - \mathbf{x}_q\| \quad \delta(l_i, l_{i^*}) = \begin{cases} 1 & \text{if } l_i = l_{i^*} \\ 0 & \text{else} \end{cases} \quad \text{kronecker delta}$$

\* $\|\cdot\|$  denotes Euclidian distance ( $L_2$  norm) but it may be another distance definition such as  $L_p$  norm:

$$\|\mathbf{x}_i - \mathbf{x}_j\|_{L_p} = \sqrt[p]{\sum_{k=1}^n |\mathbf{x}_i(k) - \mathbf{x}_j(k)|^p}$$

\*\* It is assumed that the input data is normalized at each dimension just before computing separation index.

# Some notes

1. “SI” is a normalized index between zero and one:  $\text{SmI} \in [0,1]$
2.  $SI \rightarrow 1$  (*Separation is maximum*) and  $SI \rightarrow 0$  (*Separation is minimum*)
3. “SI” counts (average of) all data points whose nearest neighbors have the same label
4. “SI” is equal to the accuracy of the nearest neighbor classifier as a non-parametric model. Hence, SI is an informative index having strong correlation with the best accuracy one can access by a model without filter process.
5. SI does not change against shift and scales of data points.  
$$\forall \beta \neq 0, \forall \alpha \neq 0, \forall \mathbf{x}_0, \forall l_0 \quad SI(\{(x^i, l^i)\}_{i=1}^m) = SI(\{(\beta x_i + \mathbf{x}_0, \alpha l_i + l_0)\}_{i=1}^m)$$
6. Separatin index of the target labels with themselves is maximum:  $SI(\{(l_i, l_i)\}_{i=1}^m) = 1$ ; it means that how input data become more similar to labels the separation index will increase.

# Two dimensional examples (binary classification)



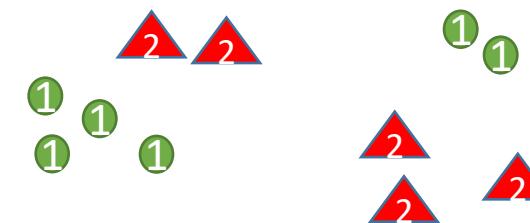
$SI=6/11$



$SI=11/11=1$



$SI=4/11$



$SI=11/11=1$

## Some notes

- To have a high SI, It is enough that **examples of each class become near and near together** in some regions
- **The number of regions** is not important but each region must have at least two members.
- The shape of each region is not important.

# The distance matrix

- To achieve SI, matrix distance of all data points must be computed (to get nearest neighbor for each data point)

$$Data = \{(\mathbf{x}_i, l_i)\}_{i=1}^m \quad \mathbf{x}^i \in R^{n \times 1}$$

Distance matrix:  $D = [d_{ij}] \quad d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$

## Steps

1- Provide data Matrix:  $X = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m]^T, \quad X \in R^{m \times n}$

2-  $M = XX^T, \quad M \in R^{m \times m}$

3-  $d = \text{diag}(M), \quad d \in R^{m \times 1}$

4-  $W = [d, d, \dots, d], \quad W \in R^{m \times m}$

5- Distance matrix is computed as follows:

$$D = W + W^T - 2M$$

# Separation index for Each Class

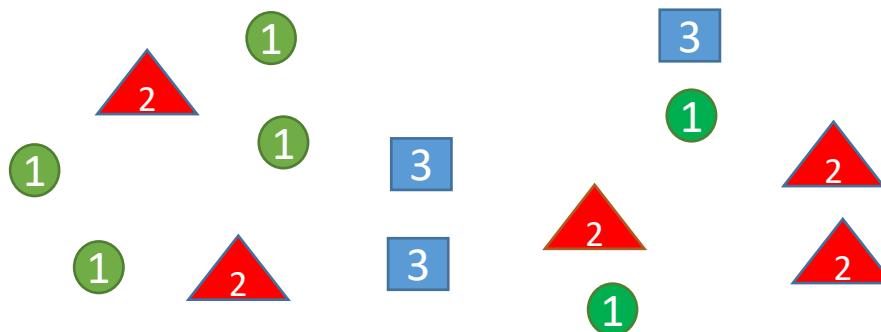
$$SI_c(Data) = \frac{1}{m_c} \sum_i \delta(l_i, c) \delta(l_i, l_{i^*}) \quad c=1,2,\dots,n_c$$

$m_c = \sum_i \delta(l_i, c)$   $m_c$ : number of all data points  $x^i$  which  $l^i = c$

Relation between "total SI" and "SI of classes"

$$SI(Data) = \frac{1}{m} \sum_{c=1}^{n_c} m_c SI_c(Data) \quad \sum_{c=1}^{n_c} m_c = m$$

A two dimensional illustrative example



$$n_c = 3, \quad c = 1,2,3$$

$$SI_1(Data) = 4/6$$

$$SI_2(Data) = 2/5$$

$$SI_3(Data) = 2/3$$

$$SI = (4+2+2)/(6+5+3) = 8/14$$

\* For when for each class  $c$ :  $m_c = \frac{m}{n_c}$  and a sufficient high number of data points are distributed with a *uniformly distributed random* variable then it is expected that  $SI \rightarrow 1/n_c$

## 2. High order SI

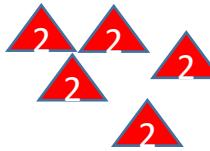
*Data* =  $\{(x_i, l_i)\}_{i=1}^m \forall i: x_i \in R^{n \times 1} \quad l_i \in \{1, 2, \dots, n_C\}$   $n_C$ : number of classes

$$SI^r(Data) = \frac{1}{m} \sum_{i=1}^m \prod_{j=1}^r \delta(l_i, l_{i_j^*}) \quad r: \text{the order of "SI"}$$

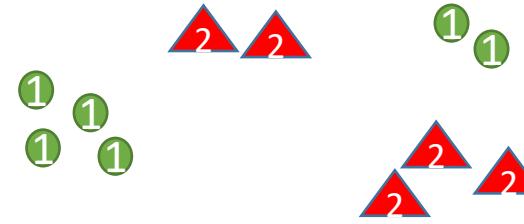
$$i_j^* = \arg \min_{\forall q \neq i, i_1^*, \dots, i_{j-1}^*} \|x_i - x_q\| \quad SI^r \in [0, 1]$$

- “ $SI^r$ ” counts (average of) all data points whose all “ $r$ ” nearest neighbors have the same label
- $SI^r$  considers more restricted condition of separation than  $SI^j$  ( $j < r$ ).
- For each “Data” we have:  $SI^r \leq SI^{r-1} \leq \dots \leq SI^1 = SI$

# Two illustrative Examples



$$\begin{aligned}SI^1 &= 11/11 \\SI^2 &= 11/11 \\SI^3 &= 11/11 \\SI^4 &= 11/11\end{aligned}$$



$$\begin{aligned}SI^1 &= 11/11 \\SI^2 &= 7/11 \\SI^3 &= 4/11 \\SI^4 &= 0\end{aligned}$$

- To increase high order SI, different regions of data points with the same label should merge together and make a hyper-circle shape distribution. In a such case, we will have  $n_C$  hyper-circle shape which can separated, linearly from each other.
- If in a classification problem, the high order SI  $SI^r(r \rightarrow \infty) \rightarrow 1$ , the data points of any pair of classes become more linearly separable.

### 3. High order soft SI

*Data* =  $\{(x_i, l_i)\}_{i=1}^m \forall i: x_i \in R^{n \times 1} \quad l_i \in \{1, 2, \dots, n_C\}$   $n_C$ : number of classes

$$SI_{\text{soft}}^r(Data) = \frac{1}{m \times r} \sum_{i=1}^m \sum_{j=1}^r \delta(l_i, l_{i_j^*})$$

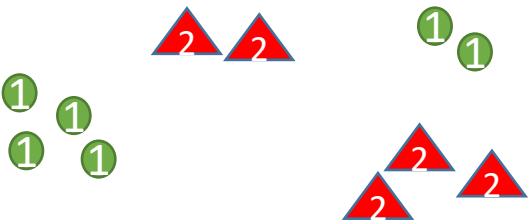
$r$ : the order of SI

$$l_{i_j^*}^* = \arg \min_{\forall q \neq i, l_1^*, \dots, l_{j-1}^*} \|x_i - x_q\| \quad SI_{\text{soft}}^r \in [0, 1]$$

- $SI_{\text{soft}}^r$  considers less restricted condition of separation than  $SI^r$

$$SI_{\text{soft}}^r \geq SI^r \quad \text{and} \quad SI_{\text{soft}}^1 = SI^1$$

# Two illustrative Examples

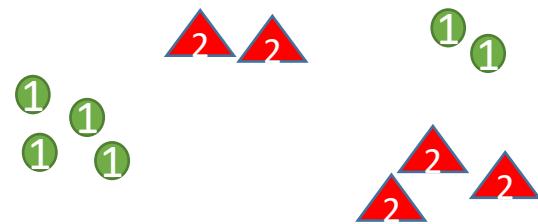


$$SI^1 = 11/11$$

$$SI^2 = 7/11$$

$$SI^3 = 4/11$$

$$SI^4 = 0$$



$$SI_{soft}^1 = 11/11$$

$$SI_{soft}^2 = (4+3+0.5+0.5)/11 = 8/11$$

$$SI_{soft}^3 = (4+3(2/3)+4*(1/3))/11 = 8.33/11$$

$$SI_{soft}^4 = (4*(3/4)+2*(1/4)+2*(1/4)+3*(2/4))/11 = 6.5/11$$

## 4. Center based Separation Index (CSI)

$Data = \{(\mathbf{x}_i, l_i)\}_{i=1}^m \forall i: \mathbf{x}^i \in R^{n \times 1} \quad l_i \in \{1, 2, \dots, n_C\} \quad n_C: \text{number of classes}$

Center of each class is the mean of all input data points having the label of that class:

$$\boldsymbol{\mu}_c = \frac{1}{m_c} \sum_{i=1}^m \mathbf{x}_i \delta(l_i, c), \quad c = 1, 2, \dots, n_C \quad m_c = \sum_{i=1}^m \delta(l_i, c)$$

$$\text{CSI}(Data) = \frac{1}{m} \sum_{i=1}^m \delta(l_i, c^*)$$

$$c^* = \arg \min_{\forall c} \|\mathbf{x}_i - \boldsymbol{\mu}_c\|$$

- CSI is computed much faster than SI because  $n_C \ll m$  and you only need to compute the distance matrix of input data points to center of classes.
- It is suggested to compute CSI instead of SI in cases that examples of each class have all exclusive features of that class and do not have any common feature with examples of other classes.

# Self supervised SI (SSSI)

- $Data = \{(\mathbf{x}_i, ?)\}_{i=1}^m \forall i: \mathbf{x}_i \in R^{n \times 1}$  *Labels are unknown*
- For each  $\mathbf{x}_i$  we generate some augmented data points:  $x_{i_h}, h \in \{1, 2, \dots, n_i\}$
- It is assumed that each  $x_i^j$  inherits at least an exclusive feature of  $\mathbf{x}_i$
- An exclusive feature of  $x_i^j$  is a feature that is sufficient to reveal the label of  $\mathbf{x}_i$ .

$$Data_{\text{aug}} = \overbrace{\{\{(\mathbf{x}_{i_h}, i)\}_{h=1}^{n_i}\}_{i=1}^m}^{\text{i-th (self) class}}$$

$$\text{SSSI}^r(Data) = \text{SI}^r(Data_{\text{aug}}), \quad n_C = m, \quad m_{\text{aug}} = \sum_{i=1}^m n_i$$

# Cross SI

$$Data = \{(\mathbf{x}_i, l_i)\}_{i=1}^m \quad D_{test} = \{(\check{\mathbf{x}}_i, \check{l}_i)\}_{i=1}^{m_{test}}$$

$$SI_{cross}(D_{test}, Data) = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} \delta(\check{l}_i, l_{i^\#})$$

$$i^\# = \arg \min_{\forall q} \|\check{\mathbf{x}}_i - \mathbf{x}_q\|$$

Cross SI measures the separation index of a test domain of dataset  $D_{test}$  based on the main domain of dataset  $Data$ .

It can be shown that:

$$SI_{cross}(D_{test}, Data) = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} SI_{cross}((\check{\mathbf{x}}_i, \check{l}_i), Data)$$

# Smoothens index (Sml)

Sml measures how much input data points make the output targets smooth

## 3.1.2 Smoothness index (SI)

A smoothness measure for regression problem

### 1. First order SI

$Data = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m \quad \forall i: \mathbf{x}_i \in R^{n \times 1}, \mathbf{y}_i \in R^{o \times 1}$   $o$  :number of outputs

\*it is assumed that Data is a measured sample with high enough diversity.

\* $\mathbf{x}_i$  and  $\mathbf{y}_i$  may have any format (video, image, time series, etc.) ; however, to compute Sml, it must be reshaped as a vector.

$$Sml(Data) = \frac{1}{m} \sum_{i=1}^m \left( \frac{d_{imax} - d_{i^*}}{d_{imax} - d_{imin}} \right)$$

$$i^* = \arg \min_{\forall q \neq i} \|\mathbf{x}_i - \mathbf{x}_q\| \quad d_{imax} = \max_{\forall q} \|\mathbf{y}_i - \mathbf{y}_q\| \quad d_{imin} = \min_{\forall q \neq i} \|\mathbf{y}_i - \mathbf{y}_q\|$$
$$d_{i^*} = \|\mathbf{y}_i - \mathbf{y}_{i^*}\|$$

\* $\|\cdot\|$  denotes Euclidian distance ( $L_2$  norm) but it may be another distance definition such as  $L_p$  norm.

\*\* It is assumed that the input and target output data are normalized at each dimension just before computing the smoothness index.

\*\*\* to avoid the effect of outliers, it is assumed that all outlier of data points are not considered in computing  $Sml$ .

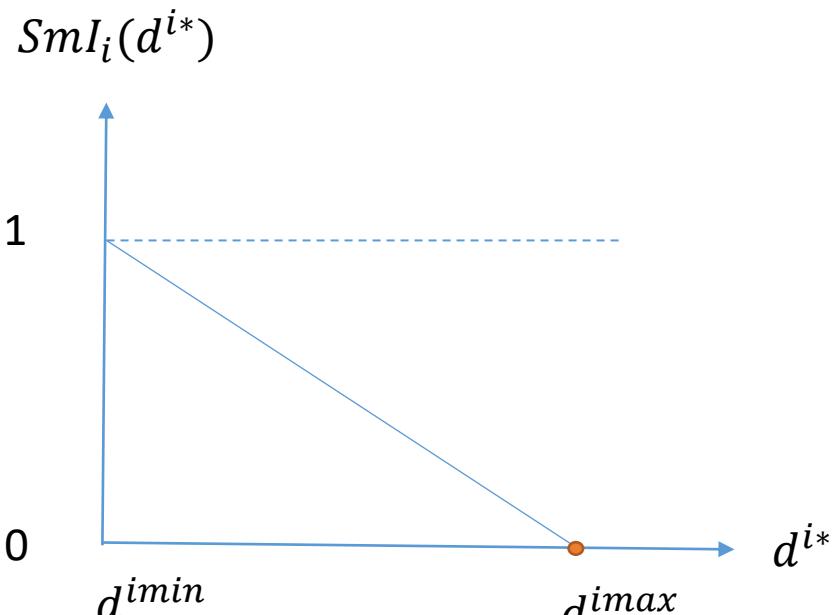
# A modified SmI (Exponential SmI)

$$\text{SmI}(Data) = \frac{1}{m} \sum_{i=1}^m \text{SmI}^i \quad \text{SmI}^i = \exp\left(-\gamma \frac{\mathbf{d}^{i*} - \mathbf{d}^{imin}}{\mathbf{d}^{imean}}\right)$$
$$\mathbf{d}^{imean} = \frac{1}{m} \sum_{q=1}^m \|\mathbf{y}^i - \mathbf{y}^q\| \quad \text{Smoothness rate } \gamma > 0$$

Some Notes:

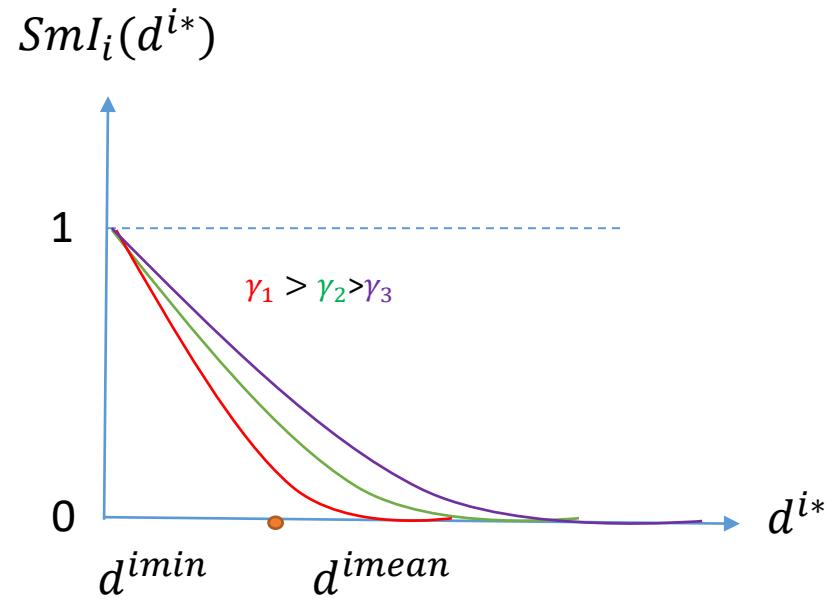
1. Exponential SmI is not sensitivities to the out layers.
2. For when  $\gamma \rightarrow \infty$ , any distance variation:  $\left(\frac{\mathbf{d}^{i*} - \mathbf{d}^{imin}}{\mathbf{d}^{imean}}\right)$  changes the smoothness, considerably.
3. Here we have exponential smoothness) versus linear influence in former definition.

# Sml Diagrams versus $d^{i*}$



Linear Sml

$$d^{i*} = \|y_i - y_{i^*}\|$$
$$d_{imax} = \max_{\forall q} \|y_i - y_q\|$$
$$d_{imin} = \min_{\forall q \neq i} \|y_i - y_q\|$$

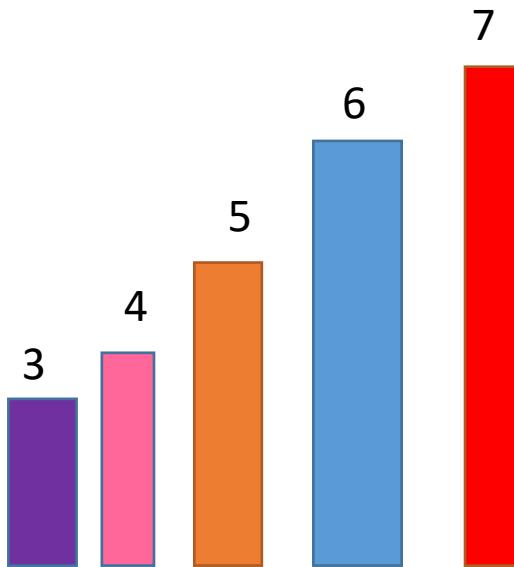


Exponential Sml

# Some notes

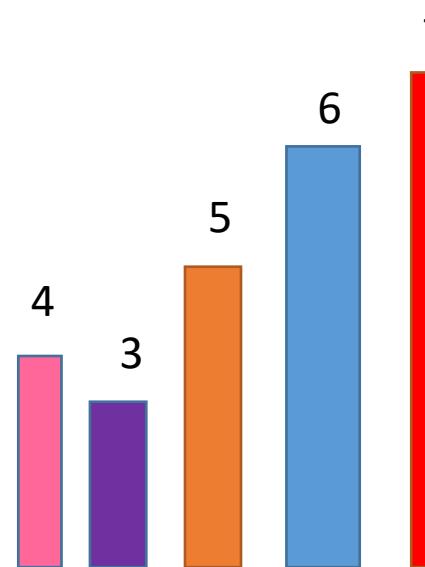
1. “Sml” is a normalized index between zero and one:  $\text{SmI} \in [0,1]$
2.  $\text{SmI} \rightarrow 1$  (*Smoothness is maximum*) and  $\text{SmI} \rightarrow 0$  (*Smoothness is minimum*)
3. “Sml” measures that how nearness of input data leads to nearness of target data.
4. Assuming, the target outputs are outputs of a classification problem in “one-hot” format, SmI is actually measure the separation index:  $\text{SmI} = \text{SI}$
5. Increasing the number of classes and considering a nearness among every two classes, SI is interpreted as a smoothness index. Actually, SmI shows in average that how neighboring examples in input space have classes with near distances in output.
6. SmI does not change for arbitrary position shift and (scalar) scale of the data  
$$\forall \beta \neq 0, \forall \alpha \neq 0, \forall x_0, \forall y_0 \quad \text{SmI}(\{(x^i, y^i)\}_{i=1}^m) = \text{SmI}(\{(\beta x_i + x_0, \alpha y_i + y_0)\}_{i=1}^m)$$
7. Smoothness index of target outputs *with themselves is maximum*:  $\text{SmI}(\{(y_i, y_i)\}_{i=1}^m) = 1$ ; it means that how input data become more similar to output the smoothness index will increase.

# One-dimensional illustrative examples



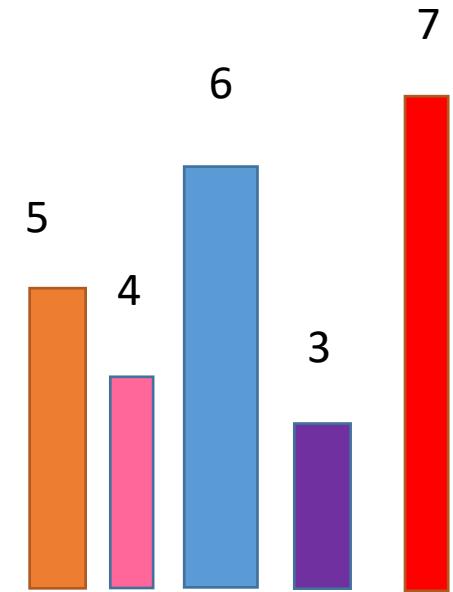
$$SmI = \frac{1}{5} \left( \frac{4-1}{4-1} + \frac{3-1}{3-1} + \frac{2-1}{2-1} + \frac{3-1}{3-1} + \frac{4-1}{4-1} \right)$$

$$SmI = 1$$



$$SmI = \frac{1}{5} \left( \frac{3-1}{3-1} + \frac{4-1}{4-1} + \frac{2-2}{2-1} + \frac{3-2}{3-1} + \frac{4-1}{4-1} \right)$$

$$SmI=0.7$$



$$SmI = \frac{1}{5} \left( \frac{2-1}{2-1} + \frac{3-1}{3-1} + \frac{3-2}{3-1} + \frac{4-3}{4-1} + \frac{4-4}{4-1} \right)$$

$$SmI=0.566$$

## 2. High order SmI

*Data* =  $\{(x_i, y_i)\}_{i=1}^m \forall i: x_i \in R^{n \times 1}, y_i \in R^{o \times 1}$

$$\text{SmI}^r(\text{Data}) = \frac{1}{m} \sum_{i=1}^m \min_{\forall j \in \{1, \dots, r\}} \left( \frac{d_{imax} - d_{i_j^*}}{d_{imax} - d_{imin_j}} \right) \quad \text{r: the order of "SmI"}$$

$$i_j^* = \arg \min_{\forall q \neq i, i_1^*, \dots, i_{j-1}^*} \|x_i - x_q\| \quad imin_j = \arg \min_{\forall q \neq i, imin_1, \dots, imin_{j-1}} \|y_i - y_q\|$$

$$d_{imin_j} = \|y_i - y_{imin_j}\| \quad d_{i^*} = \|y_i - y_{i_j^*}\|$$

- $\text{SmI}^r \in [0, 1]$
- $\text{SmI}^r$  considers more restricted condition of smoothness than  $\text{SmI}^j$  ( $j < r$ ).
- For each “Data” we have:  $\text{SmI}^r \leq \text{SmI}^{r-1} \leq \dots \leq \text{SmI}^1 \quad \text{SmI}^1 = \text{SmI}$

### 3. High order soft SmI

*Data* =  $\{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^m \quad \forall i: \mathbf{x}^i \in R^{n \times 1} \quad \mathbf{y}^i \in R^{o \times 1}$

$$\text{SmI}_{\text{soft}}^r(\text{Data}) = \frac{1}{m \times r} \sum_{i=1}^m \sum_{j=1}^r \left( \frac{d_{imax} - d_{ij}^*}{d_{imax} - d_{imin_j}} \right) \quad j = 1, 2, \dots, r \quad r: \text{the order of "SmI"}$$

- $\text{SmI}_{\text{soft}}^r \in [0, 1]$
- $\text{SmI}_{\text{soft}}^r$  considers less restricted condition of smoothness than  $\text{SmI}^r$

$$\text{SmI}_{\text{soft}}^r \geq \text{SmI}^r \quad \text{and} \quad \text{SmI}_{\text{soft}}^1 = \text{SmI}^1$$

## 4. Cross SmI

$$Data = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \quad D_{test} = \{(\check{\mathbf{x}}_i, \check{y}_i)\}_{i=1}^{m_{test}}$$

$$SmI_{cross}(D_{test}, Data) = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} \left( \frac{d_{imax} - d_{i^\#}}{d_{imax} - d_{imin_j}} \right)$$

$$i^\# = \arg \min_{\forall q} \|\check{\mathbf{x}}_i - \mathbf{x}_q\|$$

Cross SI measures the separation index of a test domain of dataset  $D_{test}$  based on the main domain of dataset  $Data$ .

It can be shown that:

$$SmI_{cross}(D_{test}, Data) = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} SmI_{cross}((\check{\mathbf{x}}_i, \check{y}_i), Data)$$

## 5. Auto Sml Matrix

$$Data = \{(v_1, v_2, \dots, v_m)\}_{i=1}^m \quad v_i \in R^{n \times 1}$$

$$SmI_{Auto}(Data) = \Omega_{m \times m} = [\omega_{ij}]$$

$$\omega_{ij} = \left( \frac{d_{jmax} - d_{ij}}{d_{jmax}} \right)$$

$$d_{ij} = \|v_j - v_i\|$$

$$d_{jmax} = \max_{\forall j_1} \|v_j - v_{j_1}\|$$

$SmI_{Auto}(v_i, v_j)$  measures how a vector  $v_i$  predicts the vector  $v_j$

Two notes:

$$1. \quad \omega_{ii} = 1$$

$$2. \quad \omega_{ij} \rightarrow 1 \text{ if } \left( \frac{d_{ij}}{d_{jmax}} \right) \rightarrow 0$$

# Homework

- Show that

$$\begin{aligned} SI^r(\{(x^i, l^i)\}_{i=1}^m) &= SI^r(\{(\Psi_1 x_i, \Psi_2 l_i)\}_{i=1}^m) \\ SmI^r(\{(x^i, y)\}_{i=1}^m) &= SmI^r(\{(\Psi_1 x_i, \Psi_2 y_i)\}_{i=1}^m) \end{aligned}$$

where

$\Psi_h$  ( $h=1,2$ ) have orthogonal columns with equal norms.

## 3.2 Analysis by Separation and Smoothness indices

3.2.1 Dataset evaluation, ranking and dividing

3.2.2 Subset Selection

3.2.3 Layer-wise Model evaluation

3.2.4 Pre-train Model ranking

3.2.5 Model Confidence and Guarantee

3.2.6 Causal relationship between two variables

### 3.2.1 Dataset evaluation, ranking and dividing

Here, based on “SI”, some methods are proposed for dataset evaluation, ranking and dividing in classification and regression problems.

# Dataset Evaluation

- Assume that a dataset:  $Data = \{(x_i, y_i)\}_{i=1}^m$  is provided for training a model in a classification ( $y_i \equiv l_i$ ) or regression problem.
- We would like to know how such a dataset is challenging and which model is more appropriate for it.

**Algorithm1: (To suggest deep or shallow for a classification or regression problem with a given dataset)**

1. Compute  $SI(Data)$  ( $SmI (Data)$ ) of the dataset.
2. If  $SI(Data)$  ( $SmI (Data)$ ) is nearer to one than to zero, the provided data is less challenging and a shallow model is suggested for the problem.
3. If  $SI(Data)$  ( $SmI (Data)$ ) is nearer to zero, the provided data is less challenging and a deep learning model with high enough complexity is suggested for the problem.

