Machine learning classification of functional brain imaging for Parkinson’s disease stage prediction

Guan-Hua Huang and Chih-Hsuan Lin

National Yang Ming Chiao Tung University, Hsinchu, Taiwan

國立交通大學 統計學研究所
NATIONAL CHIAO TUNG UNIVERSITY
INSTITUTE OF STATISTICS

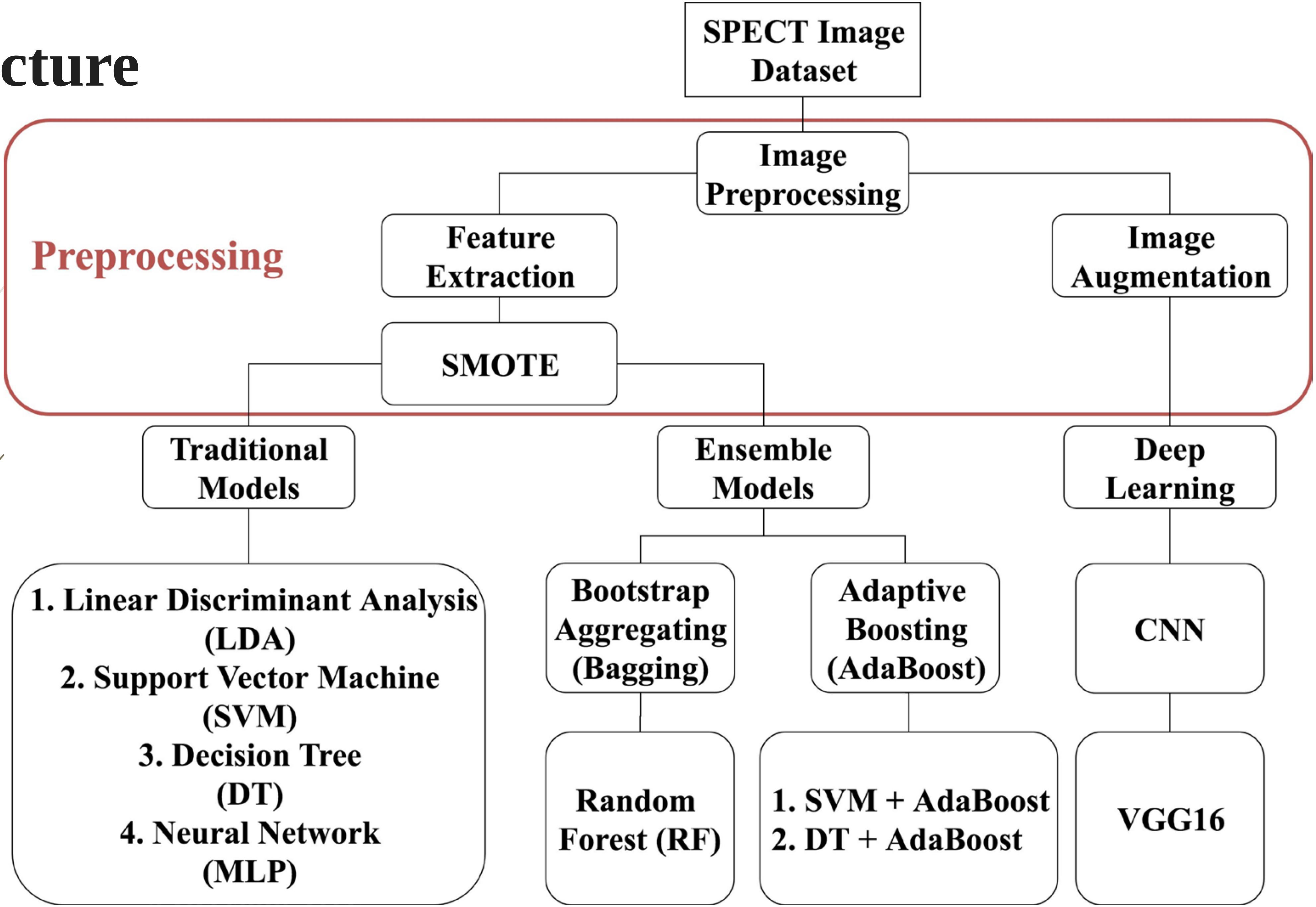
Introduction

- **Parkinson's disease (PD)**: degenerative neurological disorder related to striatal dopamine deficiency
- **Symptoms**: slow movement, muscle stiffness and shaking
- **Prevalence**:
  - 0.1%~0.2% among the general population
  - 2% among people aged over 65 years.
- **Early stage treatment**: very good effect
- **Detection of PD**: functional imaging, ex. SPECT, PET
- **Automatic discrimination of PD**: statistical or machine learning models to replace human judgement

Introduction

- Analysis Methods：
  - Voxels of the complete brain + dimensional reduction
  - Voxels of striatum + shape and intensity distribution analysis
- Researchers have developed a number of methods for classifying subjects as either healthy or suffering from PD.
- We developed system including a series of methods to deal with the multi-classes classification problem in PD stages.
- This system includes image preprocessing, imbalanced data preprocessing, and three kinds of models: traditional model, ensemble model and deep learning model.

Structure


SPECT Image Dataset
Image Preprocessing
Preprocessing
Feature Extraction
Image Augmentation
SMOTE
Traditional Models
Ensemble Models
Deep Learning
1. Linear Discriminant Analysis (LDA)
2. Support Vector Machine (SVM)
3. Decision Tree (DT)
4. Neural Network (MLP)
Bootstrap Aggregating (Bagging)
Adaptive Boosting (AdaBoost)
CNN
Random Forest (RF)
1. SVM + AdaBoost
2. DT + AdaBoost
VGG16

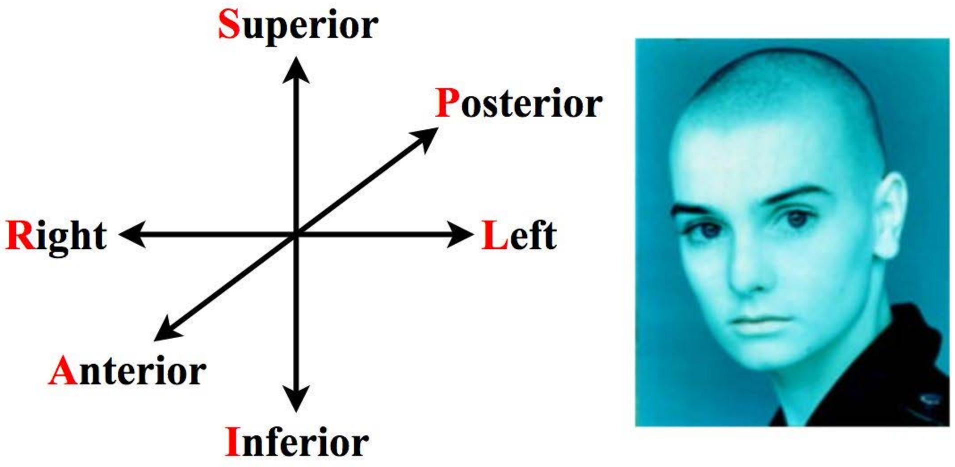
Dataset

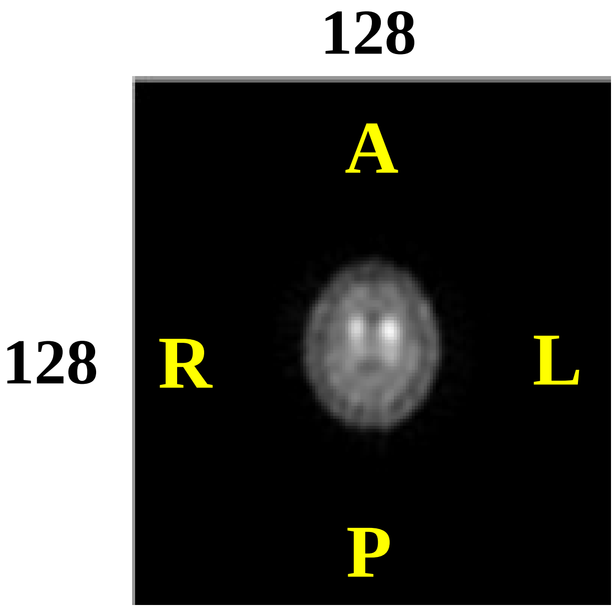
- Retrospective Experiment Designed
- Collect Time：from March 2006 to May 2014
- Data：99mTc-TRODAT-1 **SPECT** Imaging
- Imaging Format：DICOM (Digital Imaging and Communications in Medicine)
- Sample Size:：202 with 3D volume (128 pixel * 128 pixel * n slices)

Stage	Normal	Stage I	Stage II	Stage III	Stage IV	Stage V
Sample Size	6	22	27	53	87	7
Percentage	3%	11%	13%	26%	43%	3%

