ALMMo Fuzzy Systems and Deep Fuzzy

Evolving Fuzzy Inference Systems (EFIS)

- Fuzzy rule based (FRB) systems designed to learn online while also being robust and capable of adapting to abrupt changes in the data stream (non-stationary environments)
- EFIS algorithms must be computationally lightweight in terms of processing power and memory requirements

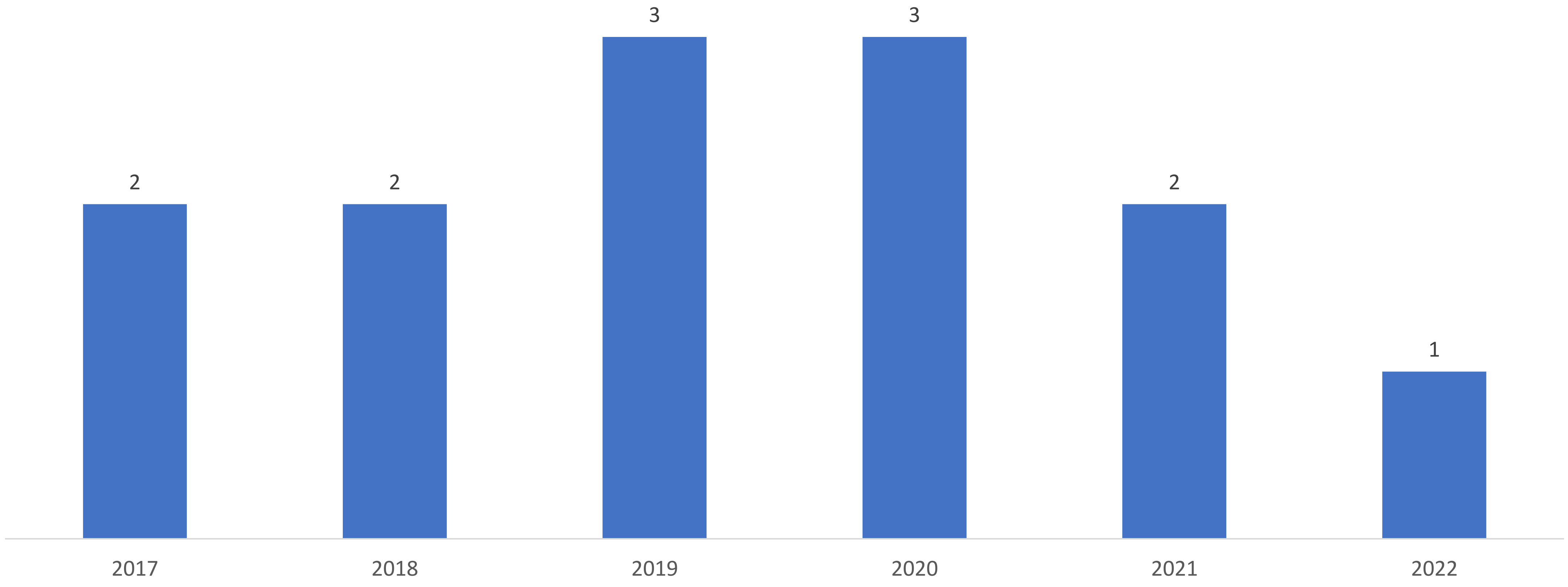
Evolving Fuzzy Inference Systems (EFIS)

- Many different algorithms proposed:
  - DENFIS [Kasabov & Song, 2002]
  - SAFIS [Rong et al. , 2006]
  - eClass [Angelov , 2008]
  - FLEXFIS [Lughofer, 2008]
  - PANFIS [Pratama et al. , 2014]
  - ALMMo [Angelov et al., 2017]
  - SOFIS [Angelov et al., 2018]

ALMMo Systems

- AnYa type fuzzy rule-based system – rules modelled by non-parametric data clouds
- Cloud structure is recursively updated based on density and distance condition-defined heuristics
- Computationally lightweight, simple architecture

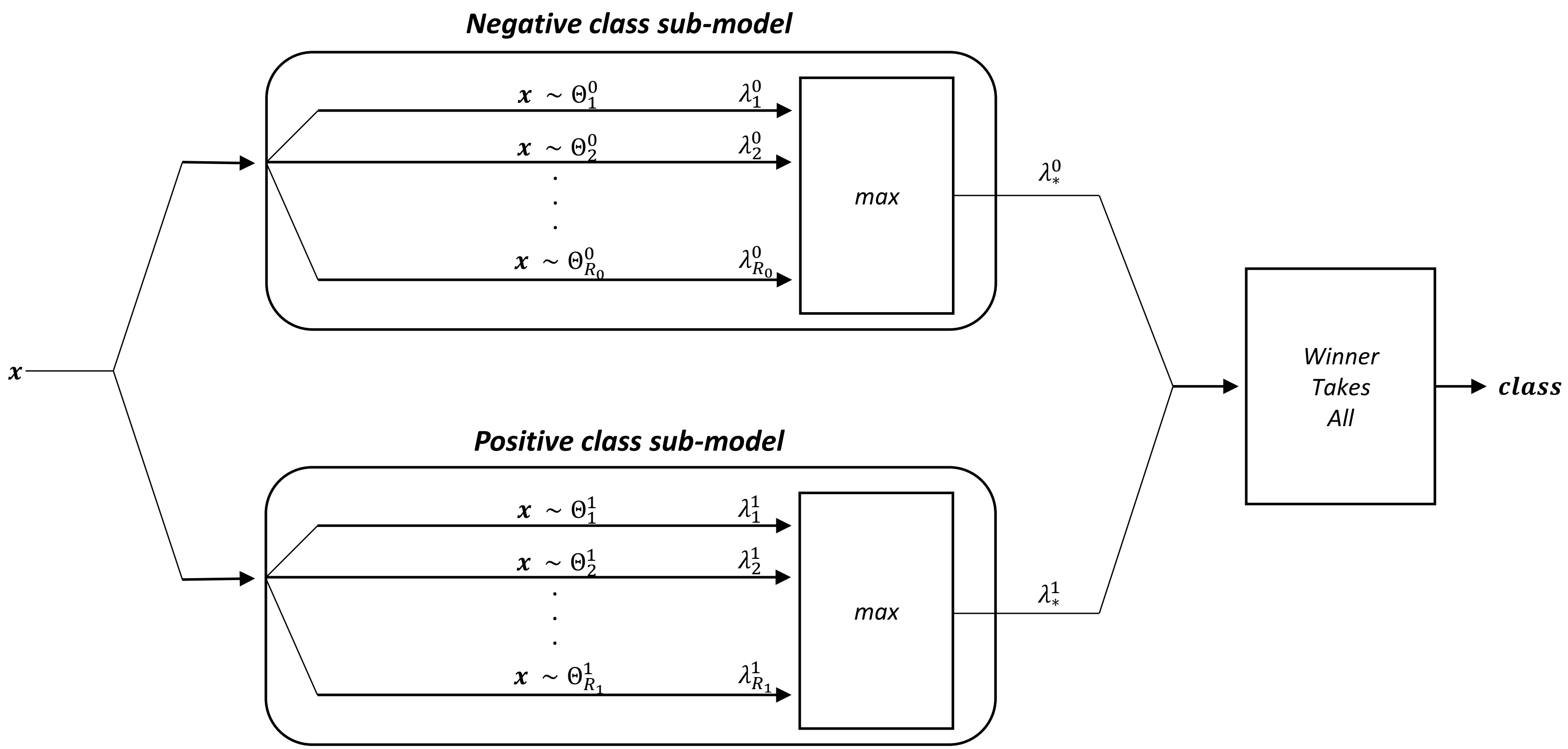
Some statistics



ALMMo Systems

	ALMMo System (ALMMo-1)	ALMMo-0 Classifier (ALMMo-0)
Antecedents	Recursively updated cloud parameters	Recursively updated cloud parameters
Consequents	Linear -> Updated using WRLS	No parameters -> Winner-takes-all approach
Problem Types	Regression, Classification	Classification
Online Cloud Quality Monitoring	Removal of stale clouds that have low utility (importance of a rule relative to the other rules)	No mechanisms introduced -> Mainly applied on non-online problems

ALMMo-0


Negative class sub-model
$x \sim \Theta_1^0$
λ_1^0
$x \sim \Theta_2^0$
λ_2^0
$x \sim \Theta_{R_0}^0$
$\lambda_{R_0}^0$
max
λ_*^0
x
Positive class sub-model
$x \sim \Theta_1^1$
λ_1^1
$x \sim \Theta_2^1$
λ_2^1
$x \sim \Theta_{R_1}^1$
$\lambda_{R_1}^1$
max
λ_*^1
Winner
Takes
All
class

ALMMo-0

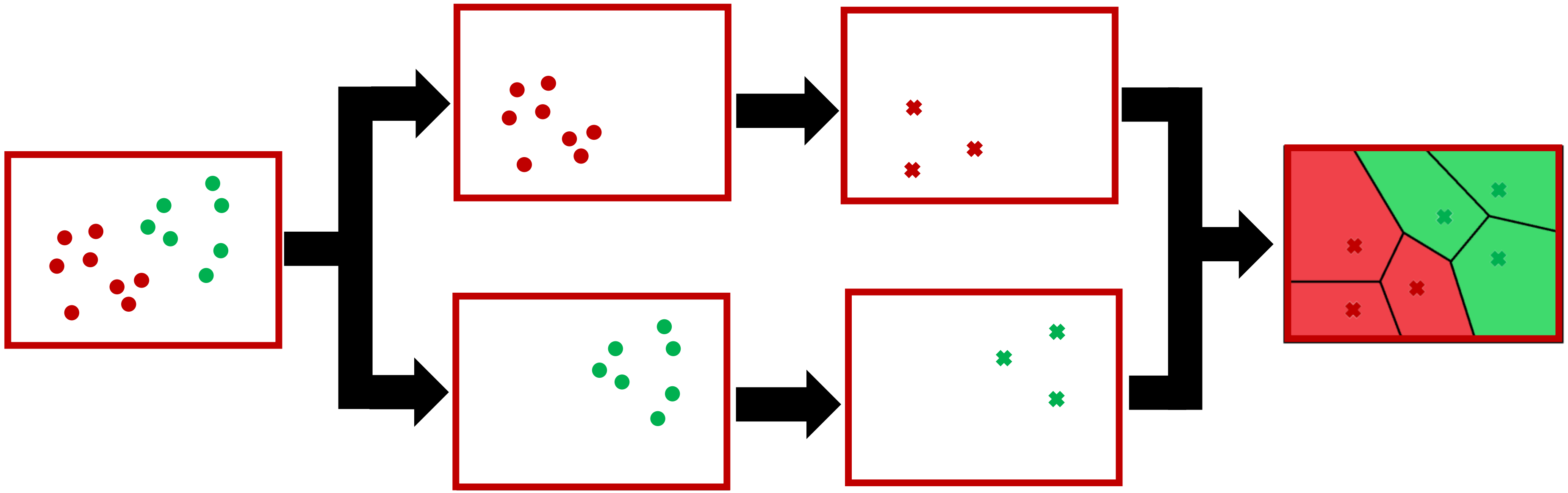
- Winner-takes-all approach for deciding on the class:

$$\textbf{IF } x \sim f_j^i \textbf{ THEN } Label^i$$

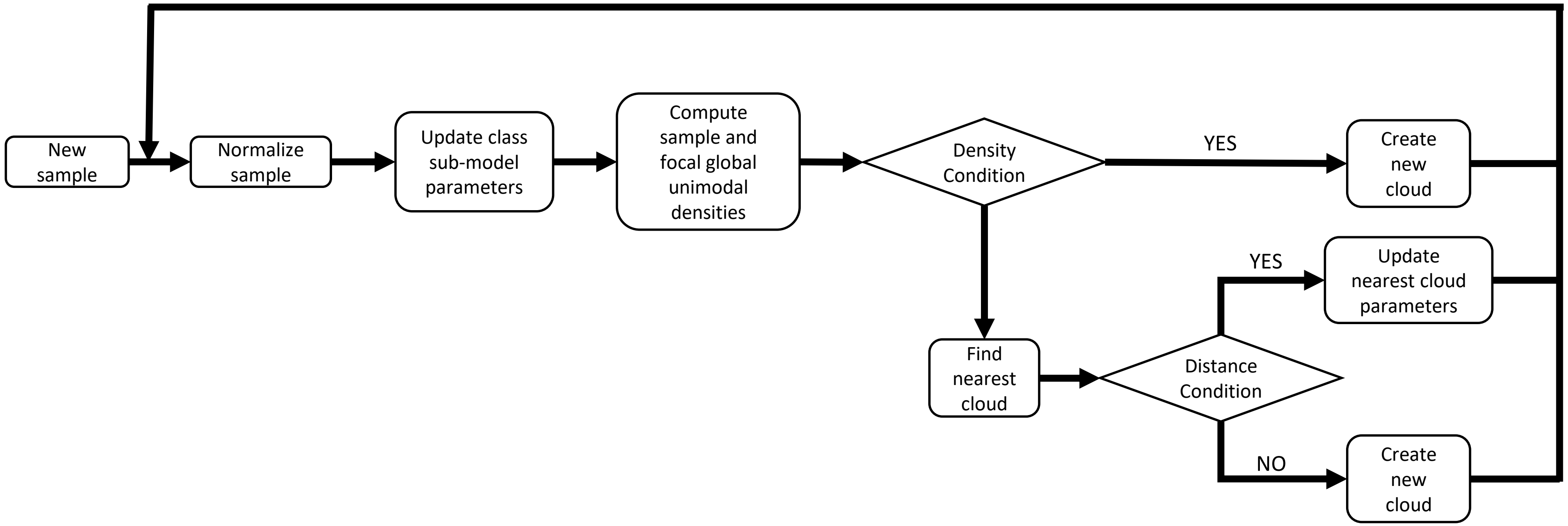
$$\lambda_j^i = \exp -\frac{1}{2} \left\| x - f_j^i \right\|^2$$

$$\hat{y} = \underset{i=1,2,\ldots,L}{\arg\max} \left(\lambda_{j*}^i \right)$$

ALMMo-0



ALMMo-0 – Training algorithm



ALMMo-1

$$\textbf{IF} \;\; x \sim \Theta_i \;\; \textbf{THEN} \;\; y_i = \boxed{u^T} \cdot a_i$$

$$u = [1, \quad x]^T$$

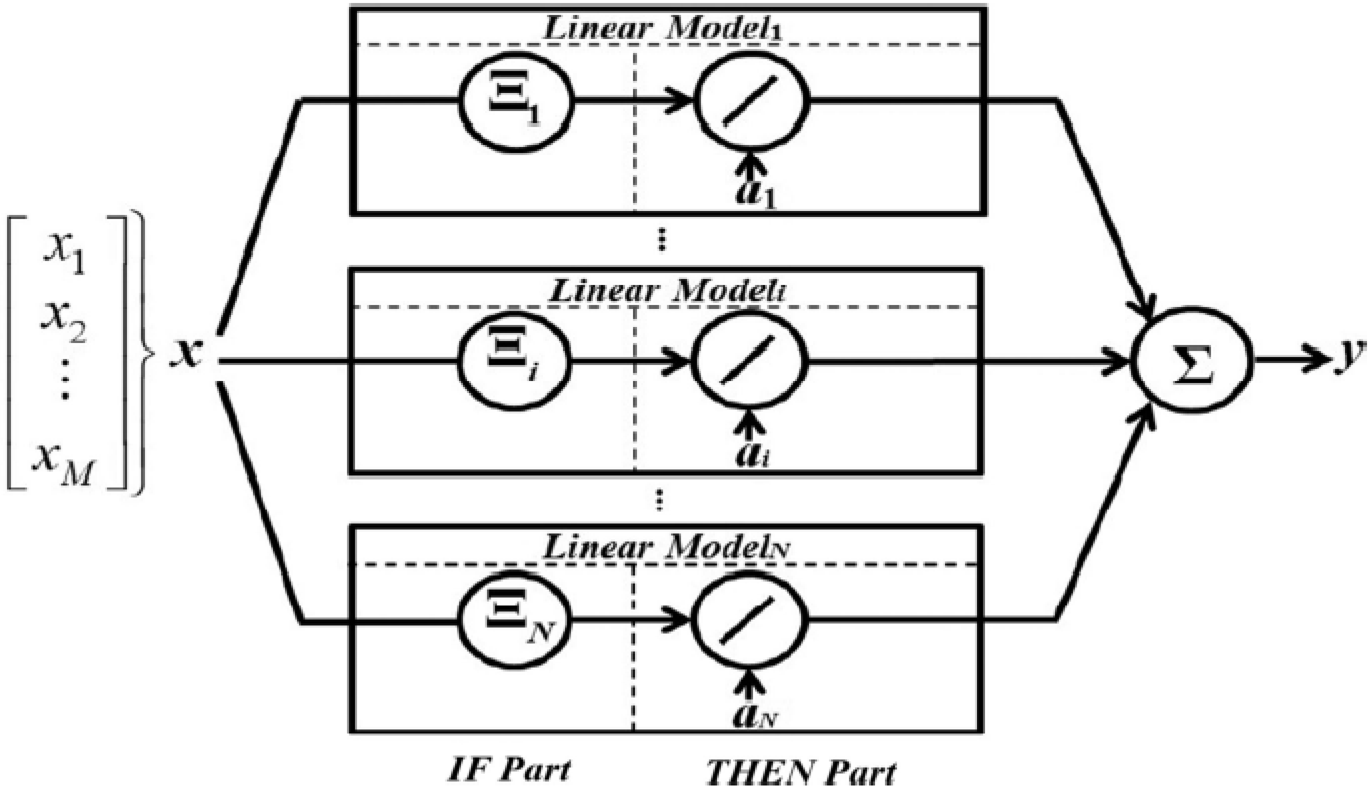
Regression:

$$y = \sum_{j=1}^{N} (y_i \lambda_i)$$

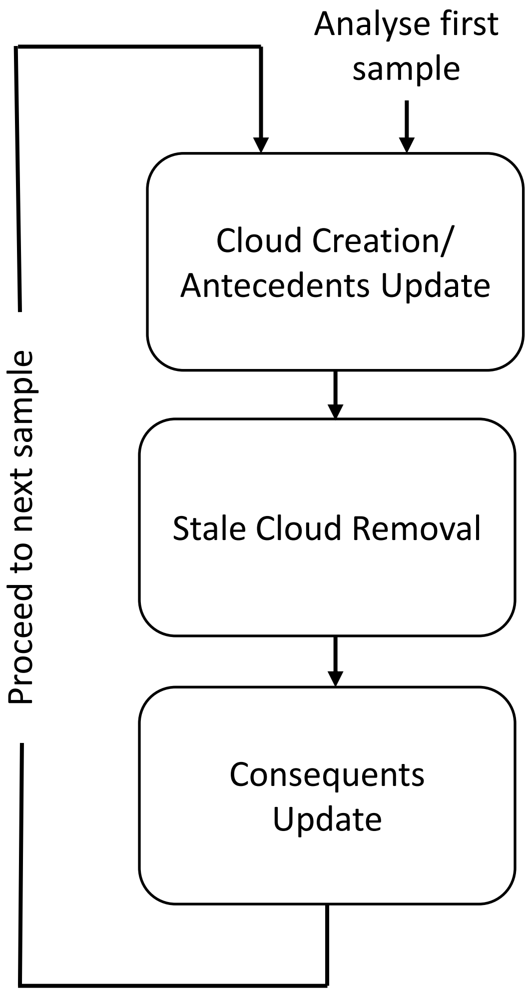
Classification

$$y_{bc} = \begin{cases} 1, y_r \geq 0.5 \\ 0, y_r < 0.5 \end{cases}$$

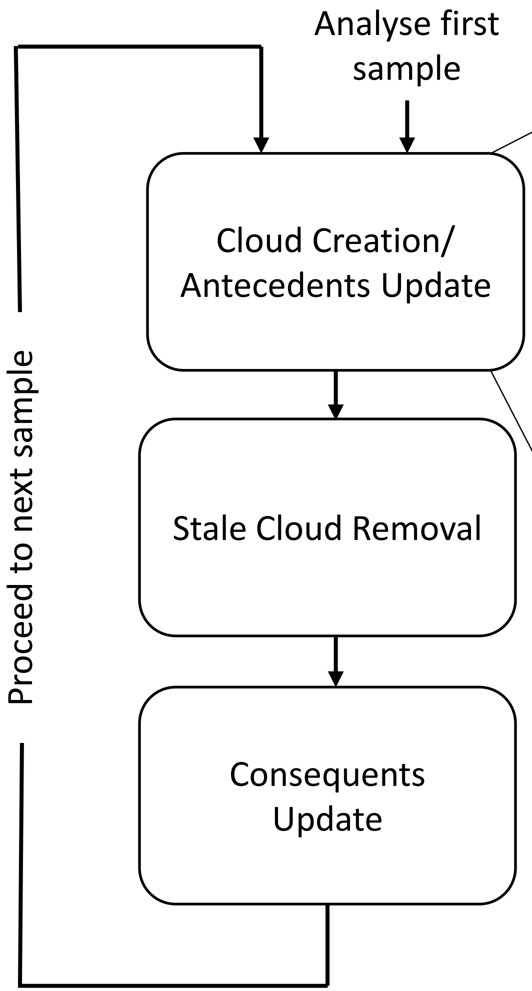
ALMMo-1


Linear Model1
Linear Modeli
Linear ModelN
Ξ_1
Ξ_i
Ξ_N
a_1
a_i
a_N
x_1
x_2
x_M
x
Σ
y
IF Part
THEN Part

ALMMo-1 – Training Algorithm



ALMMo-1 – Training Algorithm



Function:

Make clouds, match the data being fed

Methodology:

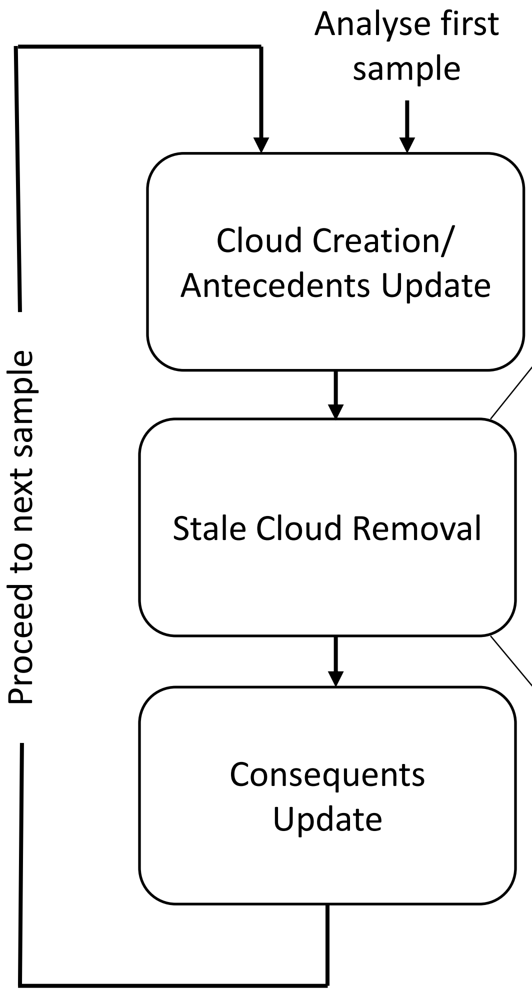
IF no cloud is nearby:

Create a new cloud and initialize the respective rule

Else:

Update the antecedents of one existing rules

Training Algorithm



Function:

Removes clouds with low utility η_i

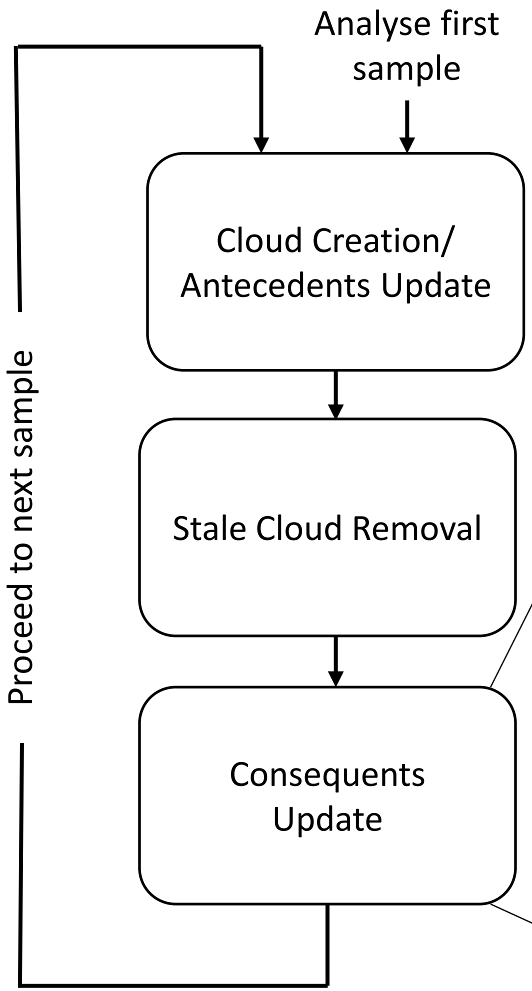
Utility Update:

Every existing cloud gets utility update on every iteration

Removal criteria:

$$\eta_i < \eta_0$$

ALMMo-1 – Training Algorithm



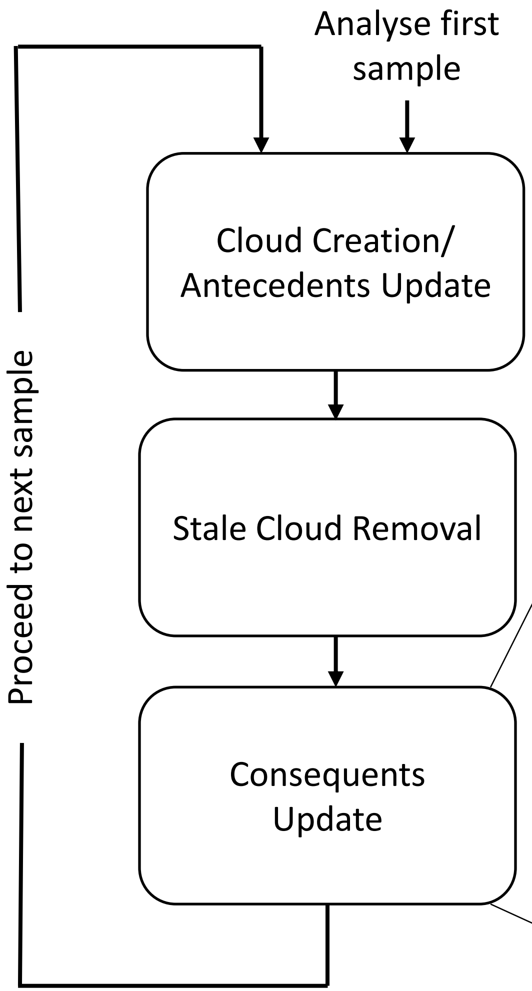
Function:

Updates the consequents of every rule

$$C_i \leftarrow C_i - \frac{\lambda_i C_i u_j u_j^T C_i}{1 + \lambda_i u_j^T C_i u_j}$$

$$a_i \leftarrow a_i + \lambda_i C_i u_j (y_j - u_j^T a_i)$$

ALMMo-1 – Training Algorithm



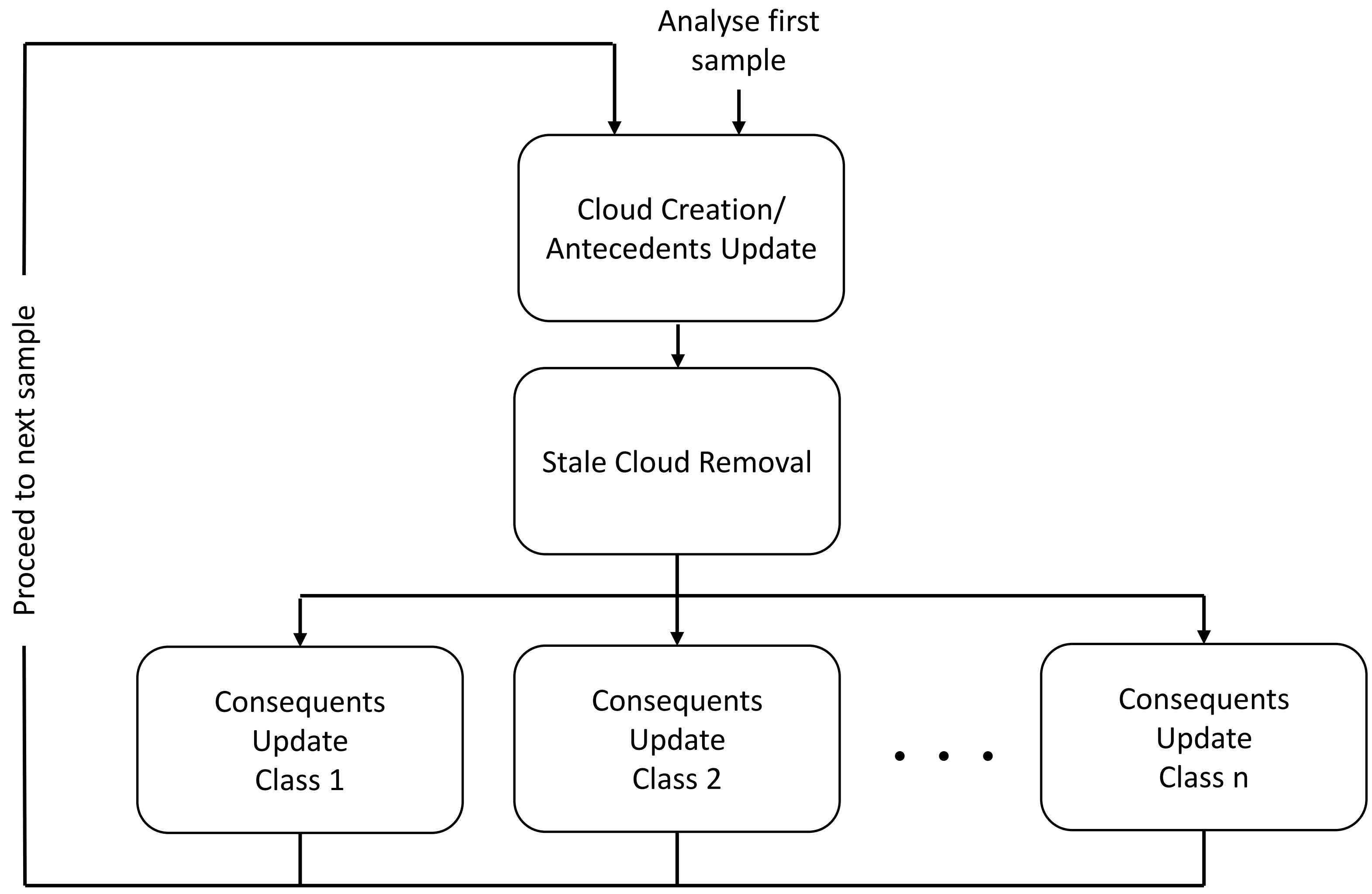
Function:

Updates the consequents of every rule

$$C_i \leftarrow C_i - \frac{\lambda_i C_i u_j u_j^T C_i}{1 + \lambda_i u_j^T C_i u_j}$$