# SI index for some known datasets

TABLE II  
Evaluation of some known classification datasets by using the separation index.

<b>Dataset</b>	<b>Number of Classes</b>	<b>Separation Index</b>	$SI_w^{*,**}$
MNIST Digits	10	0.97372	0.10
MNIST Fashion	10	0.54423	0.10
Cifar-10	10	0.2636	0.10
Cifar-100	100	0.17446	0.01

\*The expected SI is equal to  $SI_w = 1/n_C$  for when (1) each class has equal number of examples and (2) all examples are distributed with uniform random variable.

\*\* to have fair comparison among SI of different data set the it is suggested to normalize in number of classes ( $n_C$ )

$$A suggestion: SI_n = \frac{SI - 1/n_C}{1 - 1/n_C}$$

# The sensitivity of SI to the number of data points in a data-set

- Actually, the  $SI(Sml)$  is suggested to be used for a standard data-set with high enough diversity.
- For a data-set with a very low number of data points (insufficient diversity),  $SI(Sml)$  changes non-smoothly versus number of data points. In such a state, it does not show the true complexity of the data, and the sensitivity to variation of number of data points is high.
- For a data-set, while the number of data points is high enough (sufficient diversity), the  $SI(Sml)$  changes more smoothly versus number of data points (low sensitivity) .

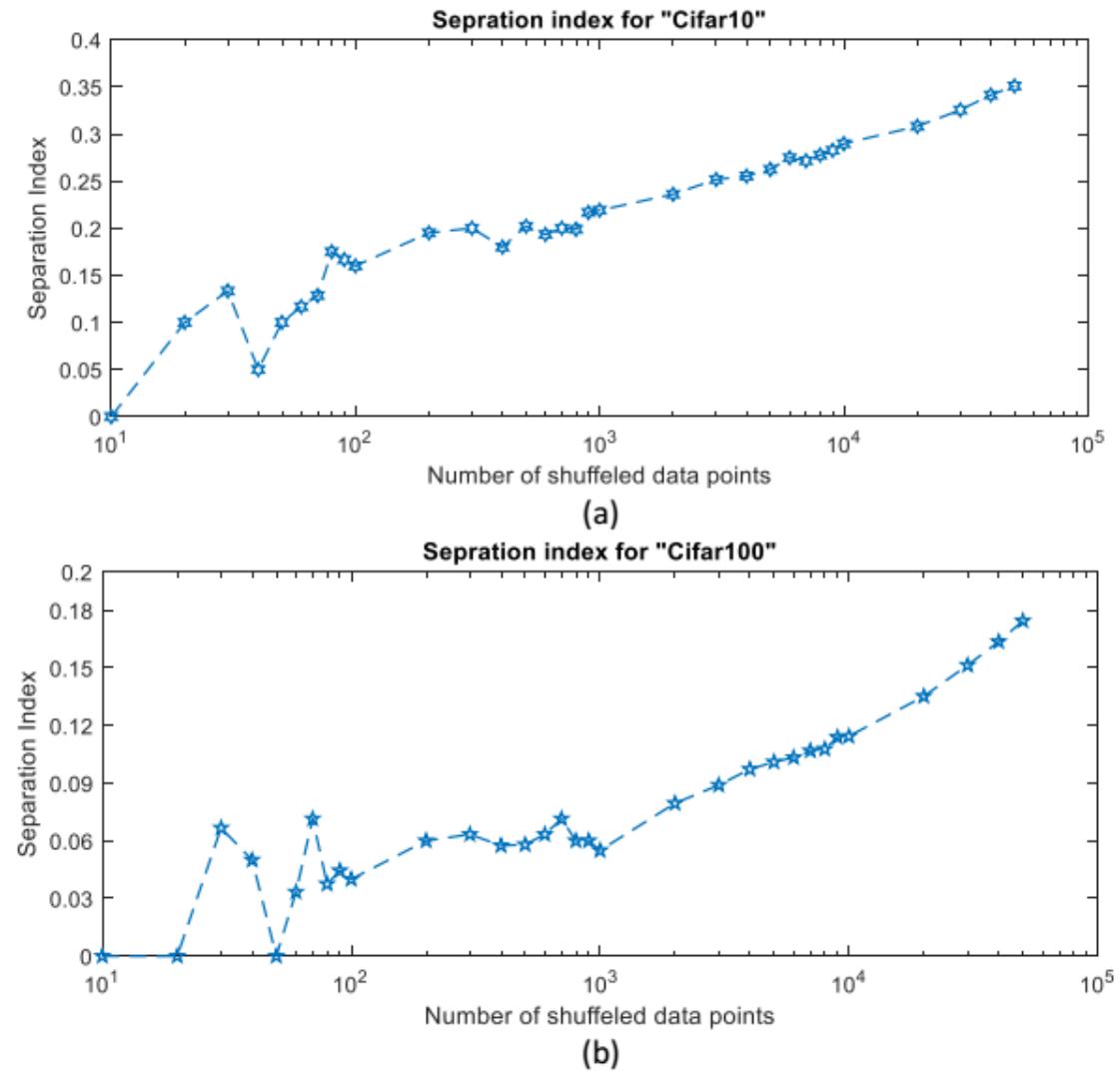


Fig. 6. The plot of separation index versus different number of shuffled data points in both "cifar10" (a) and "cifar100" (b).

# Dataset ranking

- Computing  $SI(Data)$  ( $SmI(Data)$ ) provides a solution to rank and compare standard provided datasets from challenging view point.

Cifar 100 > Cifar 10 > MNIST – Fashion > MNIST - Digits

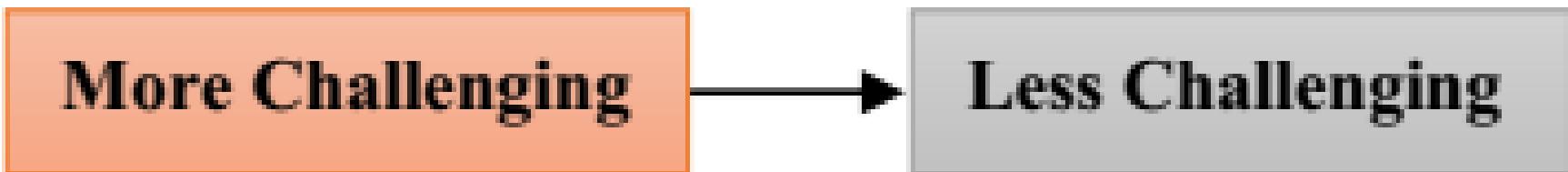


Fig. 4. The ordered classification data sets from more challenging to less challenging.

# Cross domain dataset evaluation

## 1. Classification Problems

$$Data = \{(\mathbf{x}_i, l_i)\}_{i=1}^m \quad D_{test} = \{(\check{\mathbf{x}}_i, \check{l}_i)\}_{i=1}^{m_{test}}$$

$$SI_{cross}(D_{test}, Data) = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} \delta(\check{l}_i, l_{i^*})$$

$$i^* = \arg \min_{\forall q} \|\check{\mathbf{x}}_i - \mathbf{x}_q\|$$

- ❖ if  $SI_{cross}(D_{test}, Data) \gg SI(Data)$  , then it is expected that *the training model (with "Data") will have high generalization for  $D_{test}$ .*
- ❖ if  $SI_{cross}(D_{test}, Data) \ll SI(Data)$  , then it is expected that *the training model (with "Data") will have low generalization for  $D_{test}$ .*
- ❖ The test data set is called homogenous with the training dataset when  $SI_{cross}(D_{test}, Data) \approx SI(Data)$

## 2. Regression Problems

$$Data = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \quad D_{test} = \{(\bar{\mathbf{x}}_i, \bar{y}_i)\}_{i=1}^{m_{test}}$$

$$SmI_{cross}(D_{test}, Data) = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} \left( \frac{d_{imax\#} - d_{i\#}}{d_{imax\#} - d_{imin\#}} \right)$$

$$i^\# = \arg \min_{\forall q} \|\bar{\mathbf{x}}_i - \mathbf{x}_q\| \quad d_{imax\#} = \max_{\forall q} \|\bar{y}_i - y_q\| \quad d_{imin\#} = \min_{\forall q} \|\bar{y}_i - y_q\| \quad d_{i\#} = \|\bar{y}_i - y_{i\#}\|$$

- ❖ if  $SmI_{cross}(D_{test}, Data) \gg SmI(Data)$  , then it is expected that *the training model (with "Data") will have high generalization for  $D_{test}$ .*
- ❖ if  $SmI_{cross}(D_{test}, Data) \ll SmI(Data)$  , then it is expected that *the training model (with "Data") will have low generalization for  $D_{test}$ .*
- ❖ The test data set is called homogenous with the training dataset when  $SmI_{cross}(D_{test}, Data) \approx SmI(Data)$

# Data dividing for test and training datasets

- To have high enough generalization, divide an available dataset to test and training sets in order that the  $SI_{cross}(SmI_{cross})$  of test dataset becomes almost equal to  $SI(SmI)$  of the training dataset.

$$Data_{available} \rightarrow \{D_{test}, Data\}$$

1. For classification problems

$$SI_{cross}(D_{test}, Data) \sim SI(Data)$$

2. For regression problems

$$SmI_{cross}(D_{test}, Data) \sim SmI(Data)$$

### 3.2.2 Subset Selection

Among available inputs, which ones should be selected?

Among different observations which ones should be integrated?

# Subset Selection among distinct inputs

- Assume there is  $x_{available} = \{x_1, \dots, x_{n_e}\}$  with  $n_e$  inputs.
- Among available  $n_e$  inputs, select a subset  $\mathbf{x} \subseteq x_{available}$  and define:  
$$Data = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$$
- To decrease the complexity, select  $\mathbf{x}$  in a way that the  $SI(Data)$  ( $SmI(Data)$ ) becomes maximum.
- It is aimed to remove that all non-relevant, redundancies and noise inputs, which decrease the SI(Sml) or do not increase it.
- “Forward selection”, “backward elimination” or any other “step-wise selection” algorithms can be used for this purpose.

# Choosing effective inputs by Smoothness Index

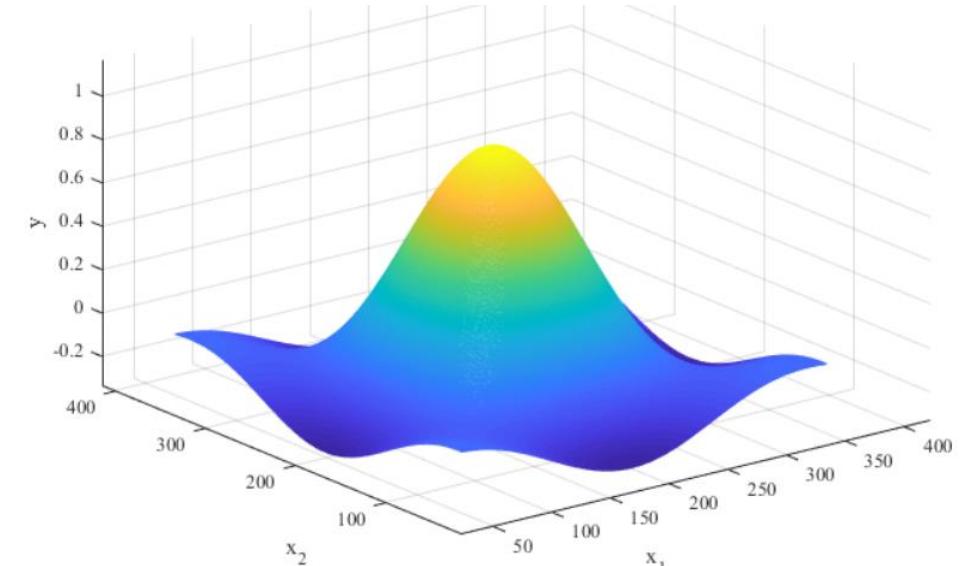
## Example1 (illustrative)

**Table 4.** SmI comparison for different subsets of handmade data

Different subsets of two main inputs and two non-related inputs			Feature Smoothness index		
Subsets /	Inputs		Linear	Exponential	
1	x	x	0.9783	0.9788	
2	x	x	0.9159	0.9249	
3	x	x	0.8314	0.8615	
4		x	0.4781	0.5972	
5		x	0.4711	0.5888	
6			0.3464	0.4929	

white noise variables with X3, and X4 features have uniform distribution

Two-dimensional synchronous function of 1000 randomly generated data points.



$$y = \frac{\sin(x_1) \sin(x_2)}{x_1 x_2},$$

$$0 < |x_1| \leq 5, 0 < |x_2| \leq 5, 0 < |x_3| \leq 5, 0 < |x_4| \leq 5$$

While we have relevant inputs the SmI is maximum so the subset selection by SmI reveals the relevant inputs.

# Choosing effective inputs by Smoothness Index

## Example2

**Table 8.** Performance evaluation using *MSE* for all models ( $\times 10^6$ )

	PCA	GUS	RFE	KBS	VT	PCC	MI	FSSmI
<b>MLR</b>	<b>0.2786</b>	0.3031	0.2764	0.2726	0.2470	0.2687	0.2655	0.2495
<b>RFR</b>	0.4912	<b>0.2347</b>	<b>0.2306</b>	0.2714	0.2520	0.3034	0.2901	<b>0.2301</b>
<b>SVR</b>	0.2916	0.2600	0.2501	<b>0.2365</b>	<b>0.2301</b>	<b>0.2400</b>	<b>0.2391</b>	0.2555
<b>KNN</b>	0.6696	0.5080	0.2576	0.3290	0.3264	0.3100	0.3337	0.2581

**Table 7.** Performance evaluation using *MAE* for all models ( $\times 10^4$ )

	PCA	GUS	RFE	KBS	VT	PCC	MI	FSSmI
<b>MLR</b>	<b>0.5232</b>	0.5499	0.5220	0.5219	0.4969	0.5179	0.5105	0.5081
<b>RFR</b>	0.6965	0.4816	0.4796	0.5203	0.5014	0.5495	0.5351	0.4986
<b>SVR</b>	0.5386	0.5066	0.4975	<b>0.4857</b>	<b>0.4781</b>	<b>0.4874</b>	<b>0.4868</b>	0.5162
<b>KNN</b>	0.8168	0.7123	0.5381	0.5727	0.5707	0.5555	0.5728	0.5167

### Selectin Algorithms

Forward selection based SmI (FSSmI)

Principle Component Analysis (PCA)

Recursive feature elimination (RFE)

Generic uni-varient selection (GUS)

Mutual Information (MI)

K-best selection (KBS)

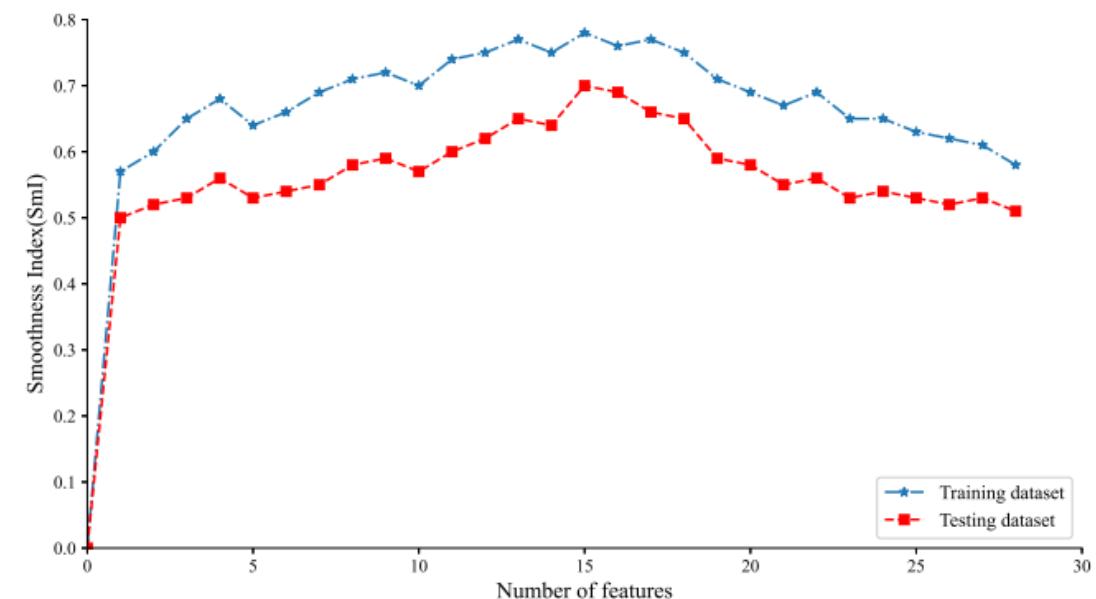
Pearson correlation coefficient (PCC)

Variance threshold (VT)

### Models

Support vector regression (SVR)  
 Multiple linear regression (MLR)  
 K nearest neighbors (KNN)  
 Random forest regression (RFR)

Yearly residential water consumption data, along with climatic characteristics, and socioeconomic factors of rural areas of Isfahan, Iran are aggregated.

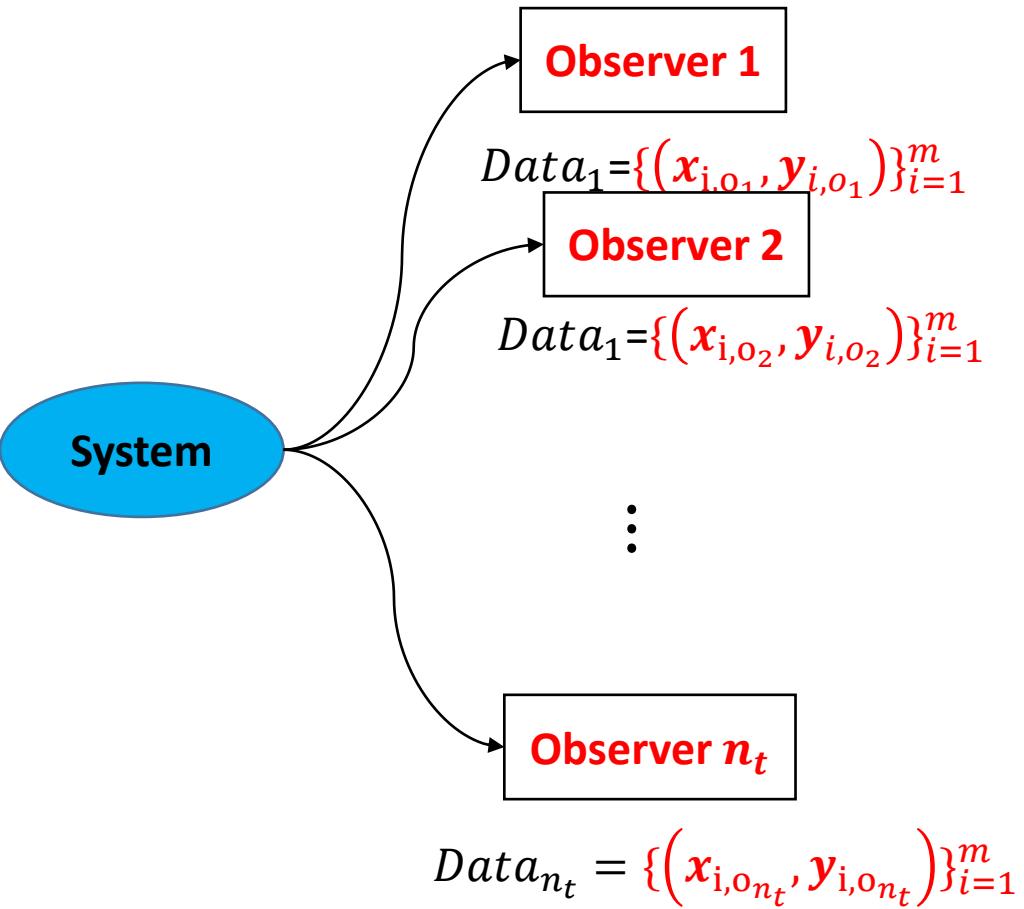


**Fig. 5.** Smoothness Index (SmI) based on number of features. There is a good correlation between the smoothness charts of training and test datasets, i.e., the selected features based on the absent data are the same as those that give the highest SmI in the training dataset.

Feature	KBS	VT	PCC	MI	GUS	FSSmI	RFE on MLR	RFE on RFR	RFE on SVR	Lasso	Ridge	Elastic
Subscriptions	×	×	×	×	×	×	×	×	×	×	×	×
Households	×	×	×	×	×	×	×	×	×	×	×	×
Average family size						×		×		×		
Female ratio	×	×	×	×	×	×	×	×	×	×	×	×
Age 0 to 9					×	×	×		×	×		
Age 10 to 19					×		×					
Age 20 to 29								×	×			
Age 30 to 39								×	×	×		
Age 40 to 49								×	×	×		
Age 50 to 59							×					
Age 60 to 69						×		×		×	×	
Age 70+						×	×	×	×	×	×	
Literacy rate						×					×	×
Employment rate						×	×					
Owner-occupied housings	×	×	×	×		×	×	×	×	×	×	×
Non-owner-occupied housings	×	×	×	×	×						×	×
Non-apartment housings						×		×			×	×
Area 50- m2		×						×			×	×
Area 51 to 75 m2	×	×	×	×					×		×	×
Area 76 to 80 m2	×	×	×	×				×			×	×
Area 81 to 100 m2	×	×	×	×								×
Area 101 to 150 m2	×	×	×	×			×	×			×	×
Area 151 to 200 m2	×	×	×	×							×	×
Area 201 to 300 m2	×	×	×	×	×					×		×
Area 301 to 500 m2		×	×		×	×				×	×	×
Area 501+		×			×	×						
Max temperature		×	×	×		×	×		×		×	×
Summer temperature		×	×	×	×	×	×	×	×	×	×	×
CDD	×	×	×	×	×	×	×	×	×	×	×	×
Number of features	12	17	15	14	14	15	15	15	15	11	16	18

**Table 5.** Selected features.  
The features, households, subscriptions, and female ratio, are selected by all the feature selection methods that show their influences on regression

# Subset selection among distinct observations



- It is aimed to select  $n_s$  observations from available  $n_t$  observations and then concatenate them (for equal events) or append (for different events), which can maximize SI (Sml).

concatenation  $x_i^* = [x_{i,o_1^*}, \dots, x_{i,o_{n_s}^*}]$ ,  $y_{i,o_1} = y_{i,o_2} \dots = y_{i,o_{n_t}}$

Appending  $x_i^* = \begin{bmatrix} x_{i,o_1^*} \\ \vdots \\ x_{i,o_{n_s}^*} \end{bmatrix}$

*For classification problems*

$$SI(\{(x_i^*, l_i)\}_{i=1}^m) \geq SI(\{\tilde{x}_i, l_i\}_{i=1}^m)$$

*or for regression problems*

$$Sml(\{(x_i^*, y_i)\}_{i=1}^m) \geq Sml(\{\tilde{x}_i, y_i\}_{i=1}^m)$$

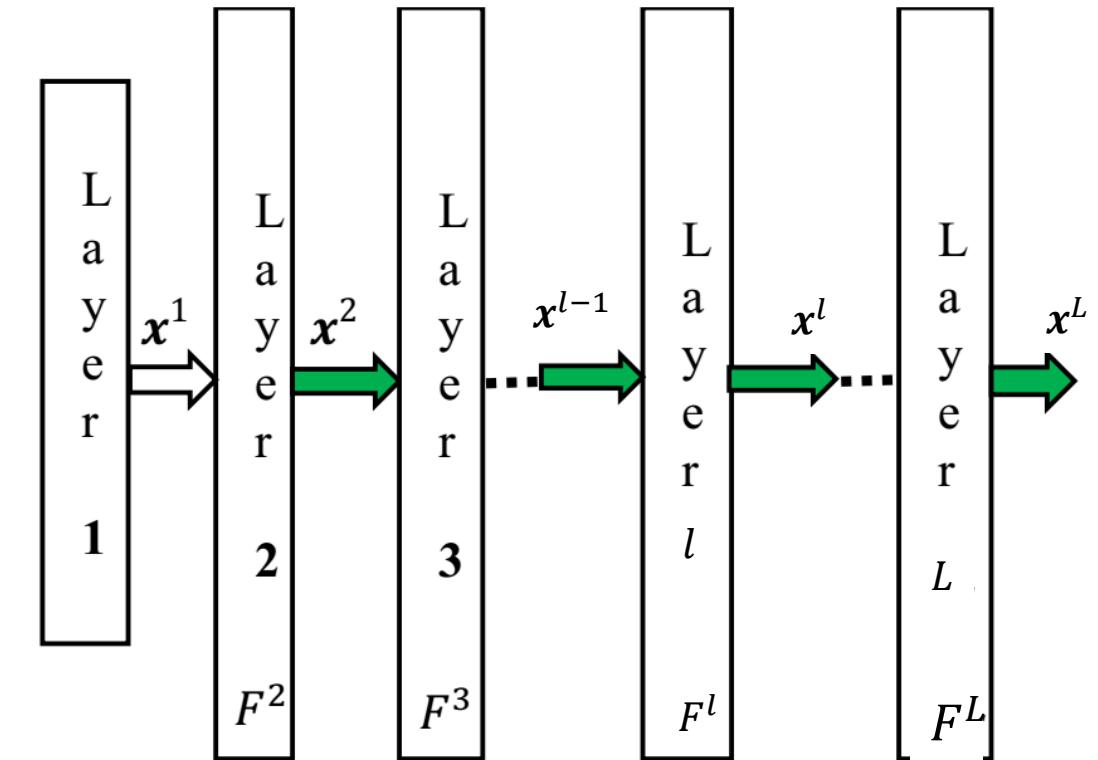
**where**  $\tilde{x}$  denotes any other concatenation or appending from available  $n_t$  different observations.

- “Forward selection”, “backward elimination” or any other “step-wise selection” algorithms can be used for this purpose.

### 3.2.3 Layer-wise Model evaluation

# The concept of dataflow or information-flow

- By applying the input data,  $\mathbf{x}$  ( $\mathbf{x} \equiv x^1$ ) to a deep neural network with  $N_L$  layers, the data will be transformed layer by layer.
- Dataflow (information-flow) denotes the data which transform by layers of a deep neural network :  
$$x^1 \rightarrow \dots x^{l-1} \rightarrow x^l \rightarrow \dots \rightarrow x^L$$
- $L$ : number of layers in the model
- $x^l$  denotes the dataflow at layer  $l$ , which is reshaped as a vector and its length is  $n_L$ .  
$$x^l \in \mathbb{R}^{n_l \times 1}$$
- One can compute SI(SMI) for dataflow at layer  $l$ :  
$$\text{Data}^l = \{(x_i^l, l_i)\}_{i=1}^m \quad l=1, 2, \dots, L$$



It seems the above DNN is a feedforward network.  
However, each layer can be a RNN or a LSTM module, too.  
In the case of RNN or LSTM, it is assumed that the hidden state  
is within the layer.

# The complexity measure SI (Sml) in DNNs

## Some definitions

- **Disturbance:** non relevant information which disturb the feature space
- **Distortion:** Uncertainties due to (1) inherent appearances of the features, (2)the environment constraints on the measuring process, and (3) different measuring parameters.
- **Common features:** In classification problems, features which are common between examples of different classes they avoid discrimination between different classes.
- **Exclusive features:** Features which discriminate different classes in a classification problem or maximize smoothness in regression problems.

## Two important notes:

1. In a DNN, it is expected that after a certain number of filter layers, a feature space with negligible disturbance, distortion, and common features is appeared.
2. Using a metric formulation, it is proved that by intensifying exclusive features and weakening disturbances, distortions, and common features (complexity), the SI(Sml) of dataflow will increase gradually through layers of a DNN.