SPECT

- Single-photon emission computed tomography





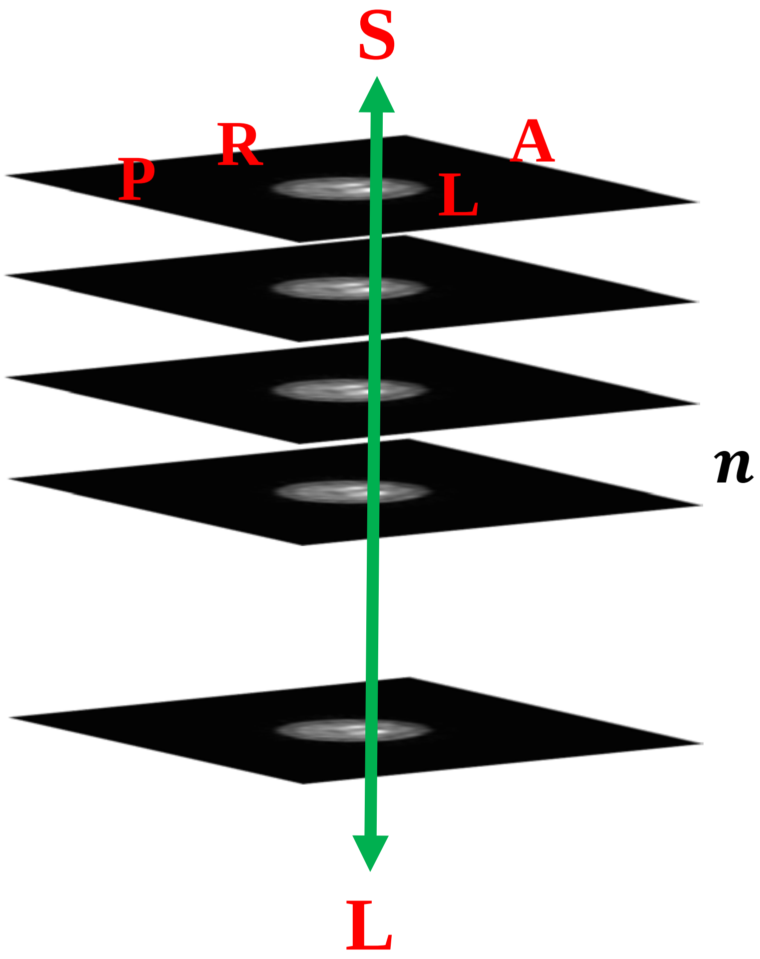
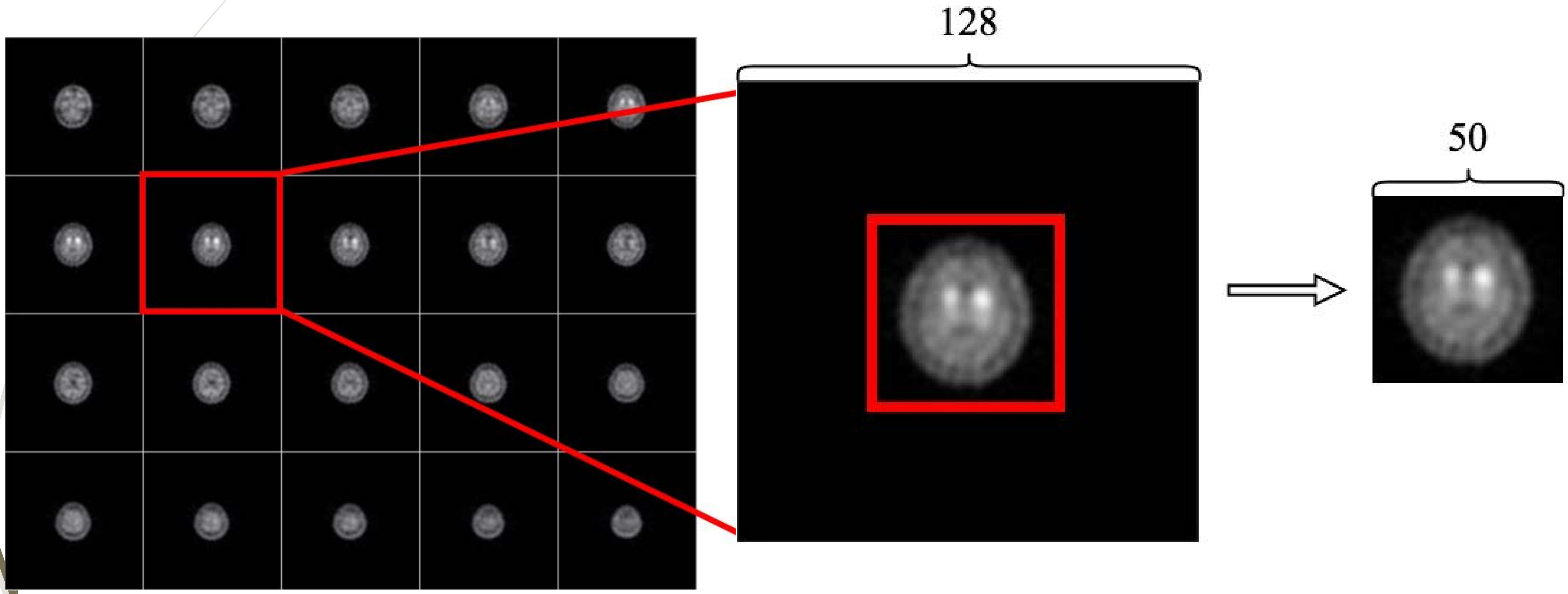
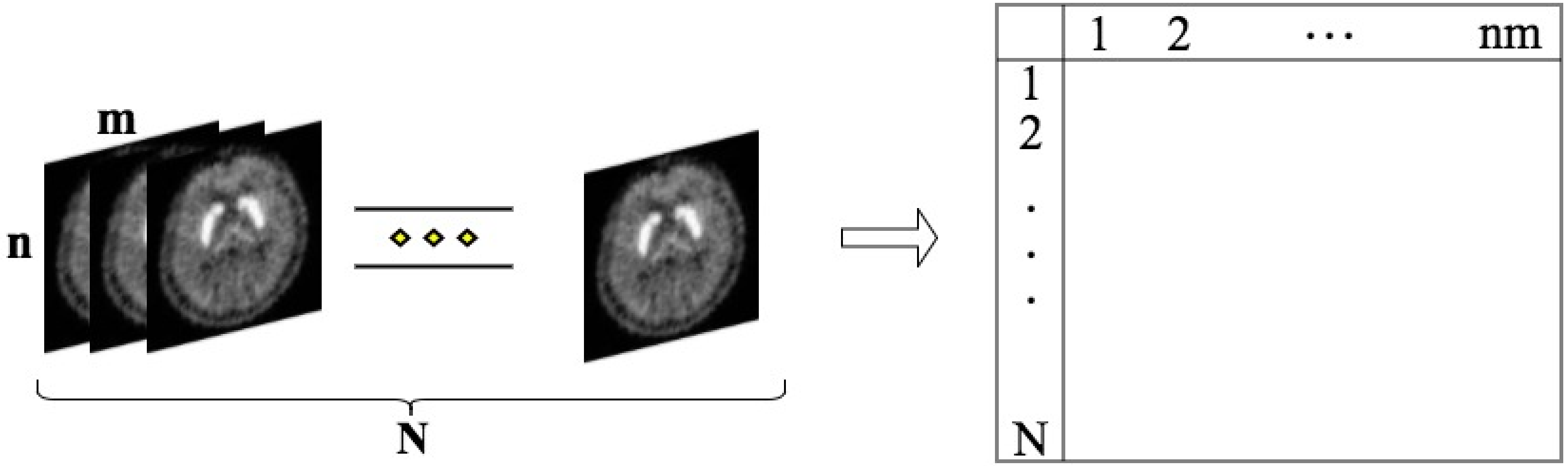


Image Preprocessing

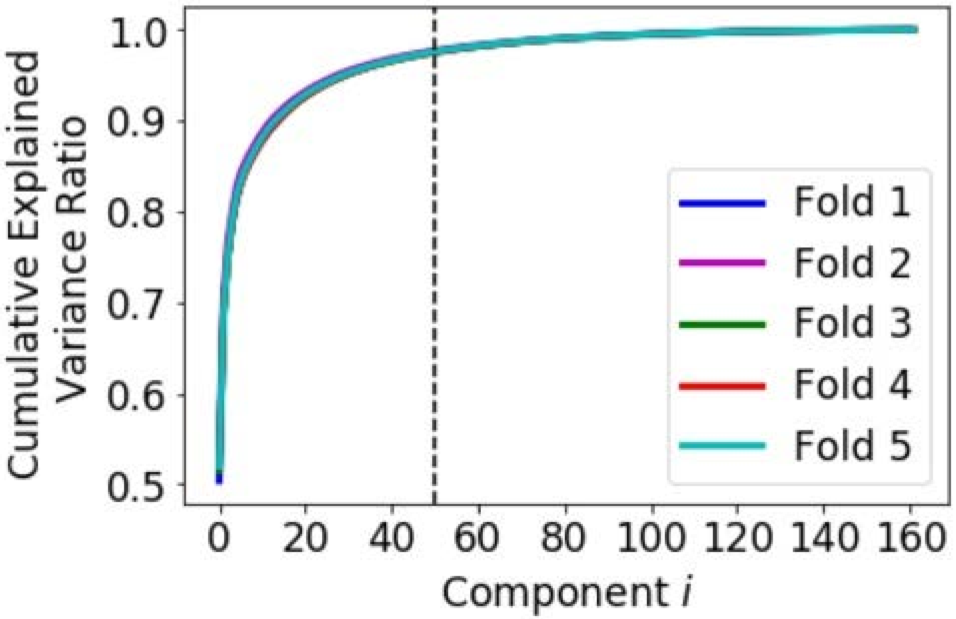


Feature Extraction – PCA

- A principle component analysis (PCA) is concerned with explaining the variance-covariance structure of a set of variables through a few “linear” combinations of these variables.
- Objectives of a principle component analysis:
  - **Dimension reduction**: the total variability of p variables can be accounted for by k principle components, where $p > k$.