$$a_i \leftarrow a_i + \lambda_i C_i u_j (y_j - u_j^T a_i)$$

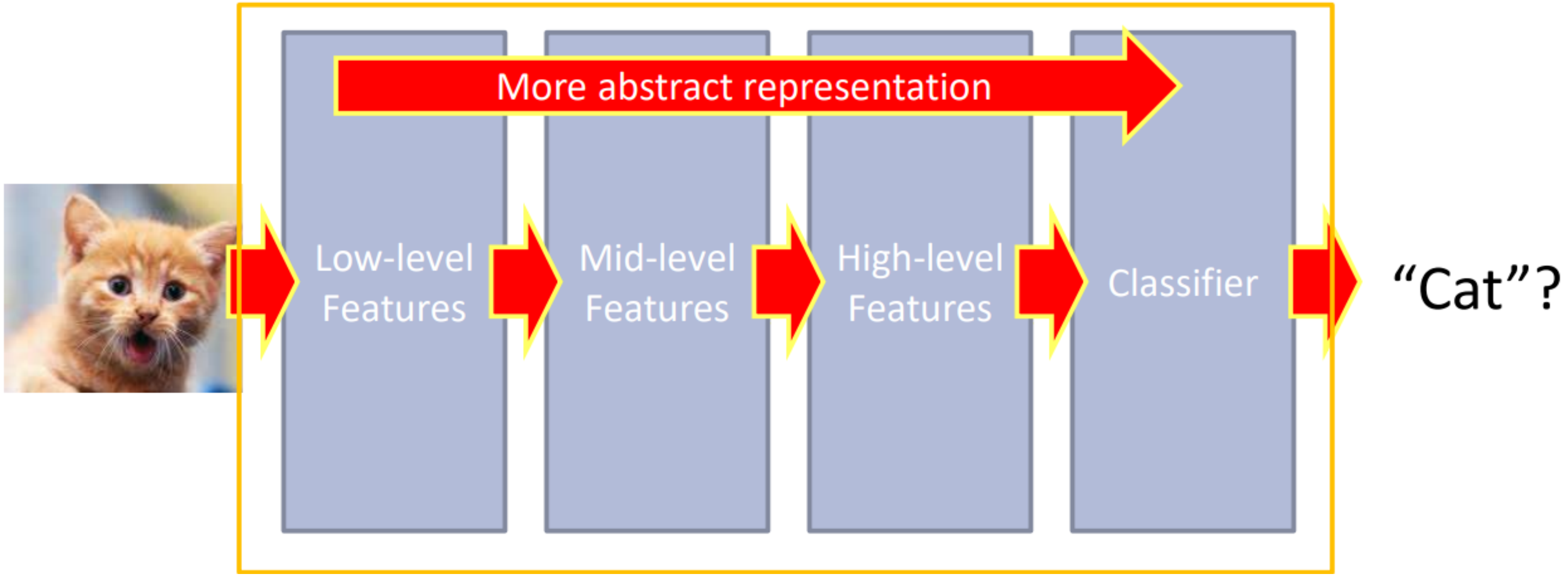
ALMMo-1 Multi Class



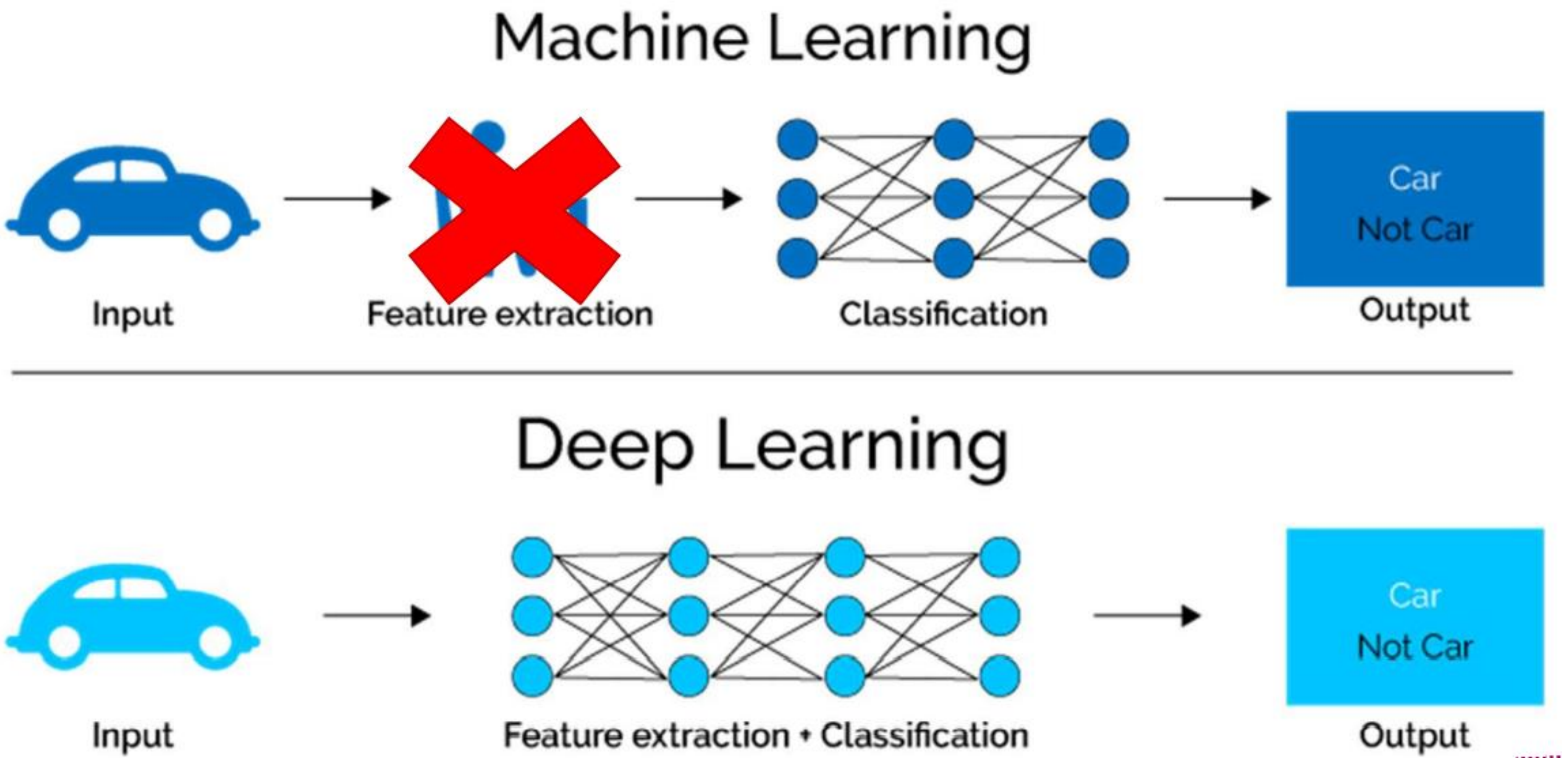
Ensemble and Multi-Layer Architectures

- Highly dimensional and non-stationary problems (such as image recognition and financial time series forecast) are generally too complex for traditional machine learning methods, as well as ALMMo systems

Ensemble and Multi-Layer Architectures



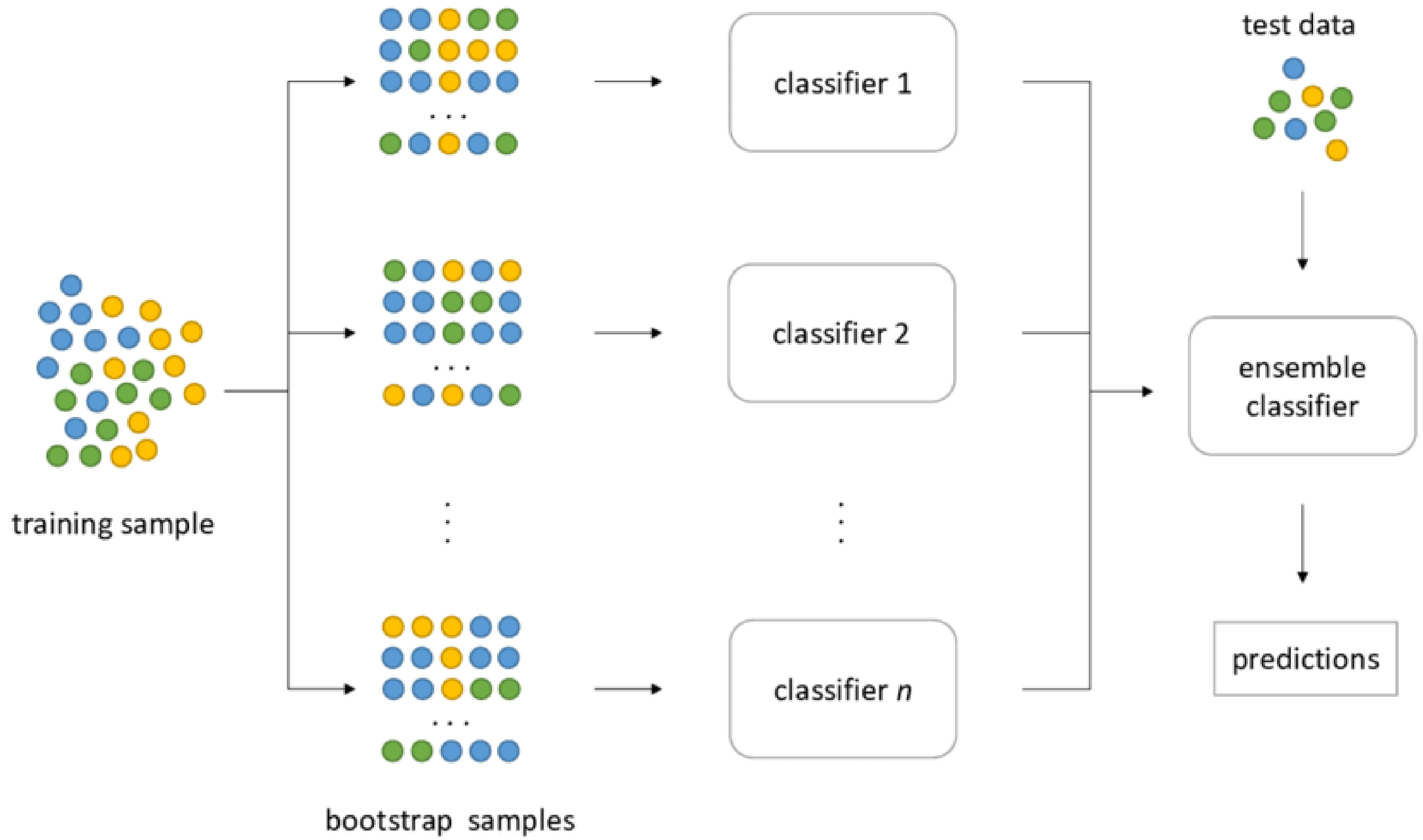
Ensemble and Multi-Layer Architectures



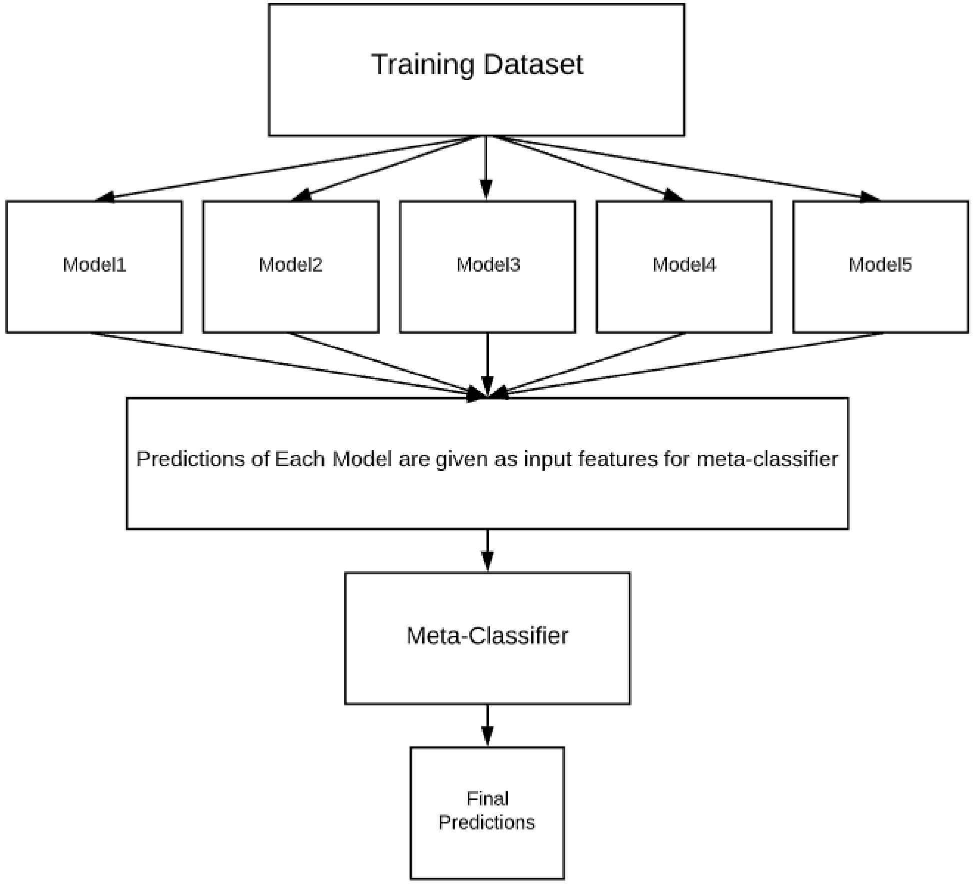
Ensemble and Multi-Layer Architectures

- Simplicity of ALMMo systems and other EFIS makes them suitable as the building blocks of multi-layer ensemble architectures

Ensemble and Multi-Layer Architectures



Ensemble and Multi-Layer Architectures



Ensemble and Multi-Layer Architectures

- Such architectures can achieve comparable or even better performances than state-of-the-art methods on problems such as image recognition and time series forecast, while remaining transparent

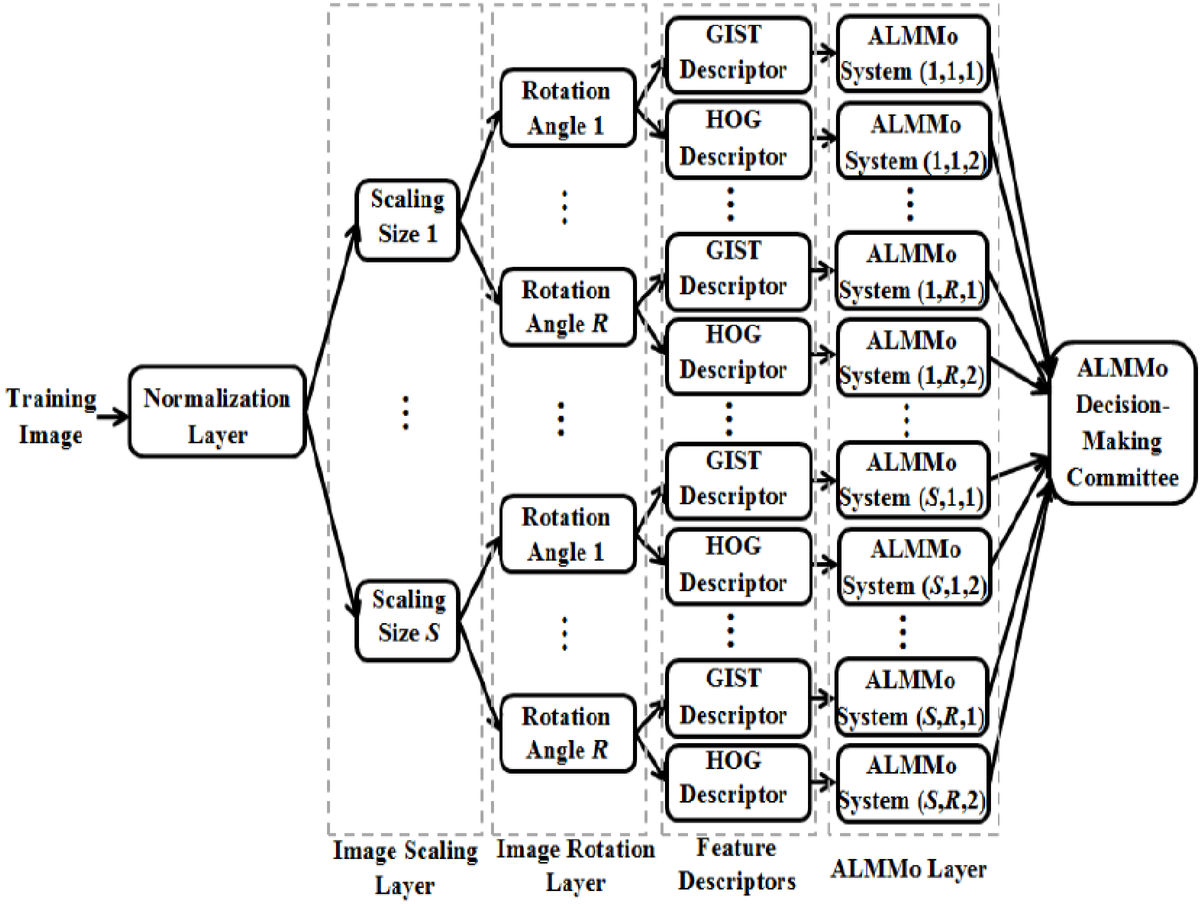
MICE

- Multi-Layer Multi-Model Images Classifier Ensemble [Angelov, 2017]
- Fast deep learning approach network for handwriting recognition
- Interpretable structure