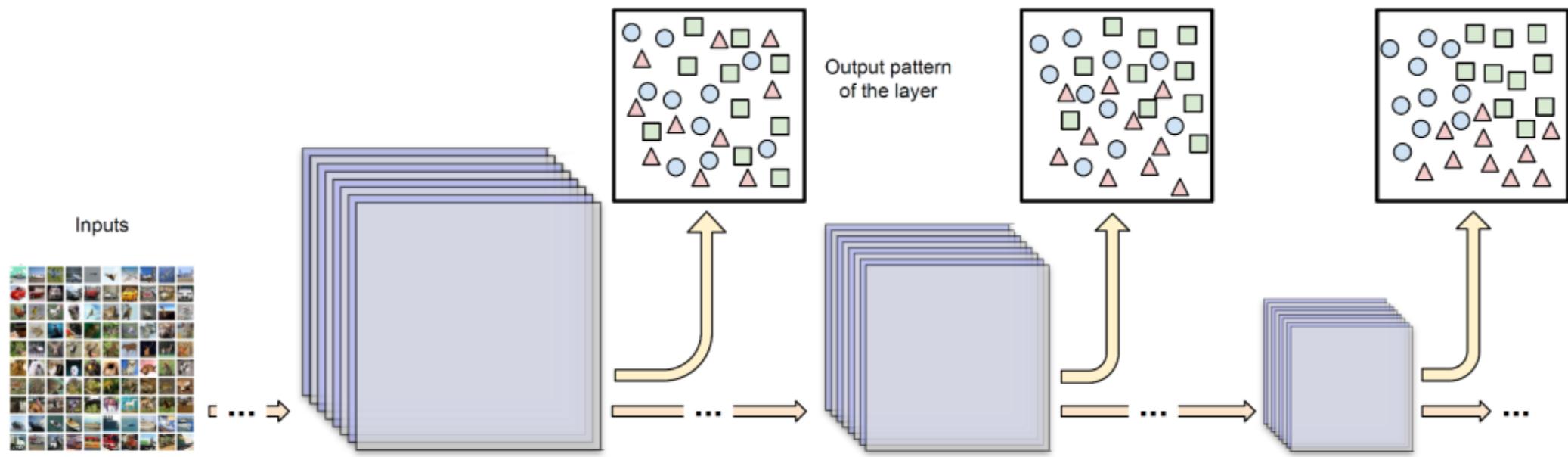
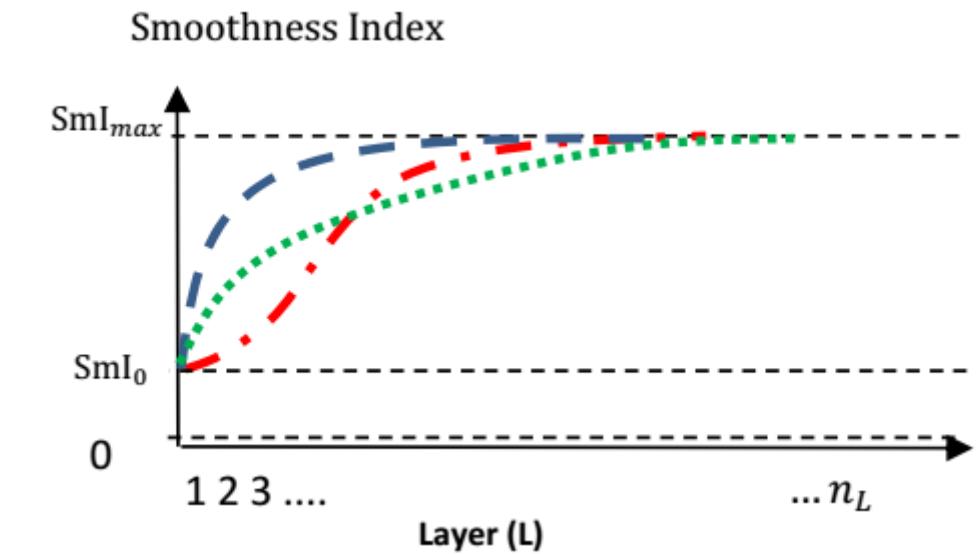
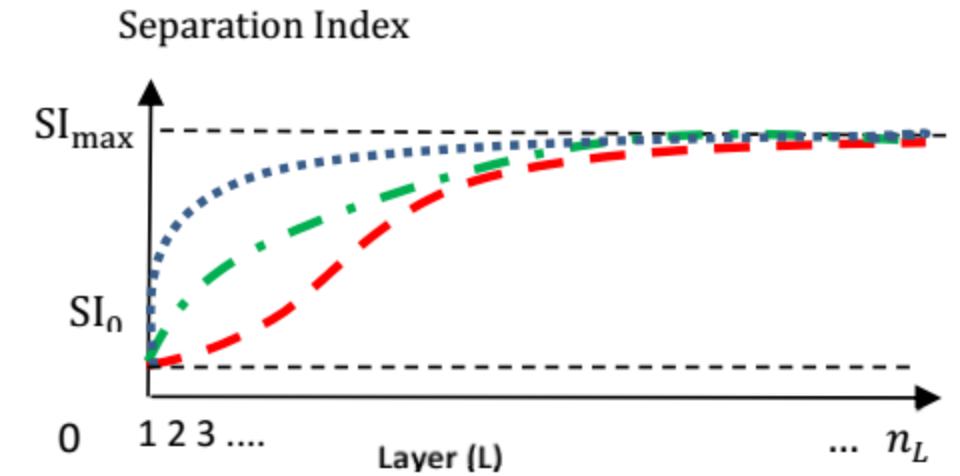


Figure 1. Output pattern of each layer through the network.

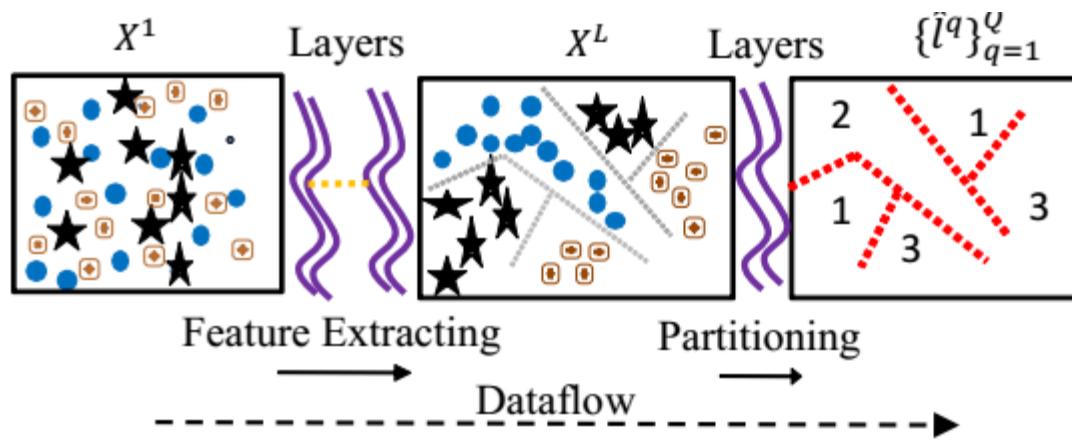
It is expected that the complexity of data decreases layer by layer in a deep neural network.

## Some important notes

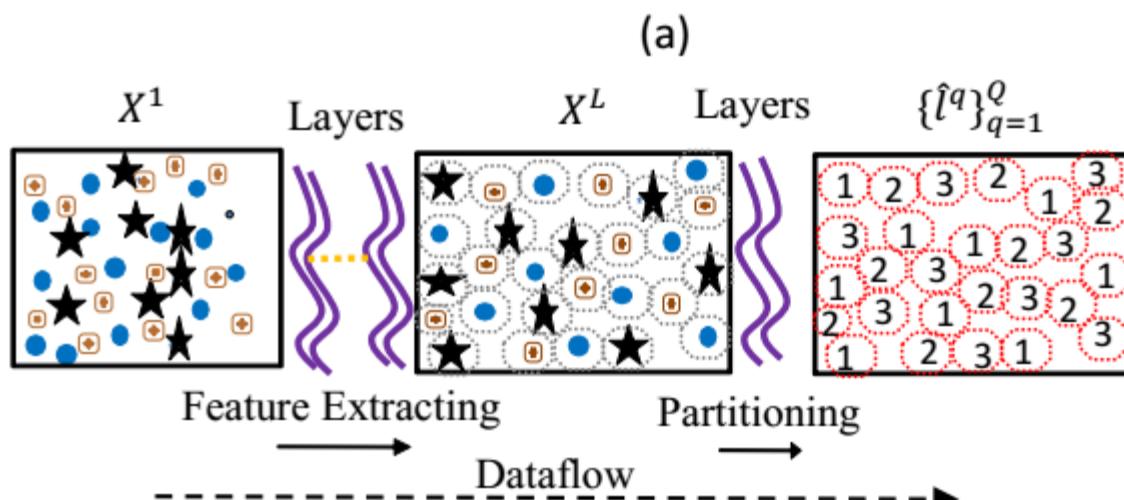
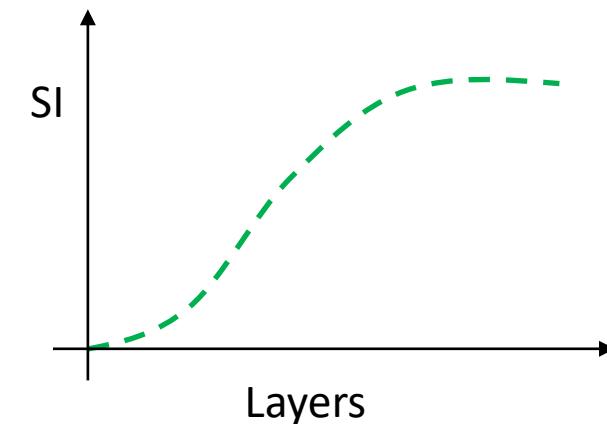
1. In fact, by decreasing the complexity of dataflow through “filter” layers, the  $SI(SmI)$  will increase.
2. The  $SI(SmI)$  may increase by different trends: different rises, different settling, with smooth or oscillatory changes.
3. Increasing  $SI(SmI)$  by Fully Connected (FC) layers is not desired. FC layers due to its redundancies make overfitting and avoid generalization.



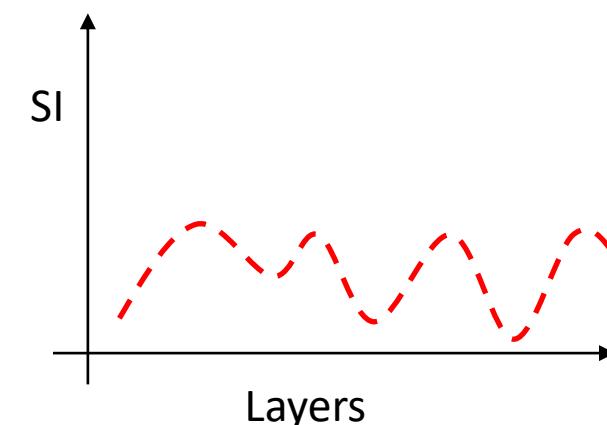
# Generalization and Separation index



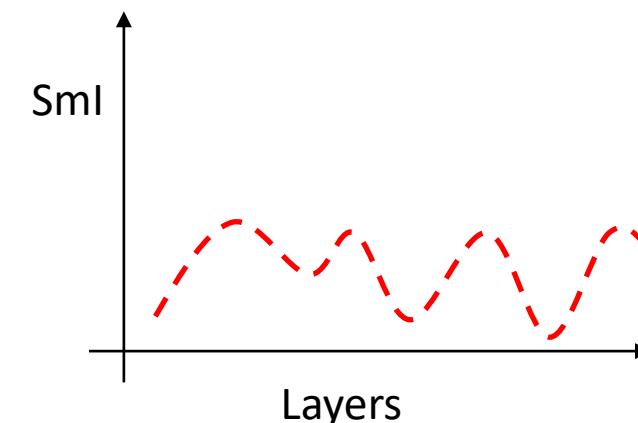
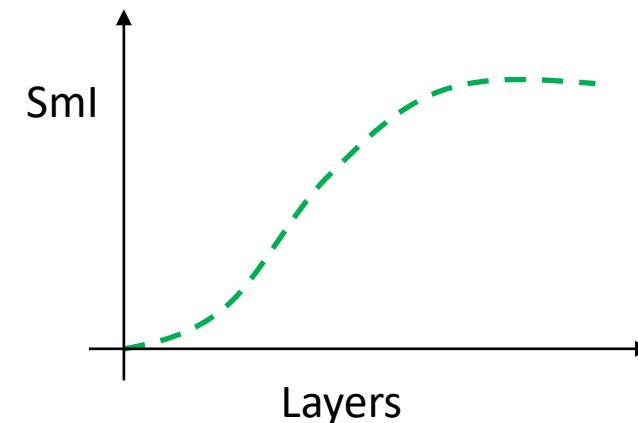
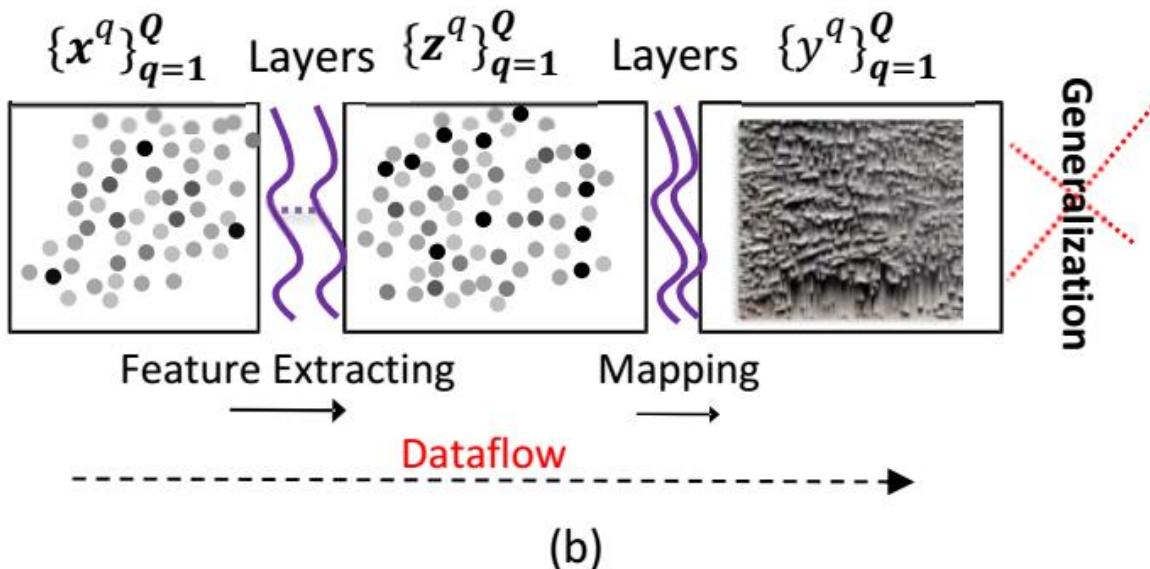
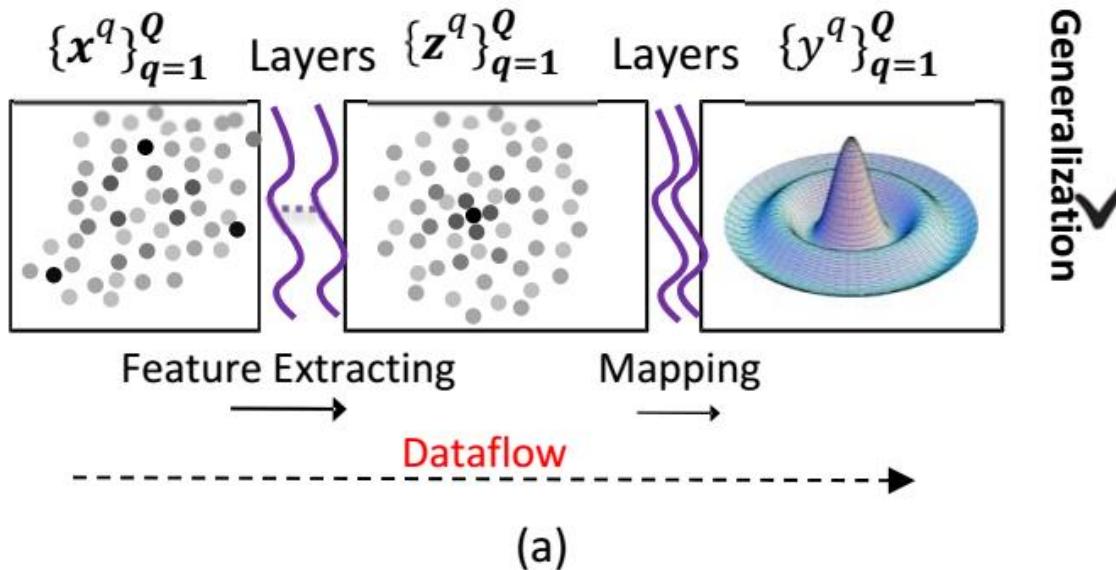
Generalization  
✓



Generalization  
✗



# Generalization and Smoothness index



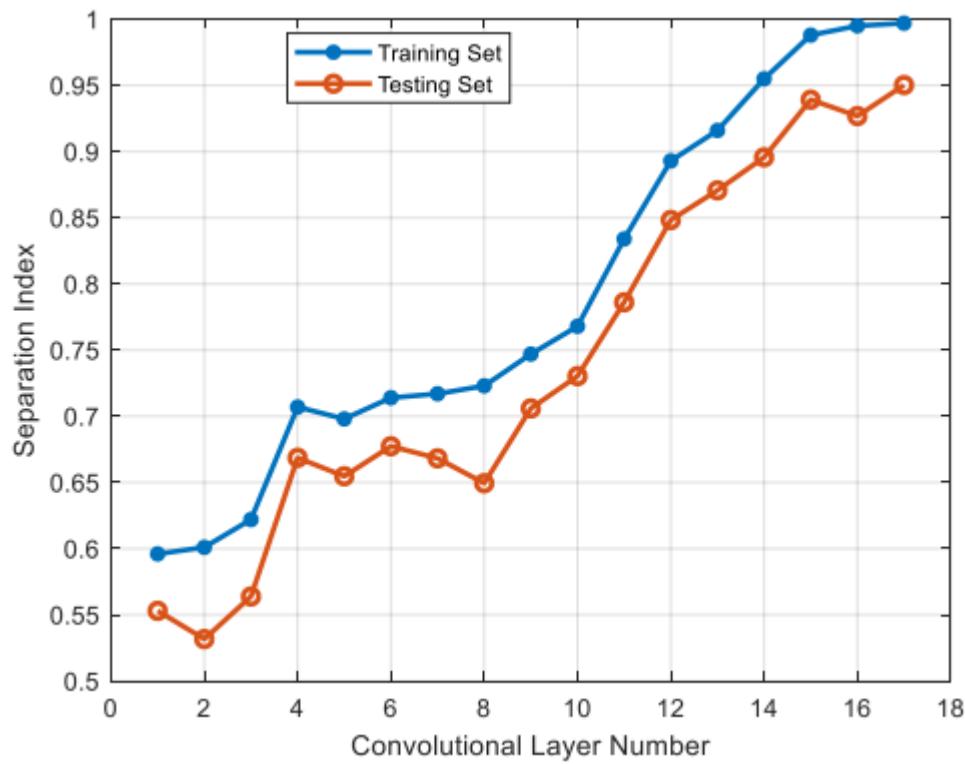
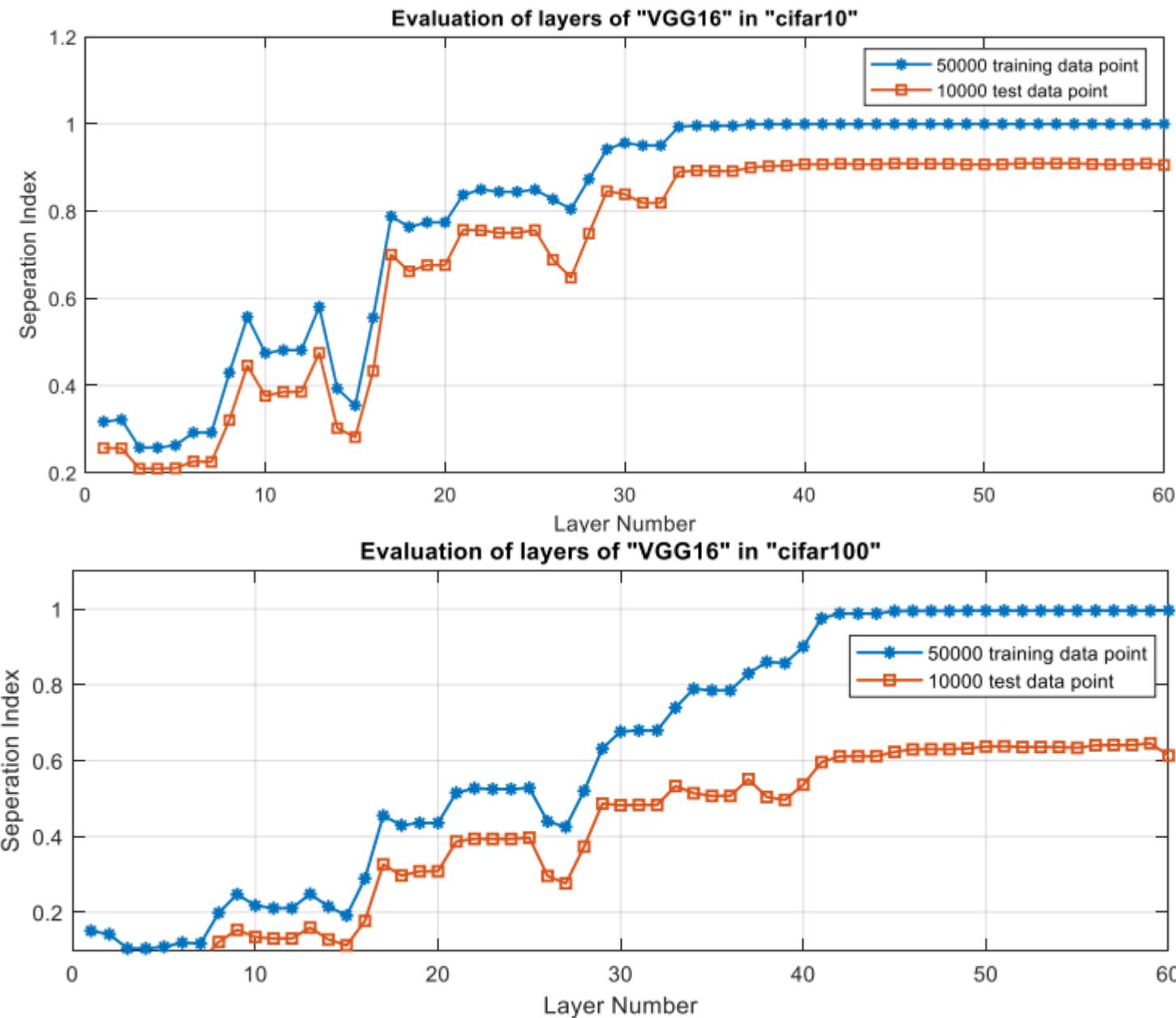


Fig. 10. Evaluation of dataflow through layers of the “Resnet18” network in the classification of Fashion-MNIST.



# Correct Classification rate and SI

To compare correct classification rate and SI:

1. After each convolution layer of a pertained network, two dense layers are added in order to predict the true labels. After two dense layers, a batch normalization layer is utilized and after them, a softmax layer is applied.
2. Considering the sum of squared error as the loss function, the “Adam” optimizer with more than 100 epochs has been utilized.
3. This process is performed on the “cifar10” dataset on pre-trained “VGG-16” and “Fashion-MNIST” dataset on trained “Resnet18” separately.

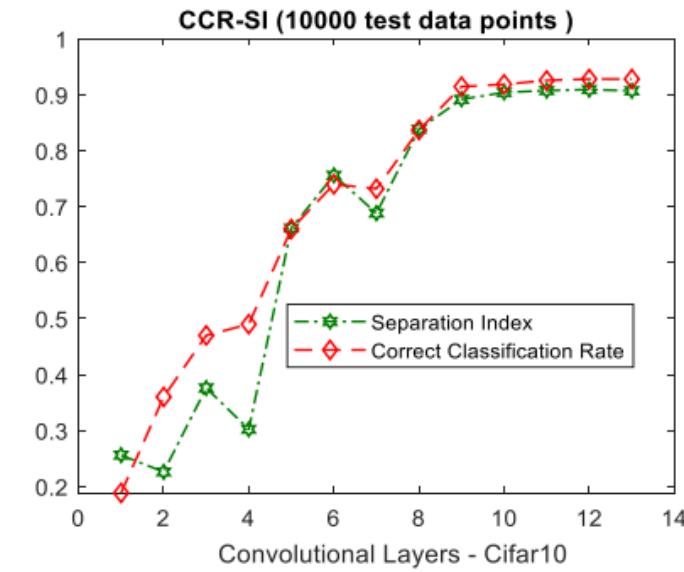
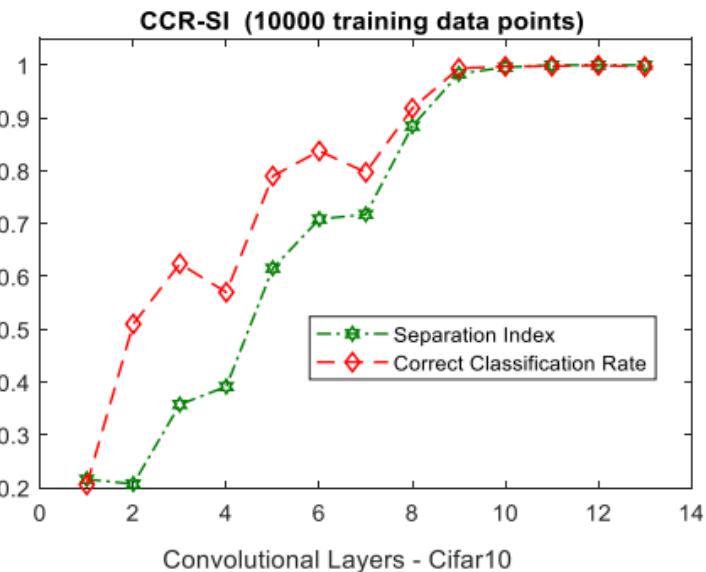


Fig. 11. The plots of separation index and correct classification rates at convolution layers in a pertained “vgg-16” network utilized for classification of “cifar10” for both training and test sets.

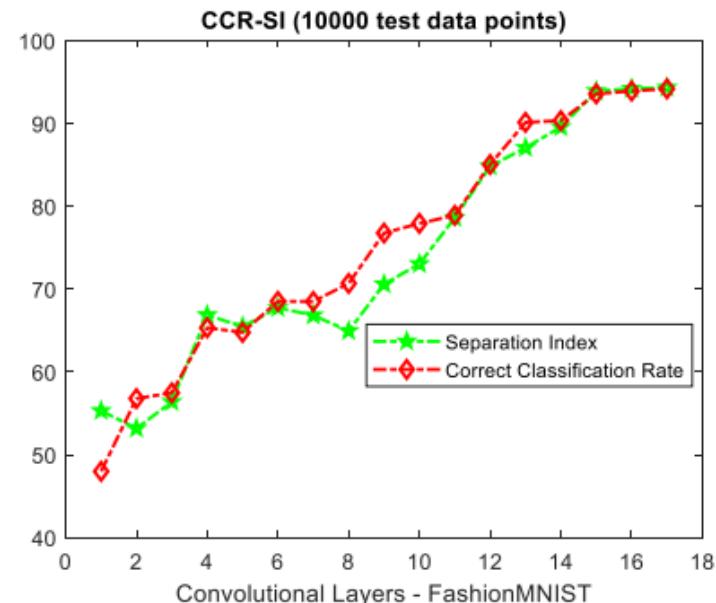
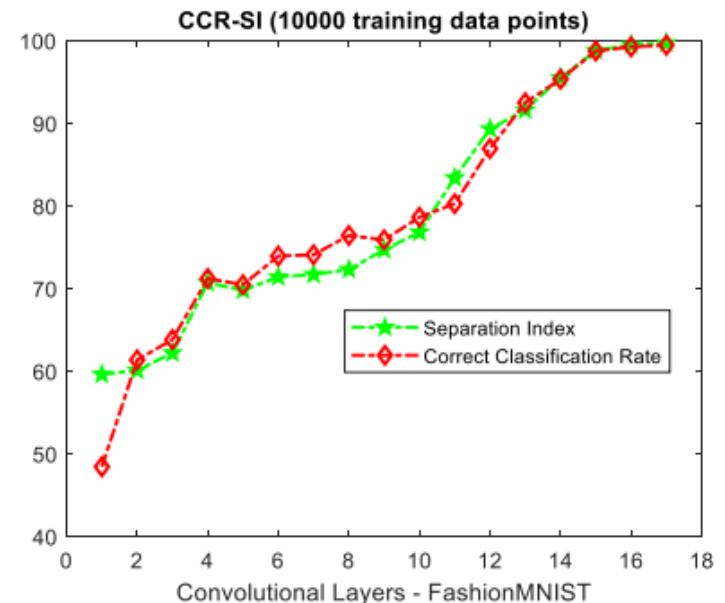


Fig. 12. The plots of separation index and correct classification rates at convolution layers in a trained “Resnet18” network utilized for classification of “Fashion-MNIST” for both training and test sets.