Results of PCA



Fold	Cumulative Explained Variance Ratio
1	97.39%
2	97.55%
3	97.33%
4	97.35%
5	97.42%

Sample Size	Training Data	Testing Data	Total
Fold 1	161	41	202
Fold 2	161	41	202
Fold 3	162	40	202
Fold 4	162	40	202
Fold 5	162	40	202

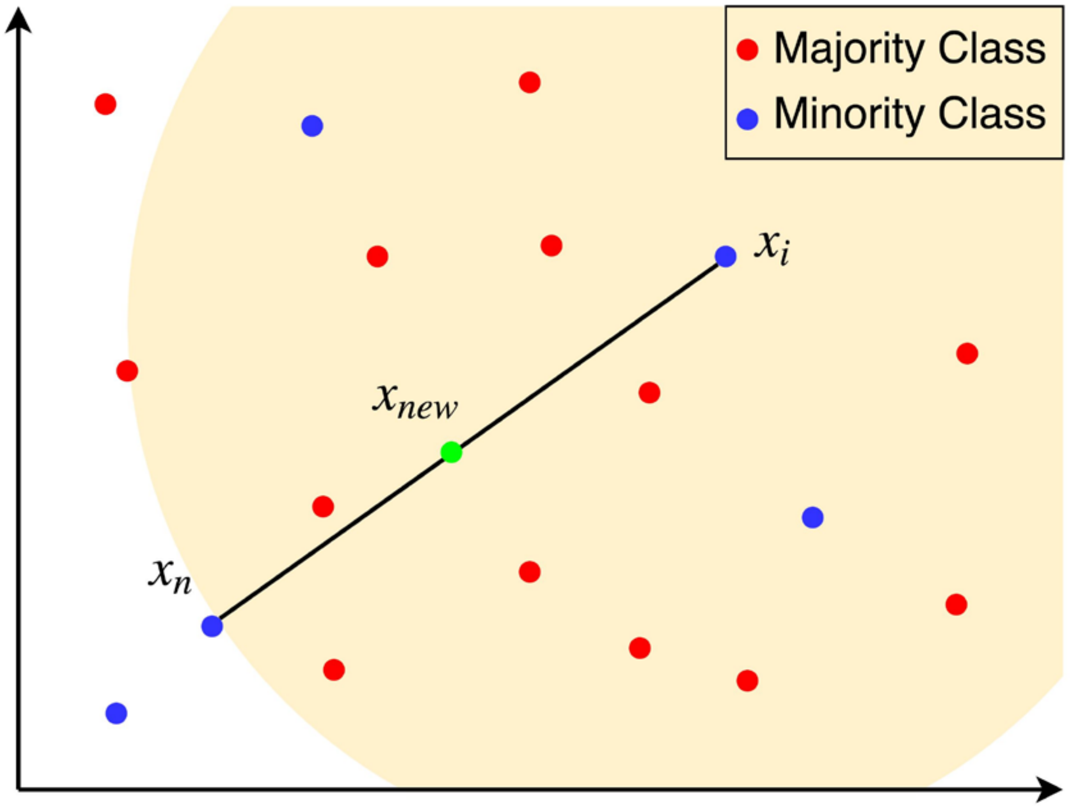
Testing Data

Stage	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
0	1	1	2	1	1
1	4	5	3	9	1
2	5	6	5	5	6
3	11	10	12	10	10
4	19	18	16	13	21
5	1	1	2	2	1
Total	41	41	40	40	40

Imbalanced Data

- **Under-sampling**：This method will random pick samples from the majority classes until each classes is balanced or reach the requirement. The rest part of the majority classes samples will be ignored.
  - Advantage：increasing the sensitivity of a classifier to minority class.
  - Disadvantage：discard potentially useful information
- **Over-sampling**：New minority class data will be drawn with replacement by the original data until each classes is balanced. It directly repeat the samples from the minority classes.
  - Advantage：Unlike under-sampling, this method leads to no information loss.
  - Disadvantage：It increases the likelihood of overfitting since it replicates the minority class events.

Over-sampling – SMOTE



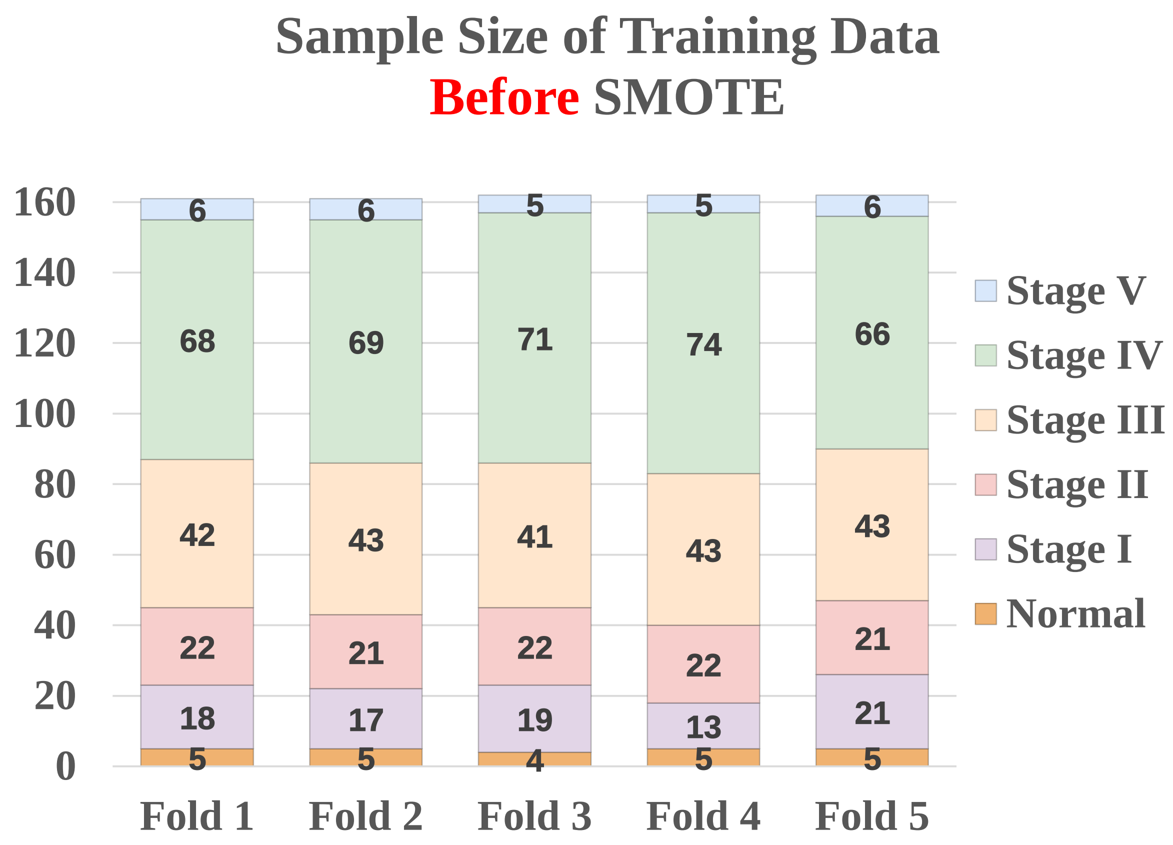
- Step 1：Considering a sample x_i belonging to minority class, select k nearest-neighbors, which also belong to minority class.
- Step 2： Randomly pick a sample x_n from these k nearest-neighbors.
- Step 3： A new sample x_{new} is generated as follows:

$$x_{new} = x_i + \lambda * |x_n - x_i|$$

where λ is a random number in the range [0, 1].

- Step 4： Repeat step 1 to step 3 until the minority sample size is reach the requirement.