MICE

- Image transformation layers (Normalization, Scaling, Rotation)
- Global feature descriptors (GIST, HOG)

MICE



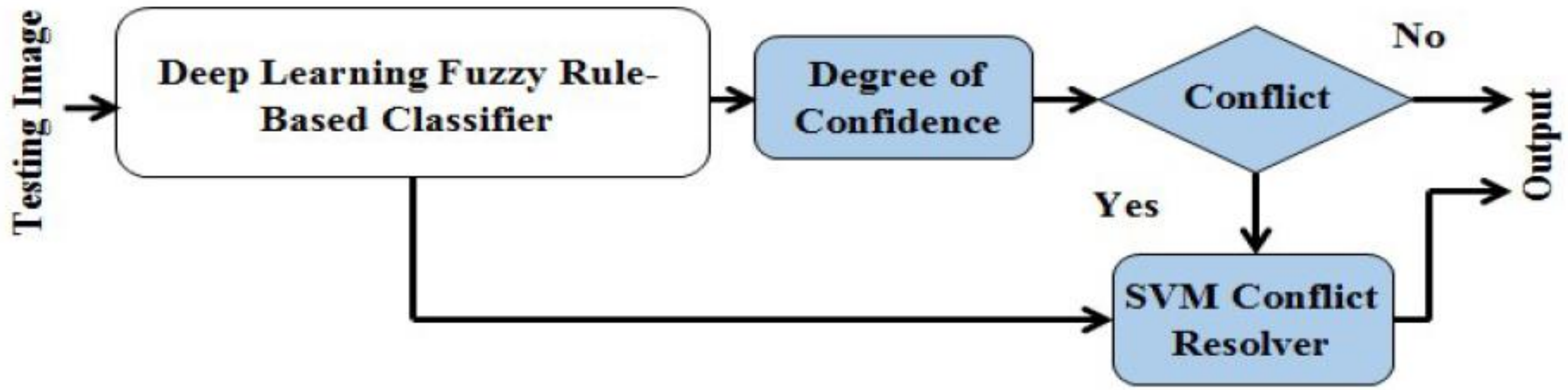
(a) The structure of the training process

MICE

Approaches	Accuracy	Training Time	PC Parameters
The Proposed MICE approach (this paper)	99.32%	*i.* Less than 1 minute per class for each member of the ALMMo committee with GIST features; *ii.* Less than 4 minutes per class for each member of the ALMMo committee with HOG features.	Core i7-4790 (3.60GHz), 16 GB DDR3
Committee of 7 Convolutional Neural Networks [3]	99.73% ± 2%	Almost 14 hours for each one of the 5 DNNs.	Core i7-920 (2.66GHz), 12 GB DDR3
Committee of 35 Convolutional Neural Networks [4]	99.77%	Almost 14 hours for each one of the 35 DNNs.	Core i7-920 (2.66GHz), 12 GB DDR3

MICE (SVM Conflict Resolver)

- Cascade of the MICE classifier and an auxiliary SVM for conflict resolution [Angelov, 2018]



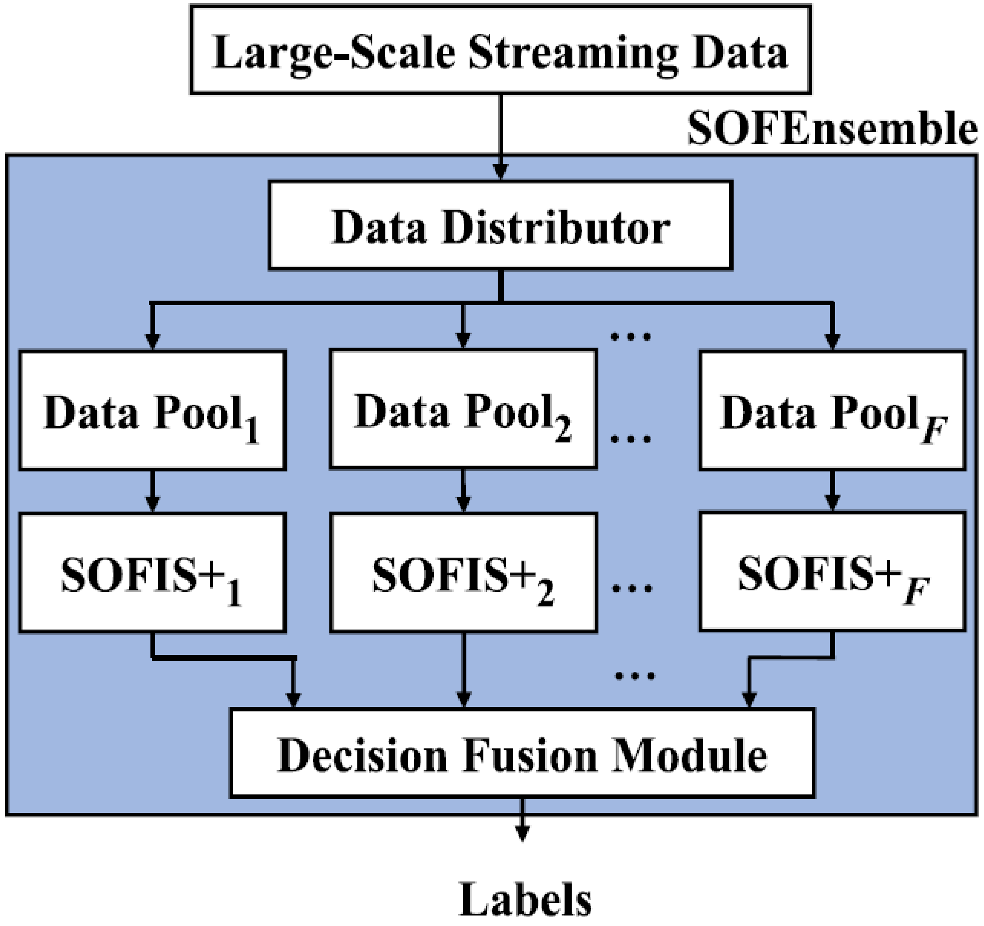
MICE (SVM Conflict Resolver)

Approaches	Accuracy
The Proposed Approach	99.55%
DLFRB Classifier	99.44%

SOFEmsemble

- Self Organizing Fuzzy Inference Ensemble System [Angelov & Gu, 2021]
- Big streaming data classification
- Uses SOFIS (Self Organizing Fuzzy Inference System) as the building blocks for the ensemble architecture

SOFEmsemble



SOFEmsemble

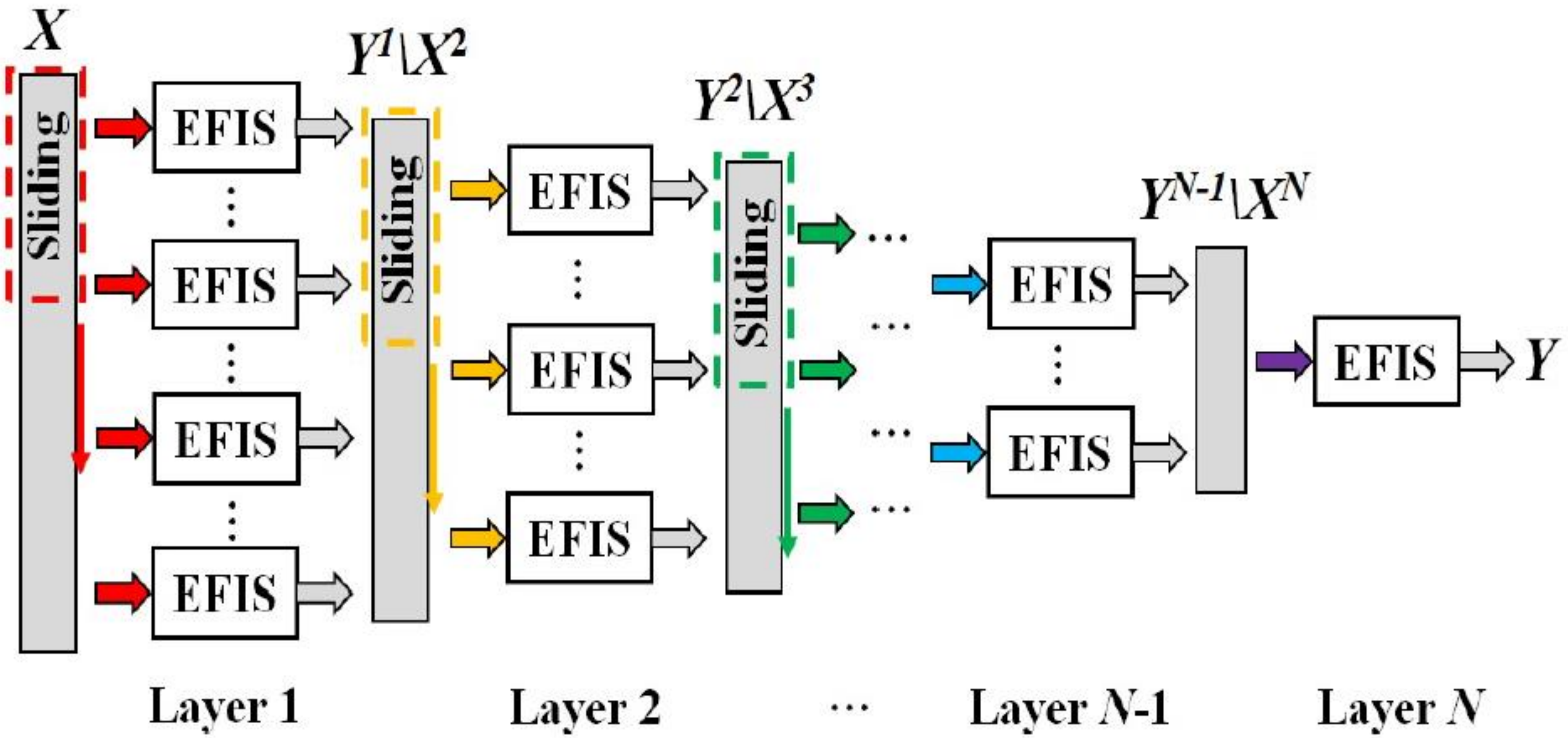
Algorithm	MNIST	FMNIST
SOFEnsemble	**0.9918**	**0.9095**
DRB	0.9914	0.9004
DCNN	0.9913	0.9078

Dataset	FC		MNIST		FMNIST	
Algorithm	*Acc*	t_{exe}	*Acc*	t_{exe}	*Acc*	t_{exe}
SOFEnsemble	0.9173	192	**0.9869**	53	**0.9017**	54
SOFIS	0.8778	1757	0.9866	956	0.8868	1085
ESAFIS	0.7359	5480	0.9830	34612	0.8962	29388
eClass0	0.3456	74	0.8386	100	0.7351	95
eClass1	0.3647	374	0.9764	10769	0.8878	10786
ALMMo0	0.8932	1717	0.9864	462	0.8882	470
ALMMo	0.7012	160	0.9748	5627	0.8735	4881
KNN	0.9107	–	0.9852	–	0.8966	–
SVM	0.7247	4641	0.9857	153	0.8936	178
DT	0.9180	38	0.9010	196	0.8088	153
RF	**0.9591**	715	0.9631	4191	0.8872	4501
SC	0.8472	17137	0.9762	1111	0.8817	1139
MLP	0.7667	214	0.6011	80	0.8289	102
ELM	0.6481	**14**	0.9424	**3**	0.8465	**3**
HP	0.9069	545	0.9864	76	0.8845	99