## Pre-train Model ranking

- Assume there are  $M$  different pre-trained models which are trained by  $M$  different source data.
- Assume for  $j=1,2,\dots,M$ , there are  $L_j$  layers before fully connected layers .
- Now, we want rank  $n_{model}$  pre-trained models to be used in transfer learning for a target data:

$Data = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ ,  $y_i$  for classification problems denotes the same label  $l_i$

### Algorithm 2: (To suggest a pre-trained model in transfer learning)

1. Apply the target data to  $j$ th pre-trained model and provide following dataflow at the last layer before fully connected layers:

$$Data^{j,L_j} = \{(\mathbf{x}_i^{j,L_j}, y_i)\}_{i=1}^m$$

2. For  $j$ th pre-trained model select the best subset of  $\mathbf{x}_i^{j,L_j}$  as  $\mathbf{x}_i^{j^*,L_j}$  which maximizes  $SI(SmI)$

$$Data^{j^*,L_j} = \{(\mathbf{x}_i^{j^*,L_j}, y_i)\}_{i=1}^m \quad \mathbf{x}_i^{j^*,L_j} \subseteq \mathbf{x}_i^{j,L_j}$$

3. Now rank the pre-trained models as best candidates for the target data in transfer learning as follows:

$$Best\_Models = \{j_1, \dots, j_M\} \text{ where } SI(Data^{j_1^*,L_{j_1}}) > SI(Data^{j_2^*,L_{j_2}}) > \dots > SI(Data^{j_M^*,L_{j_M}})$$

# Deep-HFID: Deep Neural Network Based Hands and Face bio-metric Identification System Using Metric Learning

Zahra Parvin Ashtyani  
School of Electrical and Computer Engineering  
Human and Robot Interaction Laboratory  
University of Tehran  
Tehran, Iran  
z.p.ashtyani@ut.ac.ir

Sara Ghanbari  
School of Electrical and Computer Engineering  
Human and Robot Interaction Laboratory  
University of Tehran  
Tehran, Iran  
saraghanbari@ut.ac.ir

Navid Zare  
School of Electrical and Computer Engineering  
Human and Robot Interaction Laboratory  
University of Tehran  
Tehran, Iran  
navid.zare@ut.ac.ir

Mehdi Tale Masouleh  
School of Electrical and Computer Engineering  
Human and Robot Interaction Laboratory  
University of Tehran  
Tehran, Iran  
m.t.masouleh@ut.ac.ir

Ahmad Kalhor  
School of Electrical and Computer Engineering  
University of Tehran  
Tehran, Iran  
akalhor@ut.ac.ir

**Abstract**—This paper presents a device for person identification based on hand bio-metrics. The main objective of a person recognition device is to perform accurate identification at a reasonable production cost. The so-called Deep-HFID device is designed to meet these objectives as well as to provide users with ease of use.

For the purpose of identifying individuals, this device uses hand and face bio-metrics. In terms of dimensions, this device has been designed to facilitate the use of a variety of individuals. For ease of deployment, the Deep-HFID system relies on a server for most of its processes. The routine tasks, like showing the graphical user interface are handled by a Raspberry Pi board. A dataset from the palm side of both hands from 110 individuals has been created. Using transfer learning, the six pre-trained state-of-theart neural network architecture was fine-tuned and calibrated based on the F1 score and precision criteria. An augmented dataset containing samples from the palm side of 30 individuals with between fingers angles variation is collected. **By applying the main and augmented dataset and using the center-based separation index as a complexity measure algorithm to rank pretrained deep neural networks, a compression on the last layer of the best performed neural network, EfficientNetV2B1, is done which results in a more robust system without reducement in accuracy**

### **3.2.5 Model Confidence and Guarantee**

1. Model Confidence and Guarantee by SI
2. Model Confidence and Guarantee by Sml

# Assumptions

1. There is a training dataset  $Data = \{(x_i, l_i)\}_{i=1}^m$  ( $Data = \{(x_i, y_i)\}_{i=1}^m$ ) for a classification(regression) problem.
2. The  $SI(Data)$  ( $SmI(Data)$ ) is computed for the dataset.
3. There is a test dataset which is homogenous with the training dataset.
4. Based on the Nearest Neighbor (NN) model, we want to predict the target  $l(y)$  of a new test example  $x$ , as  $\hat{l}(\hat{y})$ .
5. It is assumed that  $x_{i_j^*}$  denotes the  $j$ th nearest neighbor of input data to  $x$ .
6. There is an **uncertainty** in measuring  $x$ . It is assumed that  $x$  has maximum measuring error (distance)  $\gamma$  to its true value  $x_{true}$ :  $\|x_{true} - x\| < \gamma$ .
7. It is aimed to know that the confidence of the prediction by the NN model.
8. It is aimed to know if there is a guarantee to predict true label in classification problem or to have a limited output error in a regression problem.

# Model Confidence and Guarantee by SI (without training data)

Assume  $dw_{max}$  denotes the maximum distance of nearest neighbor example with the same label and  $db_{min}$  denotes the minimum inter distance between examples with different labels .

$Conf(\hat{l}, l_{i_1^*})$  denotes the confidence for prediction of  $\hat{l} = l_{i_1^*}$ (by the NN model).

$Guar(\hat{l}, l_{i_1^*})$  denotes the guarantee that prediction of  $\hat{l} = l_{i_1^*}$ (by the NN model ) is true.

if  $Guar(\hat{l}, l_{i_1^*}) = 1$  the guarantee exists and otherwise it does not exist.

- $Conf(\hat{l}, l_{i_1^*}) = SI(Data)$
- $Guar(\hat{l}, l_{i_1^*}) = \delta(SI, 1) * sign(1 - \frac{dw_{max} + \gamma}{db_{min} - \gamma})$

$\gamma$ : the maximum uncertainty in measuring  $x$  : $\|x_{true} - x\| < \gamma$

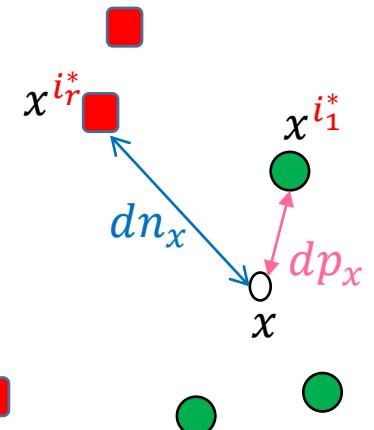
❖ If  $SI(Data) = 1$  and  $dw_{max} < db_{min}$ , the true prediction for  $x$  with NN model and for  $\gamma < 0.5(db_{min} - dw_{max})$  is guaranteed.

## Model Confidence and Guarantee by SI (with training data)

$$dp_x = \|x - x_{i_1^*}\|$$

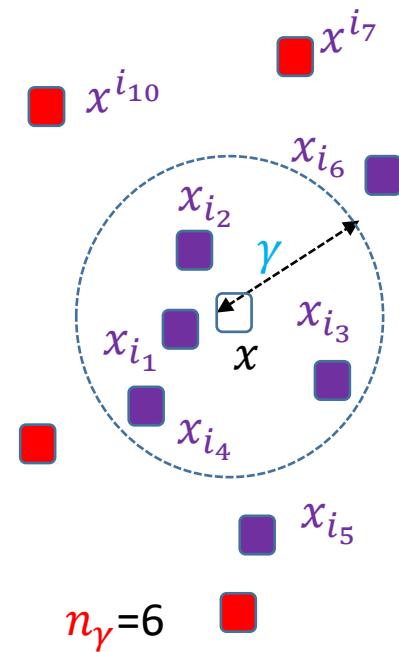
$$dn_x = \|x - x_{i_r^*}\| \quad r = \min_{j=1,\dots,m} j \text{ subject to } \delta(l_{i_j^*}, l_{i_1^*}) = 0$$

- $Conf(\hat{l}, l_{*_1}) = SI$
- $Guar(\hat{l}, l_{*_1}) = \delta(SI, 1) * sign(1 - \frac{dp_x + \gamma}{dn_x - \gamma})$
- ❖ If  $SI(Data) = 1$  the true prediction for  $x$  with NN model and  $\gamma < 0.5(dn_x - dp_x)$  is guaranteed.
- About those cases that we have  $SI^r(Data) = 1$  for  $r \gg 1$  the guarantee is satisfied for much higher  $\gamma$  than the cases which we have  $SI^1(Data) = 1$ .



# Model Confidence and Guarantee by SmI(with training data)

- Assume that for each data point  $x_i$ ,  $\alpha(i^*) = \|y_i - y_{i^*}\|$ , where  $x_{i^*}$  is the nearest neighbor of  $x_i$ .
  - Assuming  $SmI(Data) = 1$ ,  $\alpha(i^*)$  denotes the distance between the output  $y_i$  and its nearest neighbor,  $y_{imin}$ .
  - In an ideal case assume that the diversity of training dataset is so high and for each test data point  $x$  (when there is no measuring uncertainty)  $\|y - y_{i_1^*}\| \leq \alpha(i_1^*)$ .
  - Now assume,  $\gamma$  is the maximum measuring uncertainty of input  $x$ .
  - Find all training examples:  $\{x_{i_j^*}\}_{j=1}^{n_\gamma}$  that each one is a nearest neighbor of a point of the hyper-circle set  $C = \{\tilde{x}, \|x - \tilde{x}\| \leq \gamma\}$ .
  - it is guaranteed that if for each test data point  $x$ , when  $\gamma \geq 0$
- $$\|y - y_{i_1^*}\| \leq \bar{e}_y$$
- $$\bar{e}_y = \max_{j=1, \dots, n_\gamma} \left( \|y_{i_1^*} - y_{i_j^*}\| + \alpha(i_j^*) \right)$$
- About those cases that we have  $SmI^r(Data) = 1$  for  $r \gg 1$ ,  $\bar{e}_y$  is much lower than the cases which we have  $SmI^1(Data) = 1$ .



## 3.3 Layer-wise Design by Separation and Smoothness indices

3.3.1 Model Compressing

3.3.2 Forward learning in the first layer

3.3.3 Layer-wise Forward learning

3.3.7 Forward Auto Encoder Learning

3.3.4 Layer-wise branching

3.3.5 Layer-wise Fusion

3.3.6 Forward Design

3.3.8 Forward Multi-Task Design

3.3.9 Design and optimize a classifier with high order SI

3.3.10 Design and optimize a regression model with high order Sml

### 3.3.1 Model Compressing

To remove extra layers and units in filter part and then compressing flat layers

# Model Compressing

In model compressing some layers and Feature Maps(units)

Some Definitions:

**Filter Part:** The layers before flat layer which extract features from spatial or temporal correlated inputs. If in some DNNs, one or more than one RNN modules have been used, we assume that they belonged to filter part.

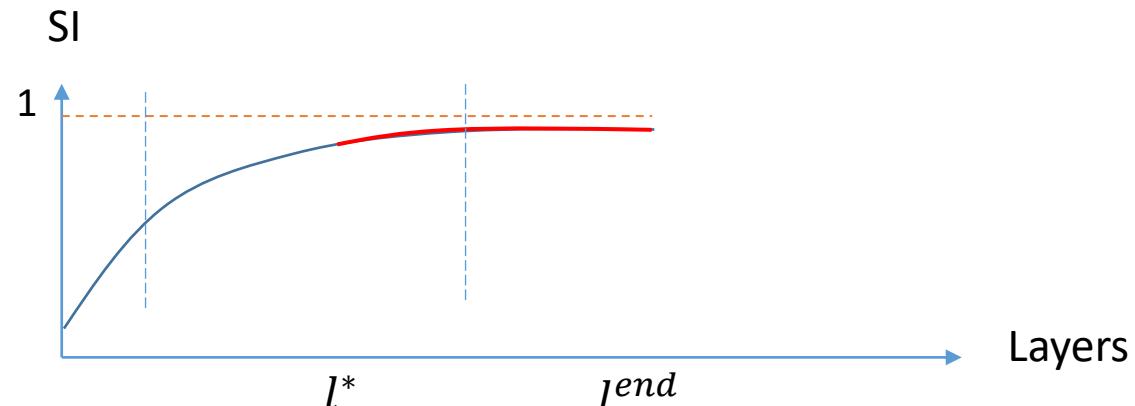
**FC Part:** one, two, or more than two fully connected layers which are defined after the “Filter Part” and the outputs of the networks.

Algorithm3:

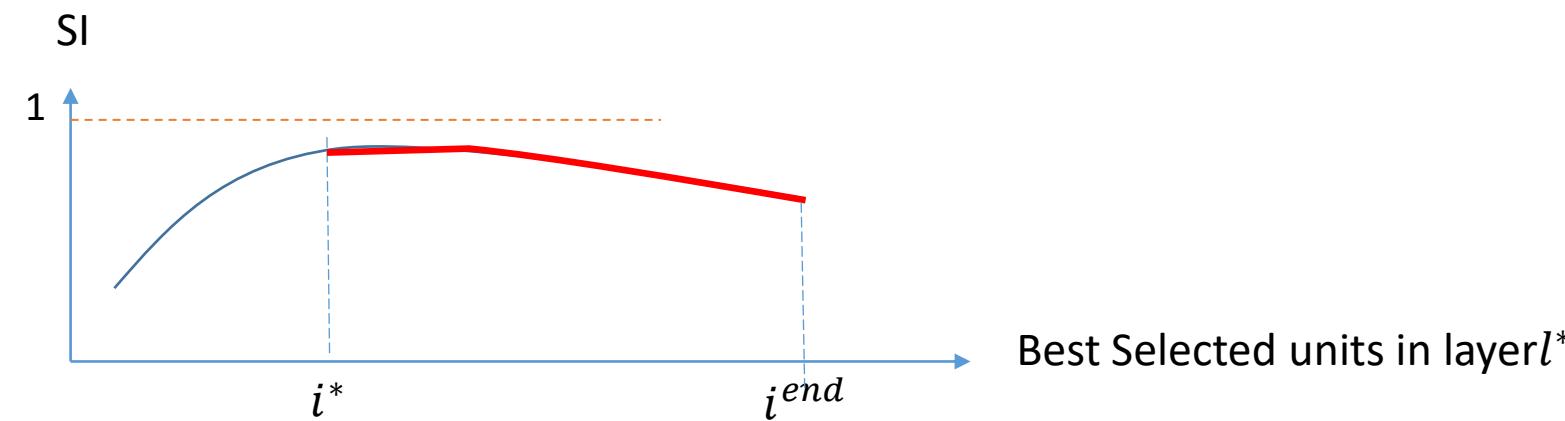
- 1- Find and remove the last layers of the “Filter Part” which do not increase the SI(SmI), significantly.
- 2- Find and remove all feature maps(units) of the last layer of the filter part which do not increase the SI(SmI), significantly.
- 3- Design a new “FC part” with respect the number of units at the last later of the “filter part”.

# Compressing Procedure

1. Compute  $SI(SmI)$  for layers of the DNN and then remove all last layers (after layer  $l^*$ ) which do no increase  $SI(SmI)$ , significantly.



2. Remove all extra feature maps(units) at layer  $l^*$  which do no increase  $SI(SmI)$ , significantly.



3. Flat the units at layer  $l^*$  and add Fully Connected layers and fine tune

TABLE VIII  
Comparison with the prior art of VGG-16 on CIFAR-10.

Compressing method	Retraining needed	FLOPs	Pruned	Accuracy
VGG16-base	-	313.7M (0.0%)	0.0%	94.04%
PFEC [17]	Yes	206M (34.3%)	63.3%	93.40%
VP [45]	Yes	190M (39.1%)	73.4%	93.18%
SS [46]	Yes	183.13M (41.6%)	77.7%	93.18%
GAL-0.05 [47]	No	189.49M (39.6%)	77.2%	92.03%
CP [48]	No	107.58M (65.1%)	77.6%	92.03%
HRFM [49]	Yes	108.61M (65.3%)	82.1%	92.34%
GAL-0.1 [47]	No	171.89M (45.2%)	82.2%	90.73%
<b>Our method</b>	<b>No</b>	<b>75.6M (76.0%)</b>	<b>87.5%</b>	<b>93.49%</b>
HRFM [49]	Yes	73.70M (76.5%)	88.2%	91.23%

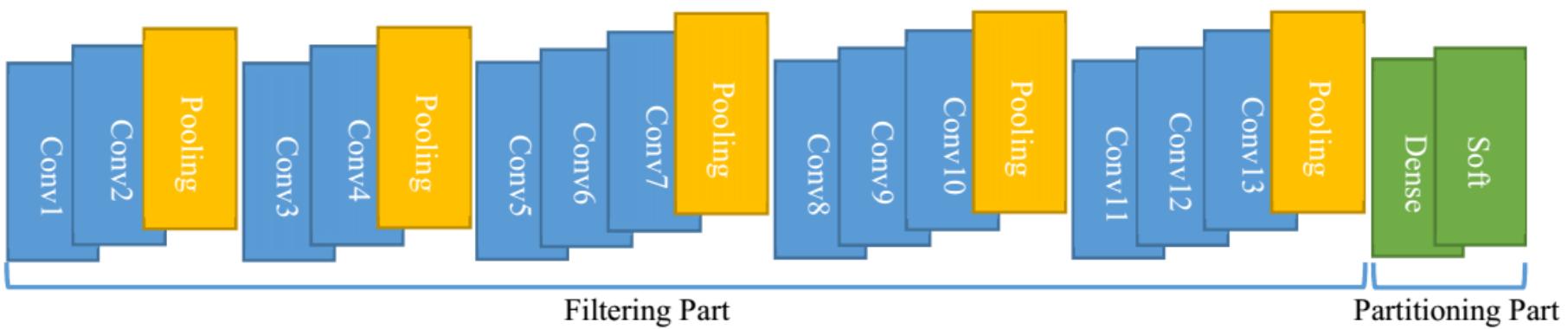


Fig. 5. The architecture of the "VGG-16" network.

TABLE IX  
Comparison with the prior art of GoogLeNet on CIFAR-10.

Compressing method	Retraining needed	FLOPS	Pruned	Accuracy
GoogleNet-base	-	1.52B(0.0%)	0.0%	95.05%
PFEC [17]	Yes	1.0B(32.9%)	42.9%	94.54%
HRFM [49]	Yes	0.69B(54.9%)	55.4%	94.53%
GAL-Apo [50]	No	0.76B(50.0%)	53.7%	92.11%
GAL-0.05 [48]	No	0.94B(38.2%)	49.3%	93.93%
HRFM [49]	Yes	0.45B(70.4%)	69.8%	94.07%
<b>Our method</b>	<b>No</b>	<b>0.39B(74.4%)</b>	<b>77.6%</b>	<b>95.00%</b>

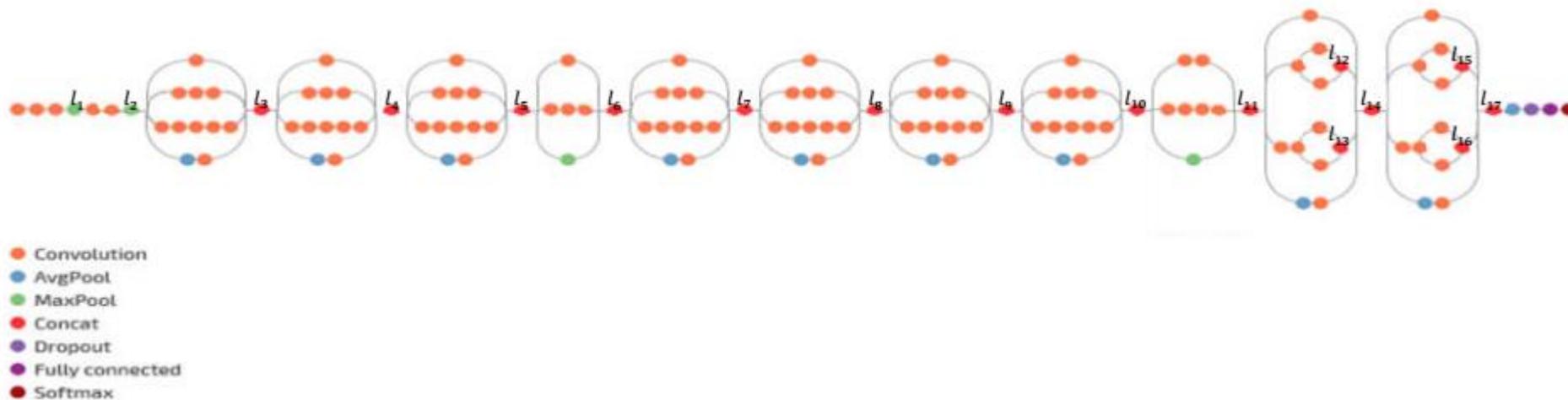
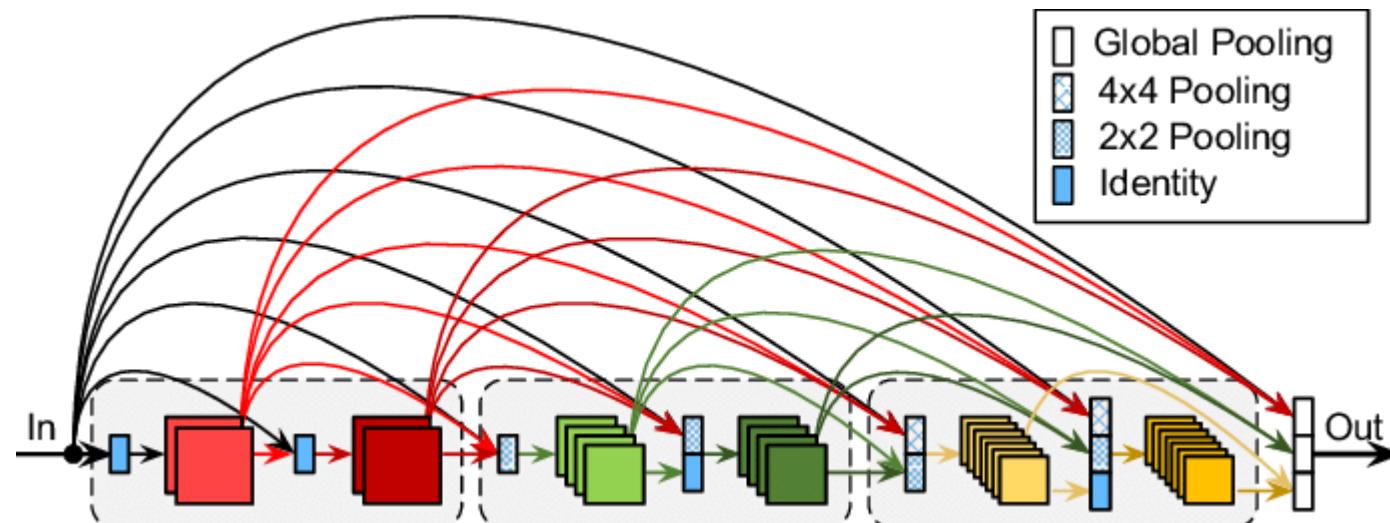


Fig. 7. The architecture of the "Inception V3" network.

TABLE X  
Comparison with the prior art of DenseNet-40 on CIFAR-10.

Compressing method	Retraining needed	FLOPS	Pruned	Accuracy
DenseNet-base	-	0.29B(0.0%)	0.0%	94.22%
ECNS [51]	Yes	0.12B(58.3%)	67.2%	94.35%
HRFM [49]	Yes	0.11B(61.8%)	55.1%	94.53%
GAL-0.05 [48]	No	0.13B(55.6%)	57.9%	92.11%
VP [45]	No	0.16B(45.8%)	60.7%	93.16%
<b>Our method</b>	<b>No</b>	<b>0.07B(76.1%)</b>	<b>78.8%</b>	<b>94.17%</b>



### 3.3.2 Forward learning in the first layer

For classification problems but the method can be generalized for regression problems

# Forward learning in the first layer

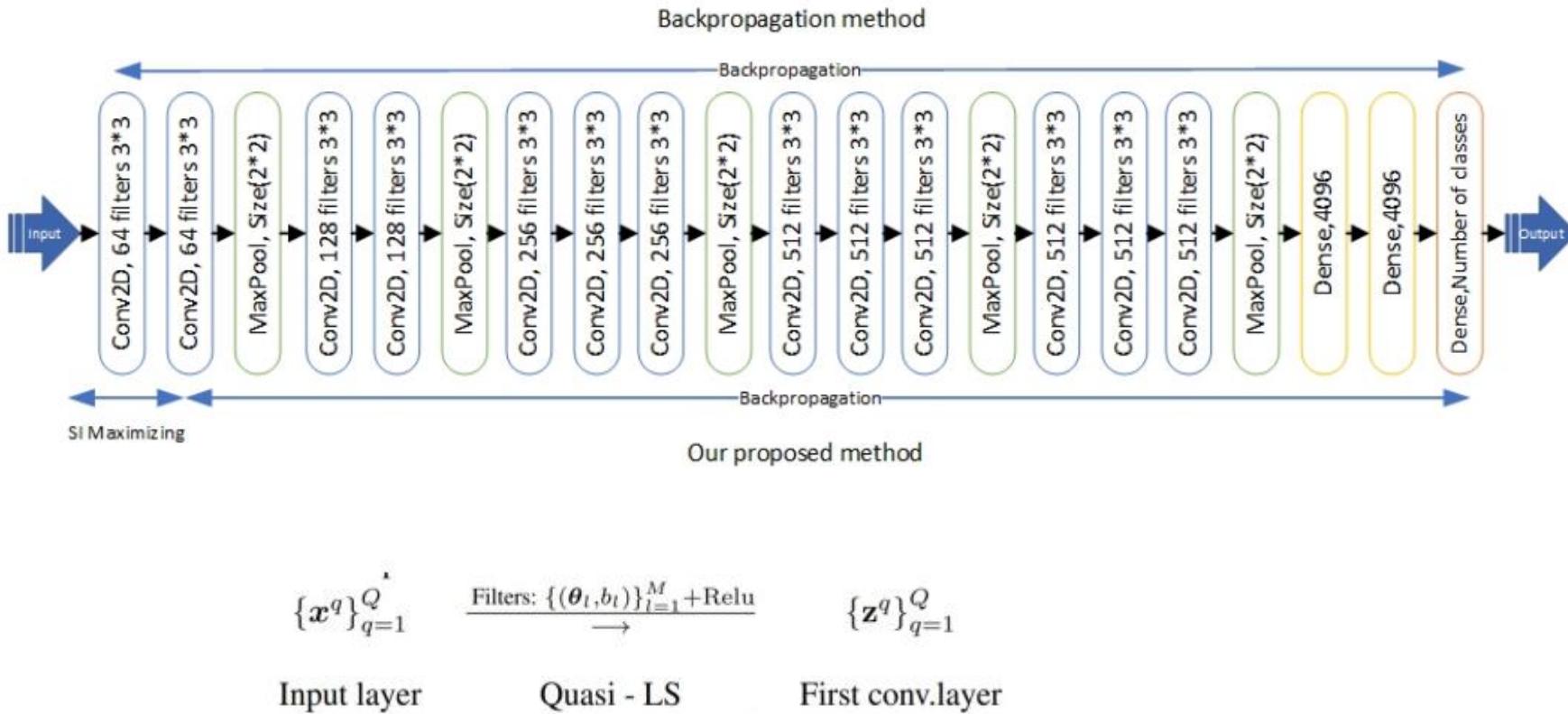


Figure 3: Applying M convolutional filters and biases to the input patterns and then activating the convolved units by Relu function the feature maps are resulted.

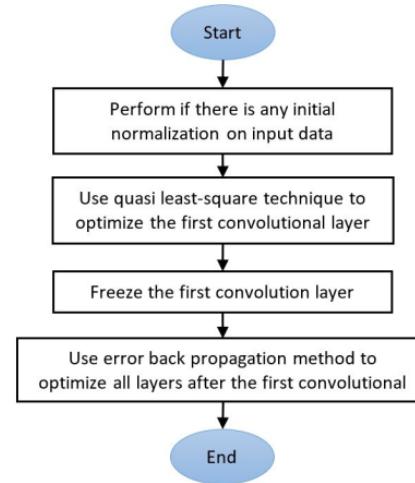


Figure 2: This flowchart indicates the learning method in which the quasi-LS is utilized to learn the first convolution layer, and other layers are learned by an error backpropagation method.