Sample Size of Training Data Before/After SMOTE



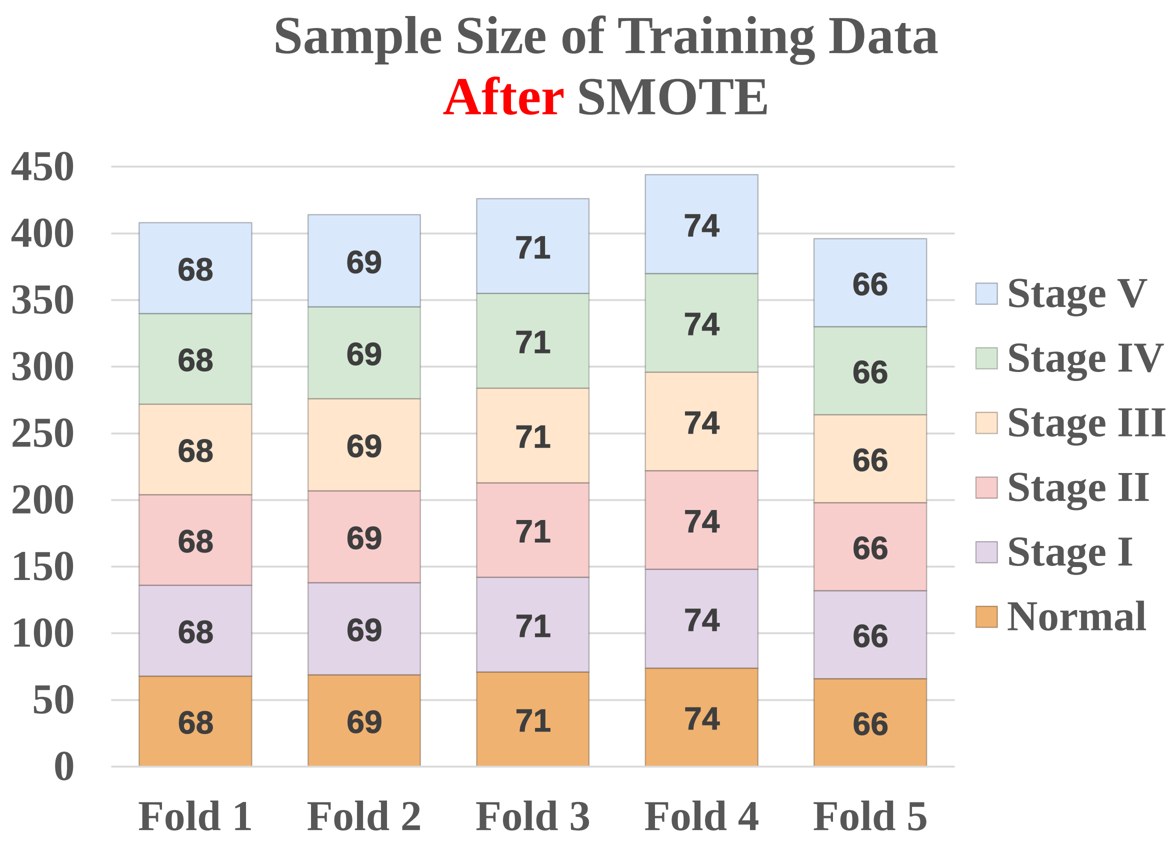
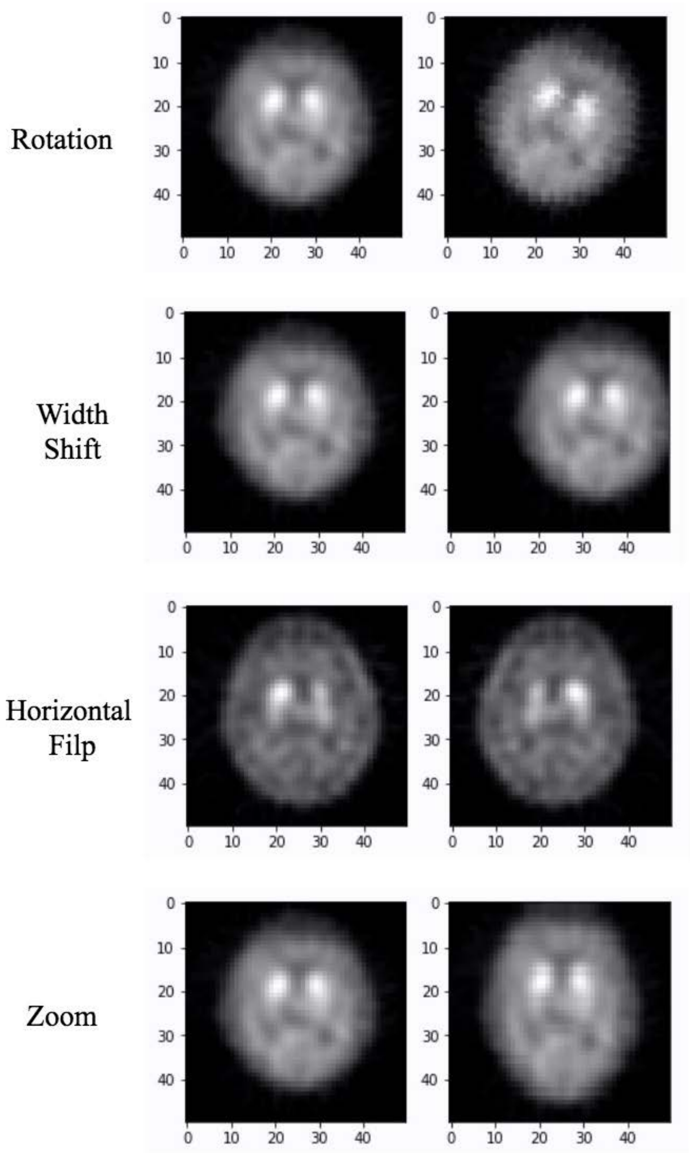


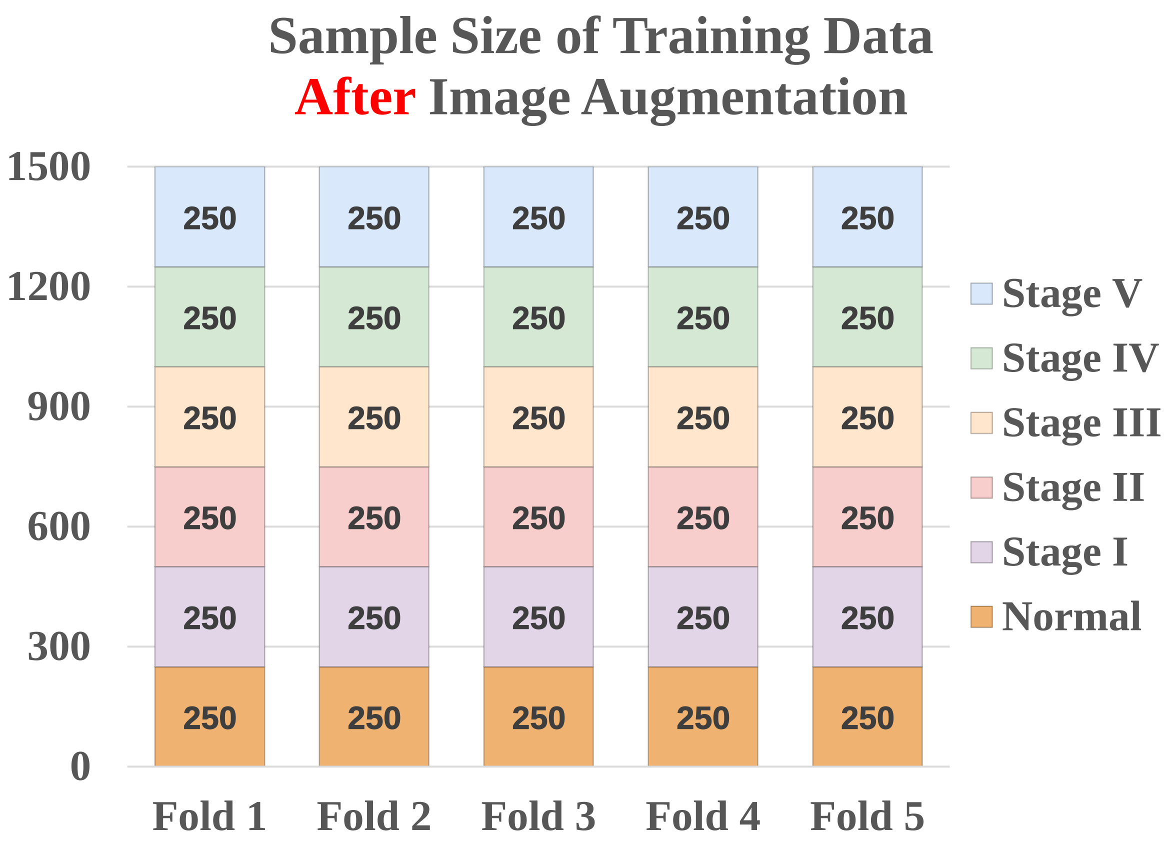
Image Augmentation

- Image augmentation **artificially** creates training images through different ways of processing or combination of multiple processing.
- Traditional transformations：using a combination of affine transformations to manipulate the training data
- For each input image, we generate **duplicate** images that are shifted, zoomed in/out, rotated, flipped, distorted, or shaded with a hue.

Origin Shift Zoom Rotate Flip Shear Coloring

Deep Learning – Image Augmentation





Traditional Model

Linear Discriminant Analysis (LDA)

Bayes' rule

$$P(y=k \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid y=k)P(y=k)}{P(\mathbf{x})} = \frac{P(\mathbf{x} \mid y=k)P(y=k)}{\sum_l P(\mathbf{x} \mid y=l) \cdot P(y=l)}$$

Multivariate Normal Distribution

$$P(\mathbf{x} \mid y=k) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)'\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right)$$

Add Nature Log

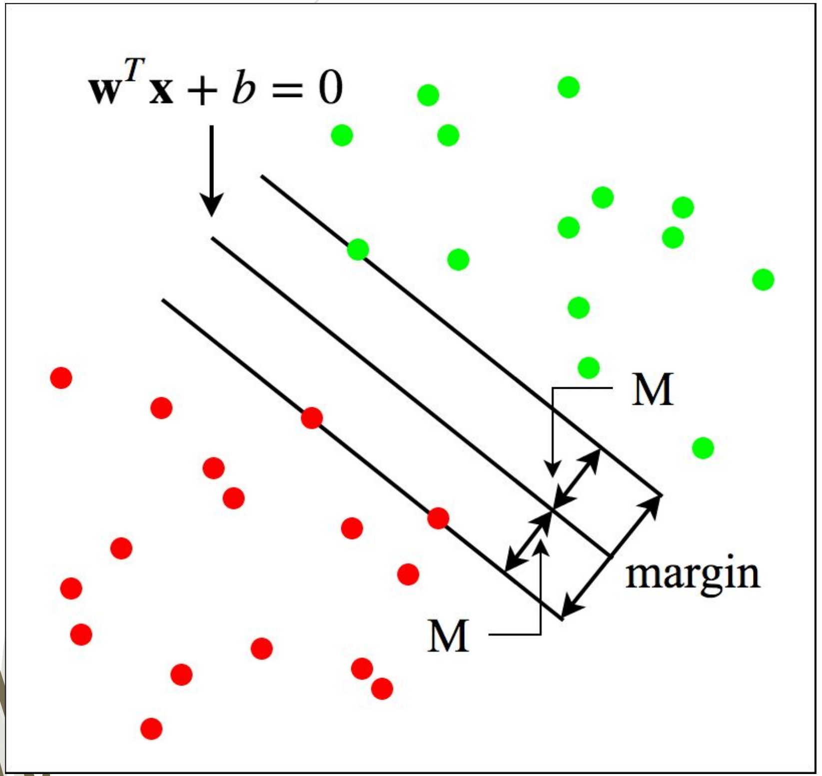
$$\begin{aligned} &\ln P(\mathbf{x} \mid y=k)P(y=k) \\ &= \ln P(y=k) - \frac{p}{2}\ln(2\pi) - \frac{1}{2}\ln|\boldsymbol{\Sigma}| - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)'\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) \\ &= \max_i \ \ln P(\mathbf{x} \mid y=i)P(y=i), \text{for } i=1,2,\ldots,g \end{aligned}$$