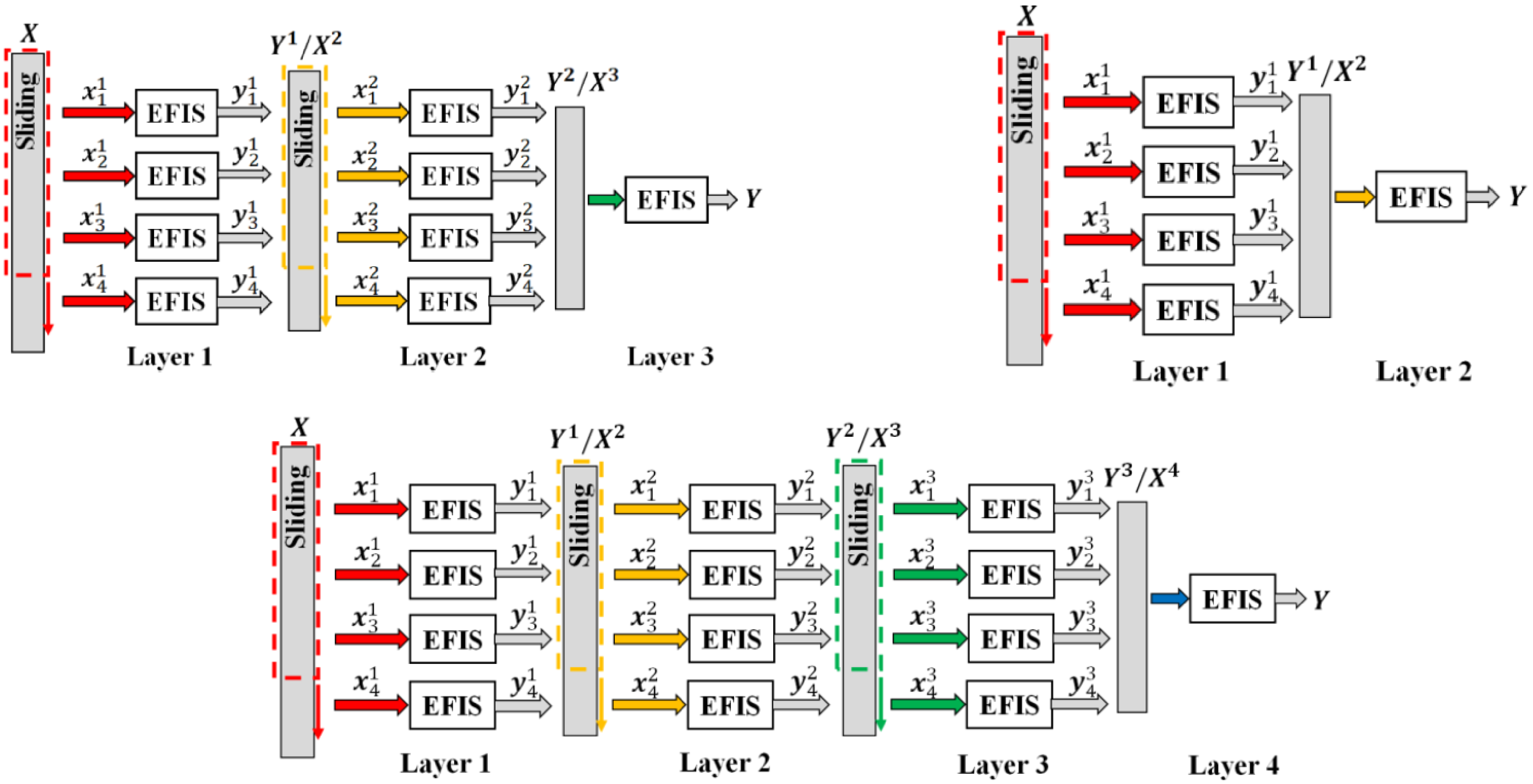
MEEFIS

- Multi-Layer Ensemble Evolving Fuzzy Inference System [Gu, 2020]
- Ensemble of multiple-input multiple-output (MIMO) first-order evolving fuzzy inference system (EFIS) organized in a multi-layered architecture
- General architecture

MEEFIS


X
Sliding
EFIS
EFIS
EFIS
EFIS
Y1\X2
Sliding
EFIS
EFIS
EFIS
Y2\X3
Sliding
EFIS
EFIS
YN-1\XN
EFIS
Y
Layer 1
Layer 2
Layer N-1
Layer N

MEEFIS



X
Sliding
x_1^1 x_2^1 x_3^1 x_4^1
EFIS
y_1^1 y_2^1 y_3^1 y_4^1
Y^1/X^2
x_1^2 x_2^2 x_3^2 x_4^2
y_1^2 y_2^2 y_3^2 y_4^2
Y^2/X^3
x_1^3 x_2^3 x_3^3 x_4^3
y_1^3 y_2^3 y_3^3 y_4^3
Y^3/X^4
Y
Layer 1
Layer 2
Layer 3
Layer 4

MEEFIS

Algorithm	MNIST (#Training Images)						Fashion MNIST (#Training Images)					
	10000	**20000**	**30000**	**40000**	**50000**	**60000**	**10000**	**20000**	**30000**	**40000**	**50000**	**60000**
MEEFIS ($\eta_0 = 0.1$)	0.9742	0.9745	0.9746	0.9748	0.9751	0.9754	0.8765	0.8786	0.8797	0.8801	0.8799	0.8810
MEEFIS ($\eta_0 = 0$)	**0.9756**	**0.9765**	**0.9773**	**0.9775**	**0.9778**	**0.9788**	**0.8834**	**0.8881**	**0.8896**	**0.8905**	**0.8913**	**0.8916**
AdaBo	0.9446	0.9447	0.9446	0.9446	0.9454	0.9451	0.8178	0.8187	0.8190	0.8196	0.8199	0.8192
RanFor	0.9477	0.9559	0.9602	0.9606	0.9626	0.9631	0.8638	0.8733	0.8778	0.8816	0.8847	0.8872
MLP	0.4543	0.4717	0.6232	0.5848	0.6351	0.5650	0.6304	0.7451	0.8028	0.7739	0.8045	0.8318
eClass0	0.8470	0.8556	0.8512	0.8603	0.8551	0.8500	0.7289	0.7303	0.7147	0.7497	0.7233	0.7167
ALMMo	0.9676	0.9695	0.9707	0.9715	0.9722	0.9723	0.8666	0.8694	0.8696	0.8699	0.8702	0.8705

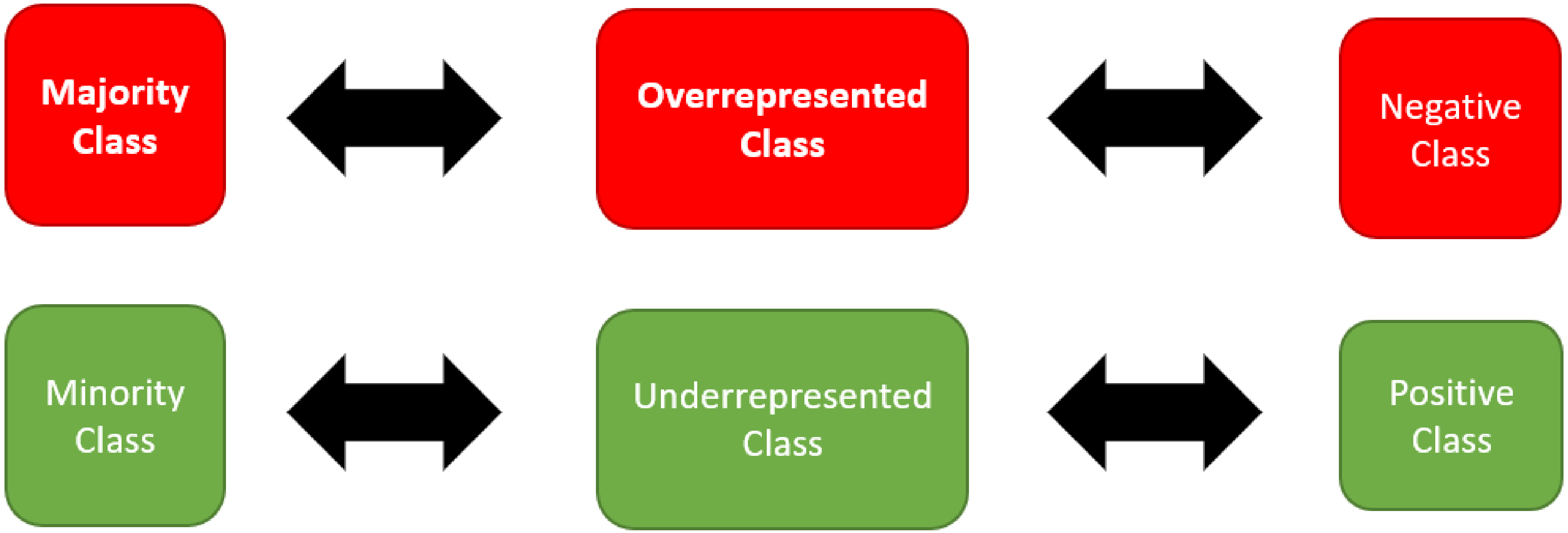
Algorithm	$NDEI$	t_{exe}	*#rules*
MEEFIS	**0.1392**	2.92	74 (8.2)
OS-Fuzzy-ELM	0.2991	0.93	5
CEFNS	0.2635	0.44	5
SAFIS [24]	0.38	/	6
ESAFIS	0.2955	5.83	6
eTS	0.3805	/	9
ALMMo	0.4437	0.46	7
SB-ALMMo	0.4402	0.86	4
GENEFIS (C) [28]	0.280	4.46	19
GENEFIS (B) [28]	0.339	3.02	9
LEOA [44]	0.2480	144.78	42

Algorithm	$NDEI$	*#rules*
MEEFIS	**0.0124**	85 (9.4)
EFIS	0.0147	15
ALMMo	0.0149	7
PANFIS [27]	0.09	4
GENEFIS [28]	0.07	2
LEOA [44]	0.1229	52
SEFS [19]	0.0182	2
EFS-SLAT [45]	0.0156	23

Imbalanced Datasets

- Datasets in which some classes (minority classes) are significantly under-represented
- Most traditional machine learning methods show a large bias towards the majority classes

Imbalanced Datasets



Imbalanced Datasets

		Prediction	
		N	P
True	N	TN	FP
	P	FN	TP

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$F_1 = 2\frac{Precision \times Recall}{Precision + Recall}$$

Imbalanced Datasets

		Prediction	
		N	P
True	N	990	0
	P	8	2

1000 patients: 990 healthy, 10 sick (1% class imbalance)

Accuracy	99.2 %
Precision	100 %
Recall	20 %
F1-Score	33.3 %

Imbalanced Datasets

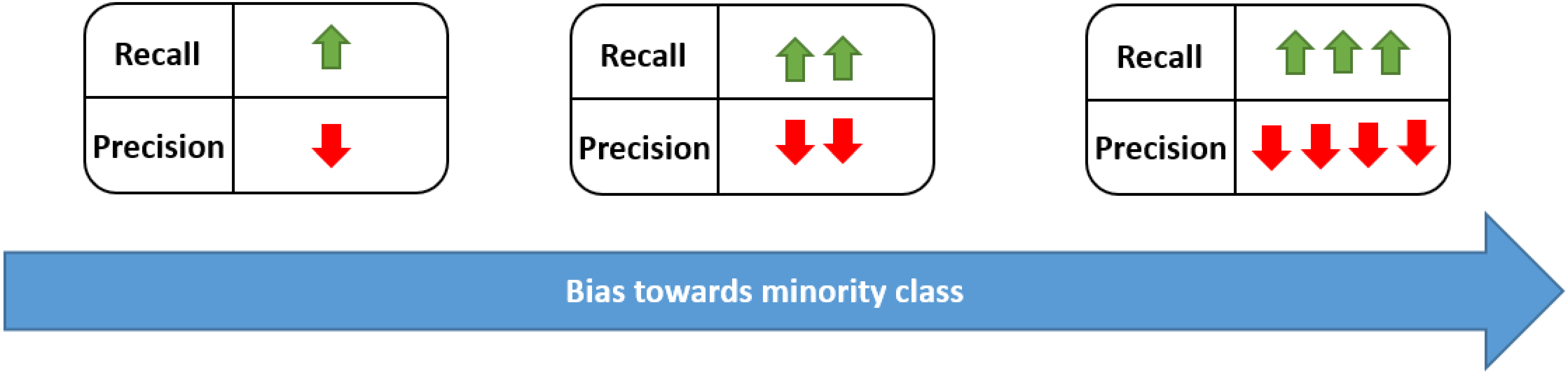
		Prediction N	Prediction P
True	N	TN	FP
True	P	FN	TP

