Part3

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

3.1 Two Data Complexity measures

- Separation index (SI)
- Smoothness index(SmI)

3.2 Layer-wise Analysis by Separation and Smoothness indices

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

3.3 Layer-wise Design by Separation and Smoothness indices

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

3.4 Related works in local Layer-wise learning

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

3.1 Two Data Complexity measures

3.1.1 Separation index

- First order SI
- High order SI
- High order soft SI
- Center Based SI

3.1.2 Smoothness index

- First order SmI
- High order SmI
- High order soft SmI

Data Complexity measures

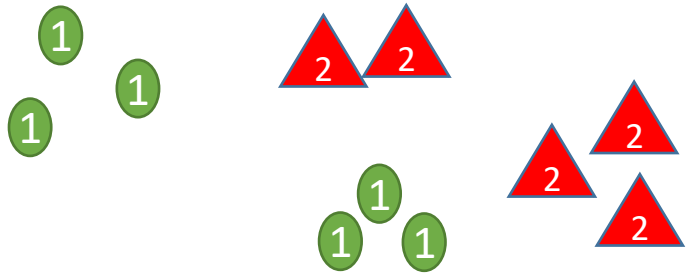
Complexity measures	Overall evaluating approach
√ Feature-based	Discovering informative features by evaluating each feature independently (Orriols-Puig et al., 2010; Cummins, 2013))
√ Linearity separation	Evaluating the linearly separation of different classes (Bottou & Lin, 2007)
√ Neighborhood	Evaluating 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

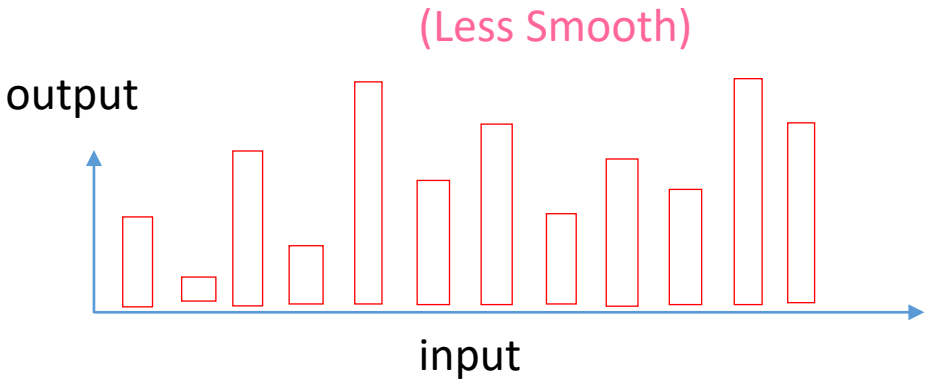
1. A separation measure (in classification problems)

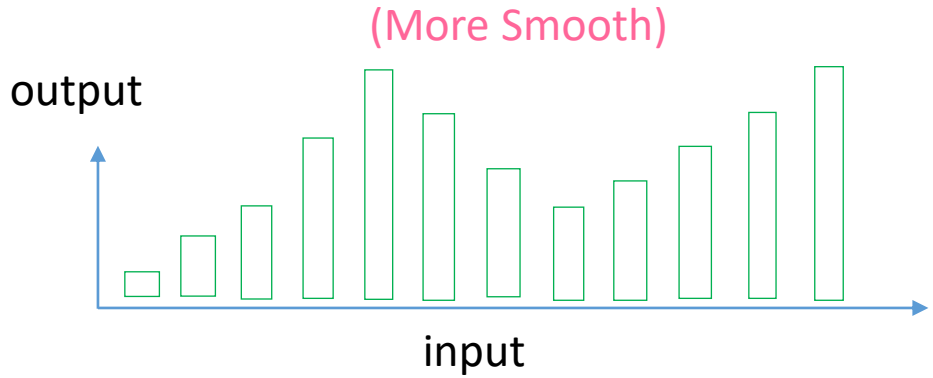
   It shows that how much input data points separate the labels from each others.



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

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

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

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

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

$$i^* = \underset{\forall q \neq i}{\arg\min} \left\| \boldsymbol{x}_i - \boldsymbol{x}_q \right\| \qquad \delta(l_i, l_{i^*}) = \begin{cases} 1 & if\ l_i = l_{i^*} \\ 0 & else \end{cases}$$

kronecker delta

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

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

** 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: $\mathrm{SmI} \in [0,1]$
2. $SI \to 1$ $(Sepration\ is\ maximmum)$ and $SI \to 0 (Sepration\ is\ minimmum)$
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.

$$\forall \beta \neq 0, \forall \alpha \neq 0, \forall x_0, \forall l_0 \qquad SI(\{(\boldsymbol{x}^i, l^i)\}_{i=1}^m) = SI(\{(\beta x_i + \boldsymbol{x}_0, \alpha l_i + l_0)\}_{i=1}^m)$$

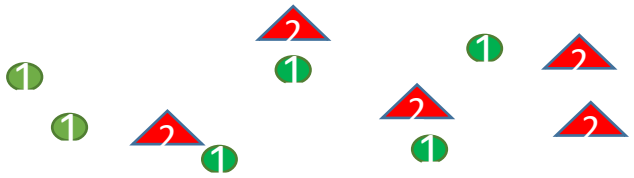
6. Separatin index of $the\ target\ labels\ with\ themselves\ is\ maximum$: $SI(\{(l_i, l_i)\}_{i=1}^m)$=1;

it means that how input data become more similar to labels the separation index will increase.

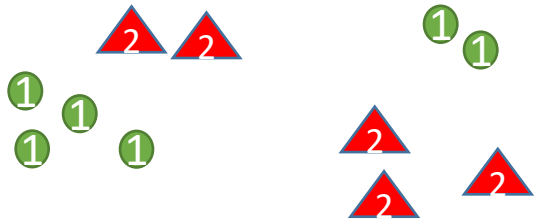
Two dimensional examples (binary classification)

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)

$$Data = \{(\boldsymbol{x}_i, l_i)\}_{i=1}^{m} \qquad \boldsymbol{x}^i \epsilon R^{n\times 1}$$

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

Steps

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

2- $\mathrm{M} = \mathrm{X}\mathrm{X}^{\mathrm{T}}, \quad \mathrm{M}\epsilon R^{m\times m}$

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

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

5- Distance matrix is computed as follows:

$$\mathrm{D} = \mathrm{W} + \mathrm{W}^{\mathrm{T}} - 2\mathrm{M}$$

Separation index for Each Class

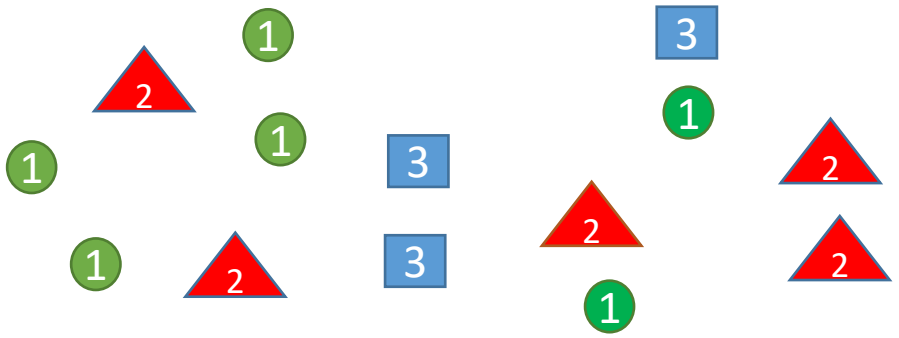
$\mathrm{SI}_c(Data) = \frac{1}{m_c}\sum_i \delta(l_i, c)\delta(l_i, l_{i^*})$ $c=1,2,..,n_C$

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

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

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

A two dimensional illustrative example



$n_C = 3, \qquad c = 1,2,3$

$SI_1(Data) = 4/6$
$SI_2(Data) = 2/5$
$SI_3(Data) = 2/3$
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 $SI \to 1/n_C$

2. High order SI

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

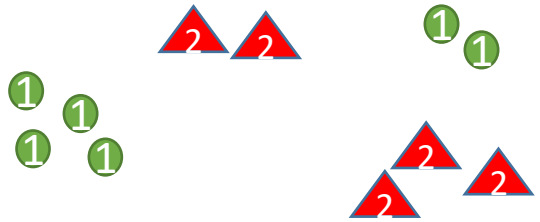
$$\mathrm{SI}^{\mathrm{r}}(Data) = \frac{1}{m}\sum_{i=1}^{m}\prod_{j=1}^{r}\delta\left(l_i, l_{i_j^*}\right) \qquad \text{r: the order of "SI"}$$

$$i_j^* = \underset{\forall q \neq i, i_1^*, \cdots, i_{j-1}^*}{\arg} min\left\|\boldsymbol{x}_i - \boldsymbol{x}_q\right\| \qquad \mathrm{SI}^{\mathrm{r}} \in [0,1]$$

- "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: $\mathrm{SI}^{\mathrm{r}} \leq \mathrm{SI}^{\mathrm{r}-1} \leq \cdots \leq \mathrm{SI}^{1}$ $\qquad \mathrm{SI}^{1} = \mathrm{SI}$

Two illustrative Examples

SI^1 =11/11
SI^2 =11/11
SI^3 =11/11
SI^4 =11/11



SI^1 =11/11
SI^2 =7/11
SI^3 =4/11
SI^4 =0

- 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.

3. High order soft SI

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

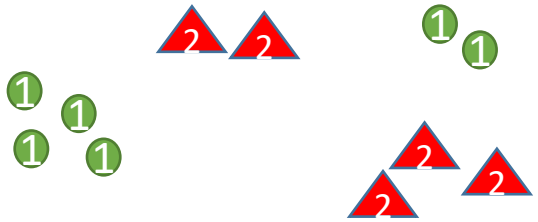
$$\mathrm{SI}^{\mathrm{r}}_{\mathrm{soft}}(Data) = \frac{1}{m\times r}\sum_{i=1}^{m}\sum_{j=1}^{r}\delta(l_i,,l_{i_j^*})$$

r: the order of SI $\quad i_j^* = \underset{\forall q\neq i, i_1^*,\cdots,i_{j-1}^*}{\arg\min} \left\|\boldsymbol{x}_i - \boldsymbol{x}_q\right\|$ $\quad \mathrm{SI}^{\mathrm{r}}_{\mathrm{soft}} \in [0,1]$

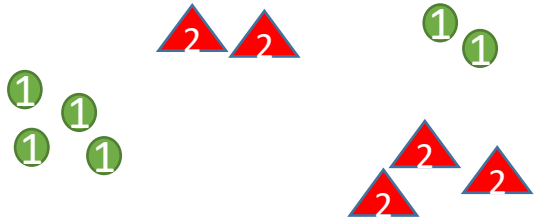
- $\mathrm{SI}^{\mathrm{r}}_{\mathrm{soft}}$ considers less restricted condition of separation than SI^{r}

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

Two illustrative Examples



SI^1 =11/11
SI^2 =7/11
SI^3 =4/11
SI^4 =0



SI^1_{soft} =11/11
SI^2_{soft}=(4+3+0.5+0.5)/11=8/11
SI^3_{soft}=(4+3(2/3)+4*(1/3)/11=8.33/11
SI^4_{soft}=(4*(3/4)+2*(1/4)+2*(1/4)+3*(2/4))/11
=6.5/11

4. Center based Separation Index (CSI)

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

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

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

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

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

- 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.

Smoothens index (SmI)

SmI 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

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

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

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

$$\mathrm{SmI}(Data) = \frac{1}{m}\sum_{i=1}^{m} \frac{\|\boldsymbol{y}_{imax} - \boldsymbol{y}_{i^*}\|}{\|\boldsymbol{y}_{imax} - \boldsymbol{y}_{imin}\|}$$

$$i^* = \underset{\forall q \neq i}{\arg\, min} \|\boldsymbol{x}_i - \boldsymbol{x}_q\| \qquad imax = \underset{\forall q}{\arg\, max} \|\boldsymbol{y}_i - \boldsymbol{y}_q\| \qquad imin = \underset{\forall q \neq i}{\arg\, min} \|\boldsymbol{y}_i - \boldsymbol{y}_q\|$$

*$\|\cdot\|$ 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 SmI.

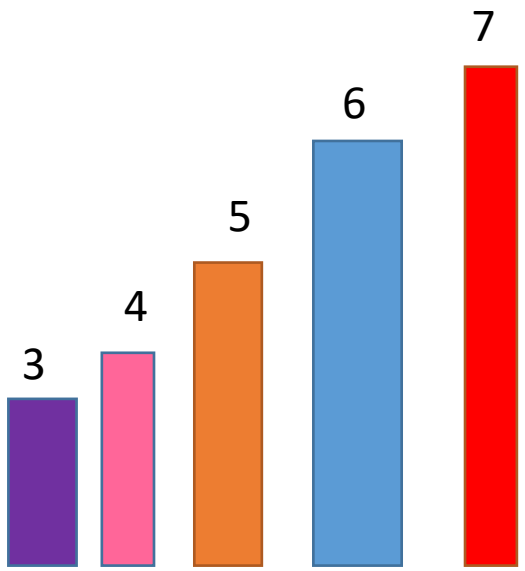
Some notes

1. "SmI" is a normalized index between zero and one: $\mathrm{SmI} \in [0,1]$
2. $SmI \to 1$ $(Smoothness\ is\ maximmum)$ and $SmI \to 0$ $(Smoothness\ is\ minimmum)$
3. "SmI" 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: $\mathrm{SmI} = \mathrm{SI}$
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

   $\forall \beta \neq 0, \forall \alpha \neq 0, \forall x_0, \forall y_0 \qquad \mathrm{SmI}(\{(x^i, y^i)\}_{i=1}^m) = SmI(\{(\beta \boldsymbol{x}_i + x_0, \alpha \boldsymbol{y}_i + y_0)\}_{i=1}^m)$

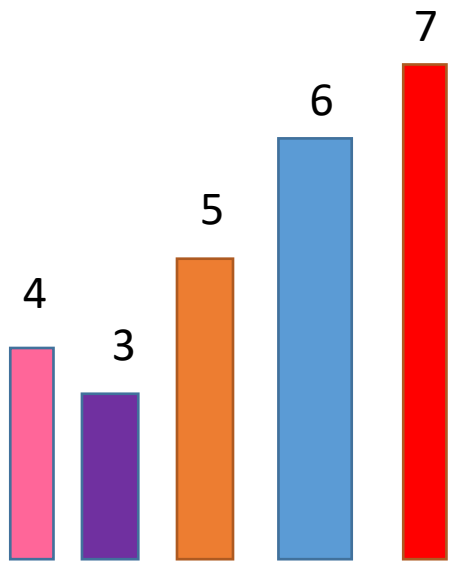
7. Smoothness index of target outputs $with\ themselves\ is$ $maximum$: $SmI(\{(\boldsymbol{y}_i, \boldsymbol{y}_i)\}_{i=1}^m)$=1; it means that how input data become more similar to output the smoothness index will increase.