Linear Discrimination Score

$$d_i(\mathbf{x}) = \boldsymbol{\mu}_i'\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_i'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i + \ln P(y=i)$$

$$\hat{d}_i(\boldsymbol{x}) = \bar{\boldsymbol{x}}_i'\boldsymbol{S}_{pooled}^{-1}\mathbf{x} - \frac{1}{2}\bar{\boldsymbol{x}}_i'\boldsymbol{S}_{pooled}^{-1}\bar{\boldsymbol{x}}_i + \ln P(y=i), \text{for } i=1,2,\ldots,g$$

Support Vector Machine (SVM)



Hyperplane

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = \sum_{i=1}^{p} w_i x_i + b = 0$$

Classification Rule

$$G(\boldsymbol{x}) = sign(\boldsymbol{w}^T\boldsymbol{x} + b)$$

$$y = \begin{cases} 1, \text{if } G(\boldsymbol{x}) \geq 0 \\ -1, \text{if } G(\boldsymbol{x}) < 0 \end{cases}$$

Optimization Problem

$$\max_{w,b,\|w\|=1} M$$

$$\text{subject to } y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq M, i = 1, \ldots, N$$

Objective Function

$$\min_{w,b} \|\boldsymbol{w}\|$$

$$\text{subject to } y_i(\boldsymbol{w}^T\boldsymbol{x} + b) \geq 1, i = 1, \ldots, N$$

Decision Tree (DT) – CART

- Step 1：Let the data at node m be represented by D^m. For each candidate split $\theta = (j, t_a)$ consisting of a feature j and threshold t_a, partition the data into $D^m_{left}(\theta)$ and $D^m_{right}(\theta)$ subsets.

$$D^m_{left}(\theta) = (j, y) | j \le t_a$$
$$D^m_{right}(\theta) D^m \backslash \mathrm{D}^m_{left}(\theta)$$

- Step 2：For each candidate split θ, the impurity at m is computed using an impurity function $H(\cdot)$. For CART, $H(\cdot)$ is Gini impurity.

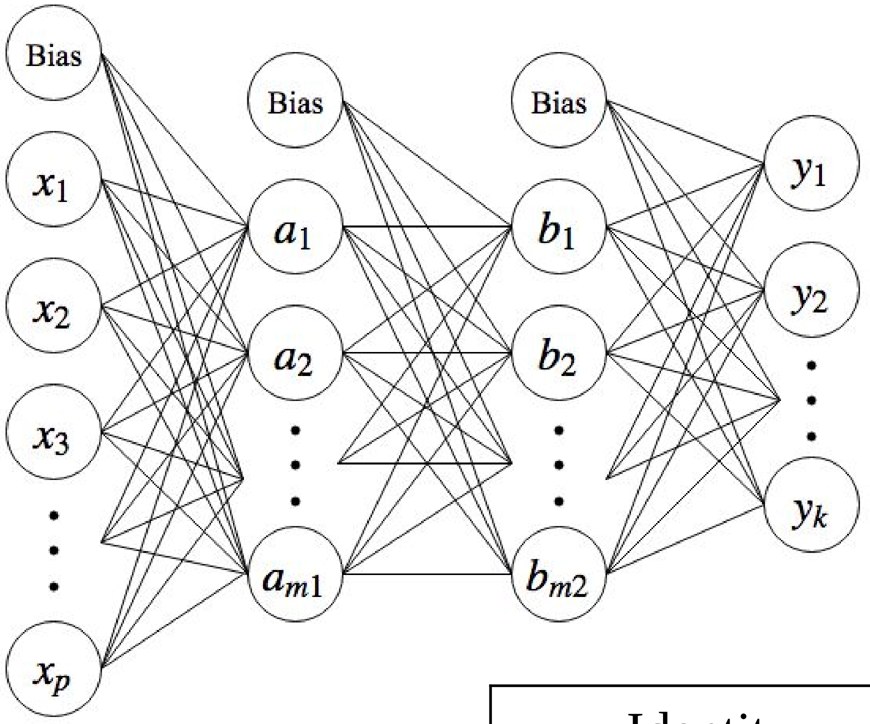
$$G(D^m, \theta) = \frac{n_{left}}{N_m} H\left(D^m_{left}(\theta)\right) + \frac{n_{right}}{N_m} H\left(D^m_{right}(\theta)\right)$$

Gini Impurity：$p_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k) \Rightarrow H(D_m) = \sum_k p_{mk}(1 - p_{mk})$

- Step 3：Select the parameters that minimize the impurity. $\theta^* = \underset{\theta}{\operatorname{argmin}}\, G(D^m, \theta)$

- Step 4：Recurse for subsets $D^m_{left}(\theta^*)$ and $D^m_{right}(\theta^*)$ until the maximum allowable depth is reached, $N_m < \underset{samples}{\min}\ or\ N_m = 1$.

Neural Network (MLP)



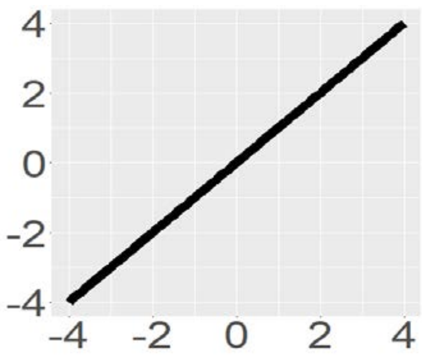
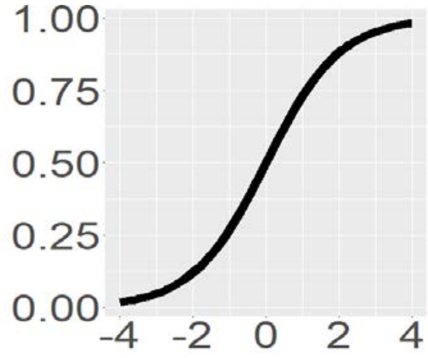
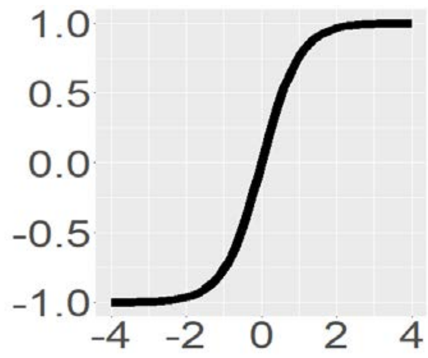
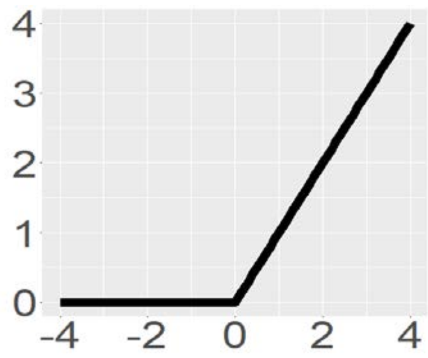
Transformed Vector Function

$$f(\boldsymbol{x}) = \boldsymbol{W}_d g(\ldots g(\boldsymbol{W}_2 g(\boldsymbol{W}_1^T \boldsymbol{x} + b_1) + b_2)\ldots) + b_{d+1}$$

Softmax Function

$$\mathrm{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{l=1}^{k} \exp(z_l)}$$

Activation Function

Identity	Sigmoid	tanh	ReLU
			

Ensemble Learning

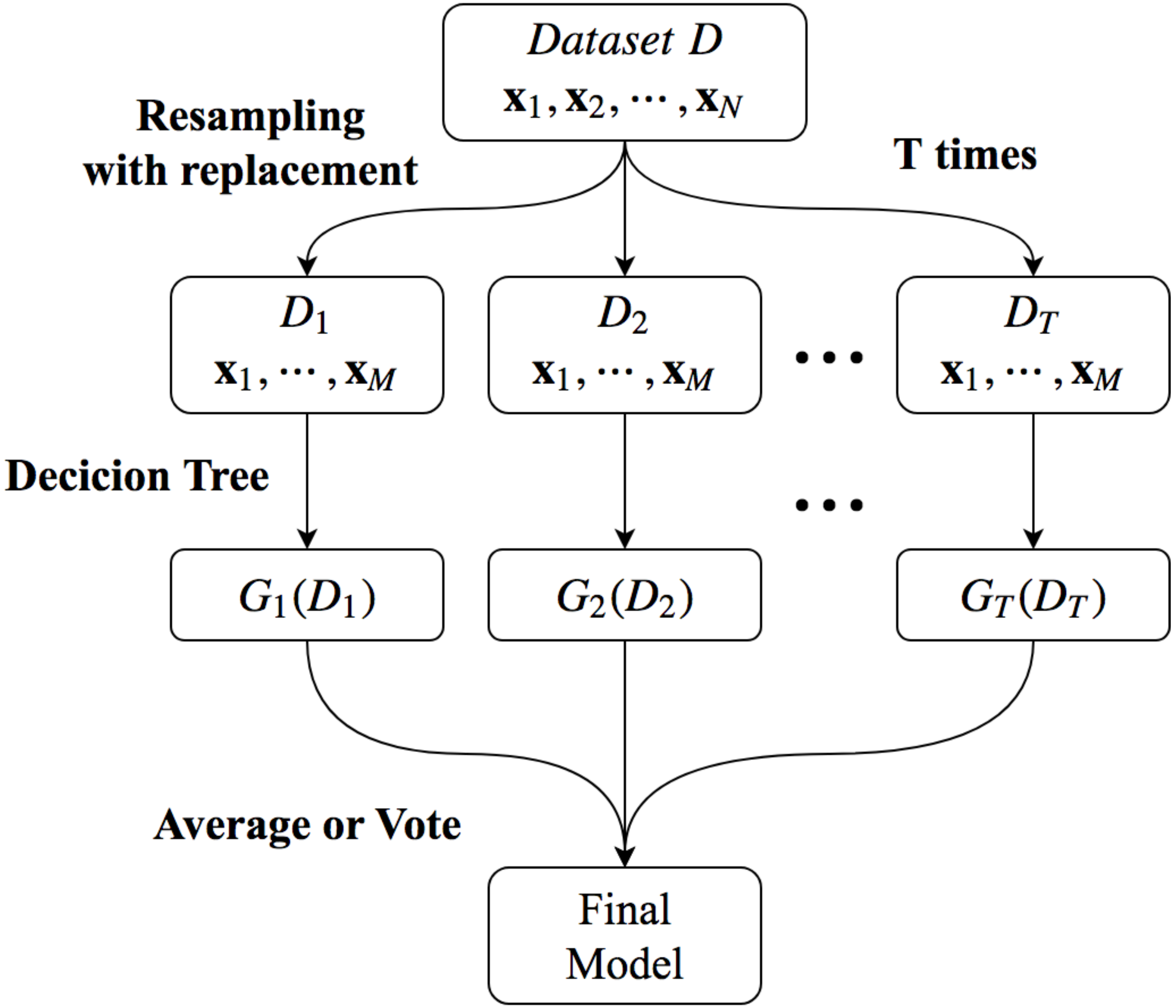
Ensemble Learning

- **Averaging methods**：The driving principle is to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced.
  - Example：Bagging, ***Random Forest***
- **Boosting methods**：Base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble.
  - Example：***Adaptive Boosting***, Gradient Tree Boosting

Random Forest (RF)

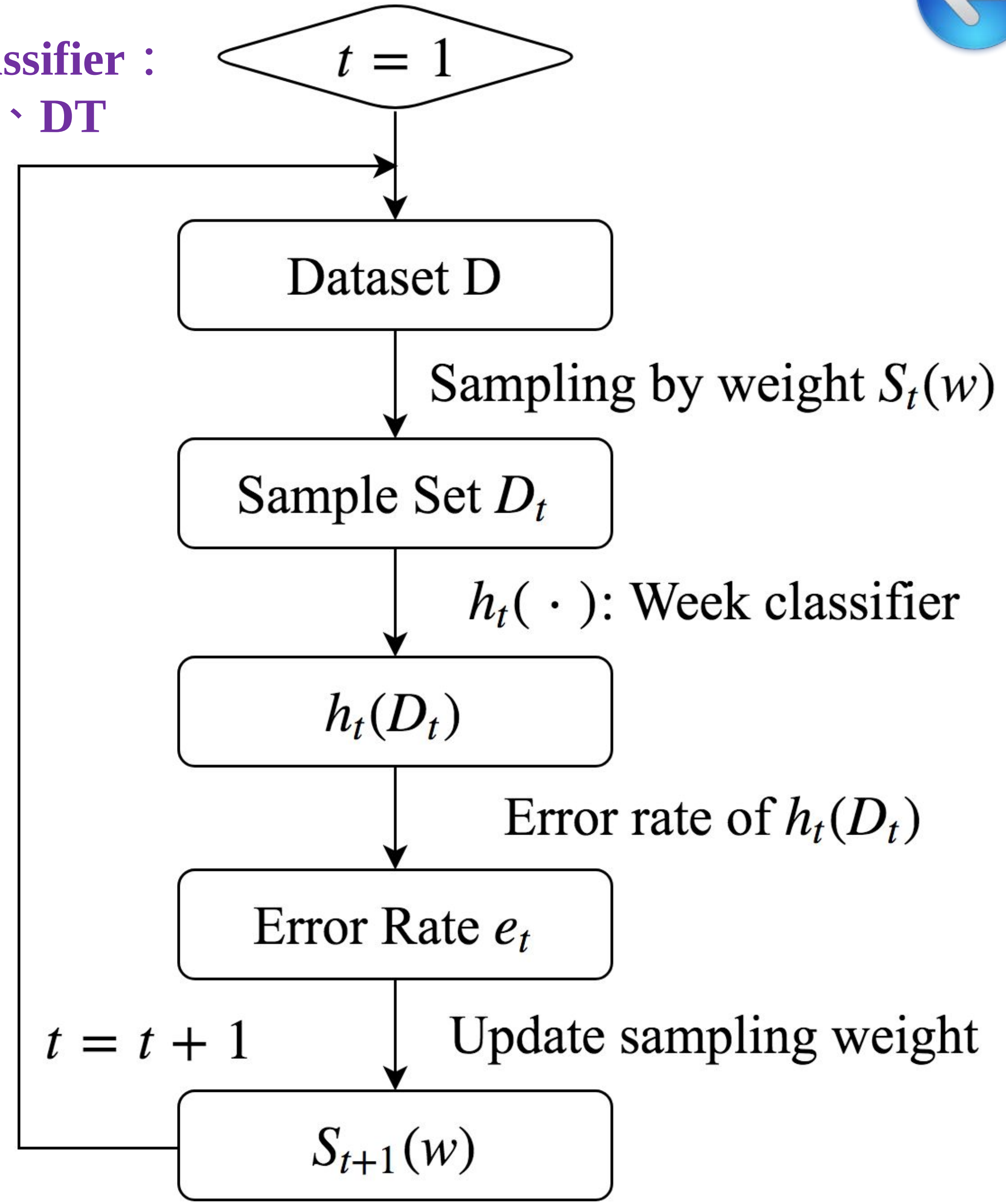
- For a given number of trees T in the forest and dataset D
- (1) For $t = 1,2,\cdots,T$ ：
  - (a) Dataset D_t is drawn **with replacement** from D at random.
  - (b) Construct decision tree $G_t(\boldsymbol{x})$ by D_t.
- (2) For classification problem, the class that the most classifier vote for is the final class.

Week classifier：
Decision Tree



Adaptive Boosting

Week classifier：
SVM、DT



Initial Sample Weight

$$S_1(\boldsymbol{w}) = (w_{11}, w_{12}, \ldots, w_{1N}), w_{1i} = \frac{1}{N} \text{ for } i = 1, 2 \ldots N$$

Error Rate

$$e_t = P(h_t(\boldsymbol{x}_i) \neq y_i) = \sum_{i=1}^{N} w_{ki} I(h_t(\boldsymbol{x}_i) \neq y_i)$$

Update sampling weight

$$w_{t+1,i} = \frac{w_{ti}}{Z_t} \exp(-\alpha_t y_i h_t(\boldsymbol{x}_i)), i = 1, 2, \ldots m$$

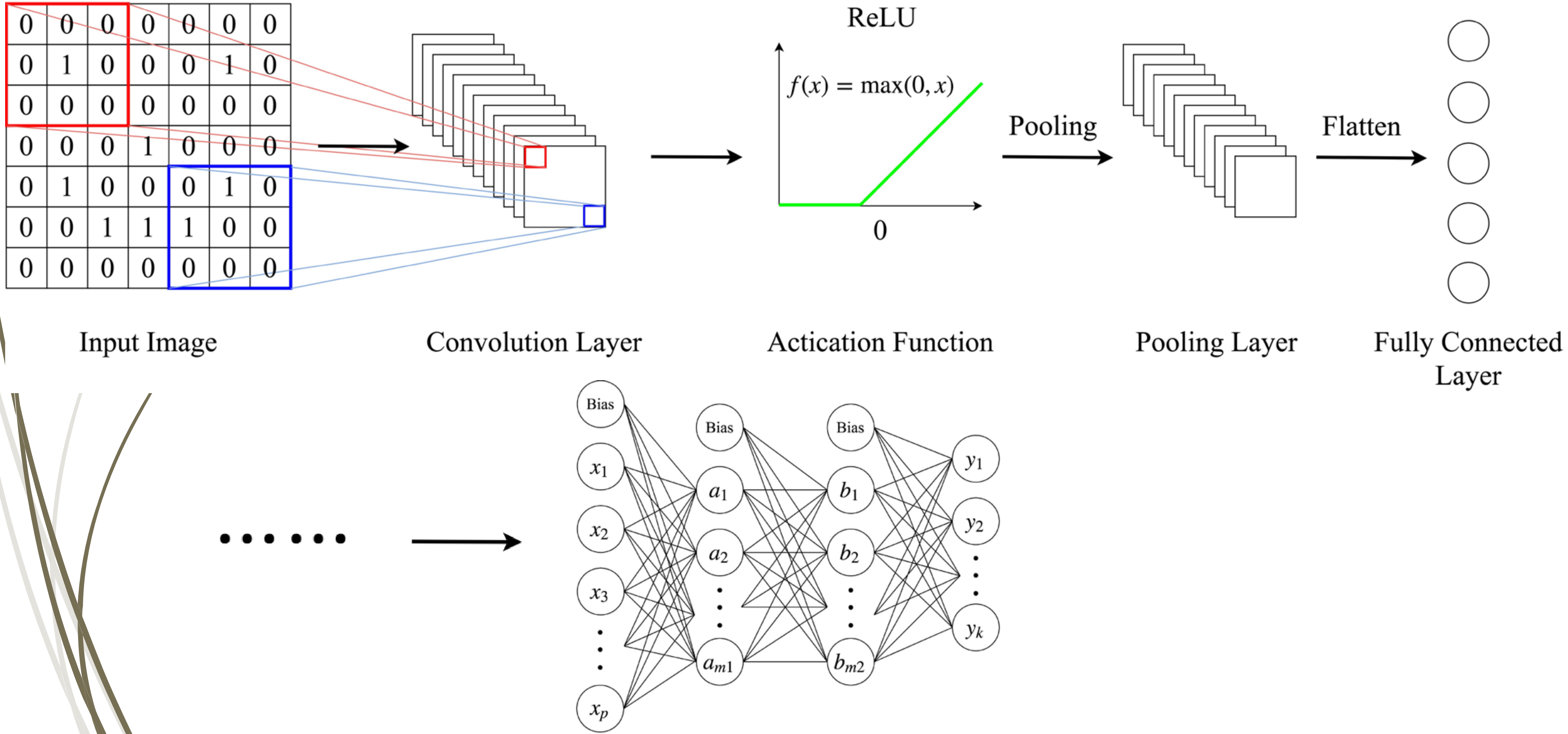
$$\alpha_t = \frac{1}{2} \log \frac{1 - e_t}{e_t} + \log(k - 1)$$

$$Z_t = \sum_{i=1}^{N} w_{ti} \exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))$$

Final Classifier $f(\boldsymbol{x}) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$

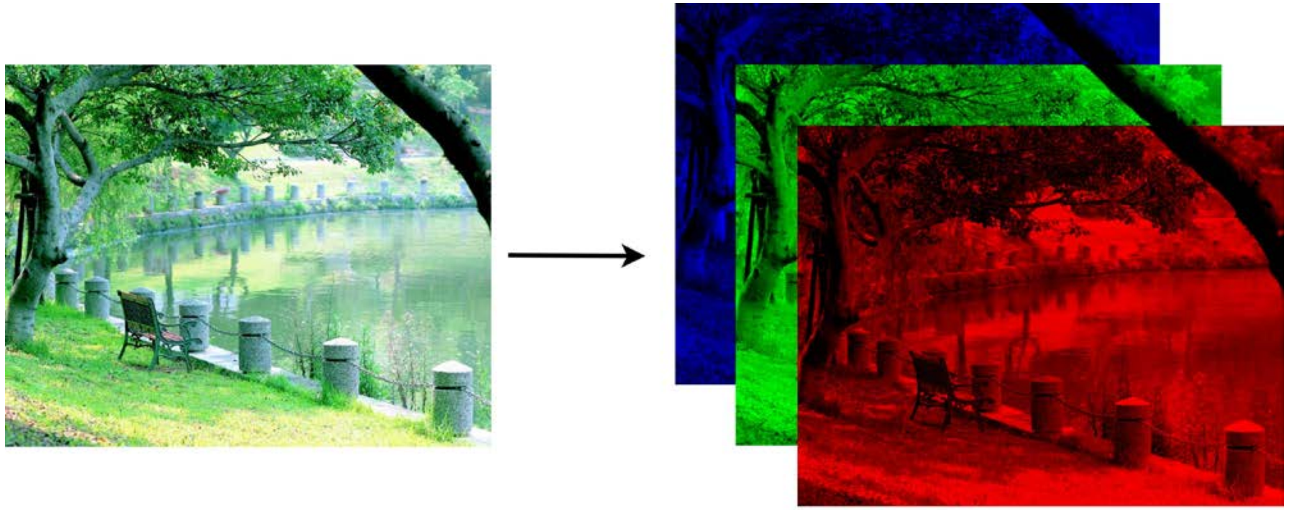
Deep Learning

Deep Learning – CNN



Convolutional Neural Network (CNN)

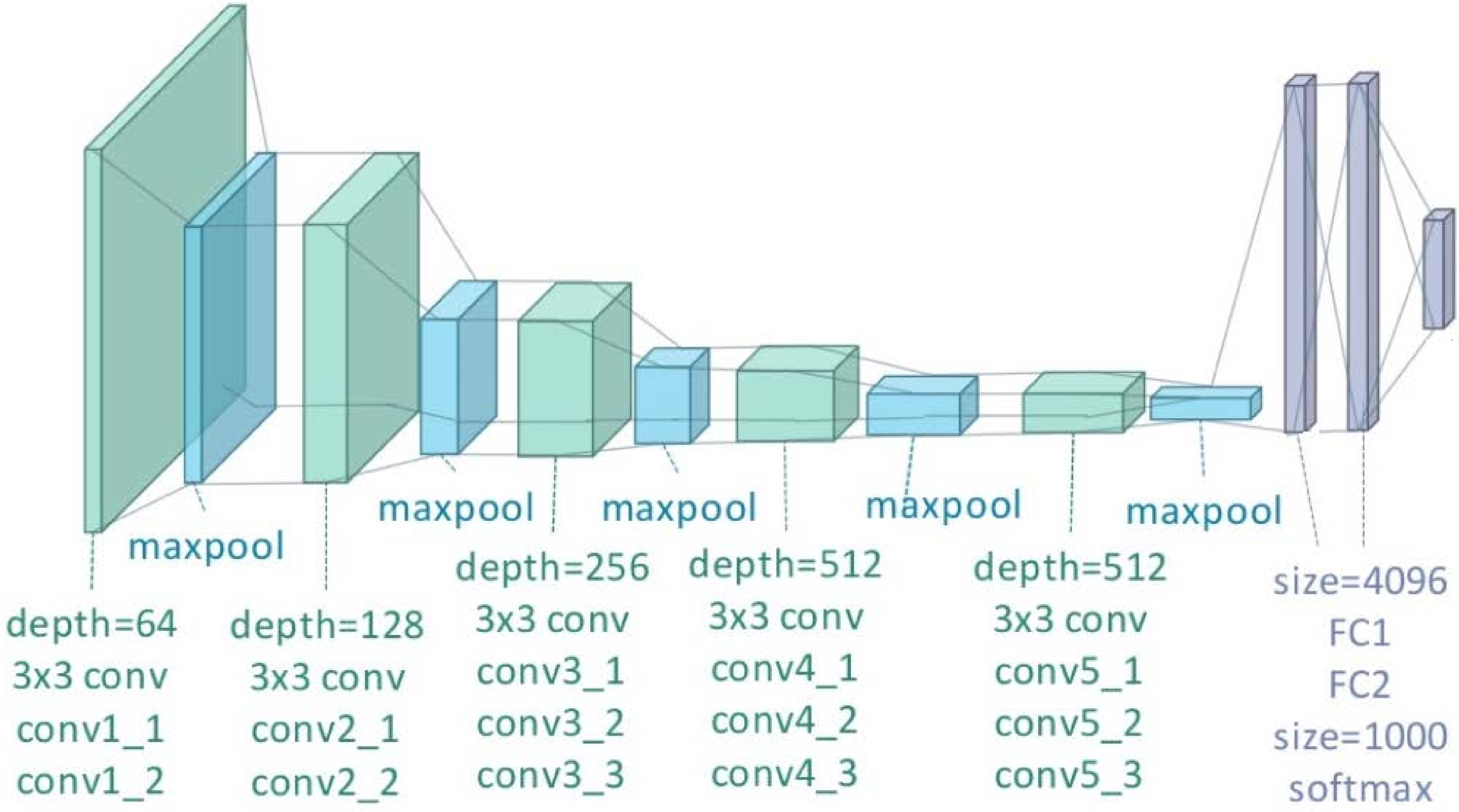
- $I = W \cdot H \cdot D$
- I ：size of input layer
- W ：width of input layer
- H ：height of input layer
- D ：depth or image channels



Original Image **Three Color Channels**

$D = 3$

Deep Learning – VGG16



Transfer Learning

- **Transfer Learning：**
  - Transfer learning is a machine learning technique where a model trained on one task is repurposed on a second related task.
  - Transfer knowledge across tasks, instead of generalizing within a specific task.
  - For example, transfer image recognition knowledge from a cat recognition app to a radiology diagnosis.
  - Speed up and optimize the learning efficiency of the model without learning from zero like most networks.

Results

Implement and Tools

python™

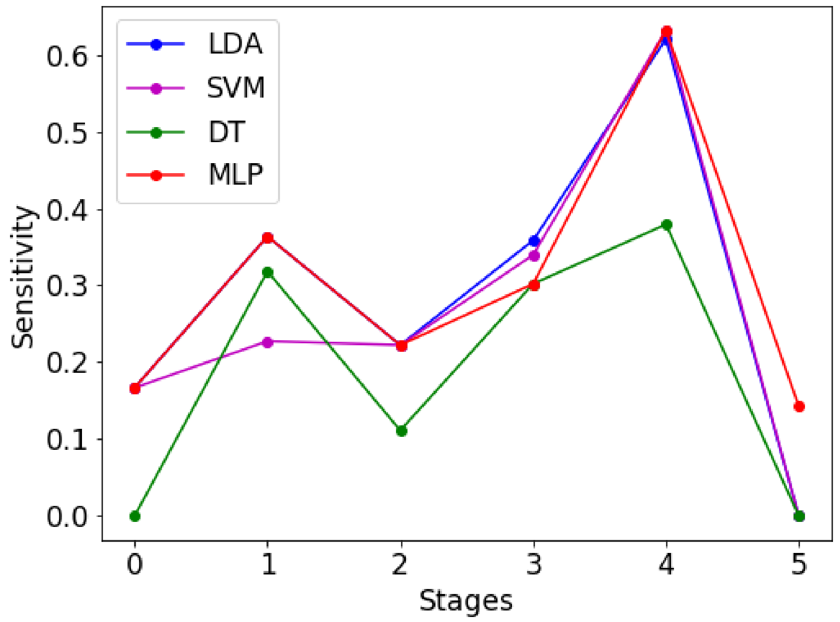
scikits learn
machine learning in Python

Keras + TensorFlow

Traditional Model

Summary

- All models perform well on Stage 4.
- All models didn’t perform well on Stage 0 & Stage 5.
- LDA, SVM and MLP outperformed DT.

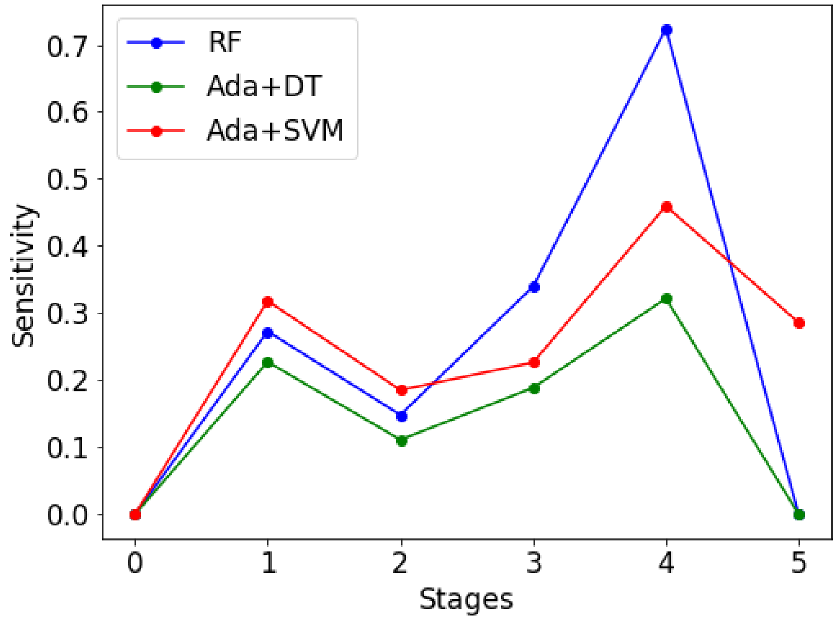


Model	Train	Test						
	Overall Accuracy	Sensitivity						Overall Accuracy
		Normal	I	II	III	IV	V	
LDA	88.27%	16.67% (1)	36.36% (8)	22.22% (6)	35.85% (19)	62.07% (54)	0.00% (0)	43.56% (88)
SVM	99.95%	16.67% (1)	22.73% (5)	22.22% (6)	33.96% (18)	63.22% (55)	0.00% (0)	42.08% (85)
DT	99.86%	0.00% (0)	31.82% (7)	11.11% (3)	30.19% (16)	37.93% (33)	0.00% (0)	29.21% (59)
MLP	100%	16.67% (1)	36.36% (8)	22.22% (6)	30.19% (16)	63.22% (55)	14.29% (1)	43.07% (87)
Total	202	6	22	27	53	87	7	202

Ensemble Model

Summary

- All models performed well on Stage 4.
- All models didn't perform well on Stage 0 & Stage 5.
- Averaging method RF outperformed other two boosting models.



Model	Train	Test						
	Overall Accuracy	Sensitivity						Overall Accuracy
		Normal	I	II	III	IV	V	
RF	100%	0.00% (0)	27.27% (6)	14.81% (4)	33.96% (18)	72.41% (63)	0.00% (0)	45.05% (91)
Ada+DT	100%	0.00% (0)	22.73% (5)	11.11% (3)	18.87% (10)	31.18% (28)	0.00% (0)	22.77% (46)
Ada+SVM	62.56%	0.00% (0)	31.82% (7)	18.52% (5)	22.64% (12)	45.98% (40)	28.57% (2)	32.67% (66)
Total	202	6	22	27	53	87	7	202

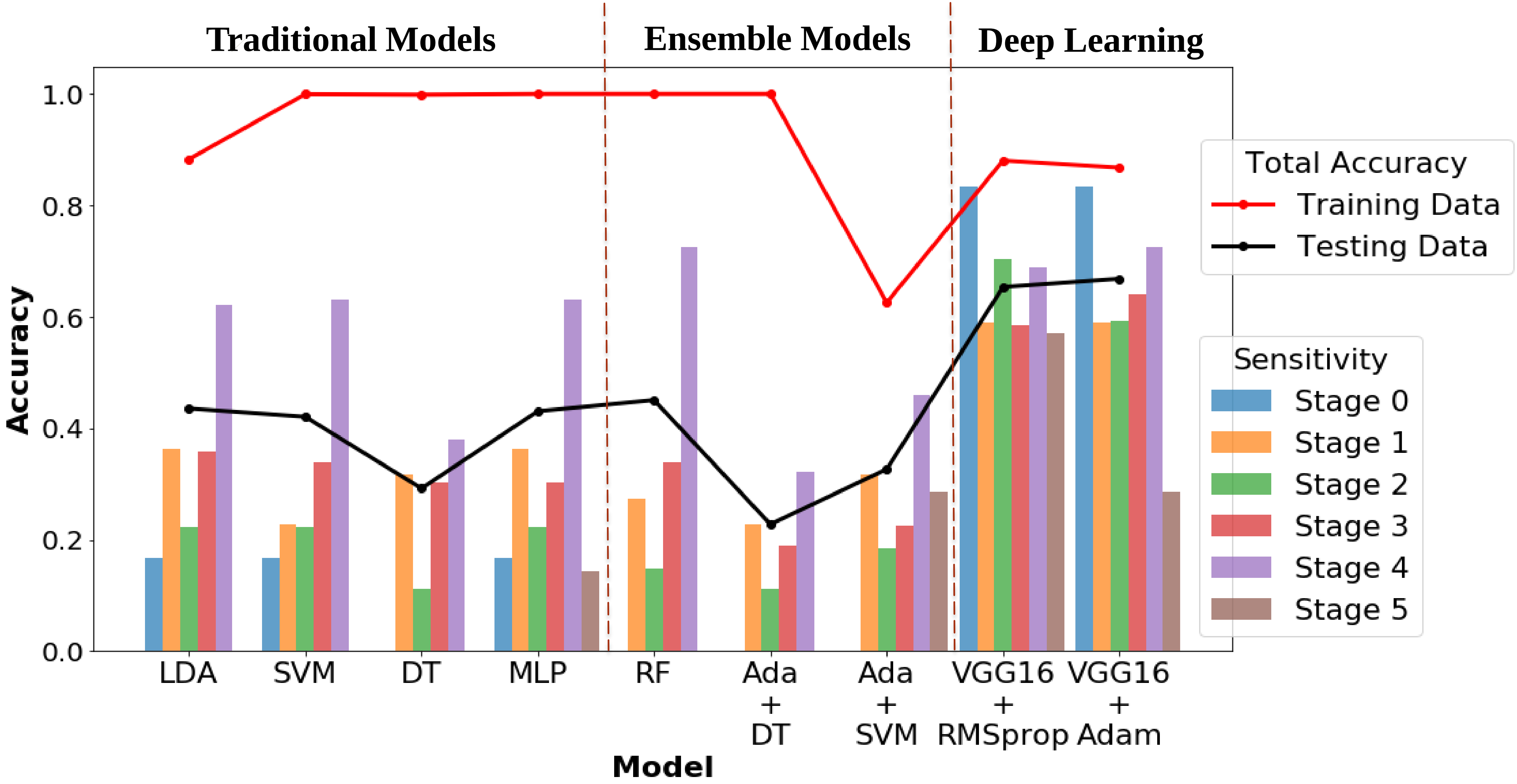
Deep Learning Model

Summary

- Almost every stage is about or higher than 60% accuracy
- Normal cases can be well separated from those whom suffering with PD.
- There are almost no difference between two optimizers.

Model	Train	Test						
	Overall Accuracy	Sensitivity						Overall Accuracy
		Normal	I	II	III	IV	V	
RMSprop	88.04%	83.33% (5)	59.09% (13)	70.37% (19)	58.49% (31)	68.97% (60)	57.14% (4)	65.35% (132)
Adam	86.77%	83.33% (5)	59.09% (13)	59.26% (16)	64.15% (34)	72.41% (63)	28.57% (2)	66.83% (135)
Total	202	6	22	27	53	87	7	202

Summary


Traditional Models
Ensemble Models
Deep Learning
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
LDA
SVM
DT
MLP
RF
Ada + DT
Ada + SVM
VGG16 + RMSprop
VGG16 + Adam
Model
Total Accuracy
Training Data
Testing Data
Sensitivity
Stage 0
Stage 1
Stage 2
Stage 3
Stage 4
Stage 5

Conclusion

Conclusion

- We developed system including a series of methods to deal with the multi-classes classification problem in PD stages.
- This system includes image preprocessing, imbalanced data preprocessing, and three kinds of models: traditional model, ensemble model and deep learning model.
- Overall, VGG16 outperforms other models.
- VGG16 and its related image preprocessing is a useful and better approach to develop multi-classes classification model.
- **Future work：**
  - Take advantage of the whole 3D brain imaging.
  - Investigate other advanced deep learning model, such as VGG19, ResNet50, Xception, Inception etc.

THE END

Thank You