$$GM = \sqrt{Recall \times Specificity}$$

$$Kappa = \frac{2(TP \times TN - FN \times FP)}{(TP + FP)(FP + TN) + (TP + FN)(FN + TN)}$$

$$Matthews = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

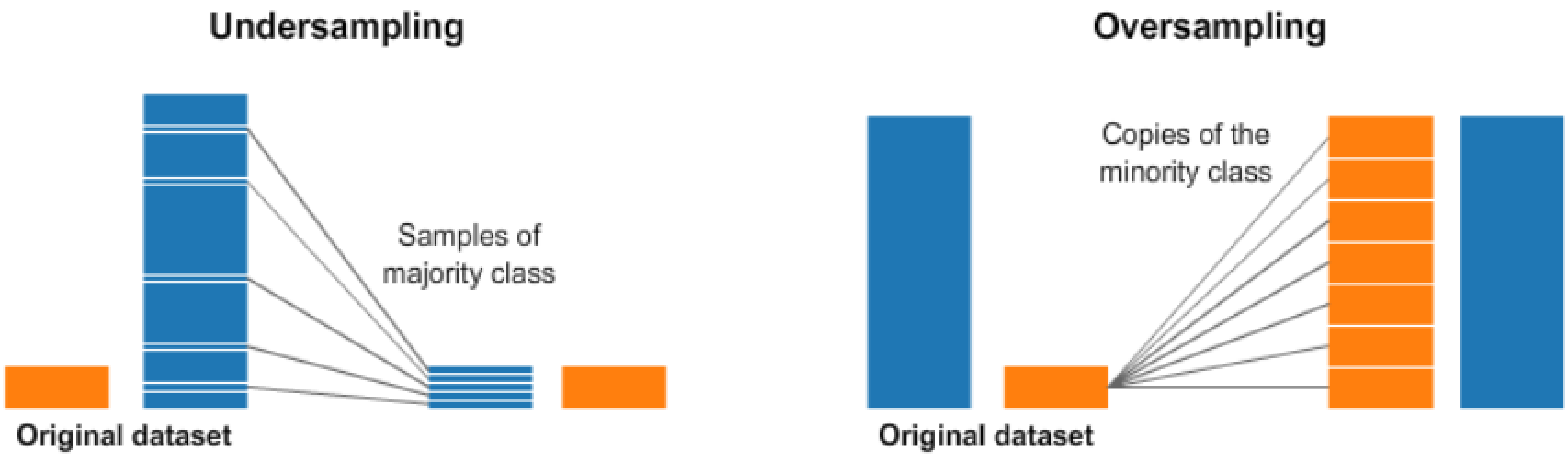
Imbalanced Datasets



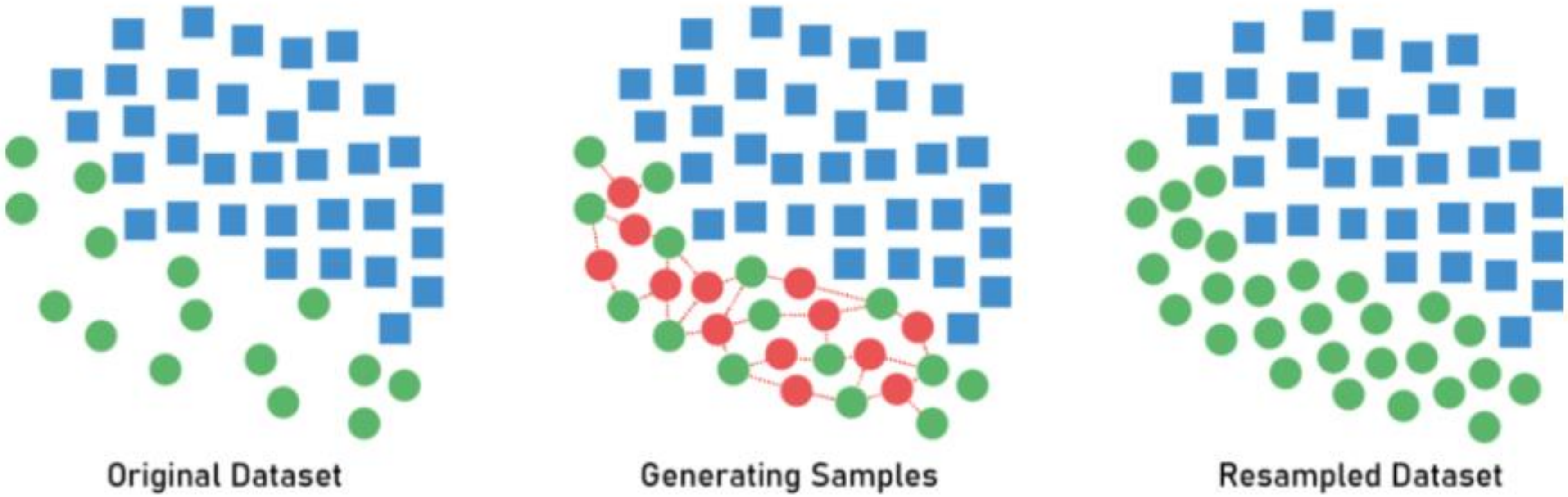
Imbalanced Datasets

- Data-based approaches:
  - Random resampling
  - Synthetic data generation
  - Feature selection
  - …

Imbalanced Datasets



Imbalanced Datasets



Imbalanced Datasets

- Model-based approaches:
  - Cost sensitive learning
  - One class classification
  - Class decision threshold tuning
  - …

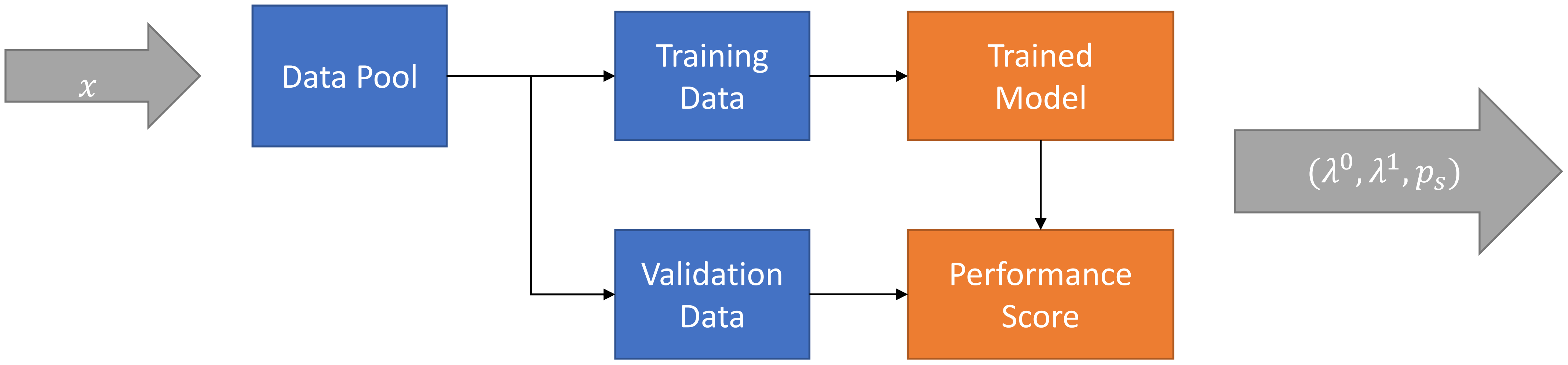
ALMMo Systems and Imbalanced Datasets

- ALMMo-0: classes trained separately, but larger number of majority class clouds that may capture some of the minority class samples (false negatives)
- ALMMo-1: regressor, large bias towards the majority class, threshold tuning and other cost sensitive approaches

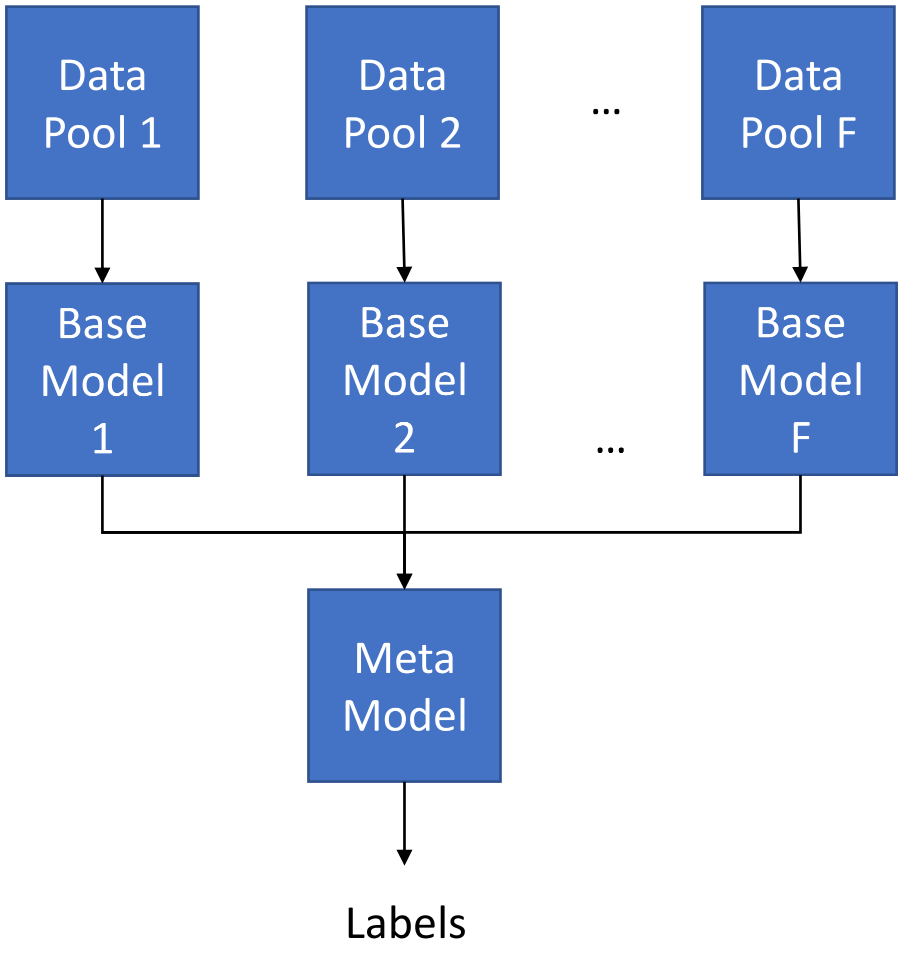
ALMMo-0-M

- Stack ensemble / meta-learning approach based on the ALMMo-0 classifier
- Train multiple base models and evaluate their performance using adequate classification metrics
- Use the base models class confidence predictions together with their performance scores to train a final meta-model that predicts the sample class

ALMMo-0-M


x
Data Pool
Training Data
Validation Data
Trained Model
Performance Score
$(\lambda^0, \lambda^1, p_s)$

ALMMo-0-M



Data
Pool 1
Data
Pool 2
...
Data
Pool F
Base
Model
1
Base
Model
2
...
Base
Model
F
Meta
Model
Labels