## Algorithm to train the first layer by Quasi Least Square (QLS) technique

1. Define the matrix of patches from the input data:  $X$
2. Subtract the matrix from their mean:  $\tilde{X} = X - \text{mean}(X)$
3. Define Auto-correlation matrix of patches:  $W_{np} = \tilde{X}^T \tilde{X}$
4. Apply SVD technique to get Eigen values and Eigen vectors:  $W = \sum_{i=1}^{n_p} \lambda_i v_i v_i^T$
5. Rank Eigen vectors by their Eigen values and then initiate all filters parameters and the biases by Eigen vectors, their negatives and the mean of patches.

$$\left\{ \{\theta^j, b^j\} \right\}_{j=1}^{n_F} \quad n_F: \text{number of filters}$$

6. Based on the filter parameters at the first (convolutional)layer compute the second layer and for each example(anchor) find nearest neighbor for both positive and negative examples and define triplet loss for all data batch.

$$J_{triplet} = \sum_{i=1}^m (\|x_i^2 - x_{i\text{pos}}^2\|^2) - \sum_{i=1}^m (\|x_i^2 - x_{i\text{neg}}^2\|^2)$$

7. Use a QLS technique to minimize  $J_{triplet}$  and update all filters.

$$\theta^j := \theta^j + \mu \tilde{\theta}^j \quad b^j := b^j + \mu \tilde{b}^j \quad \mu = \text{learning rate}$$

8. Repeat steps 6 and 7 for several epochs to get maximum possible SI on the second layer.

Now, one can train further layers by Backpropagation Algorithms.

## Comparison between our method and backpropagation algorithm

Dataset	Learning Method	AlexNet	VGG16	ResNet50	InceptionV3
CIFAR10	Backpropagation	84.61	92.95	93.17	94.20
	Our proposed method	<b>85.84</b>	<b>94.81</b>	<b>95.02</b>	<b>95.43</b>
	Percentage of improvement	1.23	1.86	1.85	1.23
CIFAR10 - (plane,truck)	Backpropagation	96.45	97.18	97.64	97.09
	Our proposed method	<b>97.09</b>	<b>98.36</b>	<b>98.49</b>	<b>97.77</b>
	Percentage of improvement	0.64	1.18	0.85	0.68
CIFAR10 - (plane,cat,bird)	Backpropagation	96.21	96.80	96.88	96.81
	Our proposed method	<b>96.62</b>	<b>97.63</b>	<b>97.16</b>	<b>97.14</b>
	Percentage of improvement	0.41	0.83	0.28	0.33
CIFAR100	Backpropagation	62.22	70.98	75.30	76.31
	Our proposed method	<b>62.45</b>	<b>71.74</b>	<b>75.58</b>	<b>76.72</b>
	Percentage of improvement	0.23	0.76	0.28	0.41
FASHION-MNIST	Backpropagation	92.53	94.17	95.24	95.78
	Our proposed method	<b>92.65</b>	<b>94.73</b>	<b>95.38</b>	<b>95.91</b>
	Percentage of improvement	0.12	0.56	0.14	0.13

Table 5: Comparing the accuracy of our proposed method in different architectures and datasets with backpropagation method

### 3.3.3 Layer-wise forward learning

# Layer Wise Forward Learning

## Algorithm

```

For L=1:Lfinal
    While (SI (Datal) may still increase )
        For l = 1:L
             $\mathbf{g}_{p^l} = \frac{\partial \text{Loss}_{\text{triplet}}(\text{Data}^l)}{p^l}$ 
             $\mathbf{p}^l := \mathbf{p}^l - \gamma \mathbf{g}_{p^l}$ 
    
```

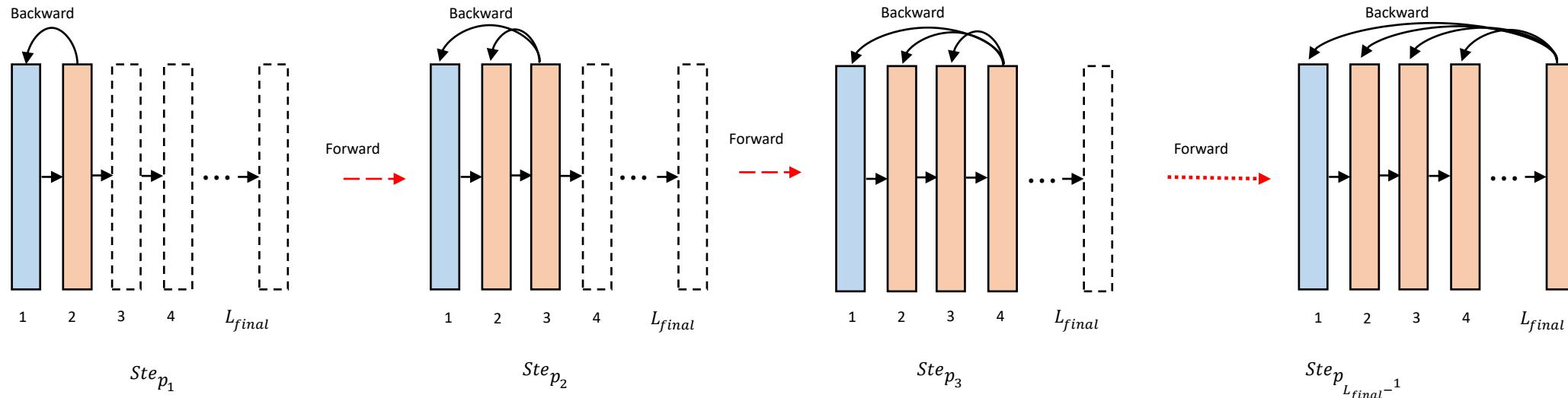
$\text{Data}^l = \{(x_i^l, y_i^l)\}$  Data at layer  $l$

## Triplet Loss

$$\text{Loss}_{\text{triplet}}(\text{Data}^L) = \frac{1}{m} \sum_{i=1}^m \left( \|x_i^L - x_{i_p}^L\|^2 - \|x_i^L - x_{i_n}^L\|^2 \right)$$

$$i_p^* = \arg \min_{\forall q \neq i} \frac{1}{\delta(y_i, y_q) + \varepsilon} \|x_i^L - x_q^L\|^2 \quad \varepsilon \rightarrow 0^+$$

$$i_n^* = \arg \min_{\forall q \neq i} \frac{1}{1 - \delta(y_i, y_q) + \varepsilon} \|x_i^L - x_q^L\|^2$$



index is feasible by training the first layers with the Forward learning approach.

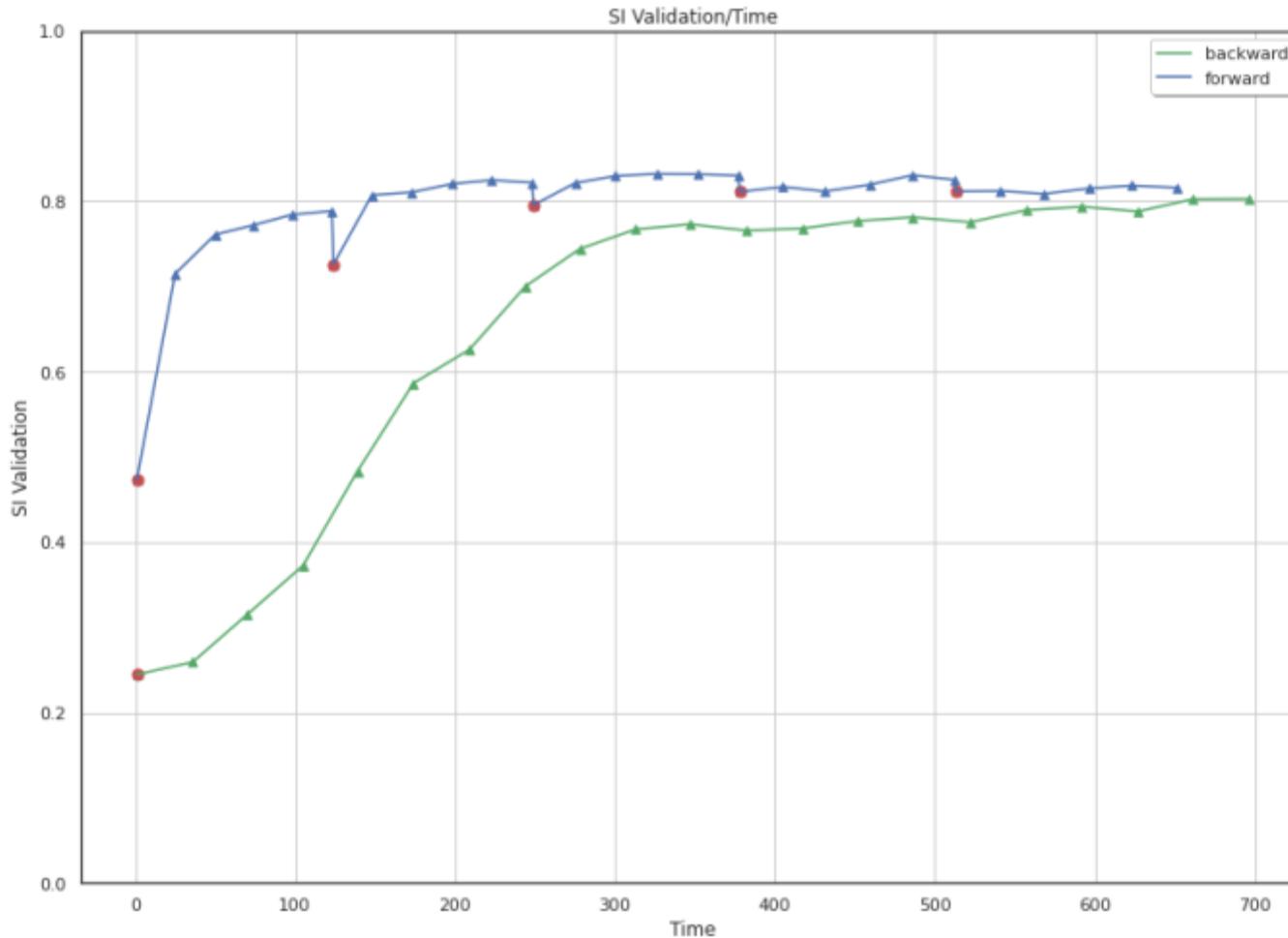


Figure 2. Comparing the separation index of the forward learning and backward learning approaches over time. Each triangle shows a single epoch in the learning process and the red dots in the Forward learning method are convolutional layers.

## Comparison between our method and backpropagation algorithm

VGG11	ResNet <sup>59</sup>	InceptionV3	EfficientNetB0		
92.95	93.17	94.20	98.11	Backpropagation	CIFAR10
93.77	94.70	94.73	98.24	Our method	
70.98	70.30	76.31	88.14	Backpropagation	CIFAR100
71.19	70.47	76.49	88.17	Our method	
94.17	95.38	95.91	97.23	Backpropagation	Fashion-MNIST
94.02	97.71	97.21	97.41	Our method	

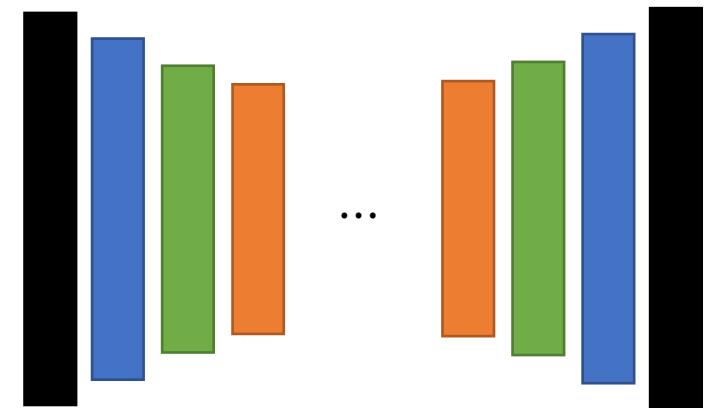
# The Text Classification Error rate

Dataset	Learning Strategy	VD-CNN(Depth=9)	VD-CNN(Depth=17)	VD-CNN(Depth=29)
AG's News	Our Method	9.35	8.71	8.72
	Backpropagation (Cross Entropy)	10.17	9.29	9.36
DBPedia	Our Method	1.53	1.30	1.23
	Backpropagation (Cross Entropy)	1.64	1.42	1.36

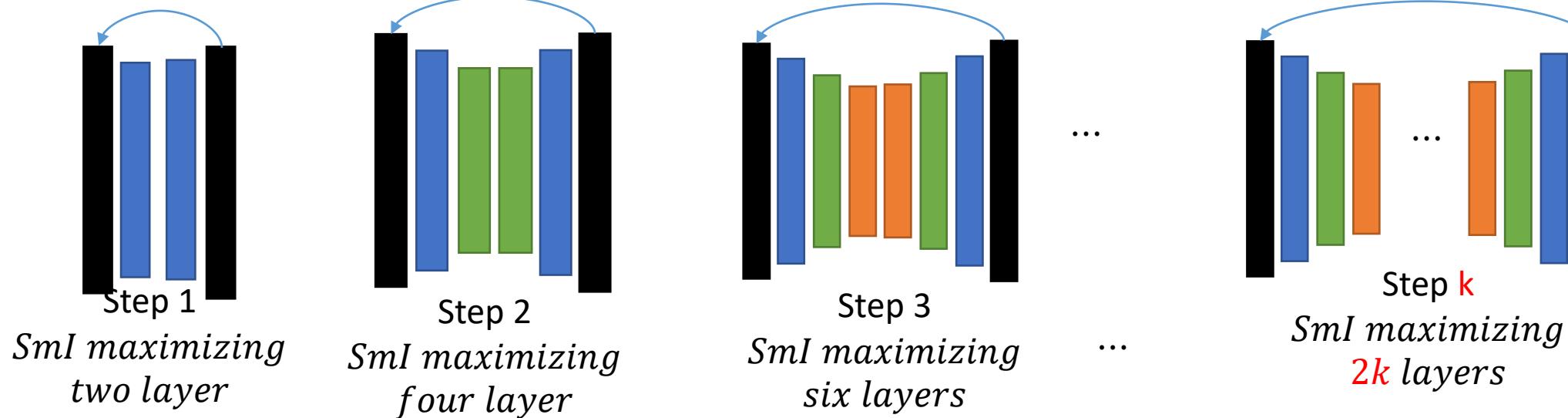
### 3.3.4 Layer-wise Forward Auto-Encoder Learning

# Layer Wise Forward Learning of an Auto-Encoder

- Smoothness index can be used to train an auto-encoder step by step in a forward manner.
- At the first step two hidden layers are trained and then by each further step two new layers are added and trained to the AE.
- At each step, the training procedure is based on “Sml” maximizing.
- In fact, the smoothness index is approximated by a triplet loss and then the loss is minimized.
- Each example at the output layer decreases(increases) its distance to one of its K nearest neighbors that has nearest (furthest) distance at the target(input) space (reconstruction)



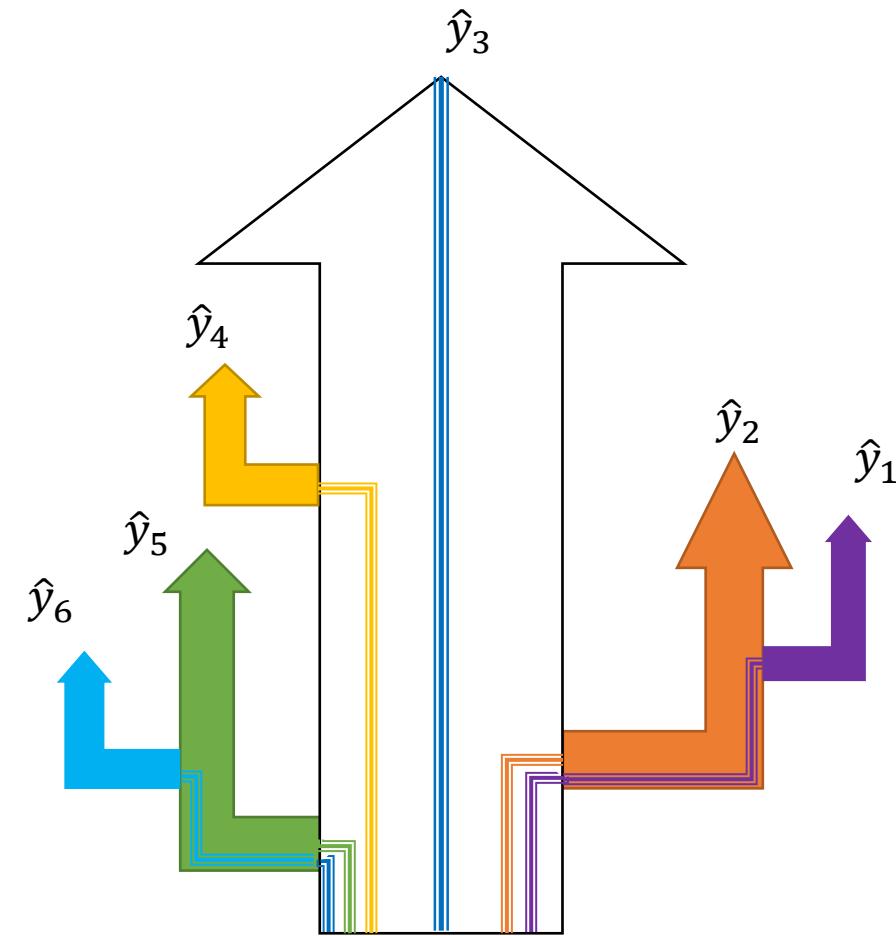
## Layer-wise Learning through Smoothness Maximizing



### **3.3.4** Layer-wise branching

# Layer-wise branching

- In a classification (regression) problem, the required features may be extracted earlier or later through layers of a DNN.
- If we design a new architecture of DNN which can conduct earlier features in some independent paths, more compact and more generalized DNNs can be learned.
- In such an architecture, there are a main body and some branches. The main body forms the main flow of data but each branch process a part of data-flow.
- We can use complexity measures like SI(Sml) to distinguish early features from other features and form a new branch for it.
- Each branch like the main body can define some sub-brances, too.
- In such a DNN, the amount of data-flow decreases through the main body just after each branching.



*Input data points*

## Some notes

- Branch in point where an increasing jump on SI(Sml) is occurred.
- Find the appropriate part of data (class/output/examples) to be branched.
- Branching is done as a forward design operation

# Utilizing SI for Branching

DukeMTMC-reID: 34183 images  
pretrained network:

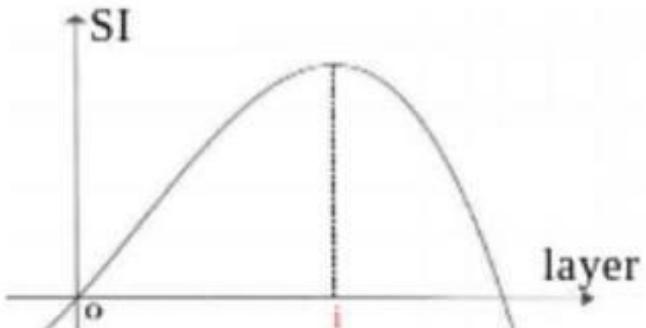
**person re identification**



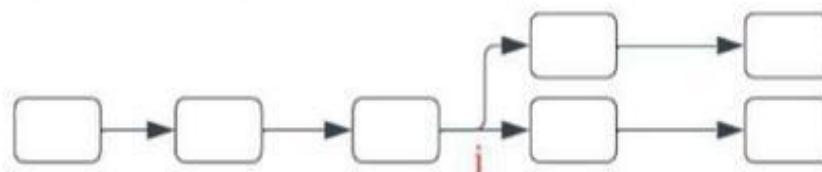
**ID,**

Task1: person reID

1. Check the information flow through network for attribute recognition



2. generating branches



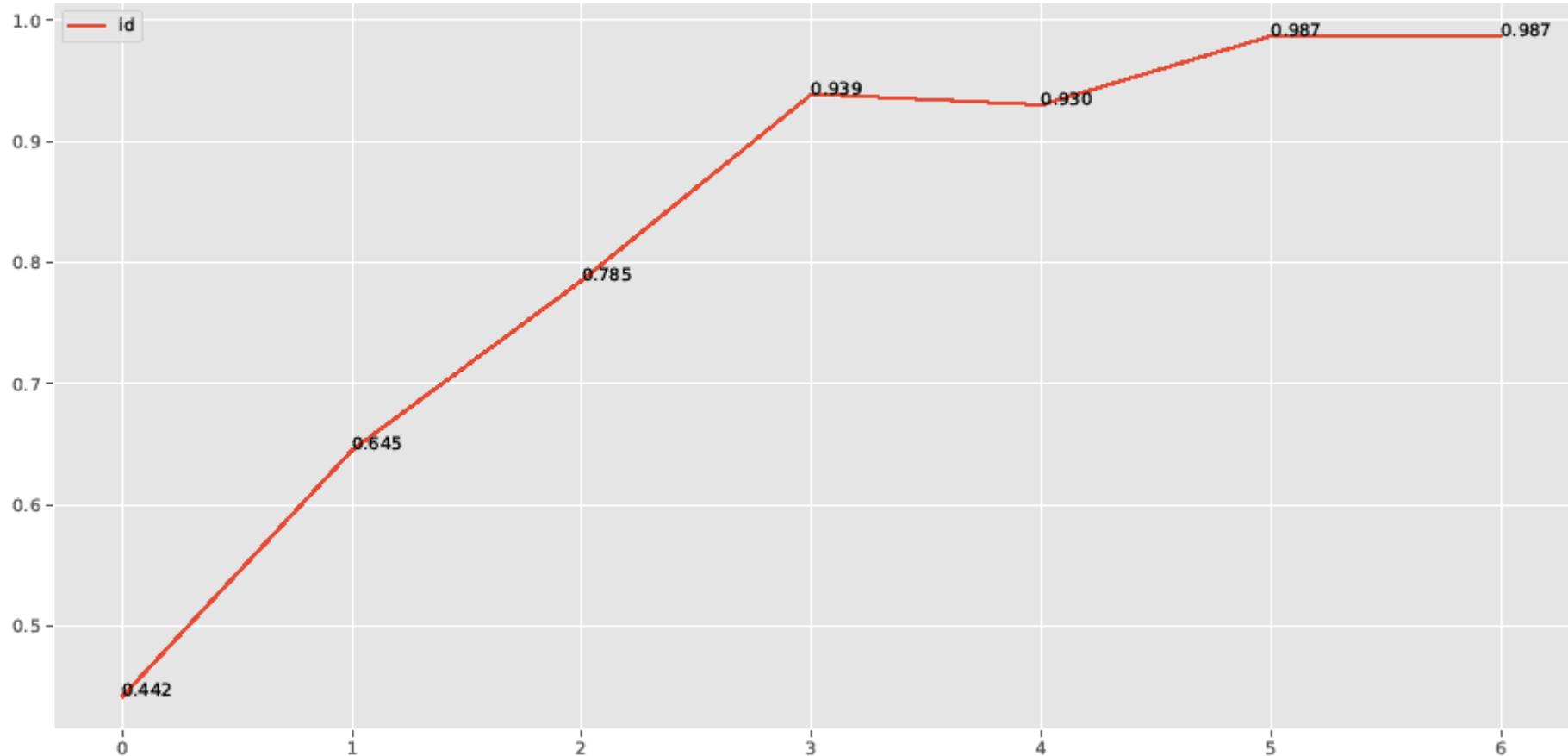
3. Train the branche for attribute recognition

**attributes,**

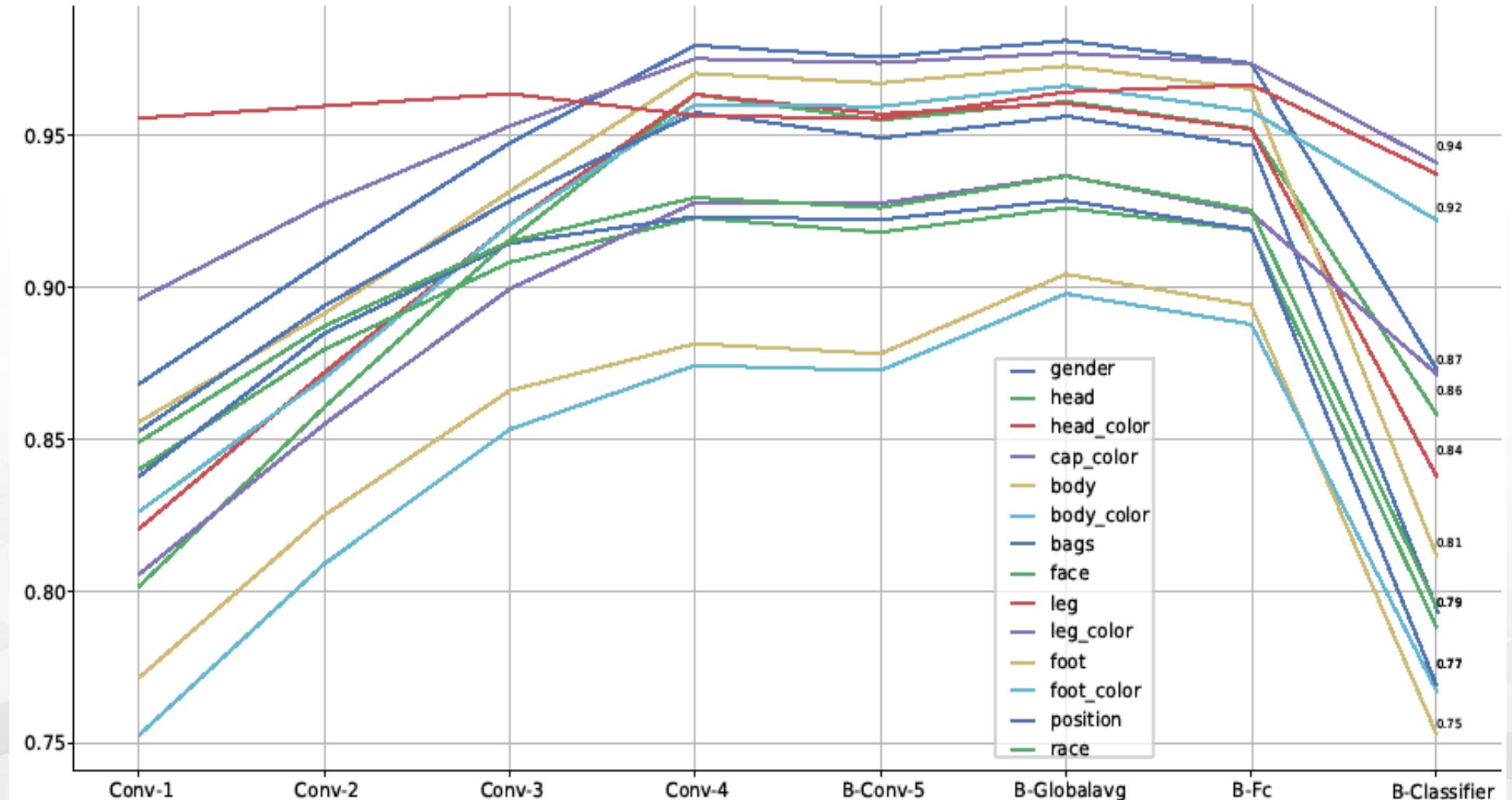


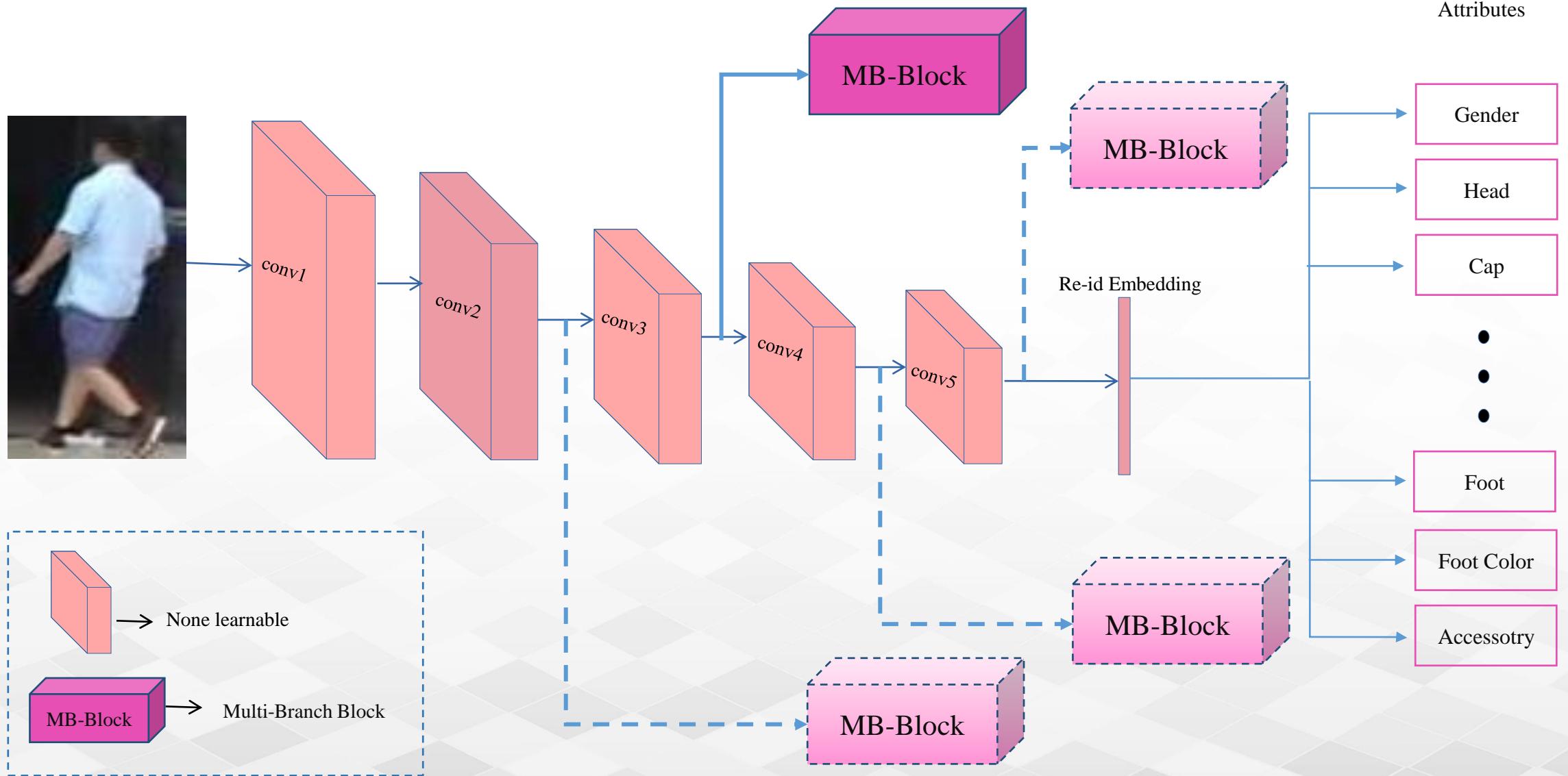
Task2:attribute recognition

# Separation Index for a classification

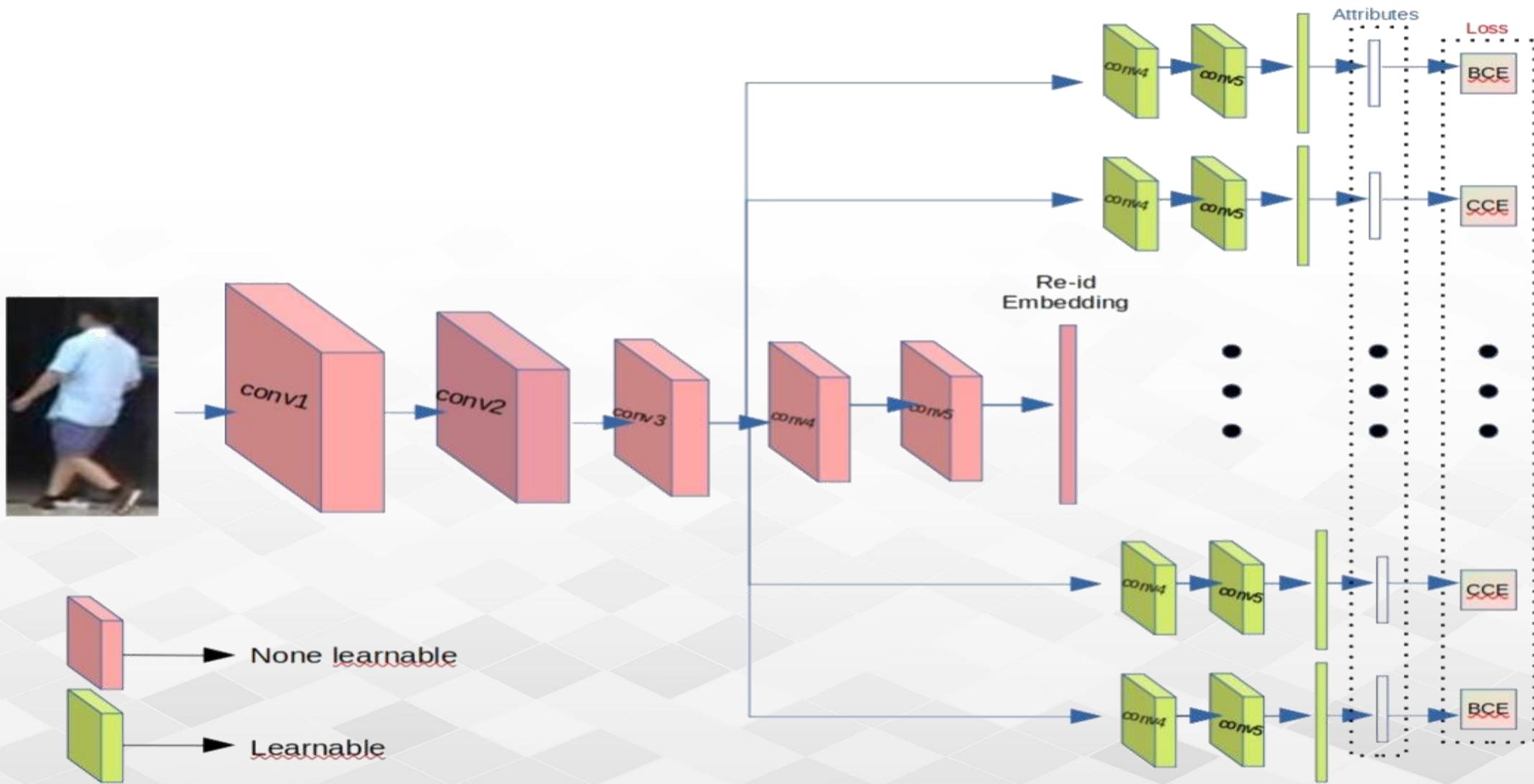


# Separation Index for Multi-label Multi classification





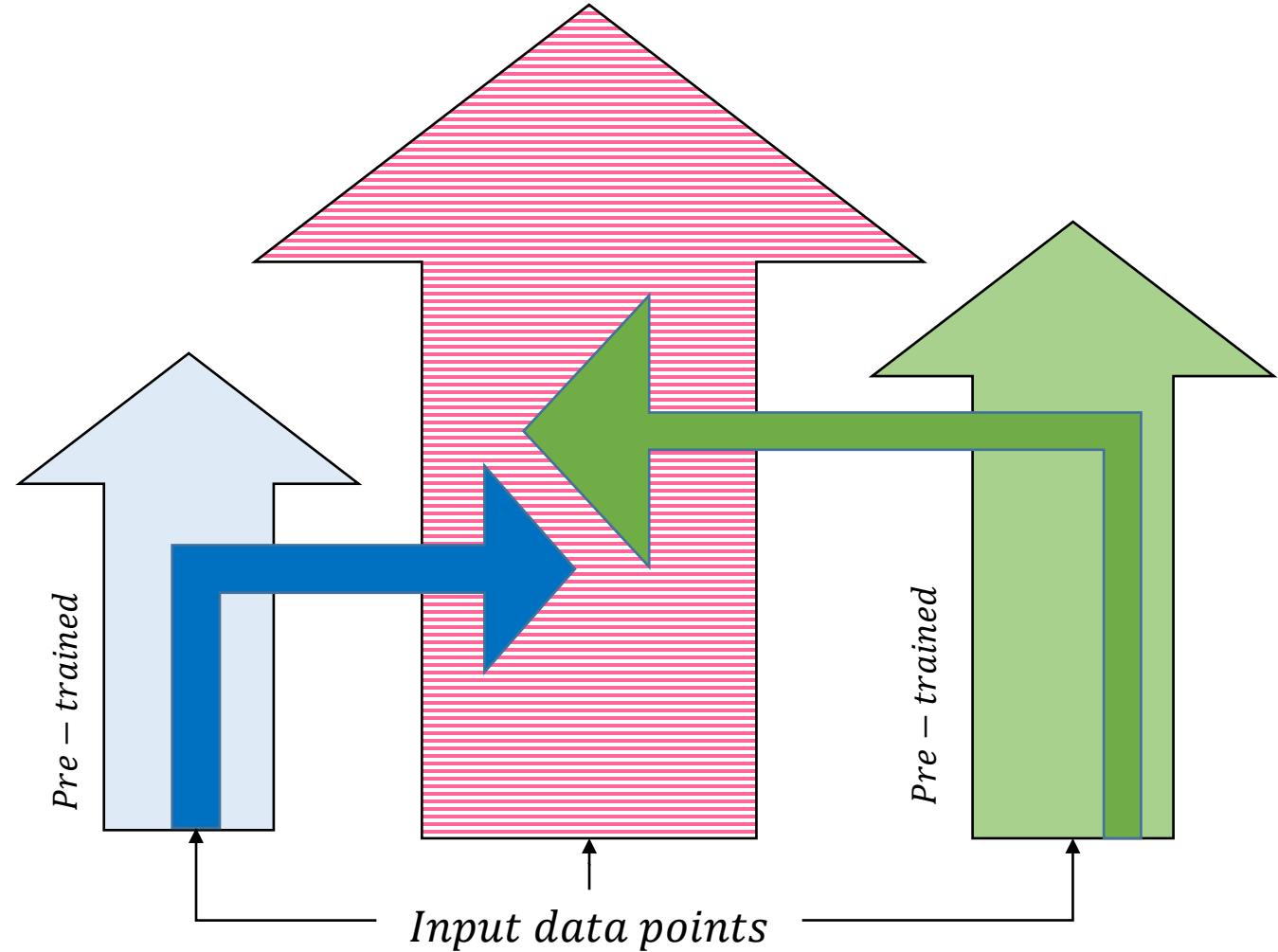
# Multi-Branch Network



### 3.3.6 Layer-wise Fusion

## Layer-wise Fusion

- In a classification (regression) problem, the required features may be extracted by fusing some prepared features from some pre-trained models.
- If we design a new architecture of DNN which can combine desired features from some pre-trained DNNs, more generalized DNNs can be learned.
- In such an architecture, there are a main body which is integrated by some branches from other DNNs at different points. The main body forms the main flow of data but each branch insert information-flow.
- We can use complexity measures like SI(Sml) to find some desired features from other pre-trained DNNs.



### Some notes

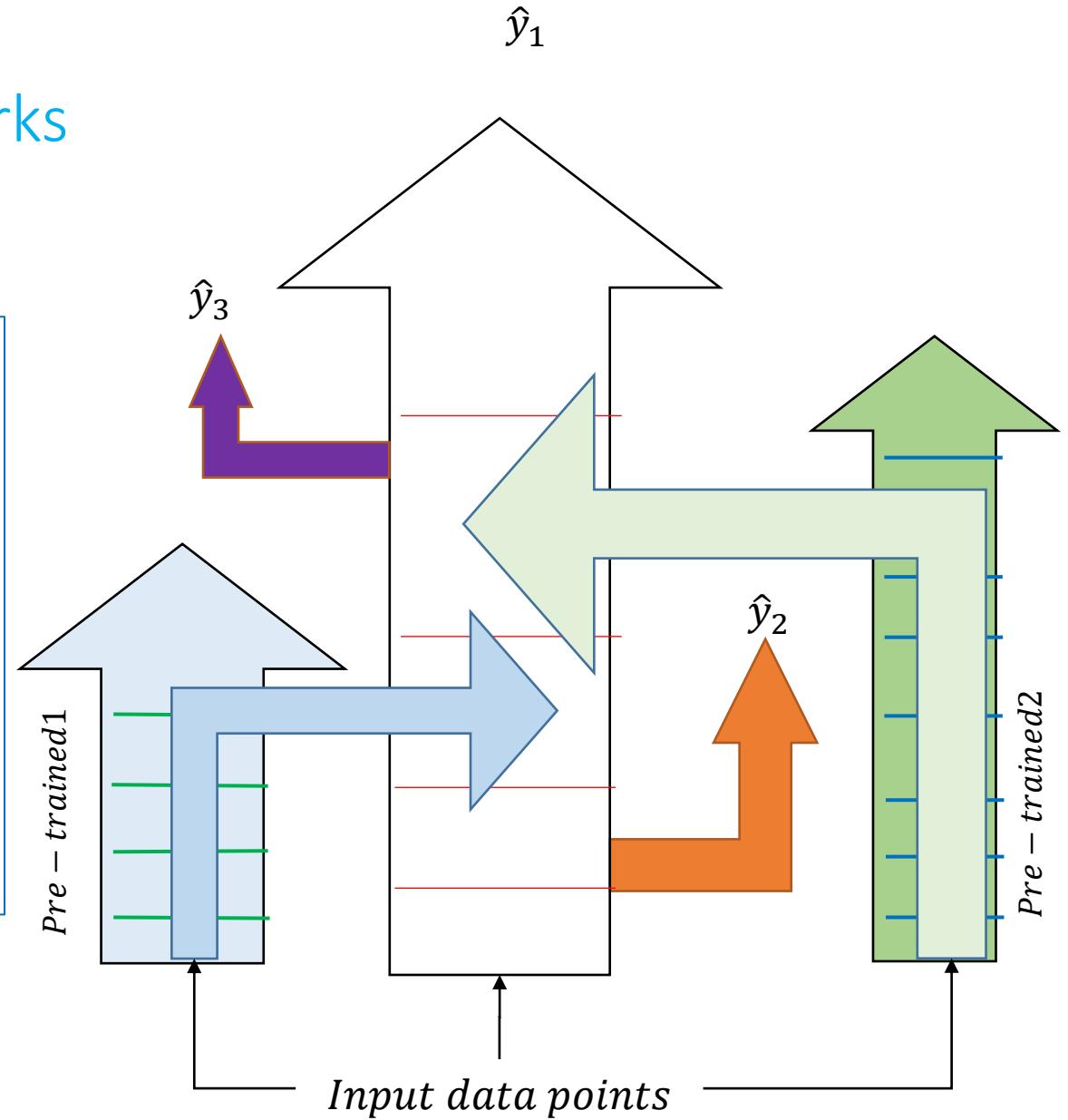
- Fuse at any point where  $SI(Sml)$  may increase significantly.
- Fusion is done in as a forward design operation.

### 3.3.7 Forward Design

## Forward Design of Deep Neural Networks

In forward design of deep neural network, flowing tasks are done in a way that SI(Sml) maximizes in a classification(regression) problem.

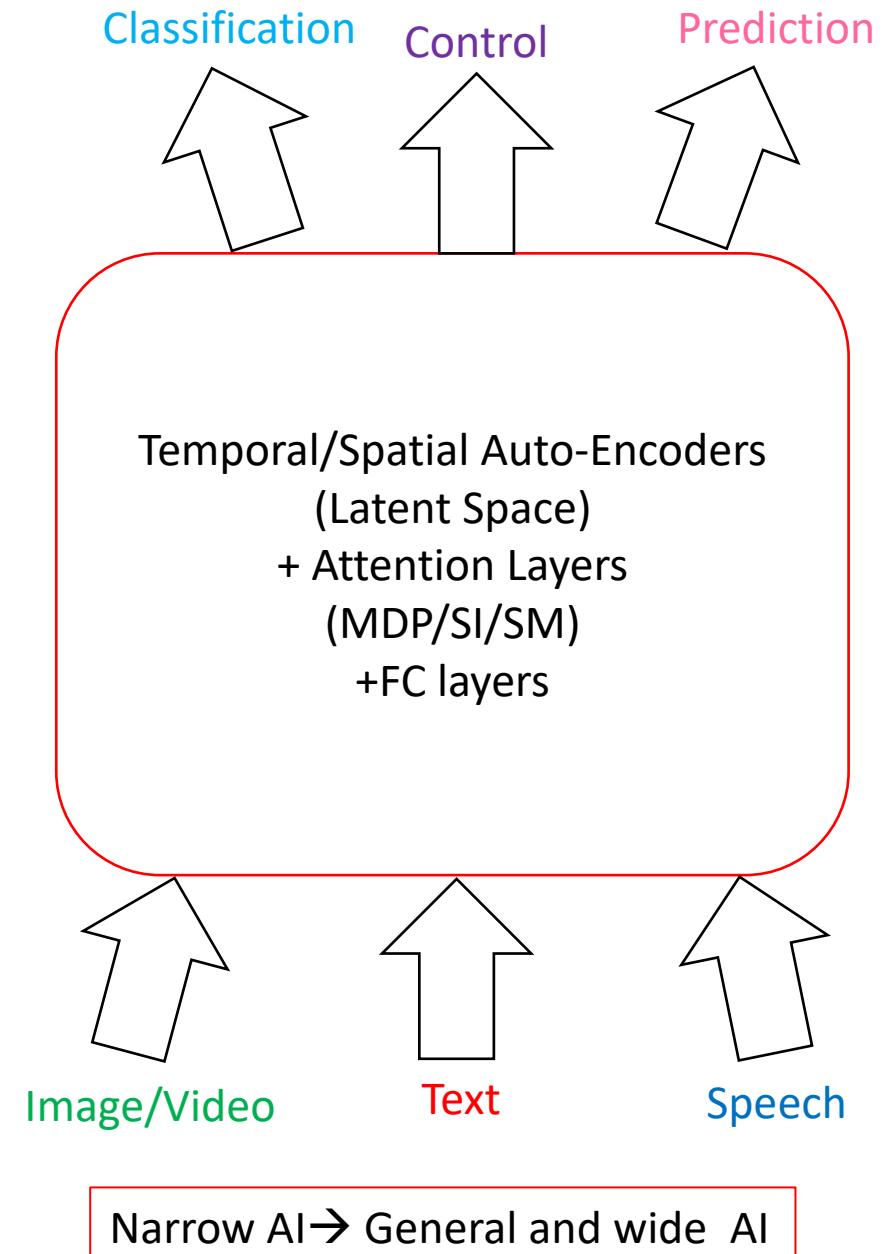
1. Organization of different layers
2. Adjusting the resolution, width, and depth of the network
3. Adding new branches on some layers where SI(Sml) increases significantly.
4. Fusing at some nodes from some other pre-trained networks through which SI(Sml) increases significantly.



### 3.3.8 Forward Multi-Task Design

A neural network which receives multi-modal data and it is trained to do multi-tasks in a supervised method. Such network works as a generalist AI agent. The key idea is that by appropriate temporal/spatial encoders, different data streams are encoded as the latent space; then the required features for each task are chosen from the latent space by using SmI(SI).

- **Language Tasks**
  - Speech recognition
  - Speech generating
  - ...
- **Vision Tasks**
  - Image recognition
  - Action Recognition
  - Text recognition
  - ...
- **Physical world tasks**
  - Motion tasks
  - Grasping tasks
  - ...



## Artificial Brain Design

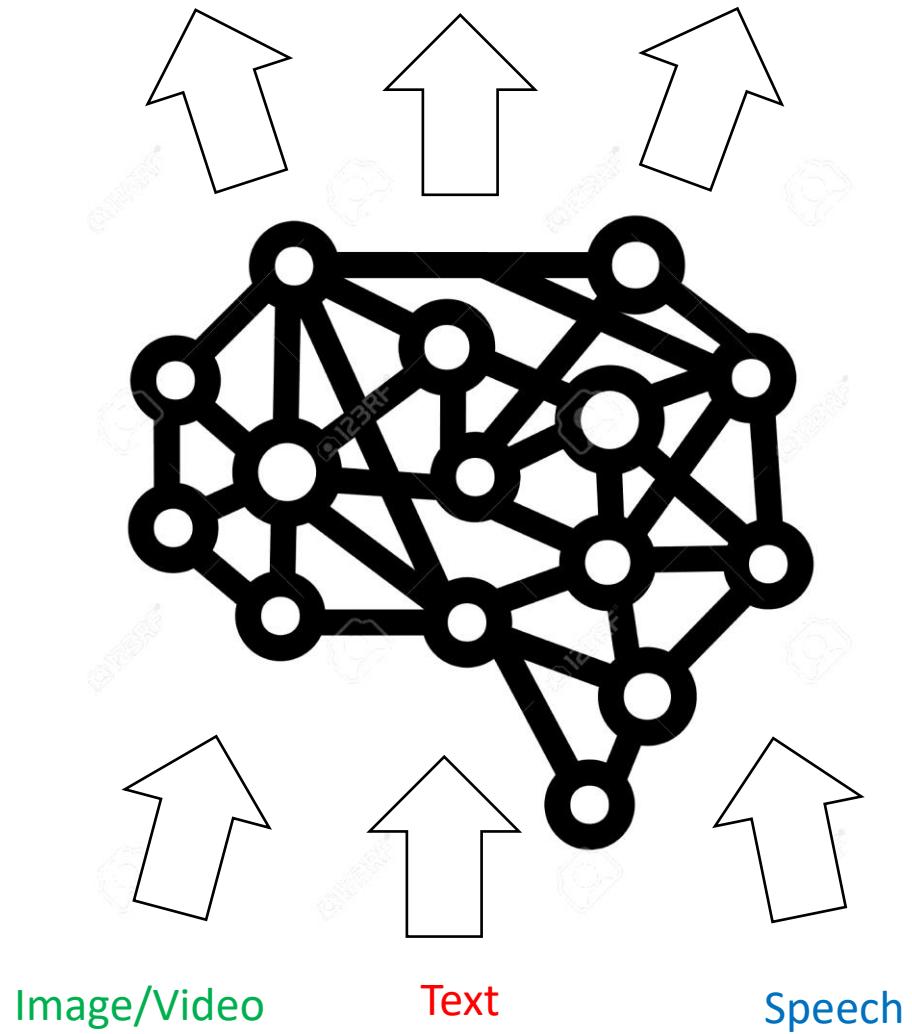
A neural network which receives multi-modal data and it is trained to do multi-tasks in an un-supervised method.

Such network works as a generalist AI agent which must learn without using labels and targets.

Some Steps

1. Receive all measuring signals: Visual, inertial, force,.....
2. Encode them as latent space by using appropriate spatial, temporal, and recurrent encoders
3. Using some attention layers and mechanisms to reveals some useful features which maximizes the gathered information from the environment
4. Learning to motion in the environment and to label objects .
5. Learning to do the required physical tasks.

Classification      Control      Prediction  
Any observed events    Motion-Grasp...    Physical variables



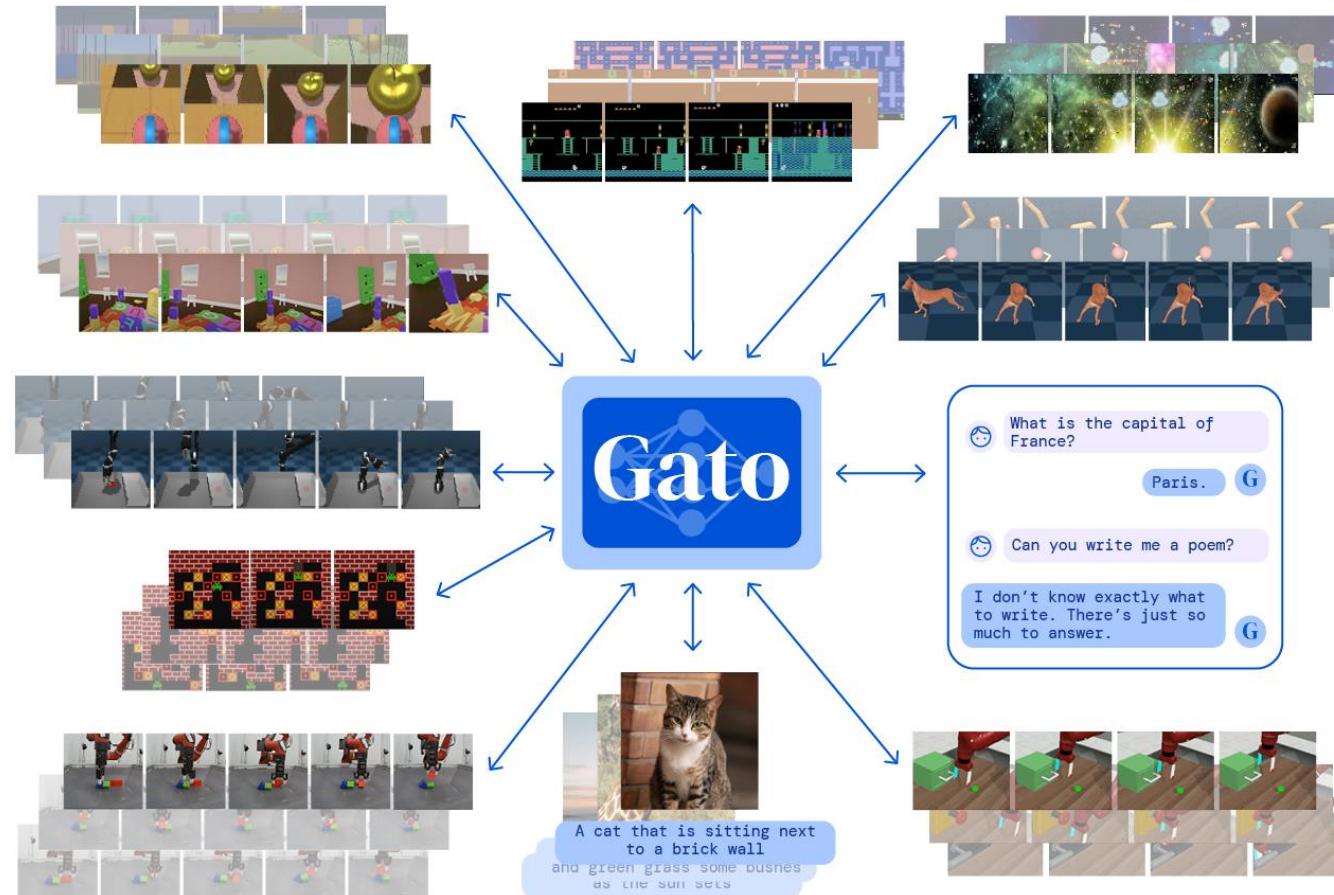
# DeepMind Introduces Gato

DeepMind Introduces Gato, a New Generalist AI AgentGato, as the agent is known, is DeepMinds's generalist AI that can perform many different tasks that humans can do, without carving a niche for itself as an expert on one task. Gato can perform more than 600 different tasks, such as playing video games, captioning images and moving real-world robotic arms. Gato is a multi-modal, multi-task, multi-embodiment generalist policy.

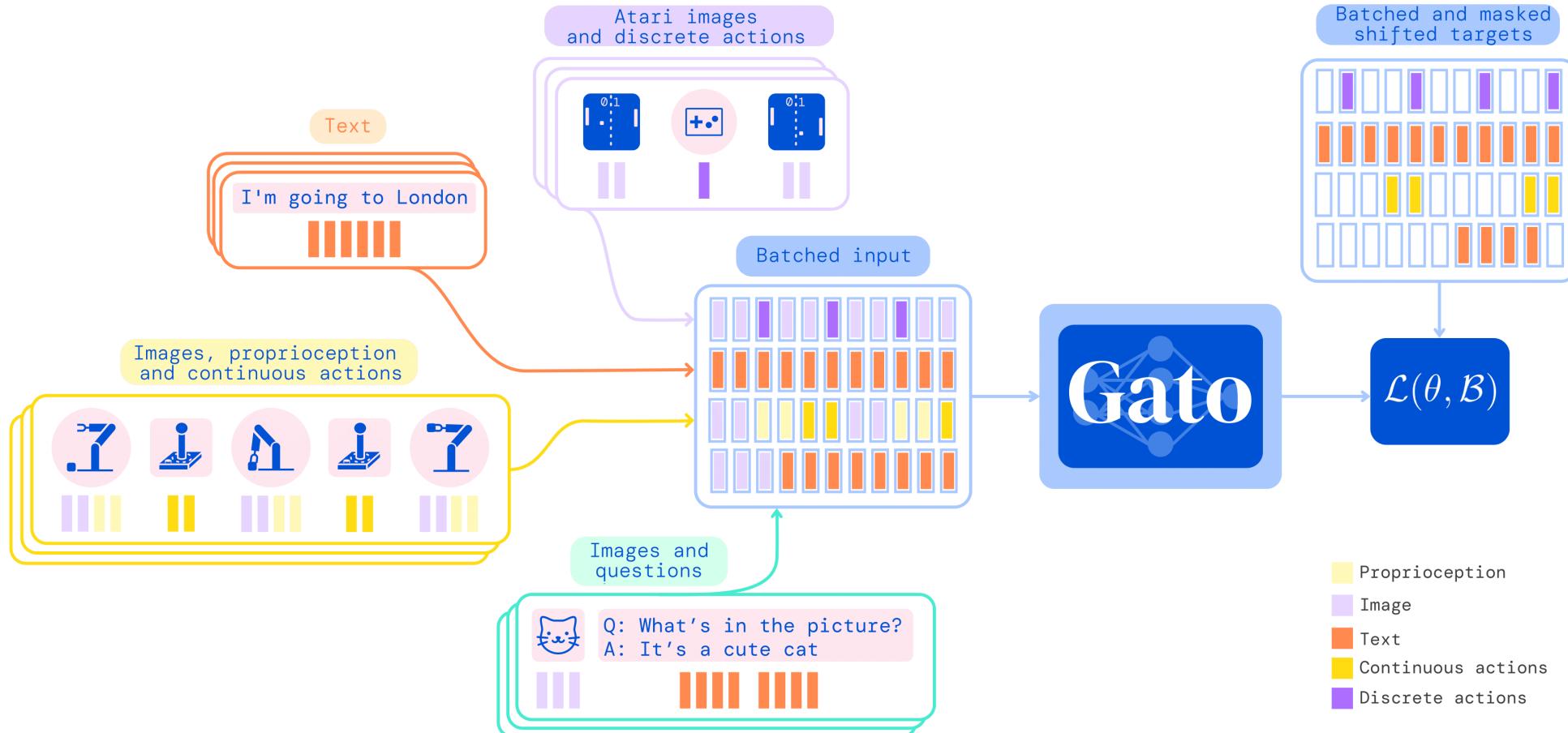
<https://www.youtube.com/watch?v=6fWEHrXN9zo&t=254s>

Integrated AI - Gato by DeepMind (May/2022) 1.2B + Asimo, GPT-3, Tesla Optimus, Boston Dynamics

Inspired by progress in large-scale language modelling, we apply a similar approach towards building a single generalist agent beyond the realm of text outputs. The agent, which we refer to as Gato, works as a multi-modal, multi-task, multi-embodiment generalist policy. The same network with the same weights can play Atari, caption images, chat, stack blocks with a real robot arm and much more, deciding based on its context whether to output text, joint torques, button presses, or other tokens.



During the training phase of Gato, data from different tasks and modalities are serialised into a flat sequence of tokens, batched, and processed by a transformer neural network similar to a large language model. The loss is masked so that Gato only predicts action and text targets.



**Token:** something serving to represent or indicate some fact, event, feeling, etc.

**3.3.9** Design and optimize a classifier with high order SI

**3.3.10** Design and optimize a regression model with high order Sml

# Applying High-order SI to design and optimize a shallow model as a classifier

- Actually a suitable classifier model should make outputs which have perfect high order SI with the labels.
- It is enough to find local regions of the data which include examples with the same labels and then to model them with a MLP with two hidden layers.
- In this idea, a fast incremental clustering method ([divide and conquer method](#)) is employed to localize examples with the same labels in some convex hyper-polygon regions .
- Next, all [convex hyper](#) polygons with the same label can join together to make the output of each class.
- The number of neurons, and parameters of [a MLP with two hidden layers](#) are optimized in this approach.
- The model then can be [compressed](#) to a model with lower neurons and parameters.

# Applying high order Sml to design and optimize a shallow model as a regression model

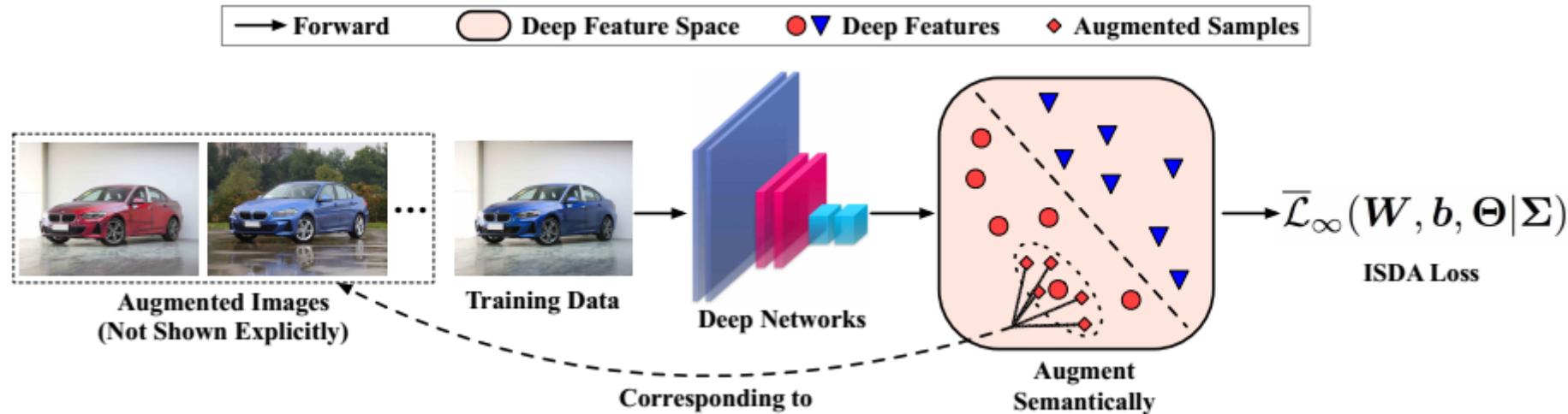
- Actually a suitable regression model should provide outputs which make **perfect high order Sml** with the targets.
- It is enough **to find local regions** of the data which can be **fitted by a local basis function** (the fitness can be **measured by the concept of Sml**)
- Then the basis function are interpolated and provide outputs which make high fitness with the target. (the fitness can be **measured by the concept of Sml**)
- Based on the divide and conquer approach, we can use a fast incremental clustering method to localize and optimize such basis functions.

### 3.4 Related works in local Layer-wise learning

- To introduce a learning approach that is more plausible with the learning in the brain
- To have better learning and design in accuracy, compactness, speed,...
- To increase the generalization and robustness of DNNs

# (1) Implicit Semantic Data Augmentation for Deep Networks

Yulin Wang et al. 2020, Department of Automation, Tsinghua University, Beijing, China



The proposed implicit semantic data augmentation (ISDA) has two important components, i.e., (1) online estimation of class-conditional covariance matrices and (2) optimization with a robust loss function.

The first component aims to find a distribution from which we can sample meaningful semantic transformation directions for data augmentation, while the second saves us from explicitly generating a large amount of extra training data, leading to remarkable efficiency compared to existing data augmentation techniques.

## Comparison Tables

*Image net →  
Cifar10 + Cifar100 ↓*

Table 1: Single crop error rates (%) of different deep networks on the validation set of ImageNet. We report the results of our implementation with and without ISDA. The better results are **bold-faced**, while the numbers in brackets denote the performance improvement achieved by ISDA. We also report the theoretical computational overhead and the additional training time introduced by ISDA in the last two columns, which is obtained with 8 Tesla V100 GPUs.

Networks	Params	Top-1 / Top-5 Error Rates (%)		Additional Cost (Theoretical)	Additional Cost (Wall Time)
		Baselines	ISDA		
ResNet-50 [4]	25.6M	23.0 / 6.8	<b>21.9<sub>(1.1)</sub> / 6.3</b>	0.25%	7.6%
ResNet-101 [4]	44.6M	21.7 / 6.1	<b>20.8<sub>(0.9)</sub> / 5.7</b>	0.13%	7.4%
ResNet-152 [4]	60.3M	21.3 / 5.8	<b>20.3<sub>(1.0)</sub> / 5.5</b>	0.09%	5.4%
DenseNet-BC-121 [5]	8.0M	23.7 / 6.8	<b>23.2<sub>(0.5)</sub> / 6.6</b>	0.20%	5.6%
DenseNet-BC-265 [5]	33.3M	21.9 / 6.1	<b>21.2<sub>(0.7)</sub> / 6.0</b>	0.24%	5.4%
ResNeXt-50, 32x4d [33]	25.0M	22.5 / 6.4	<b>21.3<sub>(1.2)</sub> / 5.9</b>	0.24%	6.6%
ResNeXt-101, 32x8d [33]	88.8M	21.1 / 5.9	<b>20.1<sub>(1.0)</sub> / 5.4</b>	0.06%	7.9%

Table 2: Evaluation of ISDA on CIFAR with different models. The average test error over the last 10 epochs is calculated in each experiment, and we report mean values and standard deviations in three independent experiments. The best results are **bold-faced**.

Method	Params	CIFAR-10	CIFAR-100
ResNet-32 [4]	0.5M	$7.39 \pm 0.10\%$	$31.20 \pm 0.41\%$
ResNet-32 + ISDA	0.5M	<b><math>7.09 \pm 0.12\%</math></b>	<b><math>30.27 \pm 0.34\%</math></b>
ResNet-110 [4]	1.7M	$6.76 \pm 0.34\%$	$28.67 \pm 0.44\%$
ResNet-110 + ISDA	1.7M	<b><math>6.33 \pm 0.19\%</math></b>	<b><math>27.57 \pm 0.46\%</math></b>
SE-ResNet-110 [34]	1.7M	$6.14 \pm 0.17\%$	$27.30 \pm 0.03\%$
SE-ResNet-110 + ISDA	1.7M	<b><math>5.96 \pm 0.21\%</math></b>	<b><math>26.63 \pm 0.21\%</math></b>
Wide-ResNet-16-8 [35]	11.0M	$4.25 \pm 0.18\%$	$20.24 \pm 0.27\%$
Wide-ResNet-16-8 + ISDA	11.0M	<b><math>4.04 \pm 0.29\%</math></b>	<b><math>19.91 \pm 0.21\%</math></b>
Wide-ResNet-28-10 [35]	36.5M	$3.82 \pm 0.15\%$	$18.53 \pm 0.07\%$
Wide-ResNet-28-10 + ISDA	36.5M	<b><math>3.58 \pm 0.15\%</math></b>	<b><math>17.98 \pm 0.15\%</math></b>
ResNeXt-29, 8x64d [33]	34.4M	$3.86 \pm 0.14\%$	$18.16 \pm 0.13\%$
ResNeXt-29, 8x64d + ISDA	34.4M	<b><math>3.67 \pm 0.12\%</math></b>	<b><math>17.43 \pm 0.25\%</math></b>
DenseNet-BC-100-12 [5]	0.8M	$4.90 \pm 0.08\%$	$22.61 \pm 0.10\%$
DenseNet-BC-100-12 + ISDA	0.8M	<b><math>4.54 \pm 0.07\%</math></b>	<b><math>22.10 \pm 0.34\%</math></b>
DenseNet-BC-190-40 [5]	25.6M	$3.52\%$	$17.74\%$
DenseNet-BC-190-40 + ISDA	25.6M	<b><math>3.24\%</math></b>	<b><math>17.42\%</math></b>

### Algorithm 1 The ISDA Algorithm.

---

```

1: Input:  $\mathcal{D}$ ,  $\lambda_0$ 
2: Randomly initialize  $\mathbf{W}$ ,  $\mathbf{b}$  and  $\Theta$ 
3: for  $t = 0$  to  $T$  do
4:   Sample a mini-batch  $\{\mathbf{x}_i, y_i\}_{i=1}^B$  from  $\mathcal{D}$ 
5:   Compute  $\mathbf{a}_i = G(\mathbf{x}_i, \Theta)$ 
6:   Estimate the covariance matrices  $\Sigma_1, \Sigma_2, \dots, \Sigma_C$ 
7:   Compute  $\bar{\mathcal{L}}_\infty$  according to Eq. (4)
8:   Update  $\mathbf{W}, \mathbf{b}, \Theta$  with SGD
9: end for
10: Output:  $\mathbf{W}, \mathbf{b}$  and  $\Theta$ 

```

---

$$\mathcal{L}_\infty(\mathbf{W}, \mathbf{b}, \Theta | \Sigma) \leq \frac{1}{N} \sum_{i=1}^N -\log\left(\frac{e^{\mathbf{w}_{y_i}^T \mathbf{a}_i + b_{y_i}}}{\sum_{j=1}^C e^{\mathbf{w}_j^T \mathbf{a}_i + b_j + \frac{\lambda}{2} (\mathbf{w}_j^T - \mathbf{w}_{y_i}^T) \Sigma_{y_i} (\mathbf{w}_j - \mathbf{w}_{y_i})}}\right) \triangleq \bar{\mathcal{L}}_\infty.$$

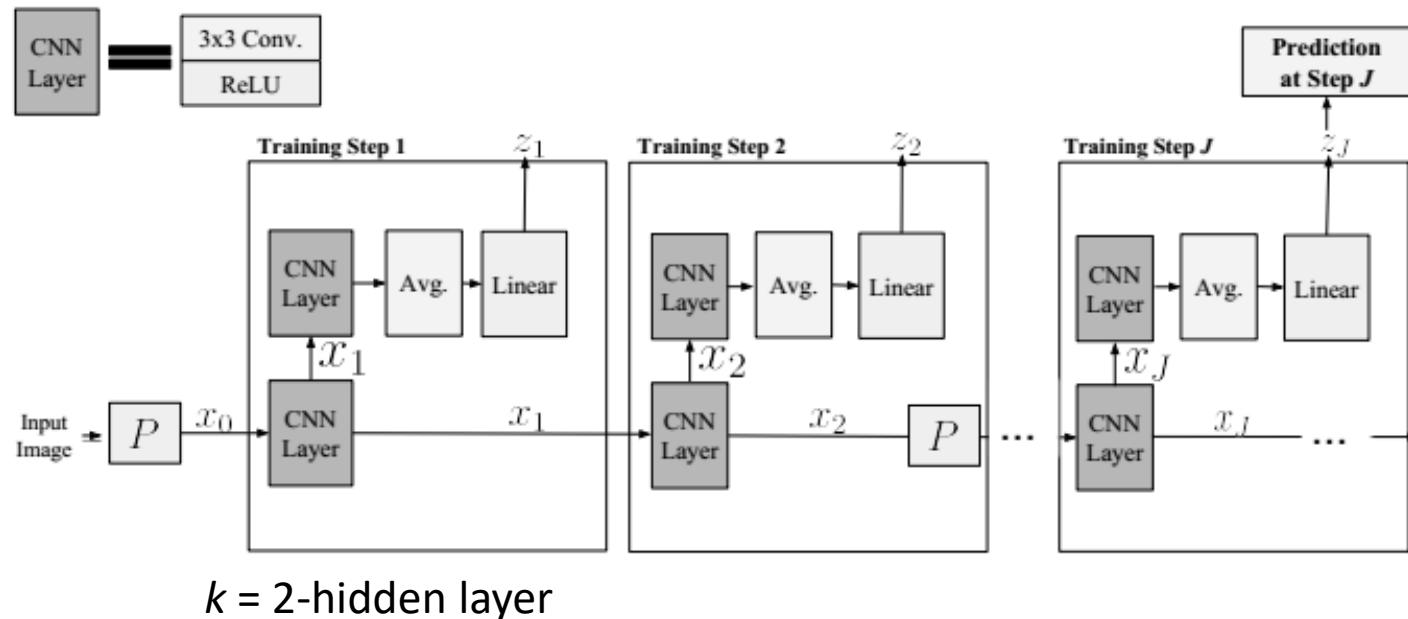
$\lambda$  is a positive coefficient to control the strength of semantic data augmentation. The strength at initial epochs is chosen low but linearly it increases:  $\lambda = (t/T) \times \lambda_0$

# (2) Greedy Layerwise Learning Can Scale to ImageNet

Bellovsky et al. (University of Montreal)-*Proceedings of the 36 th International Conference on Machine Learning*, Long Beach, California, PMLR 97, 2019.

We find that solving sequential 1-hidden-layer auxiliary problems lead to a CNN that exceeds AlexNet performance on ImageNet.

Extending this training methodology to construct individual layers by solving 2-and-3-hidden layer auxiliary problems, we obtain an 11-layer network that exceeds several members of the VGG model family on ImageNet, and can train a VGG-11 model to the same accuracy as end-to-end learning.



---

## Algorithm 1 Layer Wise CNN

---

**Input:** Training samples  $\{x_0^n, y^n\}_{n \leq N}$   
**for**  $j \in 0..J - 1$  **do**  
    Compute  $\{x_j^n\}_{n \leq N}$  (via Eq.(1))  
     $(\theta_j^*, \gamma_j^*) = \arg \min_{\theta_j, \gamma_j} \hat{\mathcal{R}}(z_{j+1}; \theta_j, \gamma_j)$   
**end for**

---

# Comparison tables

<b>Layer-wise Trained</b>	Acc. (Ens.)
SimCNN ( $k = 1$ train )	88.3 (88.4)
SimCNN ( $k = 2$ train)	90.4 (90.7)
SimCNN( $k = 3$ train)	<b>91.7 (92.8)</b>
BoostResnet (Huang et al., 2017)	82.1
ProvableNN (Malah et al., 2018) (Mosca et al., 2017)	73.4 81.6
<b>Reference e2e</b>	
AlexNet	89
VGG <sup>1</sup>	92.5
WRN 28-10 (Zagoruyko et al. 2016)	<b>96.0</b>
<b>Alternatives</b>	[Ref.]
Scattering + Linear	82.3
FeedbackAlign (Bartunov et al., 2018)	62.6 [67.6]

Table 2. Results on CIFAR-10. Compared to the few existing methods using *only* layerwise training schemes we report much more competitive results to well known benchmark models that like ours do not use skip connections. In brackets e2e trained version of the model is shown when available.

$k = 1$  :the auxiliary classifier consists of only the linear  $A$  and  $L$  operators (1-hidden layer NN)  
 $K > 1$  : $K$  hidden layers for auxiliary classifiers

	Top-1 (Ens.)	Top-5 (Ens.)
SimCNN ( $k = 1$ train)	58.1 (59.3)	79.7 (80.8)
SimCNN ( $k = 2$ train)	65.7 (67.1)	86.3 (87.0)
SimCNN ( $k = 3$ train)	69.7 (71.6)	88.7 (89.8)
VGG-11 ( $k = 3$ train)	67.6 (70.1)	88.0 (89.2)
VGG-11 (e2e train)	67.9	88.0
<b>Alternative</b>	[Ref.]	[Ref.]
DTarGetProp (Bartunov et al., 2018)	1.6 [28.6]	5.4 [51.0]
FeedbackAlign (Xiao et al., 2019)	6.6 [50.9]	16.7 [75.0]
Scat. + Linear (Oyallon et al., 2018)	17.4	N/A
Random CNN	12.9	N/A
FV + Linear (Sánchez et al., 2013)	54.3	74.3
<b>Reference e2e CNN</b>		
AlexNet	56.5	79.1
VGG-13	69.9	89.3
VGG-19	72.9	90.9
Resnet-152	78.3	94.1

Table 3. Single crop validation acc. on ImageNet. Our SimCNN models use  $J = 8$ . In parentheses see the ensemble prediction.

# (3) Hebbian Semi-Supervised Learning in a Sample Efficiency Setting?

Gabriele Lagani, 2021, Computer Science Department, University of Pisa, Pisa, Italy

## Highlights

1. A semi supervised training strategy that combines Hebbian learning with gradient descent  
**Hebbian for internal layers + SGD for the final fully connected layer**
2. All internal layers (both convolutional and fully connected) are pre-trained using an unsupervised approach based on Hebbian learning
3. The last fully connected layer (the classification layer) is trained using Stochastic Gradient Descent (SGD)
4. The learning approach has been applied to CIFAR10, CIFAR100, tiny ImageNet
5. In regimes when the number of available labeled samples is low the proposed approach outperforms the other approaches in almost all the cases

# Hebian

$$\mathbf{w}_{new} = \mathbf{w}_{old} + \Delta \mathbf{w}$$

$$\Delta \mathbf{w} = \eta y \mathbf{x}$$

“neurons that fire together wire together.”

$$\Delta \mathbf{w} = \eta y \mathbf{x} - \lambda \mathbf{w}$$

Adding Stable Term

In particular, the term  $\lambda$  can be chosen in the form  $\lambda = \eta y$

$$\Delta \mathbf{w} = \eta y \mathbf{x} - \eta y \mathbf{w} = \eta y (\mathbf{x} - \mathbf{w})$$

In a complex neural network, we need a strategy to prevent neurons from learning redundant information.

The Hebbian PCA learning rule is obtained by minimizing the representation error loss function, defined as:

$$L(\mathbf{w}_i) = E[(\mathbf{x} - \sum_{j=1}^i y_j \mathbf{w}_j)^2]$$

The classification layer is placed on top of the various internal layers ( $L_1; \dots; L_5$ )

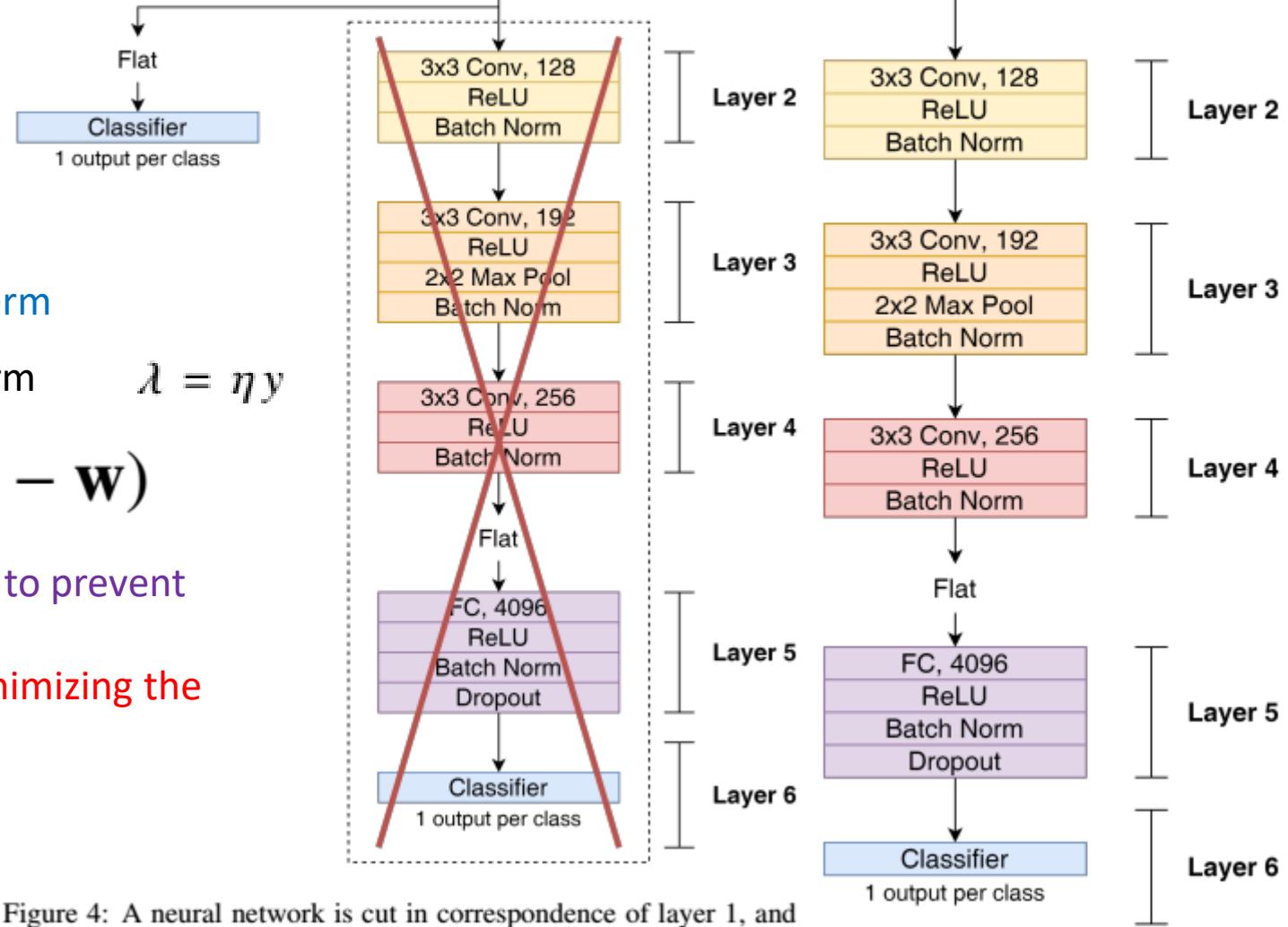


Figure 4: A neural network is cut in correspondence of layer 1, and a linear classifier is placed on top of the features extracted from that layer, in order to evaluate their quality in classification tasks.

1: The neural network used for the experiments.

## Comparison Table(CIFAR10)

Table 1: CIFAR10 accuracy (top-1) and 95% confidence intervals, obtained with a linear classifier on top of various layers, for the various sample efficiency regimes. Results obtained with supervised backprop (BP), VAE-based semi-supervised approach(VAE), Hebbian PCA (HPCA), and HPCA plus Fine Tuning (HPCA+FT) are compared. It is possible to observe that, in regimes where the number of available samples is low

(roughly between 1% and 5% of the total available samples), HPCA performs better than BP and VAE approaches in almost all the cases, leading to an improvement up to almost 5% (on layer 3, in the 1% regime) w.r.t. non-Hebbian approaches. HPCA+FT helps to further boost accuracy.

Regime: r%- where the percentage of labeled samples is r%

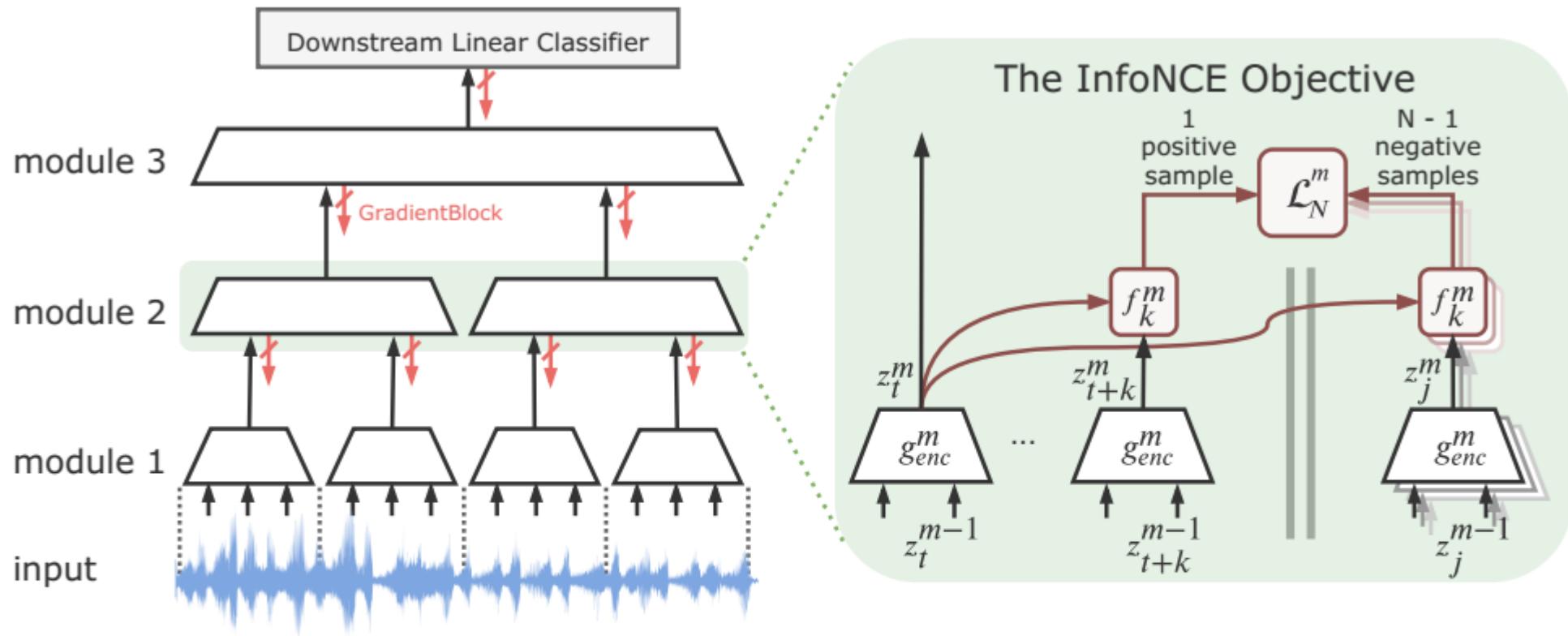
Regimes	Method	L1	L2	L3	L4	L5
1%	BP	33.27 $\pm$ 0.44	34.56 $\pm$ 0.34	36.80 $\pm$ 0.52	35.47 $\pm$ 0.58	35.18 $\pm$ 0.57
	VAE	33.54 $\pm$ 0.27	34.41 $\pm$ 0.84	29.92 $\pm$ 1.25	24.91 $\pm$ 0.66	22.54 $\pm$ 0.60
	HPCA	36.78 $\pm$ 0.46	37.26 $\pm$ 0.14	41.31 $\pm$ 0.57	39.33 $\pm$ 0.72	38.46 $\pm$ 0.44
	HPCA+FT	<b>37.01</b> $\pm$ 0.42	<b>37.65</b> $\pm$ 0.19	<b>41.88</b> $\pm$ 0.53	<b>40.06</b> $\pm$ 0.65	<b>39.75</b> $\pm$ 0.50
2%	BP	37.33 $\pm$ 0.25	39.01 $\pm$ 0.19	42.34 $\pm$ 0.51	41.50 $\pm$ 0.32	41.10 $\pm$ 0.39
	VAE	37.65 $\pm$ 0.35	39.13 $\pm$ 0.40	36.52 $\pm$ 0.47	29.39 $\pm$ 0.32	26.78 $\pm$ 0.72
	HPCA	41.13 $\pm$ 0.30	<b>41.63</b> $\pm$ 0.18	45.76 $\pm$ 0.41	44.70 $\pm$ 0.45	43.15 $\pm$ 0.45
	HPCA+FT	<b>41.60</b> $\pm$ 0.28	42.12 $\pm$ 0.24	<b>46.56</b> $\pm$ 0.38	<b>45.61</b> $\pm$ 0.19	<b>45.51</b> $\pm$ 0.43
3%	BP	40.49 $\pm$ 0.26	41.90 $\pm$ 0.40	45.13 $\pm$ 0.53	45.26 $\pm$ 0.22	44.52 $\pm$ 0.24
	VAE	41.22 $\pm$ 0.27	43.16 $\pm$ 0.44	42.60 $\pm$ 0.87	31.91 $\pm$ 0.44	29.00 $\pm$ 0.33
	HPCA	44.16 $\pm$ 0.42	44.84 $\pm$ 0.08	48.92 $\pm$ 0.17	47.70 $\pm$ 0.57	45.60 $\pm$ 0.27
	HPCA+FT	<b>44.74</b> $\pm$ 0.08	<b>45.61</b> $\pm$ 0.28	<b>49.75</b> $\pm$ 0.41	<b>48.94</b> $\pm$ 0.45	<b>48.80</b> $\pm$ 0.27
4%	BP	43.38 $\pm$ 0.22	45.43 $\pm$ 0.18	49.51 $\pm$ 0.49	48.96 $\pm$ 0.48	48.80 $\pm$ 0.24
	VAE	44.39 $\pm$ 0.30	45.88 $\pm$ 0.39	46.01 $\pm$ 0.40	34.26 $\pm$ 0.21	31.15 $\pm$ 0.35
	HPCA	46.37 $\pm$ 0.16	47.16 $\pm$ 0.28	50.70 $\pm$ 0.26	49.45 $\pm$ 0.15	47.75 $\pm$ 0.54
	HPCA+FT	<b>47.10</b> $\pm$ 0.25	<b>48.26</b> $\pm$ 0.09	<b>52.00</b> $\pm$ 0.16	<b>51.05</b> $\pm$ 0.29	<b>51.28</b> $\pm$ 0.28
5%	BP	45.11 $\pm$ 0.21	47.57 $\pm$ 0.29	50.61 $\pm$ 0.32	50.54 $\pm$ 0.23	50.42 $\pm$ 0.14
	VAE	46.31 $\pm$ 0.39	48.21 $\pm$ 0.21	48.98 $\pm$ 0.34	36.32 $\pm$ 0.35	32.75 $\pm$ 0.32
	HPCA	47.51 $\pm$ 0.65	48.69 $\pm$ 0.37	51.69 $\pm$ 0.56	50.44 $\pm$ 0.43	48.51 $\pm$ 0.32
	HPCA+FT	<b>48.49</b> $\pm$ 0.44	<b>50.14</b> $\pm$ 0.46	<b>53.33</b> $\pm$ 0.52	<b>52.49</b> $\pm$ 0.16	<b>52.20</b> $\pm$ 0.37
10%	BP	51.60 $\pm$ 0.40	54.60 $\pm$ 0.31	57.97 $\pm$ 0.28	<b>57.63</b> $\pm$ 0.23	57.30 $\pm$ 0.22
	VAE	53.83 $\pm$ 0.26	<b>56.33</b> $\pm$ 0.22	57.85 $\pm$ 0.22	52.26 $\pm$ 1.08	45.67 $\pm$ 1.15
	HPCA	52.57 $\pm$ 0.29	53.29 $\pm$ 0.25	56.09 $\pm$ 0.38	54.24 $\pm$ 0.28	52.68 $\pm$ 0.36
	HPCA+FT	<b>54.36</b> $\pm$ 0.32	56.08 $\pm$ 0.28	<b>58.46</b> $\pm$ 0.15	56.54 $\pm$ 0.23	<b>57.35</b> $\pm$ 0.18
25%	BP	60.43 $\pm$ 0.26	64.96 $\pm$ 0.18	66.63 $\pm$ 0.17	68.04 $\pm$ 0.05	68.04 $\pm$ 0.20
	VAE	<b>62.51</b> $\pm$ 0.24	<b>67.26</b> $\pm$ 0.32	<b>68.48</b> $\pm$ 0.21	<b>68.79</b> $\pm$ 0.29	<b>68.70</b> $\pm$ 0.15
	HPCA	58.30 $\pm$ 0.28	59.20 $\pm$ 0.24	59.98 $\pm$ 0.20	57.54 $\pm$ 0.20	56.46 $\pm$ 0.18
	HPCA+FT	61.45 $\pm$ 0.26	65.25 $\pm$ 0.16	64.71 $\pm$ 0.17	62.43 $\pm$ 0.13	64.77 $\pm$ 0.22
100%	BP	61.59 $\pm$ 0.08	67.67 $\pm$ 0.11	73.87 $\pm$ 0.15	83.88 $\pm$ 0.04	84.71 $\pm$ 0.02
	VAE	<b>67.53</b> $\pm$ 0.22	<b>75.83</b> $\pm$ 0.31	<b>80.78</b> $\pm$ 0.28	<b>84.27</b> $\pm$ 0.35	<b>85.23</b> $\pm$ 0.26
	HPCA	64.69 $\pm$ 0.29	65.92 $\pm$ 0.14	64.43 $\pm$ 0.21	61.24 $\pm$ 0.22	61.16 $\pm$ 0.33
	HPCA+FT	66.76 $\pm$ 0.13	75.16 $\pm$ 0.20	79.90 $\pm$ 0.18	83.55 $\pm$ 0.33	84.38 $\pm$ 0.22

## (4) Greedy InfoMax for Self-Supervised Representation Learning(GIM)

Sindy Lowe et al. University of Amsterdam

### Highlights

1. we propose a novel deep learning method for local self-supervised representation learning that does not require labels nor end-to-end backpropagation but exploits the natural order in data instead.
2. Inspired by the observation that biological neural networks appear to learn without backpropagating a global error signal, we split a deep neural network into a stack of gradient-isolated modules.
3. Each module is trained to maximize the mutual information between its consecutive outputs using the InfoNCE bound from Oord et al.(2018).



**Figure 1.** The Greedy InfoMax Learning Approach. **(Left)** For the self-supervised learning of representations, we stack a number of modules through which the input is forward-propagated in the usual way, but gradients do not propagate backwards. Instead, every module is trained greedily using a local loss. **(Right)** Every encoding module maps its inputs  $z_t^{m-1}$  at time-step  $t$  to  $g_{enc}^m(\text{GradientBlock}(z_t^{m-1})) = z_t^m$ , which is used as the input for the following module. The InfoNCE objective is used for its greedy optimization. This loss is calculated by contrasting the predictions of a module for its future representations  $z_{t+k}^m$  against negative samples  $z_j^m$ , which enforces each module to maximally preserve the information of its inputs. We employ an additional autoregressive module  $g_{ar}$ , which is not depicted here.

# Architecture & Comparison

Table 3. General outline of our architecture

Layer	Output Size (Sequence Length × Channels)	Parameters		
		Kernel	Stride	Padding
Input	20480 × 1			
Conv1	$4095^2 \times 512$	10	5	2
Conv2	$1023^2 \times 512$	8	4	2
Conv3	$512^2 \times 512$	4	2	2
Conv4	$257^2 \times 512$	4	2	2
Conv5	$128 \times 512$	1	2	1
GRU	$128 \times 256$	-	-	-

<sup>2</sup>For applying the InfoNCE objective on these layers, we randomly sample a time-window of size 128 to decrease the dimensionality.

Table 1. Results for classifying speaker identity and phone labels in the LibriSpeech dataset. All models use the same audio input sizes and the same architecture. GIM creates representations that are useful for audio classification tasks despite its greedy training and lack of a global objective.

Method	Phone	Speaker
	Classification Accuracy	Classification Accuracy
MFCC features	39.7%	17.6%
Randomly initialized	27.6%	1.9%
Supervised	74.6%	98.5%
Greedy Supervised	71.1%	84.5%
CPC (Oord et al., 2018) <sup>a</sup>	64.6%	97.4%
Greedy InfoMax (GIM)	60.0%	97.5%

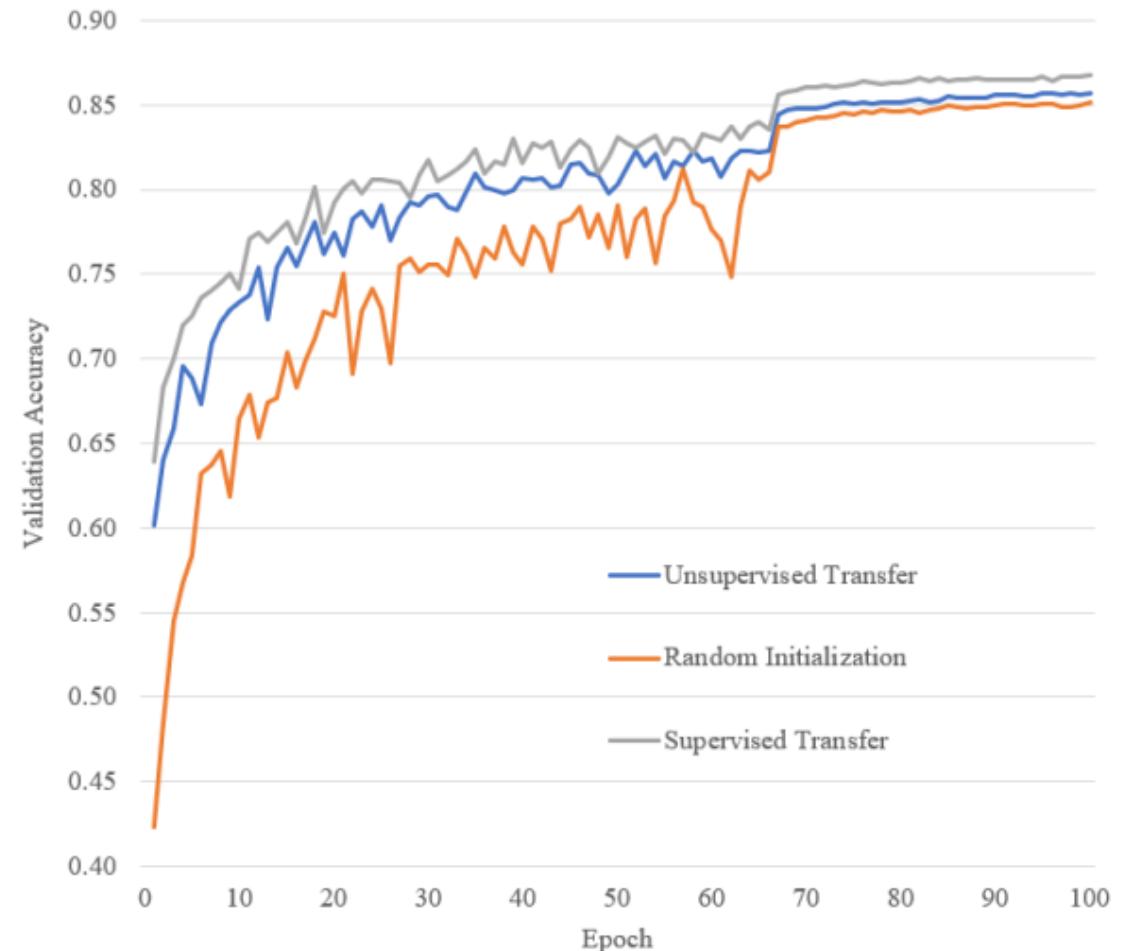
<sup>a</sup>In our reimplementation, we achieved 62.2% for the phone and 98.8% for the speaker classification task.

# (5)Layer-Wise Contrastive Unsupervised Representation Learning

Stephen Zhao, 2019

- In this work, we focus on the set of unsupervised learning approaches that Arora et al. (2019) call “contrastive unsupervised representation learning”.
- We use a layer-wise adaptation, starting within the context of images.
- We train feature representations of semantically similar images (i.e. nearby patches within the same image) to be closer than that of unrelated images (e.g. random patches), to learn convolutional neural network filters from unlabeled data.

Figure 3: Performance on CIFAR-10. The learning rate is 0.001 for the first two-thirds of the training, and is 0.0001 afterwards. Results are averaged over 3 independent runs for each line (keeping transferred filters constant across runs, but allowing for fine-tuning).



# (6) LoCo: Local Contrastive Representation Learning

Yuwen Xiong et al.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

- In this work, we discover that by overlapping local blocks stacking on top of each other, we effectively increase the decoder depth and allow upper blocks to implicitly send feedbacks to lower blocks.
- This simple design closes the performance gap between local learning and end-to-end contrastive learning algorithms for the first time.
- Aside from standard ImageNet experiments, we also show results on complex downstream tasks such as object detection and instance segmentation directly using readout features

# LoCo Architecture

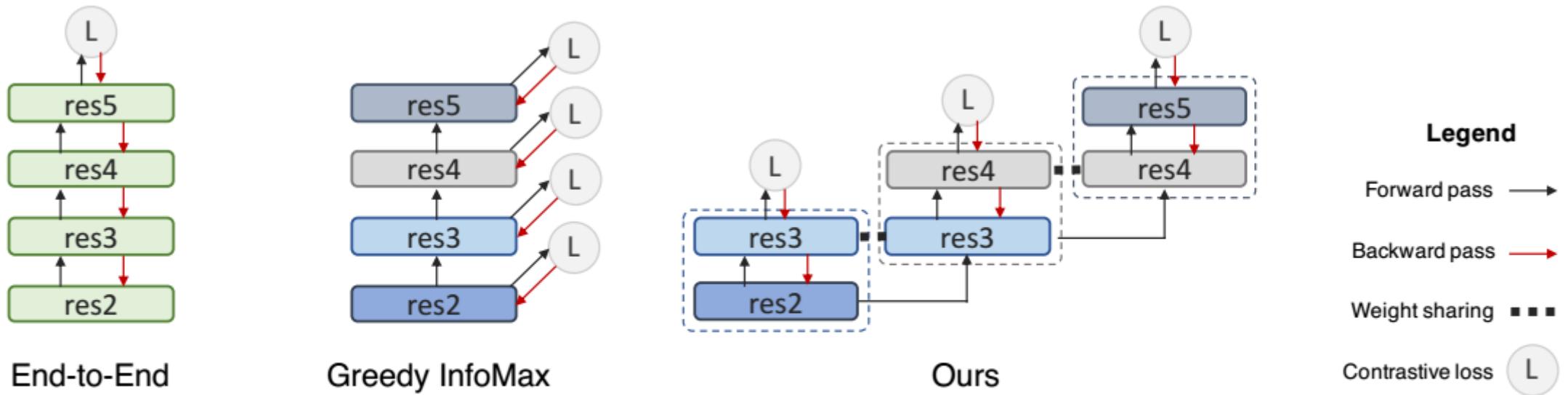


Figure 1: Comparison between End-to-End, Greedy InfoMax (GIM) and LoCo

In this paper, we used SimCLR as our main baseline, since it has superior performance on ImageNet, and we apply the changes proposed in GIM(Greedy Infomax) on top of SimCLR as our local learning baseline.

In our experiments, we find that simply applying GIM on SimCLR results in a significant loss in performance and in the next section we will explain our techniques to bridge the performance gap.

# Comparison tables

Method	Architecture	Acc.	Local
Local Agg. [62]	ResNet-50	60.2	
MoCo [22]	ResNet-50	60.6	
PIRL [45]	ResNet-50	63.6	
CPC v2 [56]	ResNet-50	63.8	
SimCLR* [11]	ResNet-50	69.3	
SimCLR [11]	ResNet-50	<b>69.8</b>	
GIM [39]	ResNet-50	64.7	✓
LoCo (Ours)	ResNet-50	69.5	✓
SimCLR [11]	ShuffleNet v2-50	69.1	
GIM [39]	ShuffleNet v2-50	63.5	✓
LoCo (Ours)	ShuffleNet v2-50	<b>69.3</b>	✓

Table 1: ImageNet accuracies of linear classifiers trained on representations learned with different unsupervised methods, SimCLR\* is the result from the SimCLR paper with 1000 training epochs.

## Greedy InfoMax (GIM)

SimCLR: T. Chen... A simple framework for contrastive learning of visual representations, 2020.

Method	Arch	COCO		Cityscapes	
		AP <sup>bb</sup>	AP	AP <sup>bb</sup>	AP
Supervised	R-50	33.9	31.3	33.2	27.1
Backbone weights with 100 Epochs					
SimCLR	R-50	32.2	29.9	33.2	28.6
GIM	R-50	27.7 (-4.5)	25.7 (-4.2)	30.0 (-3.2)	24.6 (-4.0)
Ours	R-50	32.6 (+0.4)	30.1 (+0.2)	33.2 (+0.0)	28.4 (-0.2)
SimCLR	Sh-50	32.5	30.1	33.3	28.0
GIM	Sh-50	27.3 (-5.2)	25.4 (-4.7)	29.1 (-4.2)	23.9 (-4.1)
Ours	Sh-50	31.8 (-0.7)	29.4 (-0.7)	33.1 (-0.2)	27.7 (-0.3)
Backbone weights with 800 Epochs					
SimCLR	R-50	34.8	32.2	34.8	30.1
GIM	R-50	29.3 (-5.5)	27.0 (-5.2)	30.7 (-4.1)	26.0 (-4.1)
Ours	R-50	34.5 (-0.3)	32.0 (-0.2)	34.2 (-0.6)	29.5 (-0.6)
SimCLR	Sh-50	33.4	30.9	33.9	28.7
GIM	Sh-50	28.9 (-4.5)	26.9 (-4.0)	29.6 (-4.3)	23.9 (-4.8)
Ours	Sh-50	33.6 (+0.2)	31.2 (+0.3)	33.0 (-0.9)	28.1 (-0.6)

Table 2: Mask R-CNN results on COCO and Cityscapes. Backbone networks are frozen. “R-50” denotes ResNet-50 and “Sh-50” denotes ShuffleNet v2-50.

Average Precision (AP)

Average Precision bounding box (AP<sup>bb</sup>)

# (7) Progressive Stage-wise Learning for Unsupervised Feature Representation Enhancement

Zefan Li et al., Shanghai Jiao Tong University

In this work, we explore new dimensions of unsupervised learning by proposing the **Progressive Stage-wise Learning (PSL)** framework.

For a given unsupervised task, we design multilevel tasks and define different learning stages for the deep *network*. (*Each new task has overlap with the former one*)

Early learning stages are forced to focus on low-level tasks while late stages are guided to extract deeper information through harder tasks.

We discover that by progressive stage-wise learning, unsupervised feature representation can be effectively enhanced.

Our extensive experiments show that PSL consistently improves results for the leading unsupervised learning methods.

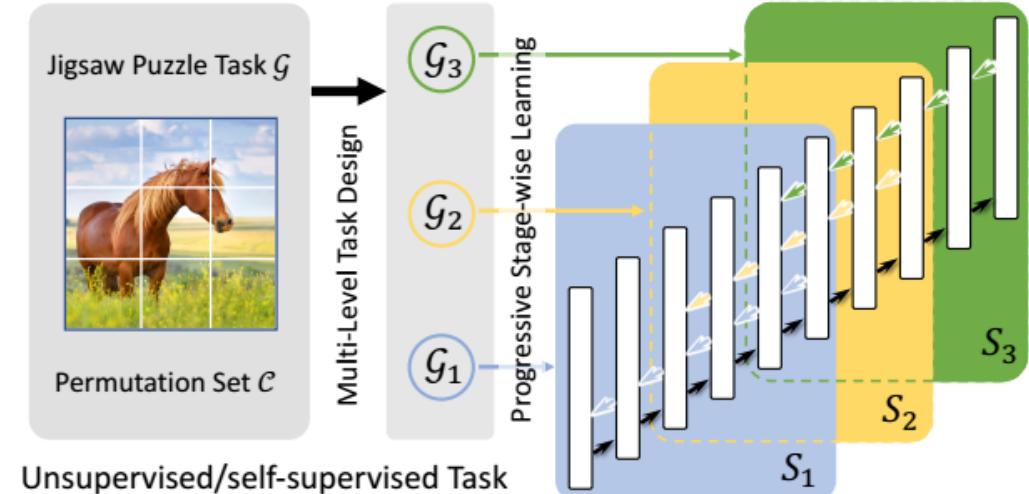


Figure 1. We present the framework of the proposed Progressive Stage-wise Learning (**PSL**) algorithm, aiming for improving unsupervised/self-supervised task. We take the jigsaw puzzle task  $\mathcal{G}$  for example. We first do multi-level task partition  $\mathcal{G} \rightarrow \{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$  with an increased task complexity and perform progressive stage-wise training for different learning stages of the network. The black arrow denotes forward pass while colored arrow represents the backward pass of each learning stage (i.e.,  $S_1$ ,  $S_2$ , and  $S_3$ ).

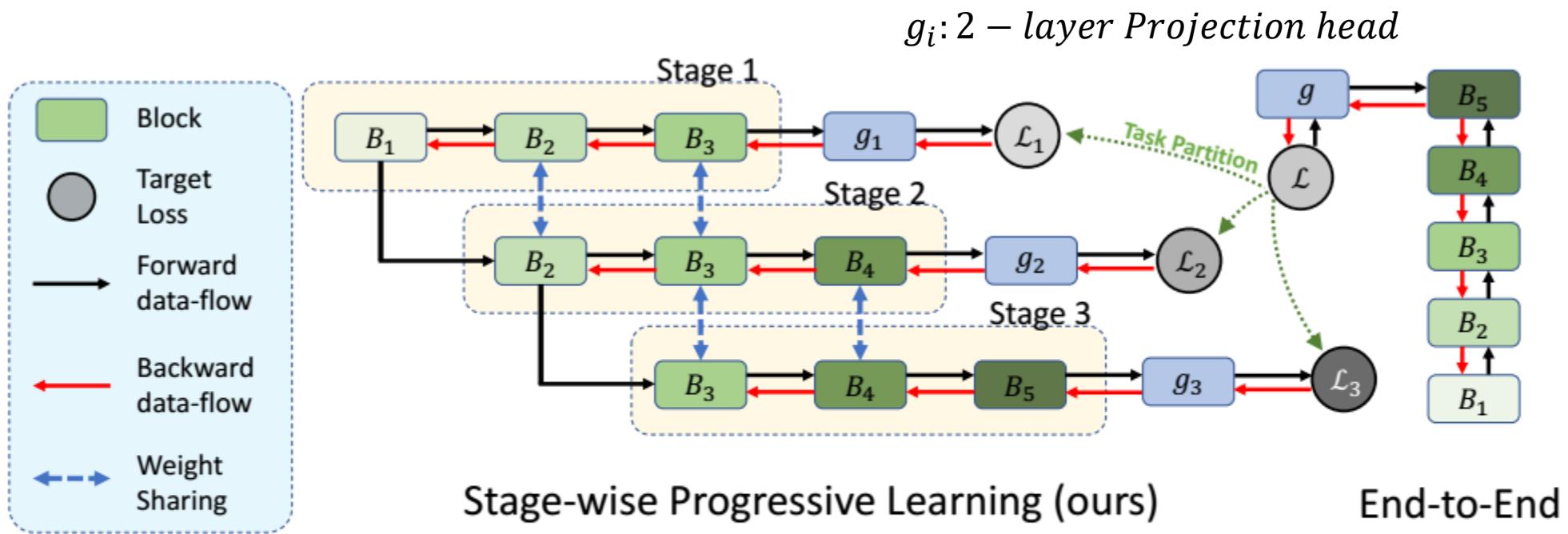


Figure 2. We present the detail of the proposed Stage-wise Progressive Learning framework. In the right is the end-to-end learning scheme while we present PSL in the middle.  $g$  and  $\{g_i\}_{i=1}^3$  are projection heads, mapping the intermediate representation to the target feature space. After the training is completed, we throw away the projection heads and use the backbone network for downstream tasks.

# Comparison Tables

Task	1% labels	10% labels
Supervised	48.4	80.4
Jigsaw [38]	45.4	79.6
Jigsaw+PSL	<b>48.7</b>	<b>83.5</b>
Rotation [21]	52.1	82.5
Rotation+PSL	<b>54.8</b>	<b>83.7</b>

**Table 7. Semi-supervised learning on ImageNet.** We use ResNet-50 as our backbone networks and report single-crop top-5 accuracy on the ImageNet validation set. All models are self-supervised trained on ImageNet and finetuned on 1% and 10% of the ImageNet training data, following [46, 37]. The supervised results are presented for reference.

Method	Arch	# Param(M)	Top 1
Colorization [49]	R50	24	39.6
BigBiGAN [15]	R50	24	56.6
LA [51]	R50	24	58.8
NPID++ [37]	R50	24	59.0
MoCo [26]	R50	24	60.6
PIRL [37]	R50	24	63.6
CPC v2 [28]	R50	24	63.8
PCL [34]	R50	24	65.9
SwAV [9]	R50	24	75.3
Jigsaw [38]	R50	24	45.7
<b>Jigsaw + PSL</b>	R50	24	<b>50.9</b>
Rotation [21]	Rv50w4×	86	47.3
Rotation*	Rv50	24	48.6
<b>Rotation+PSL</b>	Rv50	24	<b>53.3</b>
SimCLR [10]	R50	24	61.9
<b>SimCLR+PSL</b>	R50	24	<b>64.3</b>
MoCov2 [11]	R50	24	67.5
<b>MoCov2+PSL</b>	R50	24	<b>68.1</b>

**Table 4. ImageNet accuracy of linear classifiers trained on self-supervised learned representations.** All are reported as unsupervised pre-training on ImageNet, followed by supervised linear classification and evaluated on the ImageNet validation set. Note that Rotation [21] uses  $\mathcal{R}_2$  as the transformation set while Rotation\* uses  $\mathcal{R}_3$  as the transformation set. SimCLR results are obtained by 200 training epochs with batchsize 256.

Thank you