One-dimensional illustrative examples



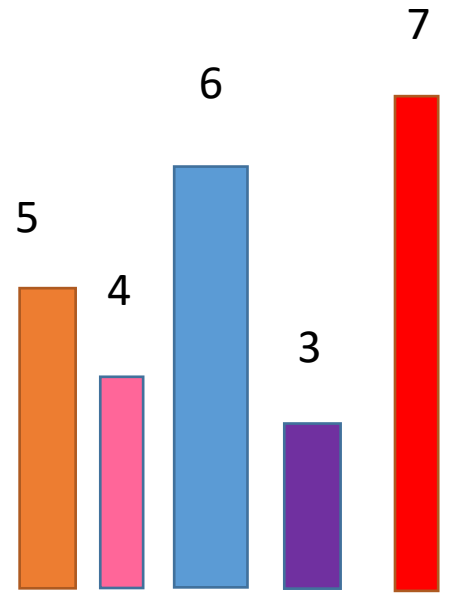
$$SmI = \frac{1}{5}\left(\frac{7-4}{7-4}+\frac{7-3}{7-3}+\frac{7-4}{7-4}+\frac{5-3}{5-3}+\frac{6-3}{6-3}\right)$$

$SmI = 1$



$$SmI = \frac{1}{5}\left(\frac{7-4}{7-4}+\frac{7-3}{7-3}+\frac{7-4}{7-3}+\frac{5-3}{5-3}+\frac{6-3}{6-3}\right)$$

SmI=0.95



$$SmI = \frac{1}{5}\left(\frac{7-4}{7-4}+\frac{7-5}{7-3}+\frac{4-3}{5-3}+\frac{7-6}{7-4}+\frac{3-3}{6-3}\right)$$

SmI=.466

2. High order SmI

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

$$\mathrm{SmI}^r(Data) = \frac{1}{m}\sum_{i=1}^{m}\prod_{j=1}^{r}\frac{\|\boldsymbol{y}_{imax}-\boldsymbol{y}_{i_j^*}\|}{\|\boldsymbol{y}_{imax}-\boldsymbol{y}_{imin_j}\|}$$

r: the order of "SmI"

$$i_j^* = \underset{\forall q\neq i, i_1^*,\cdots,i_{j-1}^*}{\arg} min\|\boldsymbol{x}_i - \boldsymbol{x}_q\| \qquad imin_j = \underset{\forall q\neq i, imin_1,\cdots,imin_{j-1}}{\arg} min\|\boldsymbol{y}_i - \boldsymbol{y}_q\|$$

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

3. High order soft SmI

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

$$\mathrm{SmI}_{\mathrm{soft}}^{\mathrm{r}}(Data) = \frac{1}{m\times r}\sum_{i=1}^{m}\sum_{j=1}^{r}\frac{\|\boldsymbol{y}_{imax}-\boldsymbol{y}_{i_j^*}\|}{\|\boldsymbol{y}_{imax}-\boldsymbol{y}_{imin_j}\|} \quad \mathrm{j} = 1,2,\ldots,\mathrm{r}$$

r: the order of "SmI"

$$i_j^* = \underset{\forall q\neq i, i_1^*,\cdots,i_{j-1}^*}{\arg} min\|\boldsymbol{x}_i - \boldsymbol{x}_q\| \qquad imin_j = \underset{\forall q\neq i, imin_1,\cdots,imin_{j-1}}{\arg} min\|\boldsymbol{y}_i - \boldsymbol{y}_q\|$$

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

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

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

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

Algortihm1: (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.

Dataset	Number of Classes	Separation Index	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)

$$SI_n = SI - 1/n_C$$

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

- Actually, the SI(SmI) 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 (SmI) 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 (SmI) changes more smoothly versus number of data points (low sensitivity) .

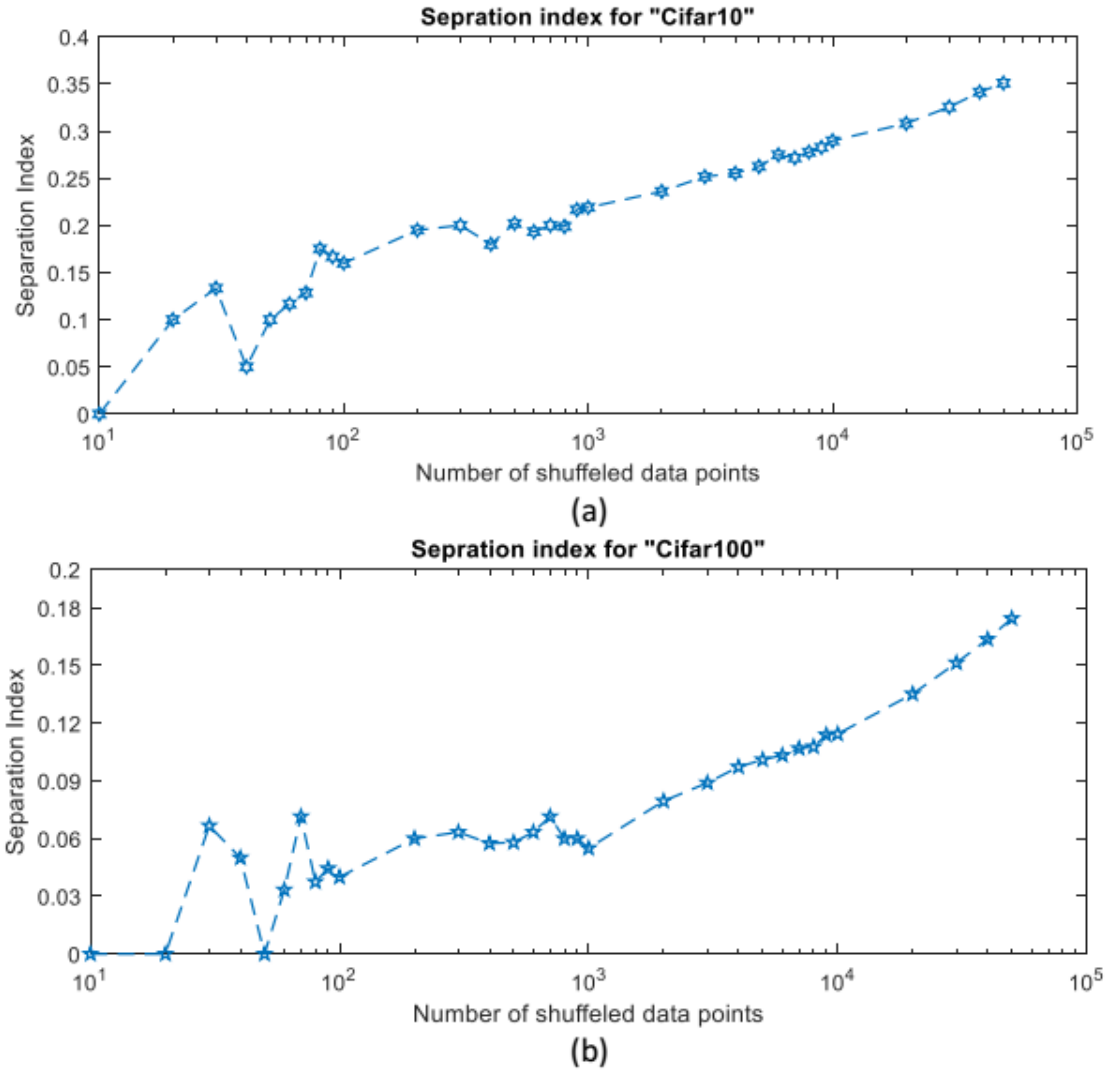


Fig. 6. The plot of separation index versus different number of shuffled data points in both "cifar10" (a) and "cifar100" (b).

Dataset ranking

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

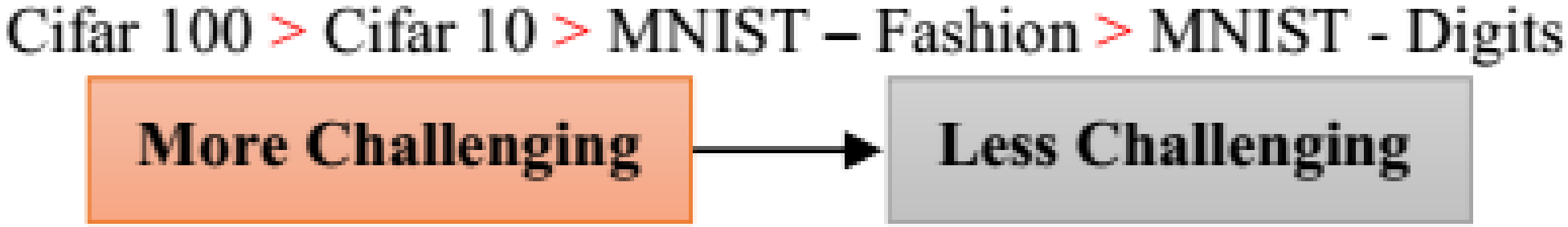


Fig. 4. The ordered classification data sets from more challenging to less challenging.

Cross domain dataset evaluation

1. Classification Problems

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

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

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

- if $SI_{cross}(D_{test}, Data) \gg SI(Data)$, then it is expected that *the training model* (with "$Data$") will have high *generlization 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 *generlization 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

$Data = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{m}$ $D_{test} = \{(\breve{\boldsymbol{x}}_i, \breve{\boldsymbol{y}}_i)\}_{i=1}^{m_{test}}$

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

$$i^{\#} = \arg\min_{\forall q} \left\| \breve{\boldsymbol{x}}_i - \boldsymbol{x}_q \right\| \quad imax\# = \arg\max_{\forall q} \left\| \breve{\boldsymbol{y}}_i - \boldsymbol{y}_q \right\| \quad imin\# = \arg\min_{\forall q} \left\| \breve{\boldsymbol{y}}_i - \boldsymbol{y}_q \right\|$$

- if $SmI_{cross}(D_{test}, Data) \gg SmI(Data)$, then it is expected that *the training model* (with "$Data$") will have high $generlization\ for\ D_{test}$.
- if $SmI_{cross}(D_{test}, Data) \ll SmI(Data)$, then it is expected that *the training model* (with "$Data$") will have low $generlization\ for\ D_{test}$.
- 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 $x_{available} = \{x_1, \dots, x_{ne}\}$ with m_e inputs.
- Among available n_e inputs, select a subset $\boldsymbol{x} \subseteq x_{available}$ and define:

$$Data = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{m}$$

- To decrease the complexity, select $\boldsymbol{x}$ in a way that the $SI(Data)$ $(SmI(Data))$ becomes maximum.
- It is aimed that all non-relevant, redundancies and noise inputs, which decrease the SI(SmI) (or do not increase it significantly), should be removed.
- "Forward selection", "backward elimination" or any other "step-wise selection" algorithms can be used for this purpose.

Choosing effective inputs by Smoothness Index

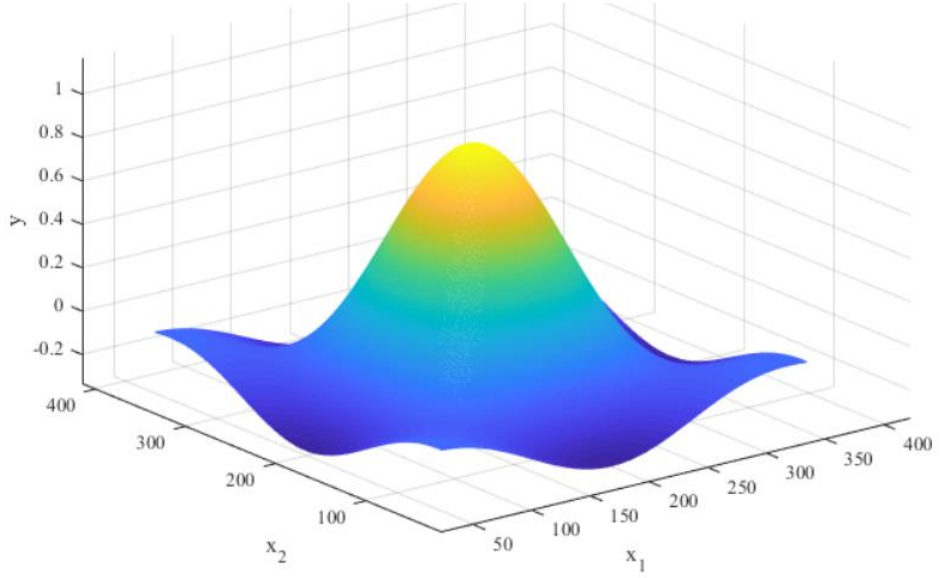
Example1 (illustrative)

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	x_1	x_2	x_3	x_4	Linear	Exponential
1	×	×			0.9783	0.9788
2	×	×	×		0.9159	0.9249
3	×	×	×	×	0.8314	0.8615
4		×	×		0.4781	0.5972
5		×	×	×	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

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

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

	PCA	GUS	RFE	KBS	VT	PCC	MI	FSSmI
MLR	**0.2786**	0.3031	0.2764	0.2726	0.2470	0.2687	0.2655	0.2495
RFR	0.4912	**0.2347**	**0.2306**	0.2714	0.2520	0.3034	0.2901	***0.2301***
SVR	0.2916	0.2600	0.2501	**0.2365**	**0.2301**	**0.2400**	**0.2391**	0.2555
KNN	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
MLR	**0.5232**	0.5499	0.5220	0.5219	0.4969	0.5179	0.5105	0.5081
RFR	0.6965	**0.4816**	**0.4796**	0.5203	0.5014	0.5495	0.5351	**0.4986**
SVR	0.5386	0.5066	0.4975	**0.4857**	***0.4781***	**0.4874**	**0.4868**	0.5162
KNN	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-variant 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)

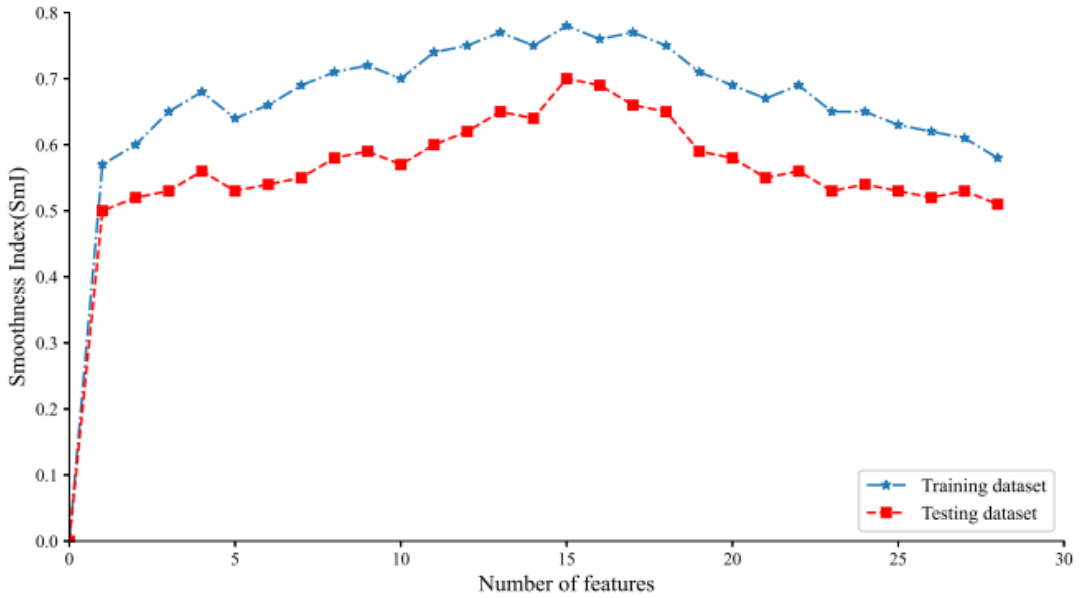
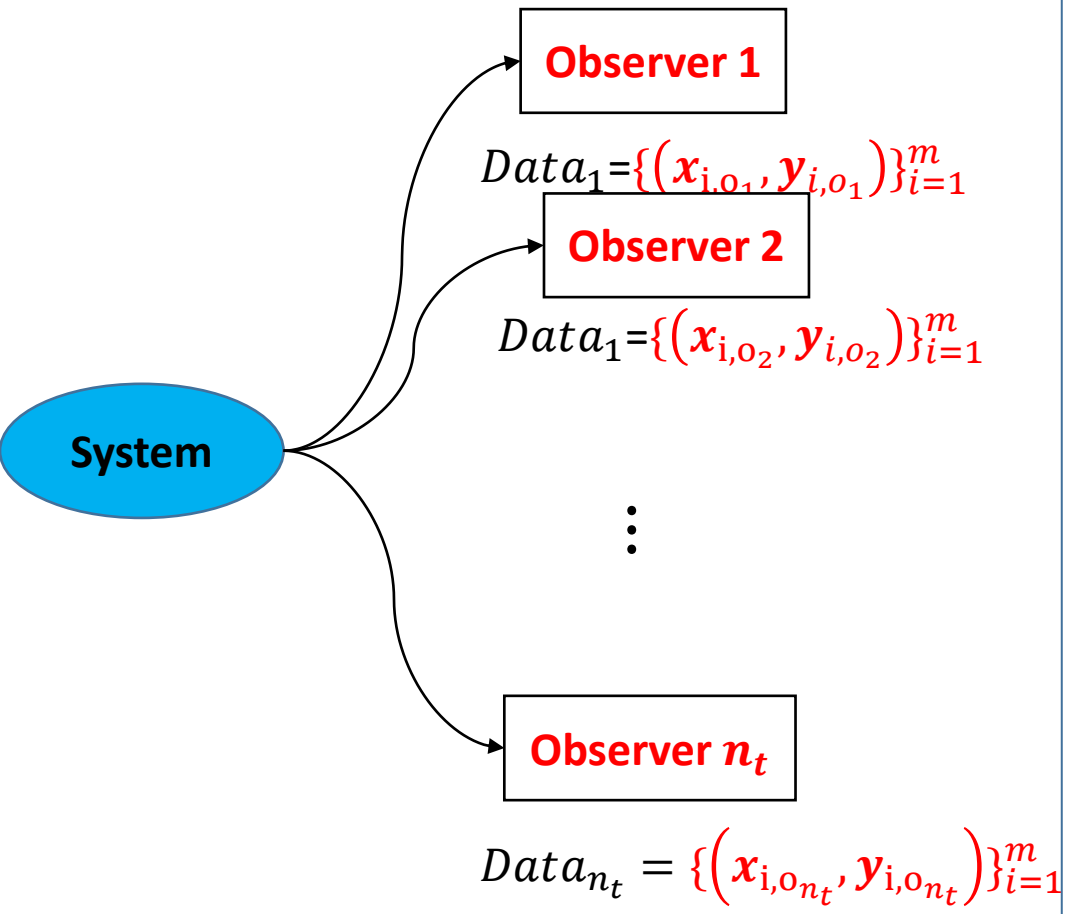


Fig. 5. Smoothness Index (SmI) based on number of features. There is a good correlation between the smoothness charts of training and test datasets, i.e., the selected features based on the absent data are the same as those that give the highest SmI in the training dataset.

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

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

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 (SmI).

concatenation $\boldsymbol{x}_i^* = [\boldsymbol{x}_{i,o_1^*},\ldots,\boldsymbol{x}_{i,o_{n_s}^*}]$, $\boldsymbol{y}_{i,o_1} = \boldsymbol{y}_{i,o_2} \ldots = \boldsymbol{y}_{i,o_{n_t}}$

Appending $\boldsymbol{x}_i^* = \begin{bmatrix} \boldsymbol{x}_{i,o_1^*} \\ \vdots \\ \boldsymbol{x}_{i,o_{n_s}^*} \end{bmatrix}$

For classification problems

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

or for *regression problems*

$$\mathrm{SmI}(\{(\boldsymbol{x}_i^*, \boldsymbol{y}_i)\}_{i=1}^{m}) \geq \mathrm{SmI}(\{(\tilde{\boldsymbol{x}}_i, \boldsymbol{y}_i)\}_{i=1}^{m})$$

where $\tilde{\boldsymbol{x}}$ 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, $\boldsymbol{x}$ ($\boldsymbol{x} \equiv \boldsymbol{x}^1$) 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 :

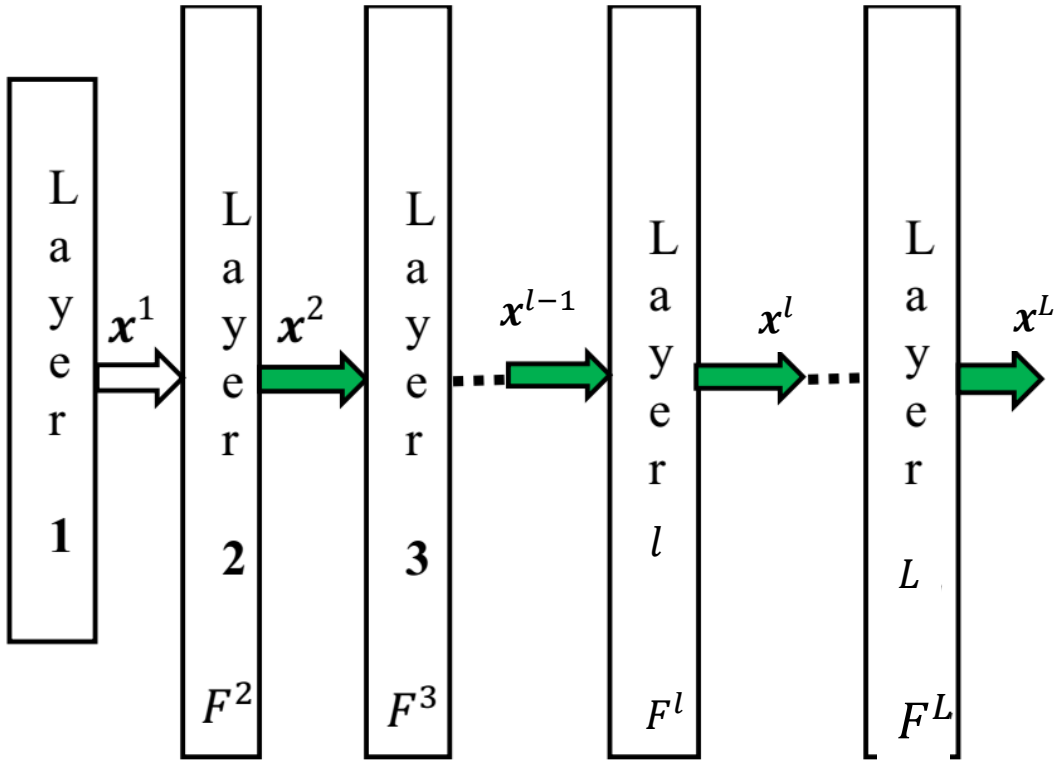
$$\boldsymbol{x}^1 \rightarrow \cdots \boldsymbol{x}^{l-1} \rightarrow \boldsymbol{x}^l \rightarrow \cdots \rightarrow x^L$$

- L: number of layers in the model
- $\boldsymbol{x}^l$ denotes the dataflow at layer l, which is reshaped as a vector and its length is n_L.

$$\boldsymbol{x}^l \boldsymbol{\epsilon} \boldsymbol{R}^{\boldsymbol{n_l} \times \boldsymbol{1}}$$

- One can compute SI(SmI) for dataflow at layer $\boldsymbol{l}$:

$Data^l = \{(\boldsymbol{x}_i^l, l_i)\}_{i=1}^m$ $\quad\quad$ $\boldsymbol{l}$=1,2, … , 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 (SmI) 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. Geometrically, it is proved that by intensifying exclusive features and weakening disturbances, distortions, and common features (complexity), the SI(SmI) of dataflow will increase gradually through layers of a DNN.

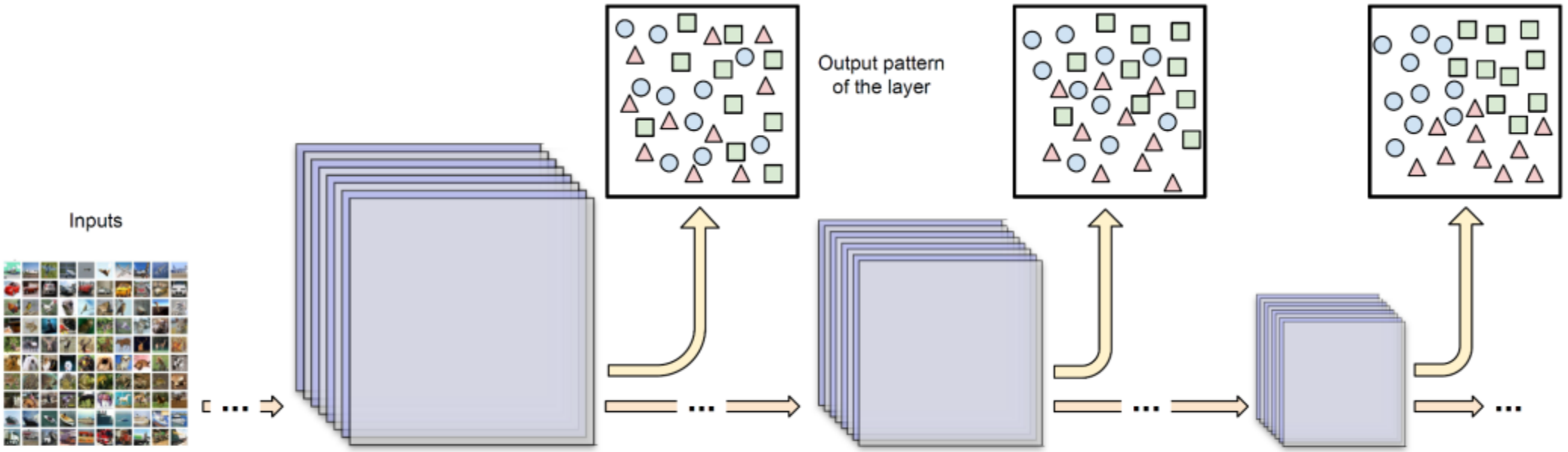
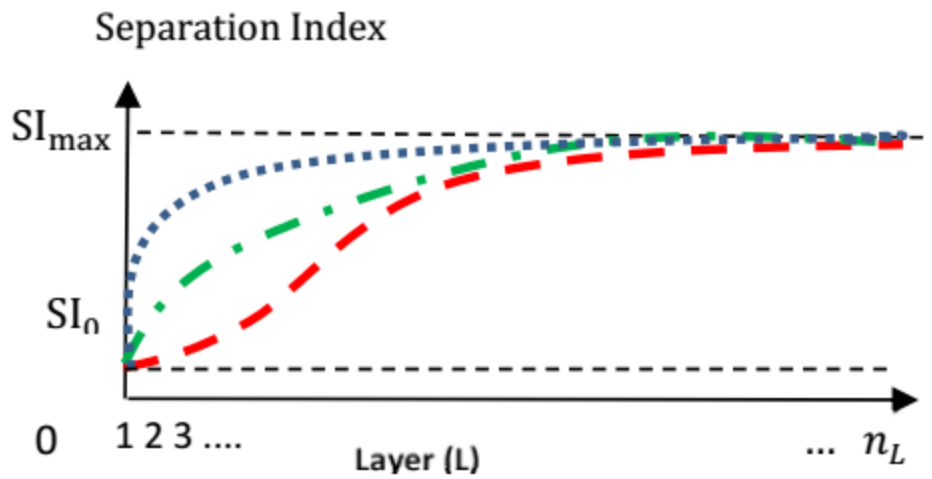


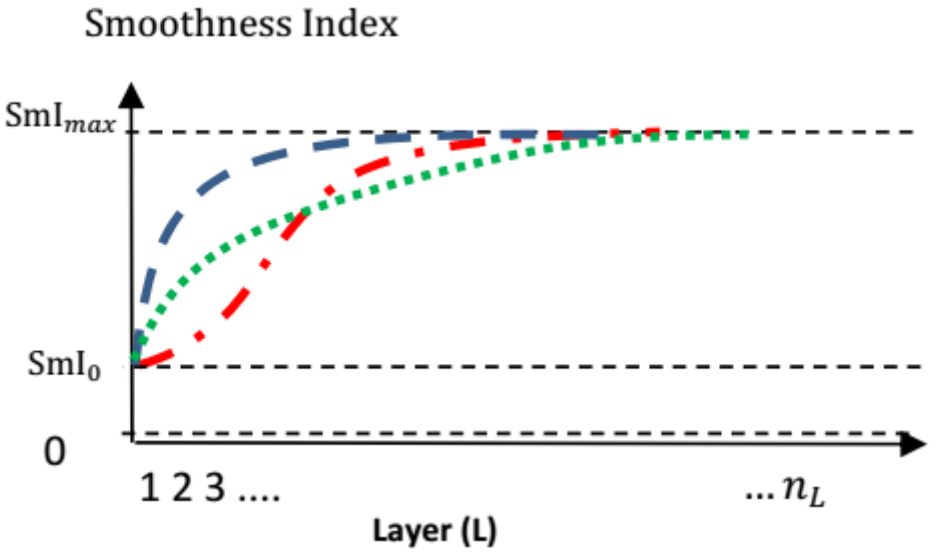
Figure 1. Output pattern of each layer through the network.

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

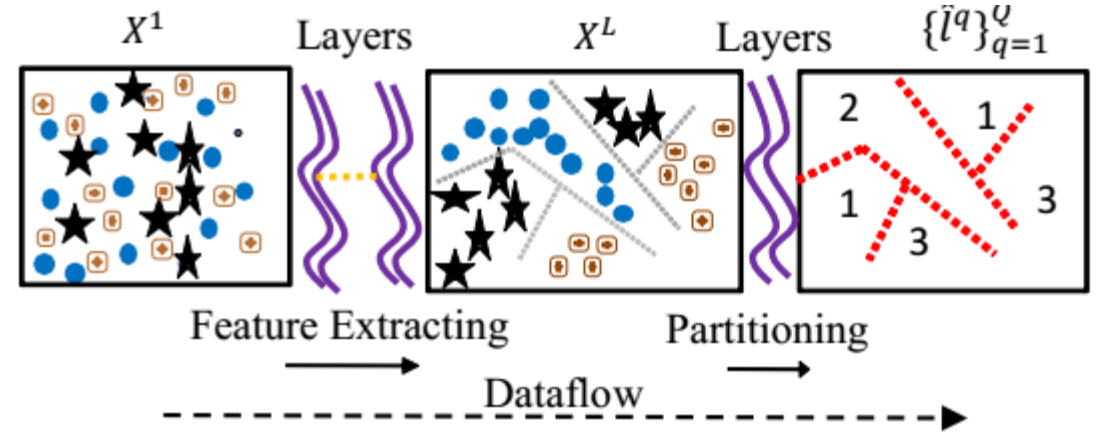
Some important notes

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.

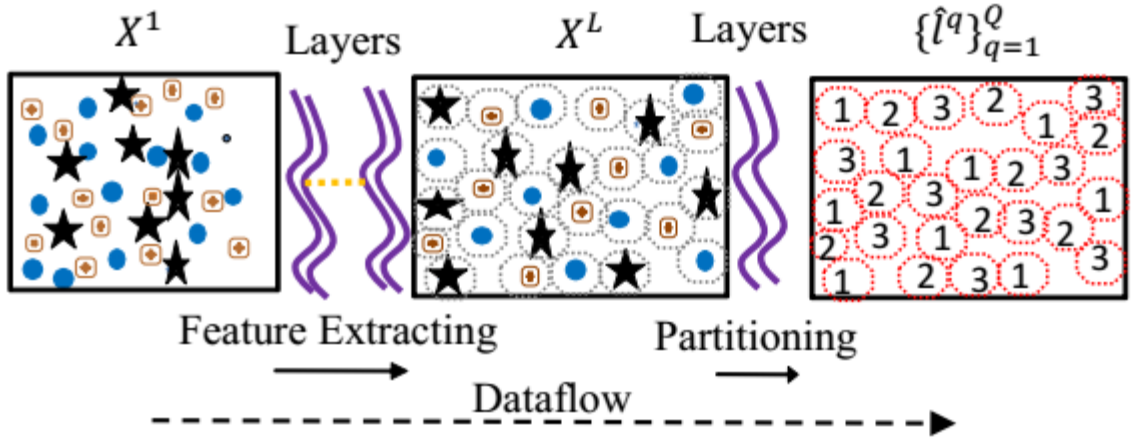




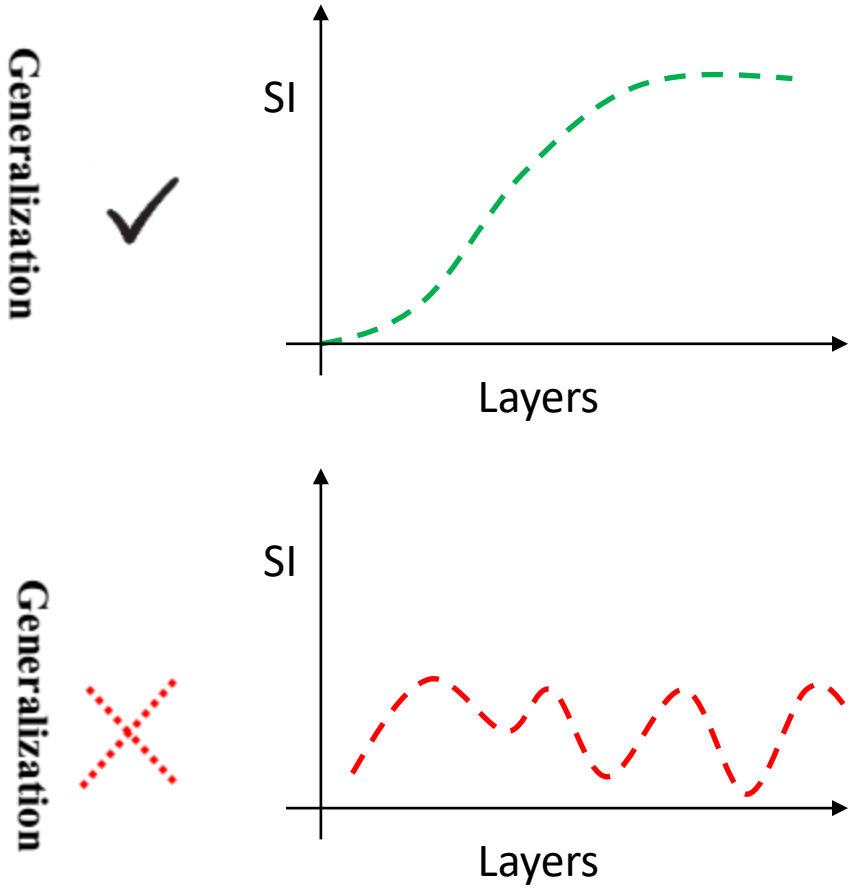
Generalization and Separation index



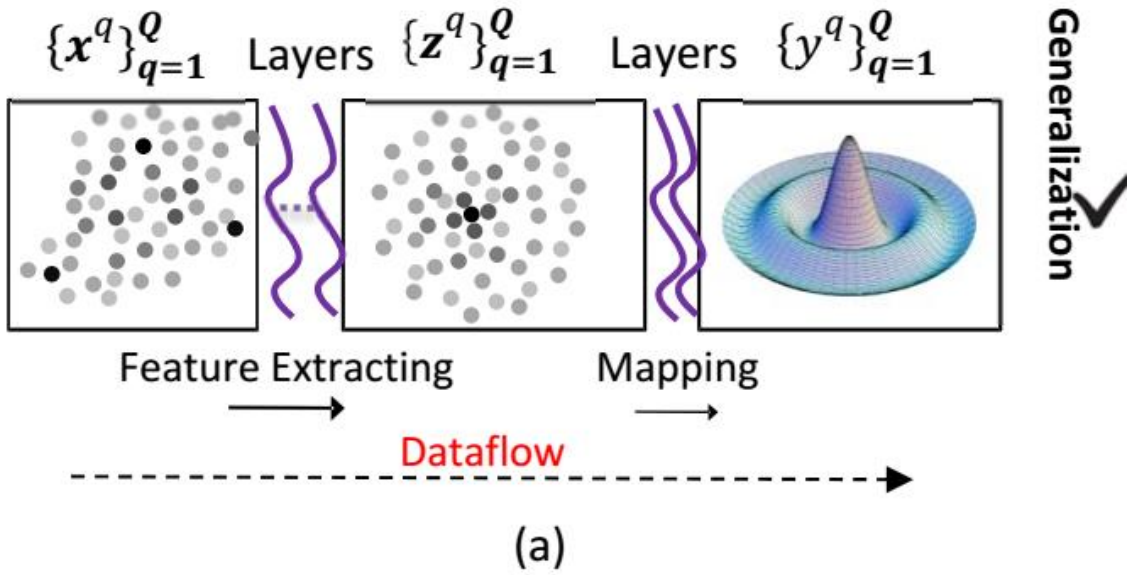
(a)



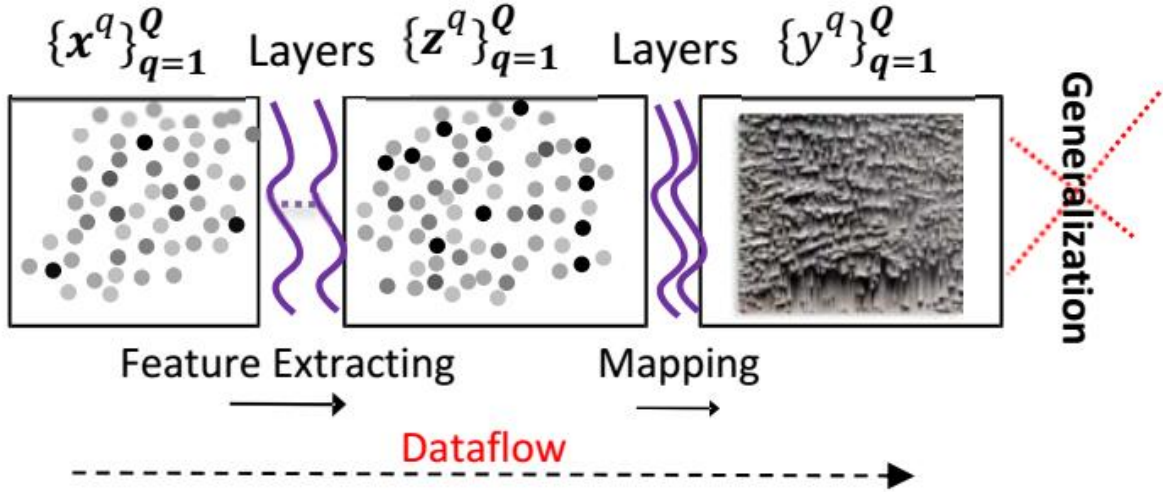
(b)



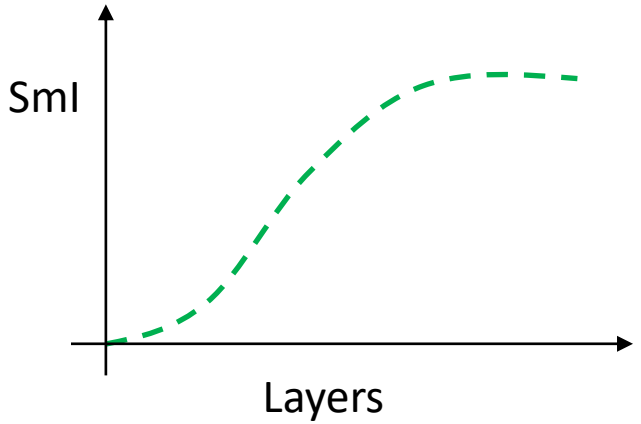
Generalization and Smoothness index

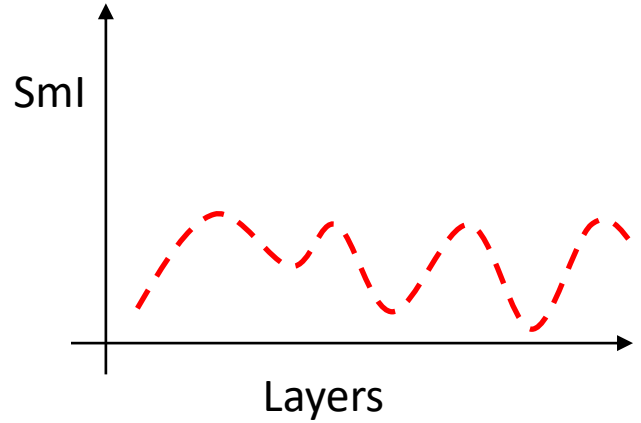


(a)



(b)





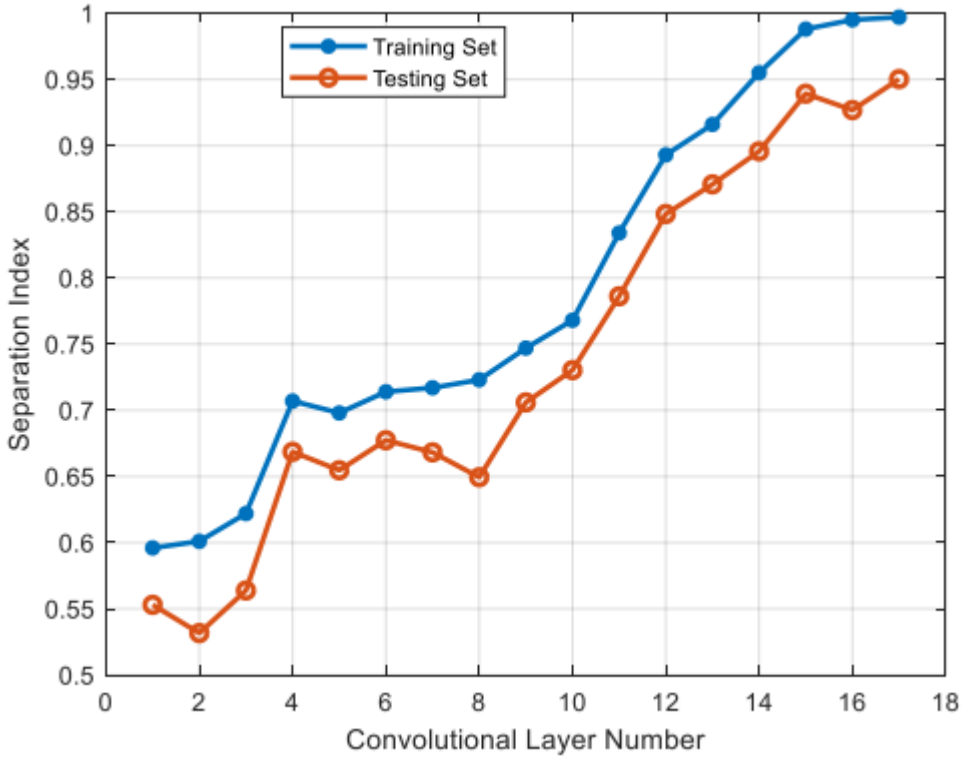
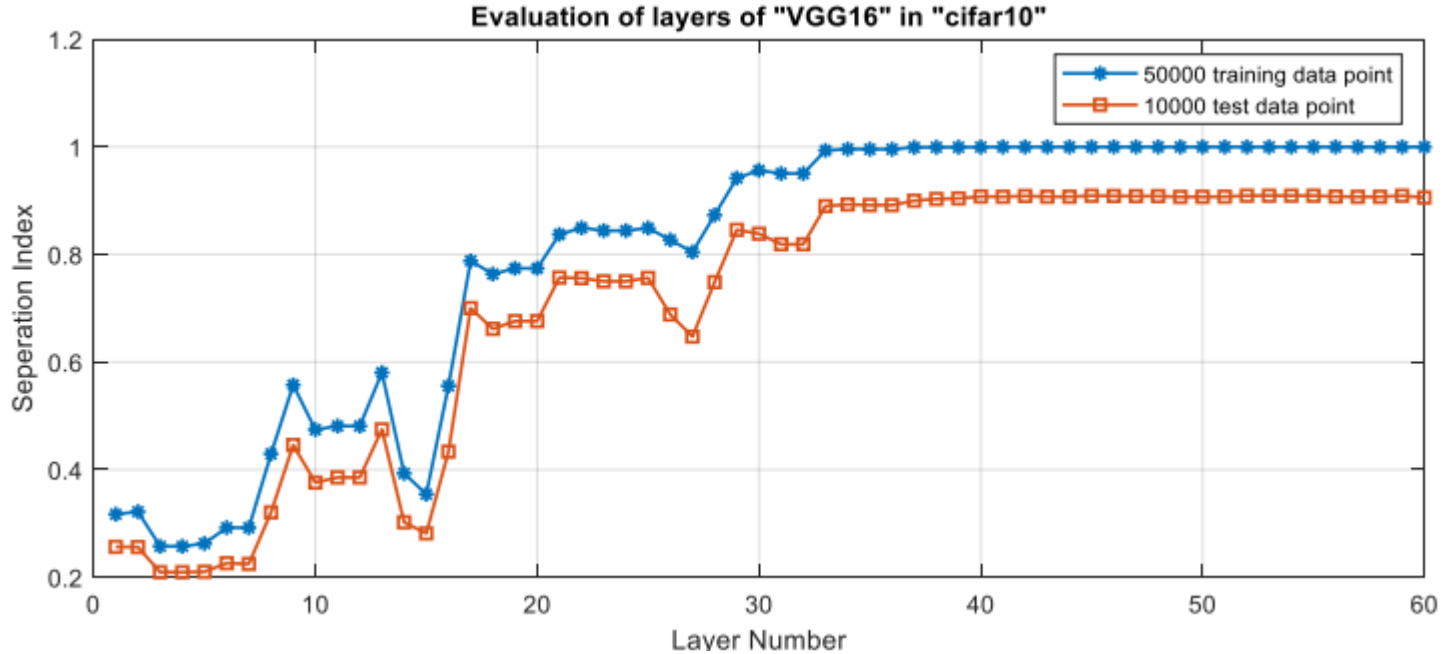
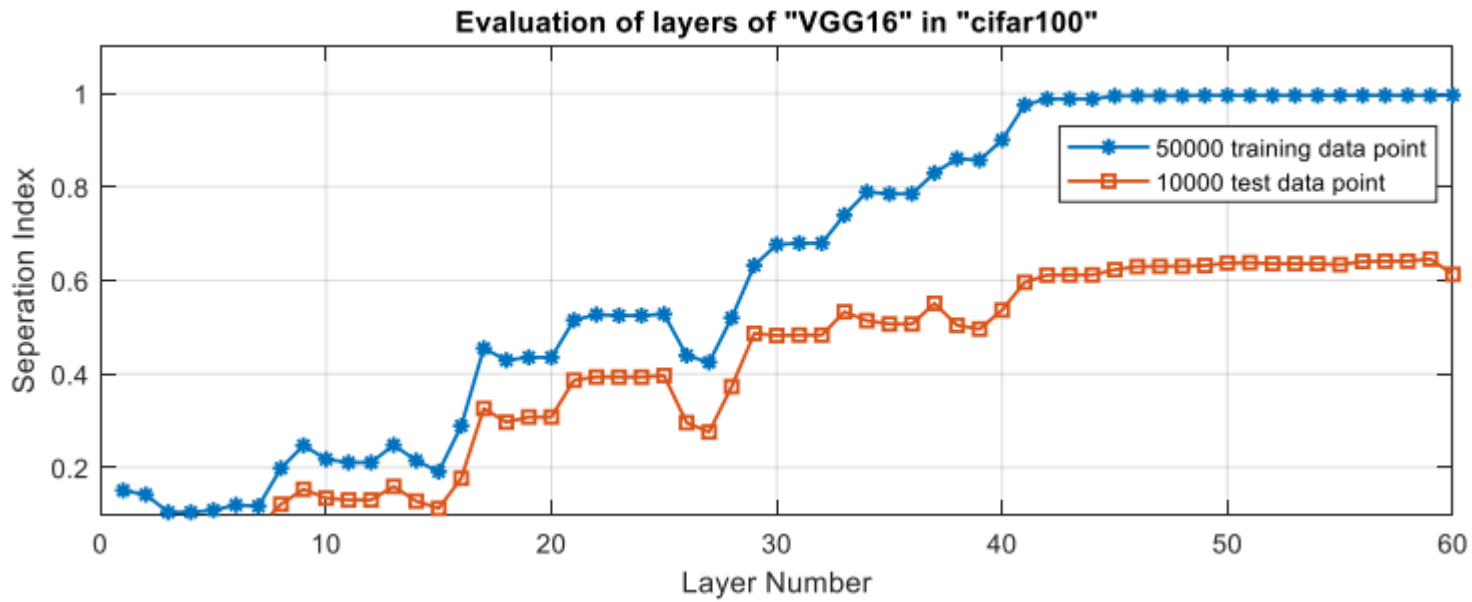


Fig. 10. Evaluation of dataflow through layers of the "Resnet18" network in the classification of Fashion-MNIST.

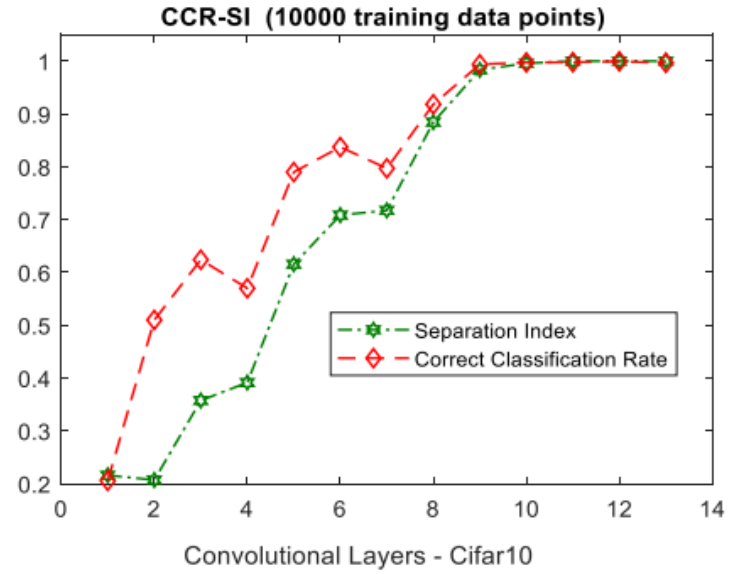




Correct Classification rate and SI

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.



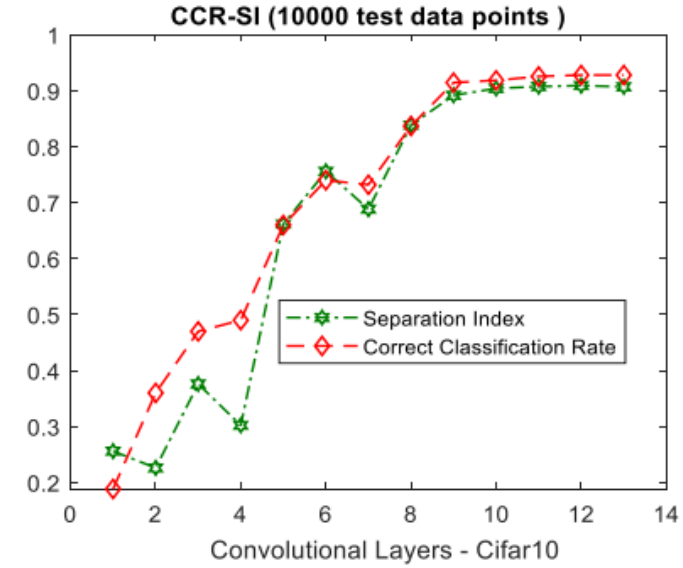
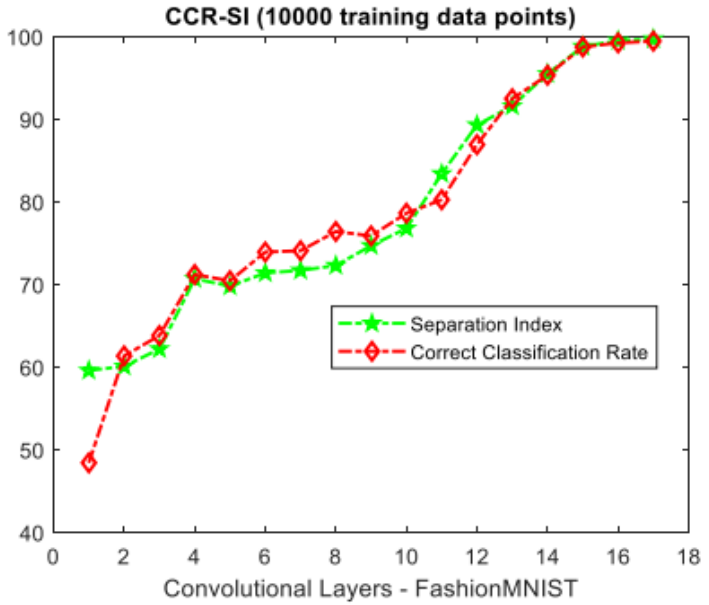


Fig. 11. The plots of separation index and correct classification rates at convolution layers in a pertained "vgg-16" network utilized for classification of "cifar10" for both training and test sets.



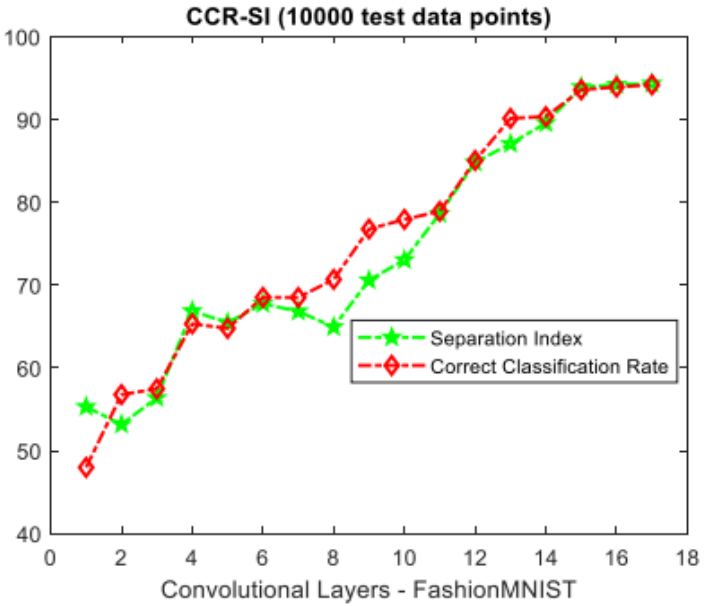


Fig. 12. The plots of separation index and correct classification rates at convolution layers in a trained "Resnet18" network utilized for classification of "Fashion-MNIST" for both training and test sets.

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,...,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:

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

Algortihm2: (To suggest a pre-trained model in transfer learning)

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

$$Data^{j,L_j}=\left\{\left(x_i^{j,L_j}, y_i\right)\right\}_{i=1}^{m}$$

2. For jth pre-trained model select the best subset of x_i^{j,L_j} as $x_i^{j^*,L_j}$ which maximizes $SI(SmI)$

$$Data^{j^*,L_j}=\left\{\left(x_i^{j^*,L_j}, y_i\right)\right\}_{i=1}^{m} \qquad x_i^{j^*,L_j} \subseteq x_i^{j,L_j}$$

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

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

3.2.5 Model Confidence and Guarantee

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

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 $l\ (\boldsymbol{y})$ of a new test example $\boldsymbol{x}$, as $\hat{l}(\hat{\boldsymbol{y}})$.
5. It is assumed that $\boldsymbol{x}_{i_j^*}$ denotes the jth nearest neighbor of input data to x.
6. There is an extra uncertainty in measuring x . It is assumed that x has maximum distance γ with 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 intra distance between examples 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.

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

γ: 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.

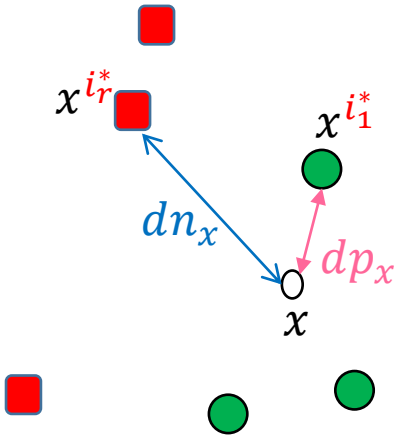
Model Confidence and Guarantee by SI (with training data)

$$dp_x = \left\| x - x^{i_1^*} \right\|$$
$$dn_x = \left\| x - x^{i_r^*} \right\| \quad r = \min_{j=1,..,m} j \text{ subjec to } \delta(l^{i_j^*}, l^{i_1^*}) = 0$$

- $Conf(\hat{l}, l^{i_1^*}) = SI$
- $Guar(\hat{l}, l^{i_1^*}) = \delta(SI, 1) * sign(1 - \frac{dp_x + \gamma}{dn_x - \gamma})$

❖ If $SI(Data) = 1$ the true prediction for x $with$ NN $model$ and $\gamma < 0.5(dp_x - dn_x)$ is guaranteed.

- About those cases that we have $SI^r(Data) = 1$ for $r \gg 1$ the guarantee is satisfied for much higher γ than the cases which we have $SI^1(Data) = 1$.



Model Confidence and Guarantee by Sm I(with training data)

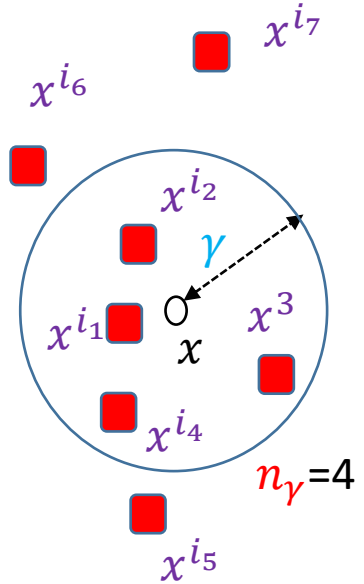
- Assume that for each data point x^i, $\alpha(i^*)=\|y^i - y^{i^*}\|$, where x^{i^*} is the nearest neighbor of x^i.
- Assuming $SmI(Data) = 1$, $\alpha(i^*)$ denotes the distance between the output y^i and its nearest neighbor, y^{imin}.
- In an ideal case a*ssume that the divesity of training dataset 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, γ is the maximum measuring uncertainty of input x.
- Find all training examples: $\{x^{i_j}\}_{j=1}^{n_\gamma}$, which $\|x - x^{i_j}\| \leq \gamma$.

❖ it is guaranteed that if *for each test data pint*, 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

3.3.1 Model Compressing

To remove extra layers and units in both filter and flat layers

Model Compressing

In model compressing some layers and units

Some Definitions:

Filter Part: The layers before fully connected layers which extract features from spatial or temporal inputs. If in some DNNs, one or more than on 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 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".

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%)	٧٣,٢%	٩٣,٠٢%
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%
Our method	**No**	**75.6M (76.0%)**	**87.5%**	**93.49%**
HRFM **[49]**	Yes	73.70M (76.5%)	88.2%	91.23%

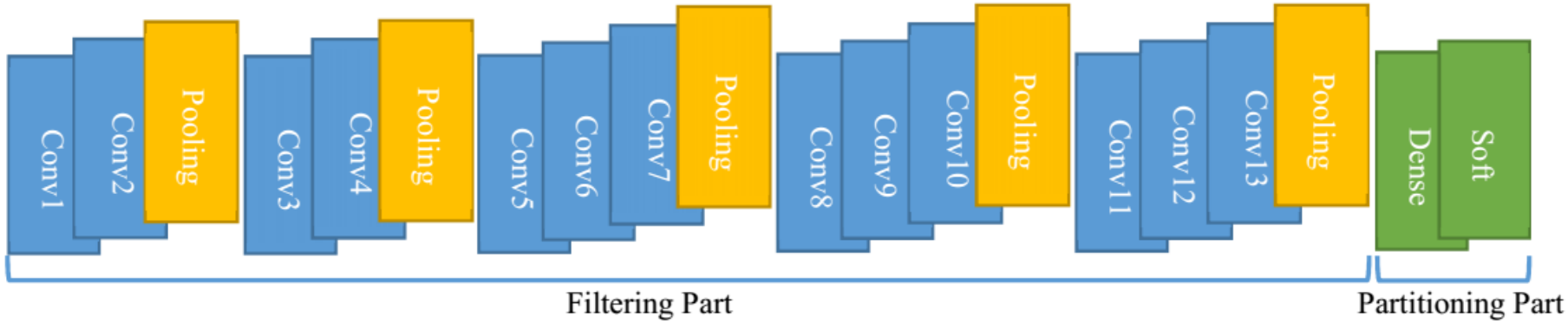


Fig. 5. The architecture of the "VGG-16" network

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%
Our method	**No**	**0.39B(74.4%)**	**77.6%**	**95.00%**

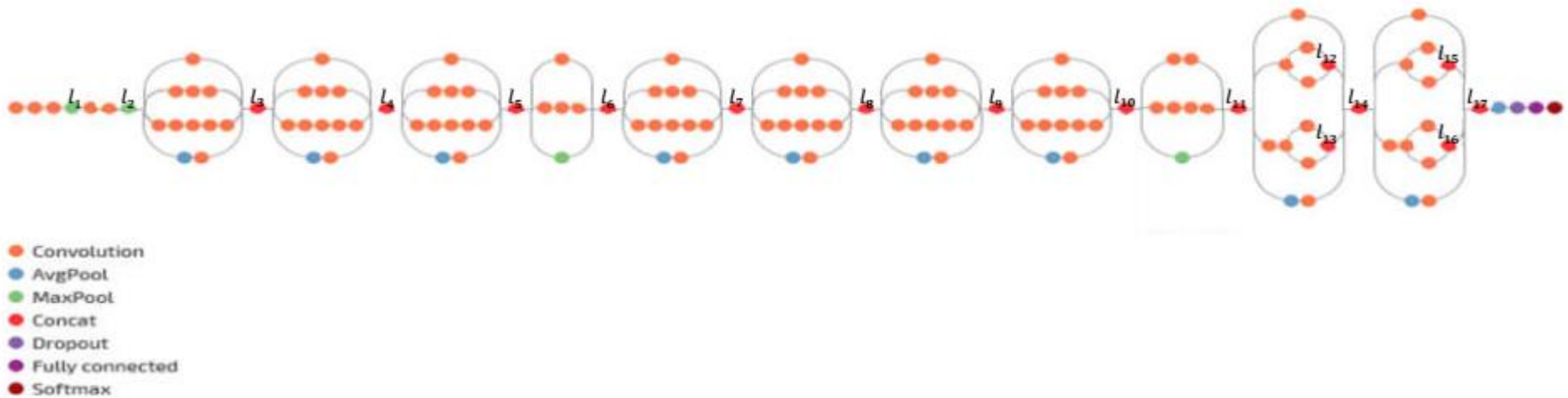


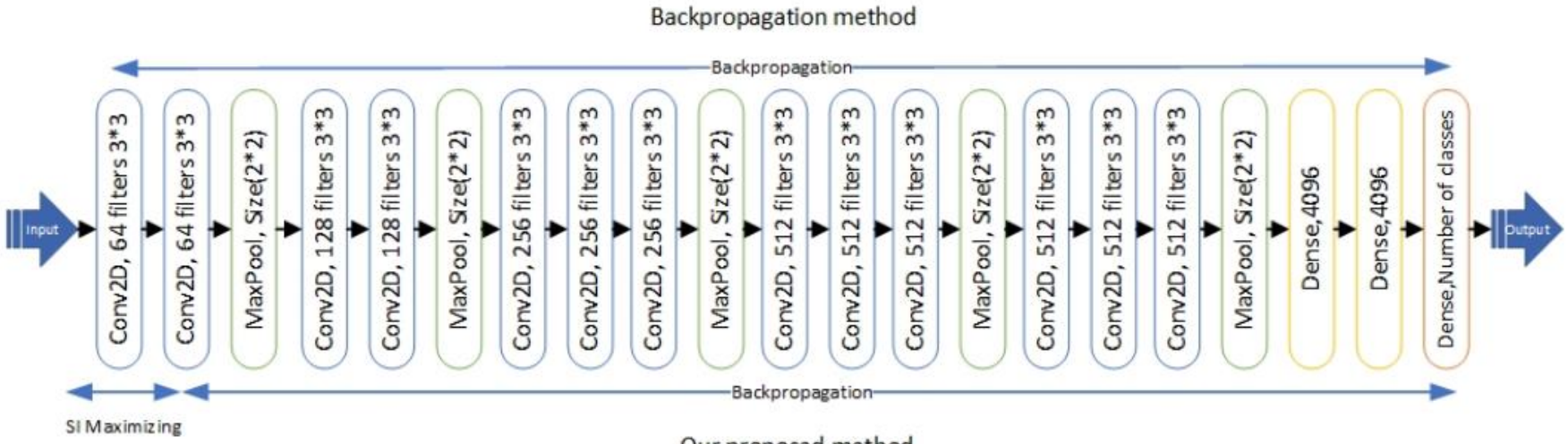
Fig. 7. The architecture of the "Inception V3" network.

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%
Our method	**No**	**0.07B(76.1%)**	**78.8%**	**94.17%**

3.3.2 Forward learning in the first layer

Forward learning in the first layer



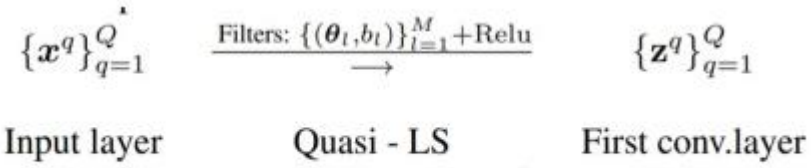


Figure 3: Applying M convolutional filters and biases to the input patterns and then activating the convolved units by Relu function the feature maps are resulted.

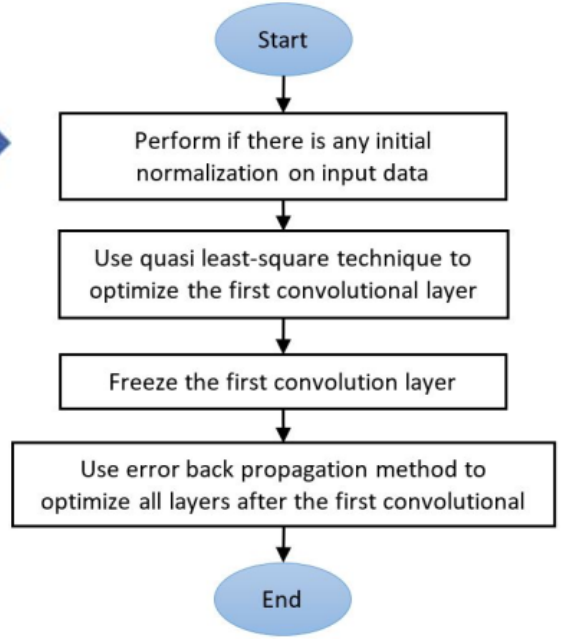


Figure 2: This flowchart indicates the learning method in which the quasi-LS is utilized to l
first convolution layer, and other layers are learned by an error backpropagation method.

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

1. Define the matrix of patches from the input data.
2. Subtract the matrix from their mean
3. Define Auto-correlation matrix of patches
4. Apply SVD technique to get Eigen values and Eigen vectors
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.
6. Based on the filter parameters at the first layer compute the second layer and for each example find nearest neighbor for both positive and negative examples.
7. Use a QLS technique and update all filters.
8. Repeat steps 6 and 7 for several epochs to get maximum possible SI.

Dataset	Learning Method	**AlexNet**	**VGG16**	**ResNet50**	**InceptionV3**
CIFAR10	Backpropagation	84.61	92.95	93.17	94.20
	Our proposed method	**85.84**	**94.81**	**95.02**	**95.43**
	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	**97.09**	**98.36**	**98.49**	**97.77**
	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	**96.62**	**97.63**	**97.16**	**97.14**
	Percentage of improvement	0.41	0.83	0.28	0.33
CIFAR100	Backpropagation	62.22	70.98	75.30	76.31
	Our proposed method	**62.45**	**71.74**	**75.58**	**76.72**
	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	**92.65**	**94.73**	**95.38**	**95.91**
	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:L_{final}

While (SI ($Data^l$) may still increase)

For $l = 1{:}\,L$

$$\boldsymbol{g}_{\boldsymbol{p}^l} = \frac{\partial \text{Loss}_{\text{triplet}}(Data^l)}{\boldsymbol{p}^l}$$

$$\boldsymbol{p}^l := \boldsymbol{p}^l - \gamma \boldsymbol{g}_{\boldsymbol{p}^l}$$

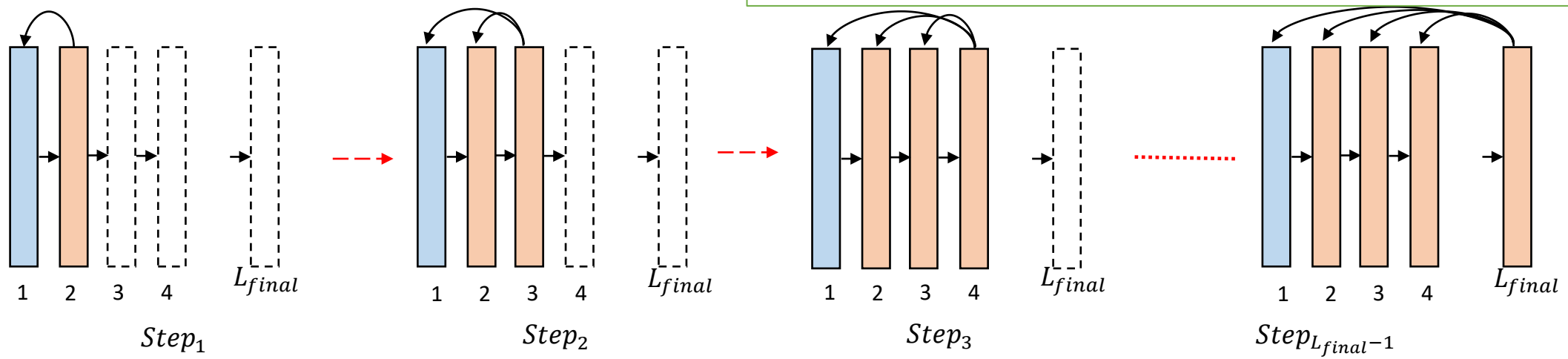
$Data^l = \{(\boldsymbol{x}_i^L, \boldsymbol{y}_i^L)\}$ *Data at layer l*

Triplet Loss

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

$$i_p^* = \underset{\forall q \neq i}{\arg\, min} \frac{1}{\delta(\boldsymbol{y}_i, \boldsymbol{y}_q) + \varepsilon}\left\|\boldsymbol{x}_i^L - \boldsymbol{x}_q^L\right\|^2 \quad \varepsilon \to 0^+$$

$$i_n^* = \underset{\forall q \neq i}{\arg\, min} \frac{1}{1 - \delta(\boldsymbol{y}_i, \boldsymbol{y}_q) + \varepsilon}\left\|\boldsymbol{x}_i^L - \boldsymbol{x}_q^L\right\|^2$$



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

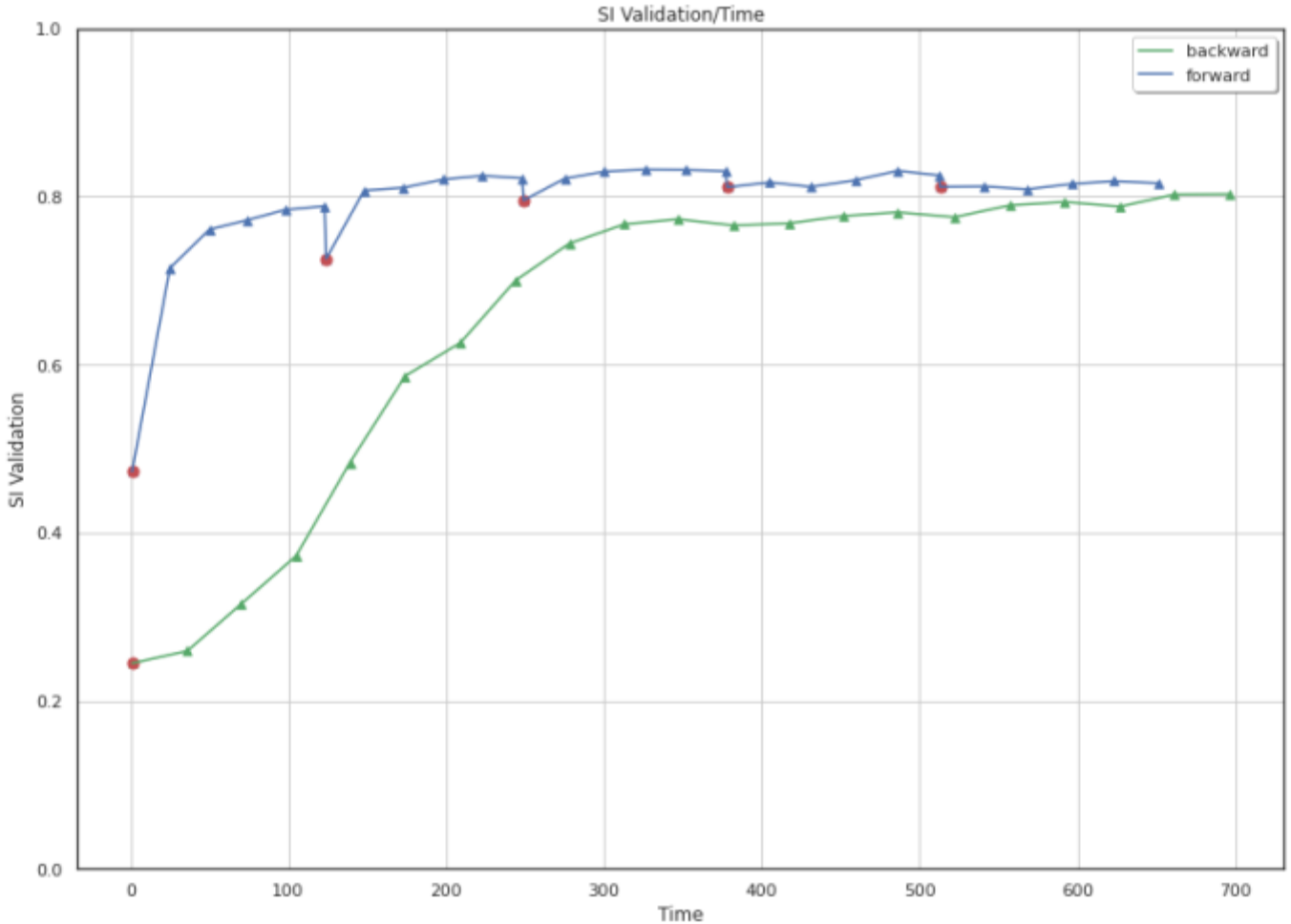
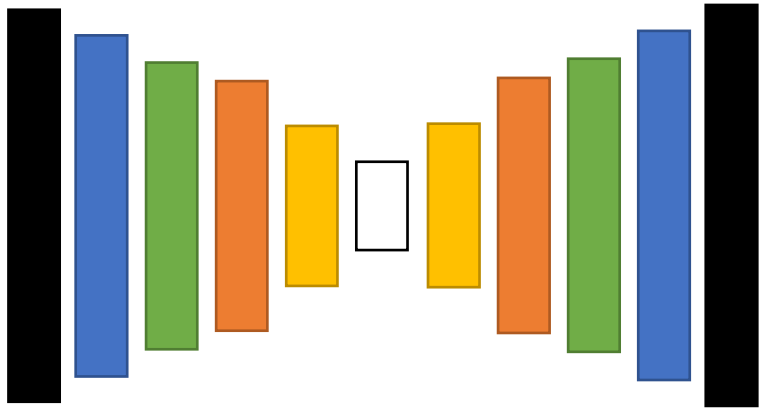


Figure 2. Comparing the separation index of the forward learning and backward learning approaches over time. Each triangle shows a single epoch in the learning process and the red dots in the Forward learning method are convolutional layers.

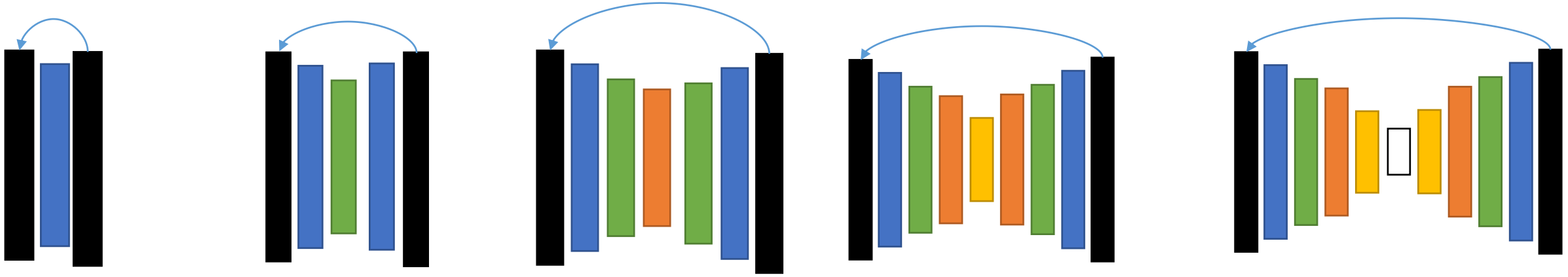
VGG١٦	ResNet٥٩	InceptionV٣	EfficientNetB٠		
٩٢.٩٥	٩٣.١٧	٩٤.٢٠	٩٨.١١	Backpropagation	CIFAR١٠
٩٣.٦٧	**٩٤.٦٥**	**٩٤.٦٣**	**٩٨.٢٤**	Our method	
٧٠.٩٨	٧٥.٣٠	٧٦.٣١	٨٨.١٤	Backpropagation	CIFAR١٠٠
٧١.١٩	**٧٥.٤٧**	**٧٦.٤٩**	**٨٨.١٧**	Our method	
٩٤.١٧	٩٥.٣٨	٩٥.٩١	٩٧.٢٣	Backpropagation	Fashion-MNIST
٩٤.٥٢	**٩٦.٦١**	**٩٦.٢١**	**٩٧.٤١**	Our method	

3.3.4 Layer-wise Forward Auto-Encoder Learning

Layer Wise Forward Auto-Encoder Learning



Smoothness Maximizing

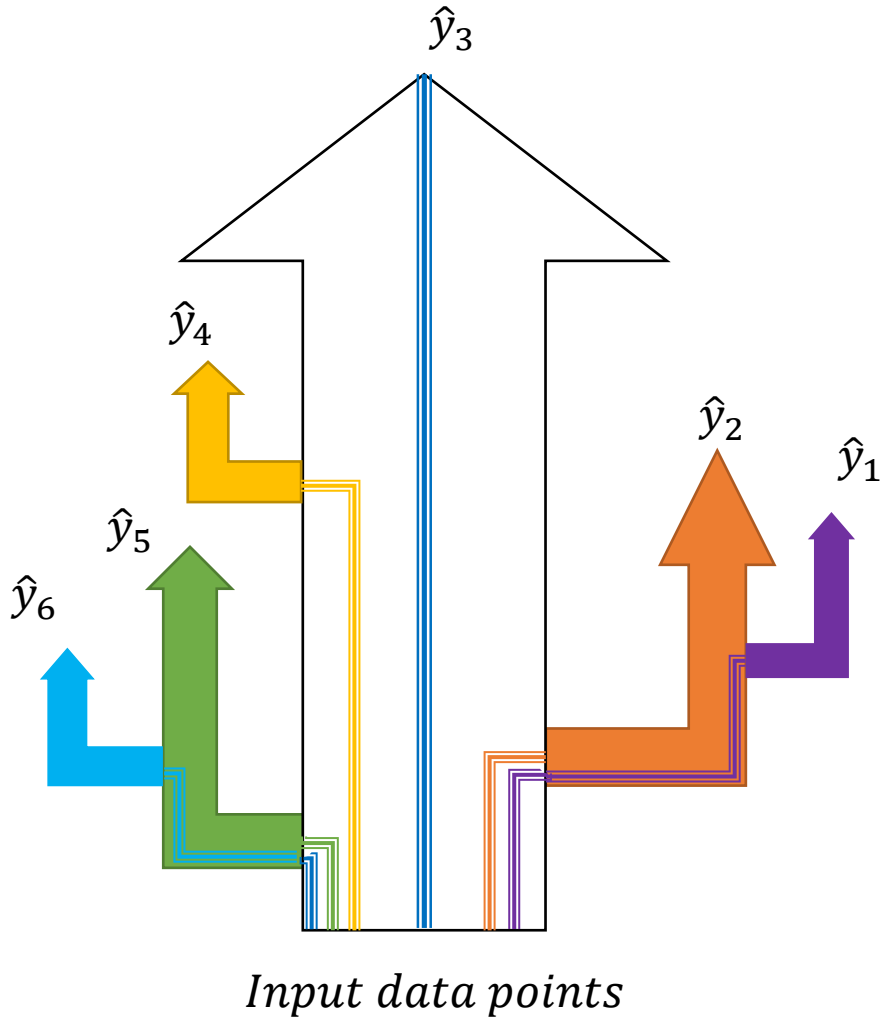


Step1 Step2 Step3 Step4 Step5

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(SmI) to distinguish early features from other features and form a new branch for it.
- Each branch like the man body can define some sub-braches, too.
- In such a DNN, the amount of data-flow decreases through the main body just after each branching.

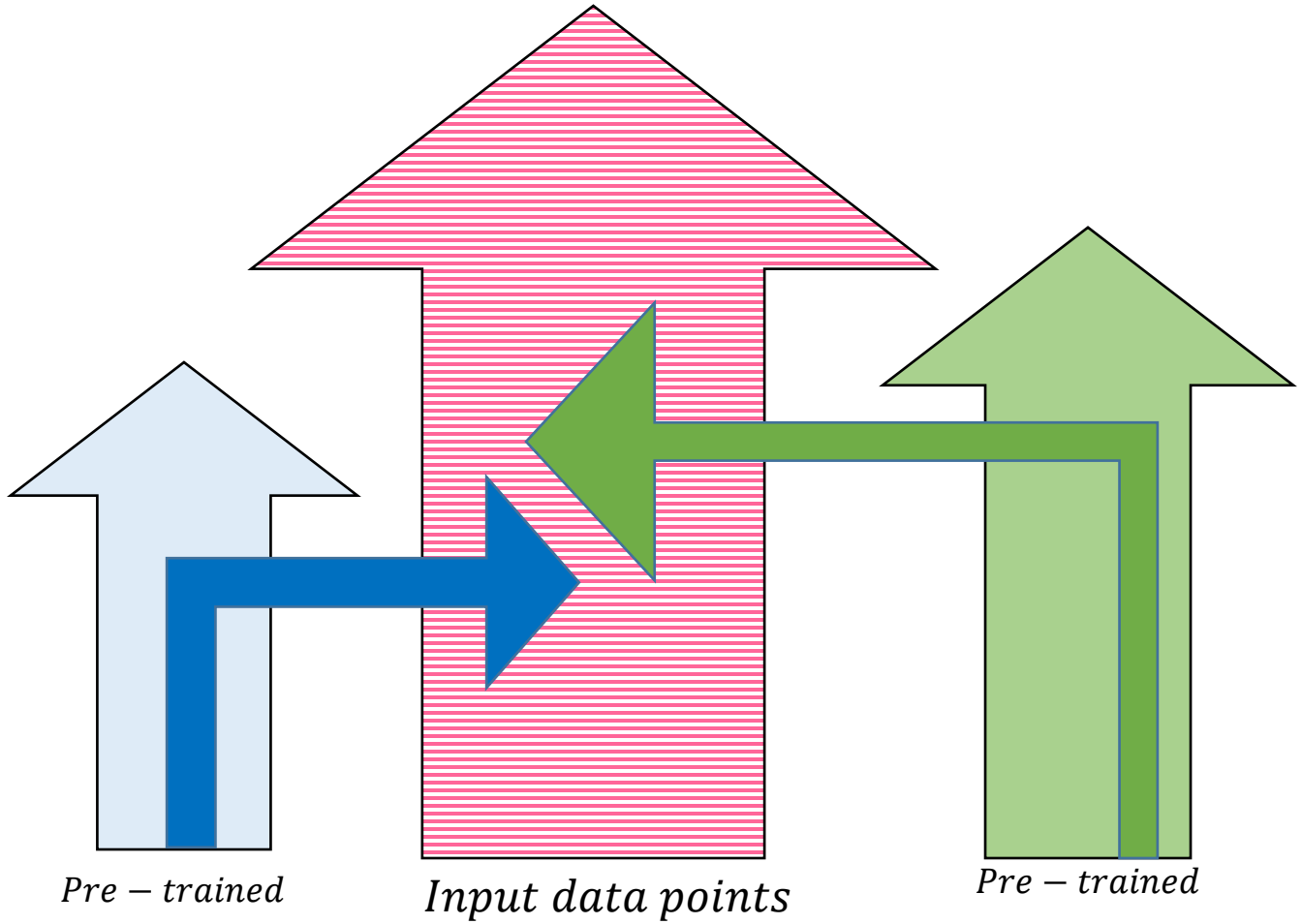


- Branch in point where a jump on SI(SmI) is occurred.
- Find the appropriate part of data to be branched.
- Branching is done in a forward design

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(SmI) to find some desired features from other pre-trained DDNs.

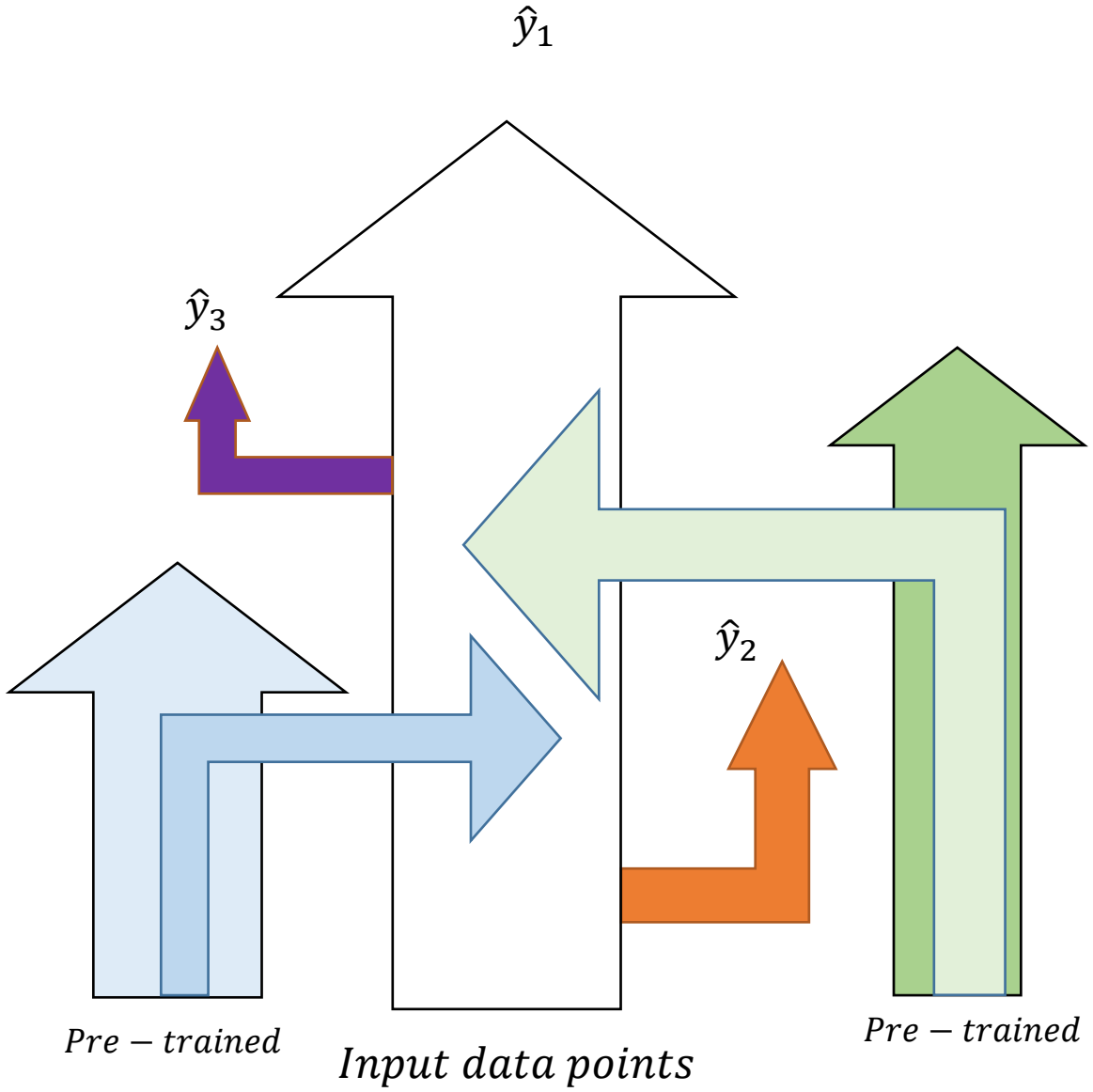


- Fuse in points where SI(SmI) increases significantly.
- Fusion is done in a forward design

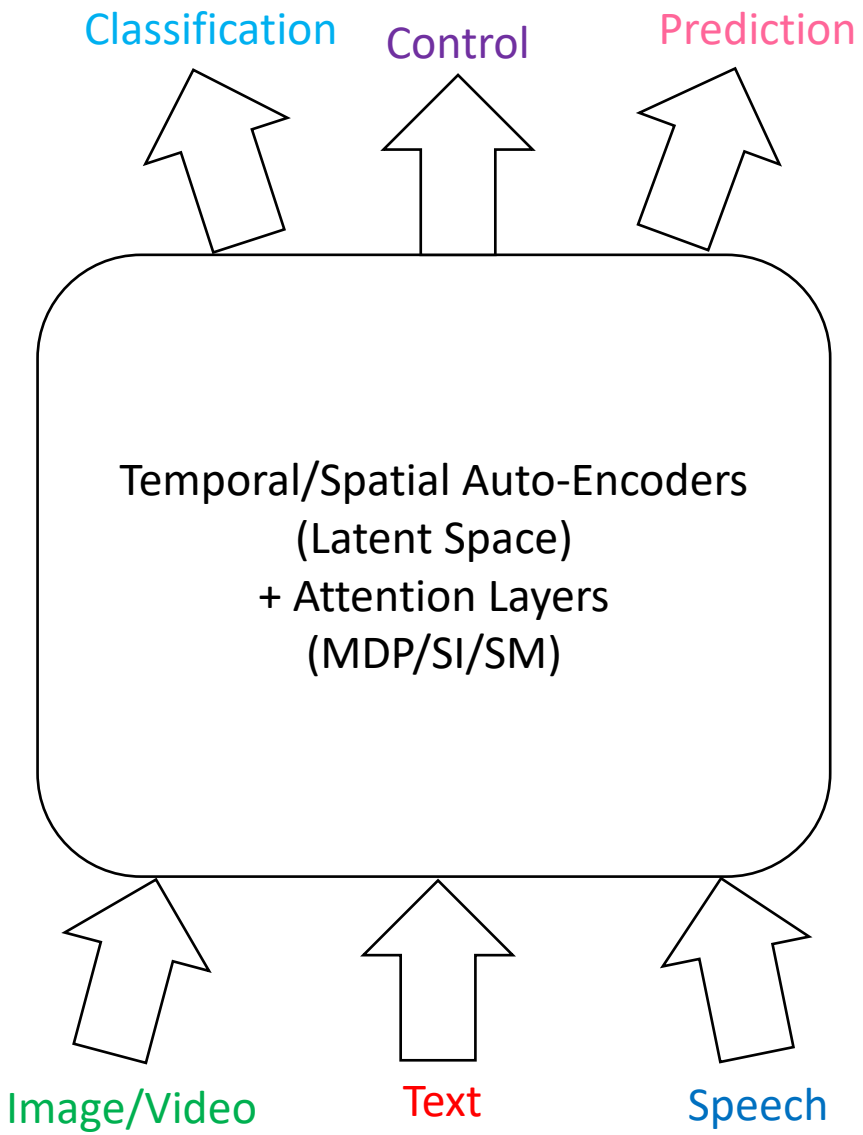
3.3.7 Forward Design

Forward Design

- Branching
- Fusion



3.3.8 Forward Multi-Task Design


Classification
Control
Prediction
Temporal/Spatial Auto-Encoders
(Latent Space)
+ Attention Layers
(MDP/SI/SM)
Image/Video
Text
Speech