ALMMo-0-M

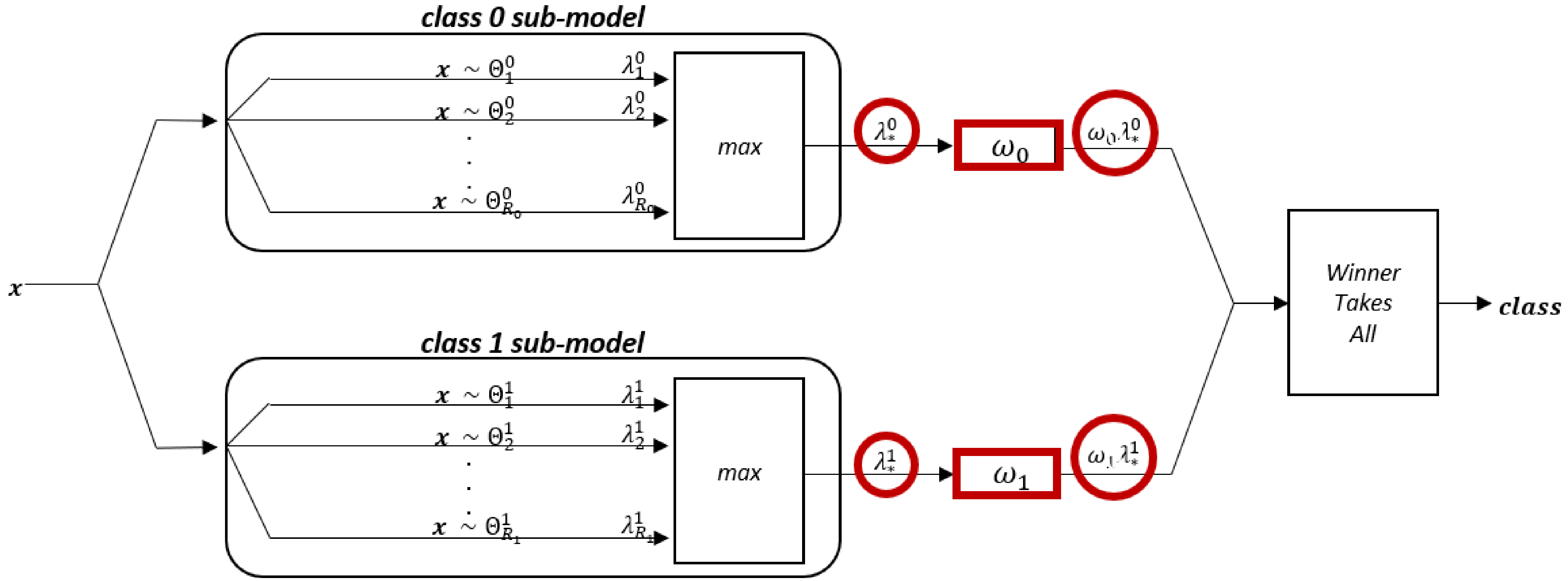
Metric	ALMMo-0-M			
	GM	**F1**	**Kappa**	**Matthews**
Accuracy	342/205/223 0.00	342/203/225 0.00	341/202/227 0.00	344/200/226 0.00
Recall	242/345/183 0.00	241/349/180 0.00	240/349/181 0.00	240/350/180 0.00
Precision	307/242/221 0.00	306/244/220 0.00	310/244/216 0.00	309/243/218 0.00
Specificity	298/260/212 0.00	295/264/211 0.00	296/265/209 0.00	296/264/210 0.00

TABLE X: Win/Tie/Loss results and respective Wilcoxon pairwise tests p-values of the proposed ALMMo-0-M against the original ALMMo-0 algorithm

ALMMo-0-W

- Cost sensitive approach that assigns a weight to each class
- Class confidence levels are weighted before the winner-takes-all approach is used to decide on the label

ALMMo-0-W

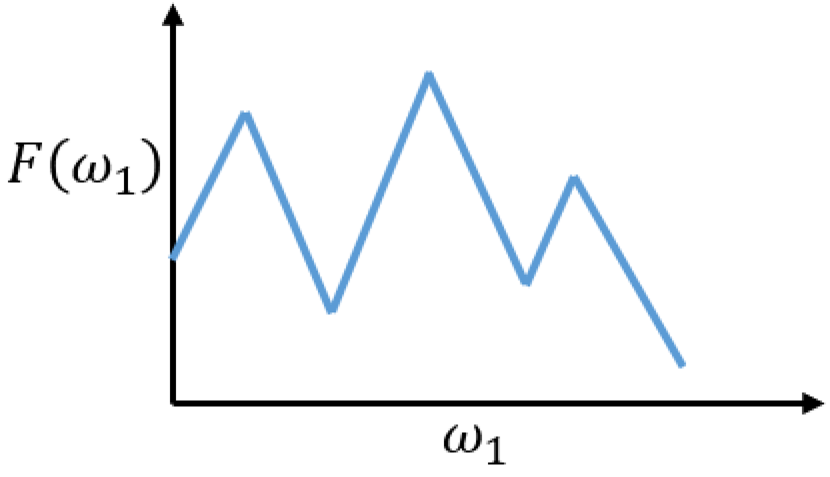


class 0 sub-model
$x \sim \Theta_1^0$
λ_1^0
$x \sim \Theta_2^0$
λ_2^0
$x \sim \Theta_{R_0}^0$
$\lambda_{R_0}^0$
max
λ_*^0
ω_0
$\omega_0.\lambda_*^0$
x
class 1 sub-model
$x \sim \Theta_1^1$
λ_1^1
$x \sim \Theta_2^1$
λ_2^1
$x \sim \Theta_{R_1}^1$
$\lambda_{R_1}^1$
max
λ_*^1
ω_1
$\omega_1.\lambda_*^1$
Winner Takes All
class

ALMMo-0-W

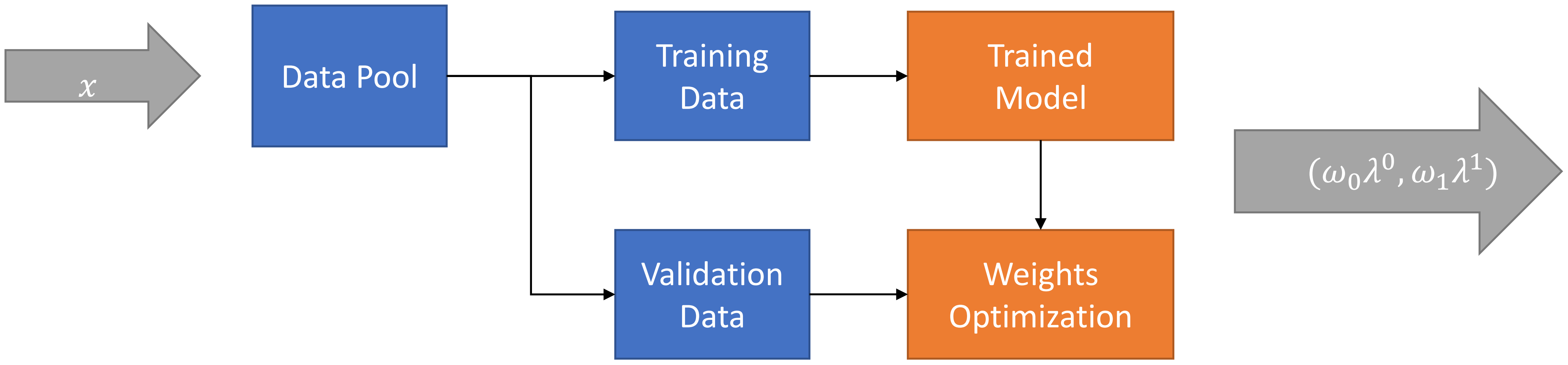
$$0.5 \leq \omega_1 \leq 1.0$$
$$0.0 \leq \omega_0 \leq 0.5$$
$$\omega_0 + \omega_1 = 1.0$$

$$\underbrace{\boxed{F(\omega_1)}}_{\text{Cost Function}} = F(\underbrace{\boxed{\hat{y}(\omega_1)}}_{\text{Weighted Model Predictions}}, \underbrace{\boxed{y}}_{\text{True Labels}})$$



Optimize using Bayesian Optimization

ALMMo-0-W

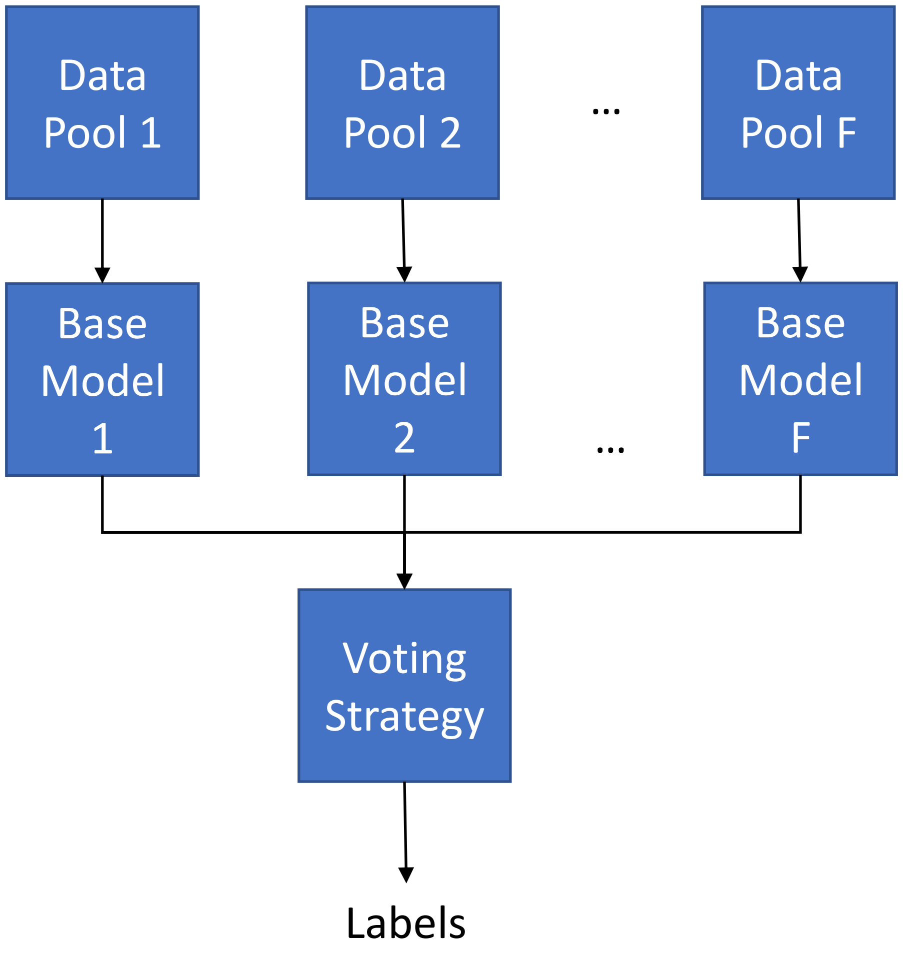


x
Data Pool
Training Data
Validation Data
Trained Model
Weights Optimization
$(\omega_0\lambda^0, \omega_1\lambda^1)$

ALMMo-0-W

- Create an ensemble of class-weighted base models and use an adequate voting strategy to decide on the class

ALMMo-0-W


Data Pool 1
Data Pool 2
...
Data Pool F
Base Model 1
Base Model 2
...
Base Model F
Voting Strategy
Labels

ALMMo-0-W

Metric	ALMMo-0-W			
	GM	F1	Kappa	Matthews
Accuracy	121/203/446 1.00	206/216/348 1.00	229/224/317 1.00	190/220/360 1.00
Recall	388/305/77 0.00	311/335/124 0.00	304/332/134 0.00	323/335/112 0.00
Precision	158/225/387 1.00	233/242/295 1.00	251/244/275 0.99	226/243/301 1.00
Specificity	72/238/460 1.00	156/263/351 1.00	170/267/333 1.00	142/263/365 1.00

TABLE IX: Win/Tie/Loss results and respective Wilcoxon pairwise tests p-values of the proposed ALMMo-0-W against the original ALMMo-0 algorithm

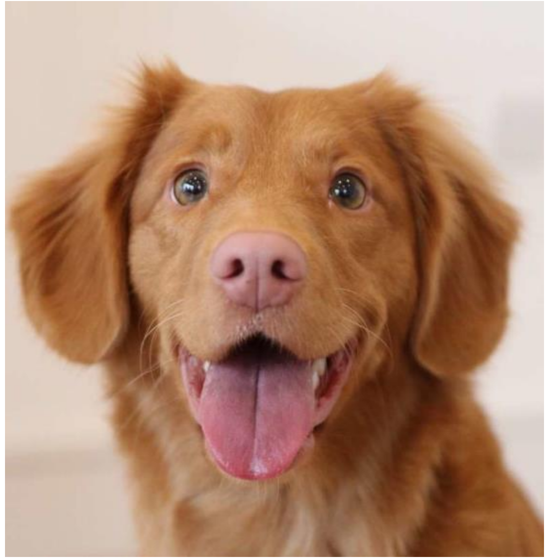
Future Work

- Combine the proposed methods with more complex architectures to improve performance on high dimensional and nonstationary imbalanced classification problems
- …

Future Work

- Image recognition for imbalanced datasets:

10 samples



990 samples

Future Work

- Class imbalance is time dependent: