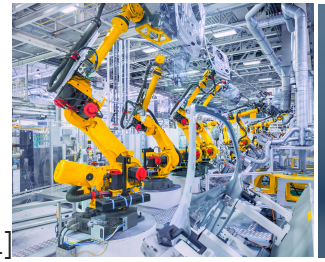
Cedric Kulbach

Online Automated Machine Learning

— Motivation

Motivation

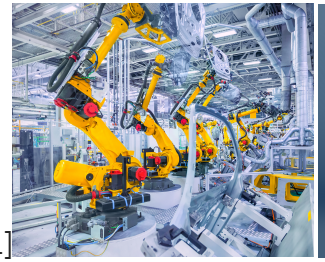
Data Stream


[1]


[2]

Motivation

Data Stream


[1]


[2]

Automated ML

Auto-Sklearn

TPOT

H_2O.ai

GAMA

Deep Learning

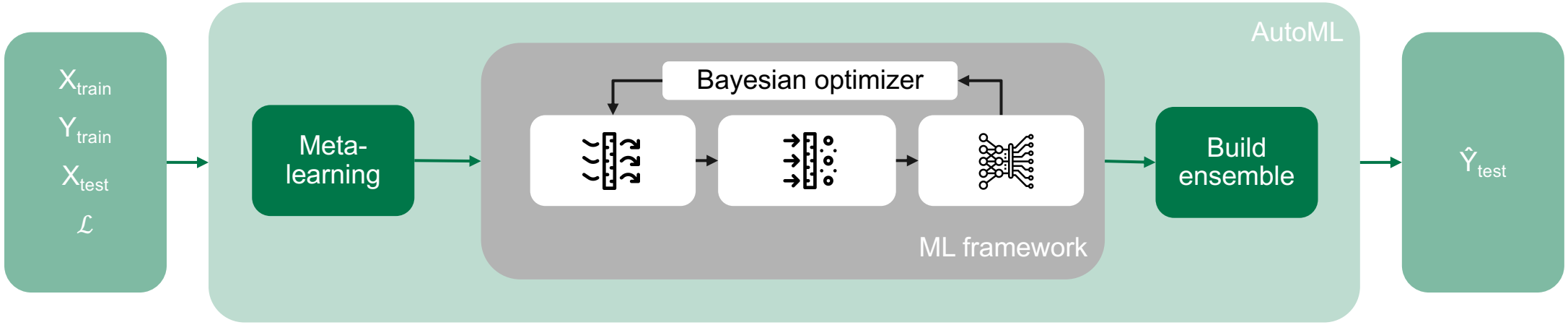
PyTorch [11][12]

Keras [13]

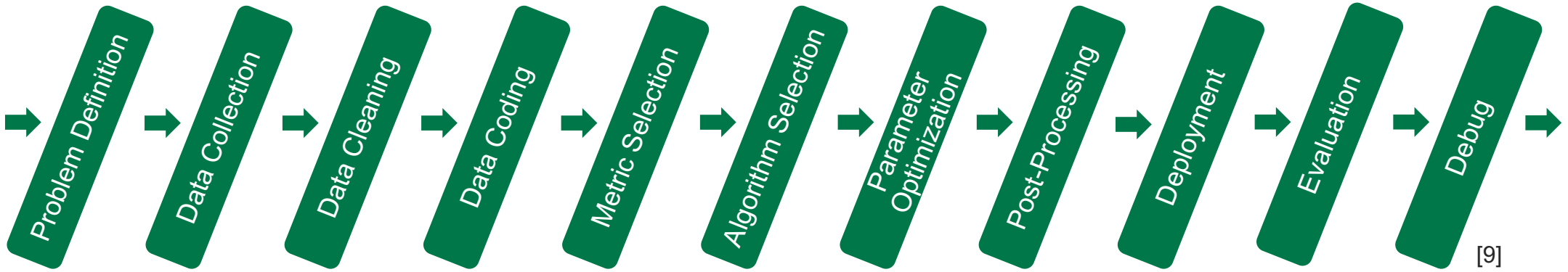
TensorFlow [14]

01 Automated Machine Learning

— Automated Machine Learning

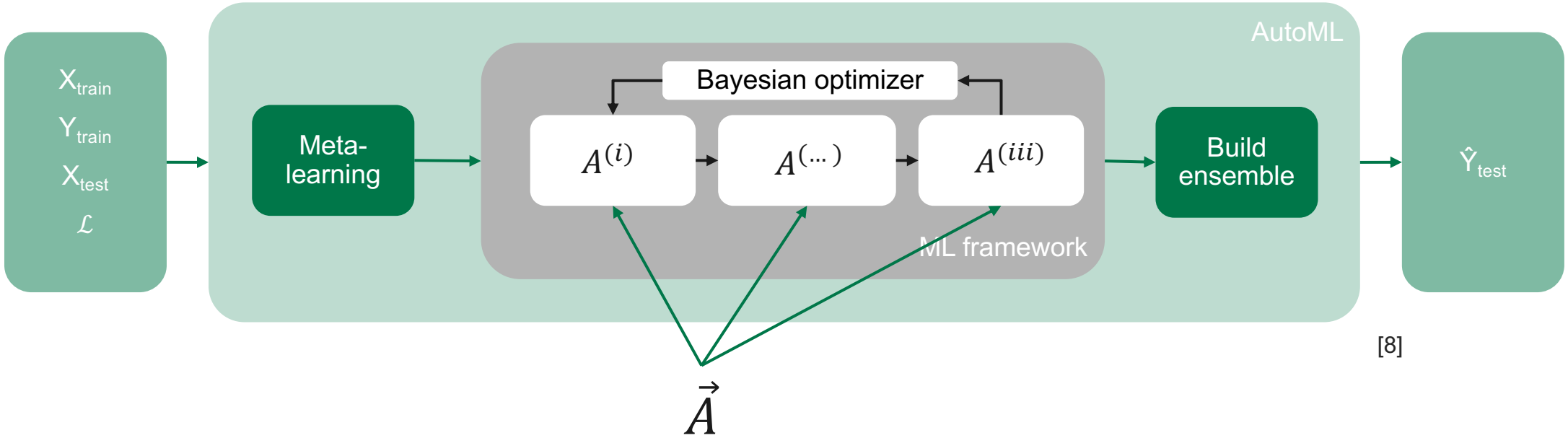


[8]



[9]

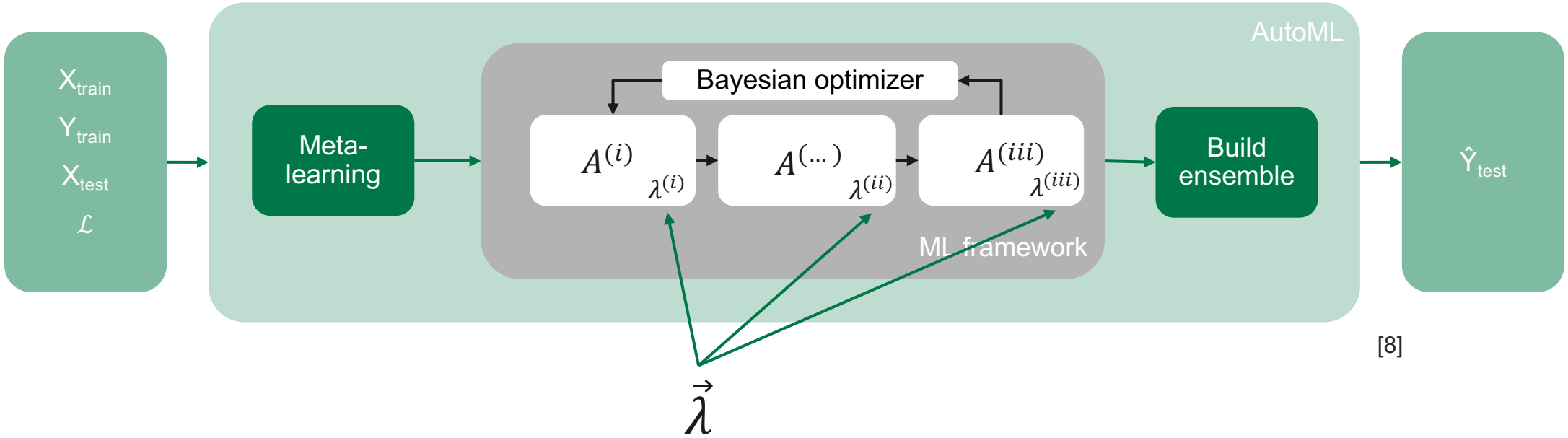
— Automated Machine Learning



[8]

$P_{g,\vec{A},\vec{\lambda}}$	Configured ML Pipeline
g	ML Pipeline Structure
$\vec{A}$	Algorithm Types
$\vec{\lambda}$	Hyperparameter Configuration
$\mathcal{L}$	Metric / Loss
$D^{(i)}$	Dataset

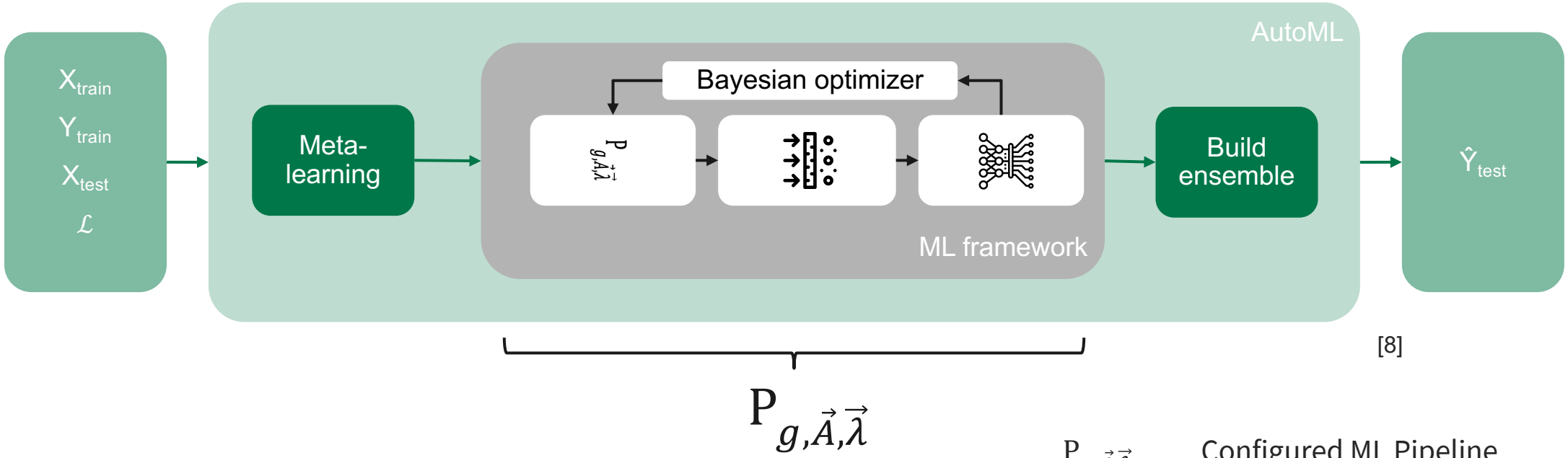
— Automated Machine Learning



[8]

$P_{g,\vec{A},\vec{\lambda}}$	Configured ML Pipeline
g	ML Pipeline Structure
$\vec{A}$	Algorithm Types
$\vec{\lambda}$	Hyperparameter Configuration
$\mathcal{L}$	Metric / Loss
$D^{(i)}$	Dataset

— Automated Machine Learning

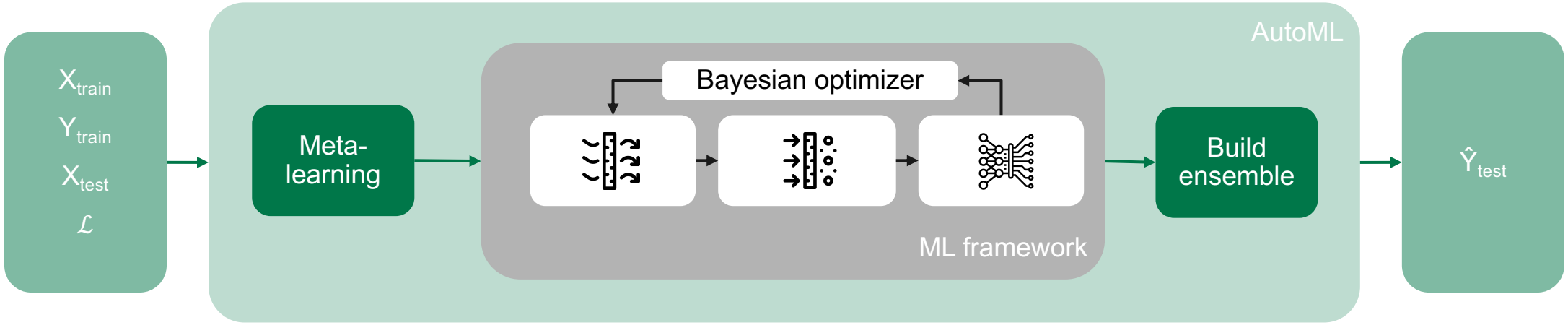


[8]

$$P_{g,\vec{A},\vec{\lambda}}$$

$P_{g,\vec{A},\vec{\lambda}}$	Configured ML Pipeline
g	ML Pipeline Structure
$\vec{A}$	Algorithm Types
$\vec{\lambda}$	Hyperparameter Configuration
$\mathcal{L}$	Metric / Loss
$D^{(i)}$	Dataset

— Automated Machine Learning



[8]

Loss on $D_{valid}^{(i)}$

CASH-Problem

$$g^*, \vec{A}^*, \vec{\lambda}^* \in \underbrace{\underset{P^{(j)} \in \mathrm{P}, \lambda \in \Lambda^{(j)}}{\arg\min}}_{\text{Minimize the Loss of a Pipeline configuration}} \underbrace{\frac{1}{K} \sum_{i=1}^{K}}_{\text{Sum of Losses on k-folds}} \mathcal{L}\left(\underbrace{\mathrm{P}_{g,\vec{A},\vec{\lambda}}\left(D_{train}^{(i)}\right)}_{\text{Pipeline fitted on } D_{train}^{(i)}}, D_{valid}^{(i)}\right)$$

$\mathrm{P}_{g,\vec{A},\vec{\lambda}}$	Configured ML Pipeline
g	ML Pipeline Structure
$\vec{A}$	Algorithm Types
$\vec{\lambda}$	Hyperparameter Configuration
$\mathcal{L}$	Metric / Loss
$D^{(i)}$	Dataset

— Online Automated Machine Learning

- Error Estimation
  - **Holdout:** Create holdout sets to test an train $\mathrm{P}_{g,\vec{A},\vec{\lambda}}$.
  - **Interleaved test-then-train:** Each sample is used to test and then train $\mathrm{P}_{g,\vec{A},\vec{\lambda}}$.
  - **Prequential:** test-then-train with sliding window so that recent examples are more important.
  - **Interleaved chunks:** test-then-train with chunks in sequence.

Loss on S^V

Online CASH-Problem

$$g^*, \vec{A}^*, \vec{\lambda}^* \in \underset{P^{(j)} \in \mathrm{P}, \lambda \in \Lambda^{(j)}}{\arg\min} \mathcal{L}\left(\mathrm{P}_{g,\vec{A},\vec{\lambda}}(S^T), S^V\right)$$

Minimize the Loss of a Pipeline configuration

Pipeline fitted on S^T

$\mathrm{P}_{g,\vec{A},\vec{\lambda}}$	Configured ML Pipeline
g	ML Pipeline Structure
$\vec{A}$	Algorithm Types
$\vec{\lambda}$	Hyperparameter Configuration
$\mathcal{L}$	Metric / Loss
S_i	Streaming Sample i $\{x_i, y_i\}$

$S^T \cap S^V = \emptyset$

— Approach

EvO AutoML

Input

Algorithm 1 EvO AutoML Training

Input:
Data stream S, population size P, sampling rate f_{SS}, loss function $\mathcal{L}$, configuration space $\mathcal{A}, \Lambda, G$
Output:
Set of suited algorithms configurations:
$p^* = \{\mathcal{P}^{(1)}, \ldots, \mathcal{P}^{(P)}\}$

$p \leftarrow \emptyset$ ▷ Initialization
while $|p| < P$ **do**
  $\mathcal{P} \leftarrow \text{Random}(G, \mathcal{A}, \Lambda)$
  $p \leftarrow p \cup \mathcal{P}$
end while
$t \leftarrow 0$
if e_t **then** ▷ Start Datastream
  if $t \mod f_{SS} == 0$ **then**
    $\mathcal{P}^{best} \leftarrow \min_{\mathcal{P} \in p} \mathcal{L}(\mathcal{P}(S^T), S^V)$
    $\mathcal{P}^{weak} \leftarrow \max_{\mathcal{P} \in p} \mathcal{L}(\mathcal{P}(S^T), S^V)$
    $\mathcal{P}^{mut} \leftarrow \text{Mutate}(\mathcal{P}^{best})$
    $p \leftarrow p \cup \mathcal{P}^{mut}$
    $p \leftarrow p \backslash \mathcal{P}^{weak}$
  end if
  $\omega \leftarrow Poisson(6)$
  for $\mathcal{P} \in p$ **do** ▷ Update Population
    loop ω
      $\mathcal{P}.\text{fit}(e_t)$
    end loop
  end for
  $t \leftarrow t + 1$
end if

Evolutionary Approach

Input :

Population size: P
Sampling rate: f_{SS}
loss function: $\mathcal{L}$
configuration space: A, Δ, G

— Approach

EvO AutoML

Input

Output

Algorithm 1 EvO AutoML Training

Input:
Data stream S, population size P, sampling rate f_{SS}, loss function $\mathcal{L}$, configuration space $\mathcal{A}, \Lambda, G$
Output:
Set of suited algorithms configurations:
$p^* = \{\mathcal{P}^{(1)}, \ldots, \mathcal{P}^{(P)}\}$

$p \leftarrow \emptyset$ ▷ Initialization
while $|p| < P$ **do**
  $\mathcal{P} \leftarrow \text{Random}(G, \mathcal{A}, \Lambda)$
  $p \leftarrow p \cup \mathcal{P}$
end while
$t \leftarrow 0$
if e_t **then** ▷ Start Datastream
  if $t \mod f_{SS} == 0$ **then**
    $\mathcal{P}^{best} \leftarrow \min_{\mathcal{P} \in p} \mathcal{L}(\mathcal{P}(S^T), S^V)$
    $\mathcal{P}^{weak} \leftarrow \max_{\mathcal{P} \in p} \mathcal{L}(\mathcal{P}(S^T), S^V)$
    $\mathcal{P}^{mut} \leftarrow \text{Mutate}(\mathcal{P}^{best})$
    $p \leftarrow p \cup \mathcal{P}^{mut}$
    $p \leftarrow p \backslash \mathcal{P}^{weak}$
  end if
  $\omega \leftarrow Poisson(6)$
  for $\mathcal{P} \in p$ **do** ▷ Update Population
    loop ω
      $\mathcal{P}.\text{fit}(e_t)$
    end loop
  end for
  $t \leftarrow t + 1$
end if

Evolutionary Approach

Output: p^*

— Approach

EvO AutoML

Input

Output

Initialization Population

Algorithm 1 EvO AutoML Training

Input:
Data stream S, population size P, sampling rate f_{SS}, loss function $\mathcal{L}$, configuration space $\mathcal{A}, \Lambda, G$
Output:
Set of suited algorithms configurations:
$p^* = \{\mathcal{P}^{(1)}, \ldots, \mathcal{P}^{(P)}\}$

$p \leftarrow \emptyset$ ▷ Initialization
while $|p| < P$ **do**
  $\mathcal{P} \leftarrow \text{Random}(G, \mathcal{A}, \Lambda)$
  $p \leftarrow p \cup \mathcal{P}$
end while
$t \leftarrow 0$
if e_t **then** ▷ Start Datastream
  if $t \mod f_{SS} == 0$ **then**
    $\mathcal{P}^{best} \leftarrow \min_{\mathcal{P} \in p} \mathcal{L}(\mathcal{P}(S^T), S^V)$
    $\mathcal{P}^{weak} \leftarrow \max_{\mathcal{P} \in p} \mathcal{L}(\mathcal{P}(S^T), S^V)$
    $\mathcal{P}^{mut} \leftarrow \text{Mutate}(\mathcal{P}^{best})$
    $p \leftarrow p \cup \mathcal{P}^{mut}$
    $p \leftarrow p \backslash \mathcal{P}^{weak}$
  end if
  $\omega \leftarrow Poisson(6)$
  for $\mathcal{P} \in p$ **do** ▷ Update Population
    loop ω
      $\mathcal{P}.\text{fit}(e_t)$
    end loop
  end for
  $t \leftarrow t + 1$
end if

Evolutionary Approach

Initialization:

Generating a random population from the configuration space

— Approach

EvO AutoML

Algorithm 1 EvO AutoML Training

1: **Input:**
2: Data stream S, population size P, sampling rate f_{SS}, loss function $\mathcal{L}$, configuration space $\mathcal{A}, \Lambda, G$
3: **Output:**
4: Set of suited algorithms configurations:
5: $p^* = \{\mathcal{P}^{(1)}, \ldots, \mathcal{P}^{(P)}\}$
6:
7: $p \leftarrow \emptyset$ ▷ Initialization
8: **while** $|p| < P$ **do**
9: $\mathcal{P} \leftarrow \text{Random}(G, \mathcal{A}, \Lambda)$
10: $p \leftarrow p \cup \mathcal{P}$
11: **end while**
12: $t \leftarrow 0$
13: **if** e_t **then** ▷ Start Datastream
14: **if** $t \mod f_{SS} == 0$ **then**
15: $\mathcal{P}^{best} \leftarrow \min_{\mathcal{P} \in p} \mathcal{L}(\mathcal{P}(S^T), S^V)$
16: $\mathcal{P}^{weak} \leftarrow \max_{\mathcal{P} \in p} \mathcal{L}(\mathcal{P}(S^T), S^V)$
17: $\mathcal{P}^{mut} \leftarrow \text{Mutate}(\mathcal{P}^{best})$
18: $p \leftarrow p \cup \mathcal{P}^{mut}$
19: $p \leftarrow p \backslash \mathcal{P}^{weak}$
20: **end if**
21: $\omega \leftarrow Poisson(6)$
22: **for** $\mathcal{P} \in p$ **do** ▷ Update Population
23: **loop** ω
24: $\mathcal{P}.\text{fit}(e_t)$
25: **end loop**
26: **end for**
27: $t \leftarrow t + 1$
28: **end if**

Input

Output

Initialization Population

Mutation

Evolutionary Approach

Mutation:

1. Select best & weekest Pipeline
2. Generate a new configuration by mutating the best one
3. Remove the weakest configuration

— Approach

EvO AutoML

Algorithm 1 EvO AutoML Training

Input

1: **Input:**
2: Data stream S, population size P, sampling rate f_{SS}, loss function $\mathcal{L}$, configuration space $\mathcal{A}, \Lambda, G$

Output

3: **Output:**
4: Set of suited algorithms configurations:
5: $p^* = \{\mathcal{P}^{(1)}, \ldots, \mathcal{P}^{(P)}\}$
6:

Initialization Population

7: $p \leftarrow \emptyset$ ▷ Initialization
8: **while** $|p| < P$ **do**
9: $\mathcal{P} \leftarrow \text{Random}(G, \mathcal{A}, \Lambda)$
10: $p \leftarrow p \cup \mathcal{P}$
11: **end while**
12: $t \leftarrow 0$
13: **if** e_t **then** ▷ Start Datastream

Mutation

14: **if** $t \mod f_{SS} == 0$ **then**
15: $\mathcal{P}^{best} \leftarrow \min_{\mathcal{P} \in p} \mathcal{L}(\mathcal{P}(S^T), S^V)$
16: $\mathcal{P}^{weak} \leftarrow \max_{\mathcal{P} \in p} \mathcal{L}(\mathcal{P}(S^T), S^V)$
17: $\mathcal{P}^{mut} \leftarrow \text{Mutate}(\mathcal{P}^{best})$
18: $p \leftarrow p \cup \mathcal{P}^{mut}$
19: $p \leftarrow p \backslash \mathcal{P}^{weak}$
20: **end if**

Training

21: $\omega \leftarrow Poisson(6)$
22: **for** $\mathcal{P} \in p$ **do** ▷ Update Population
23: **loop** ω
24: $\mathcal{P}.\text{fit}(e_t)$
25: **end loop**
26: **end for**
27: $t \leftarrow t + 1$
28: **end if**

Evolutionary Approach

Training:

Train all configurations in p with an online bagging approach.

— Results

Single algorithms

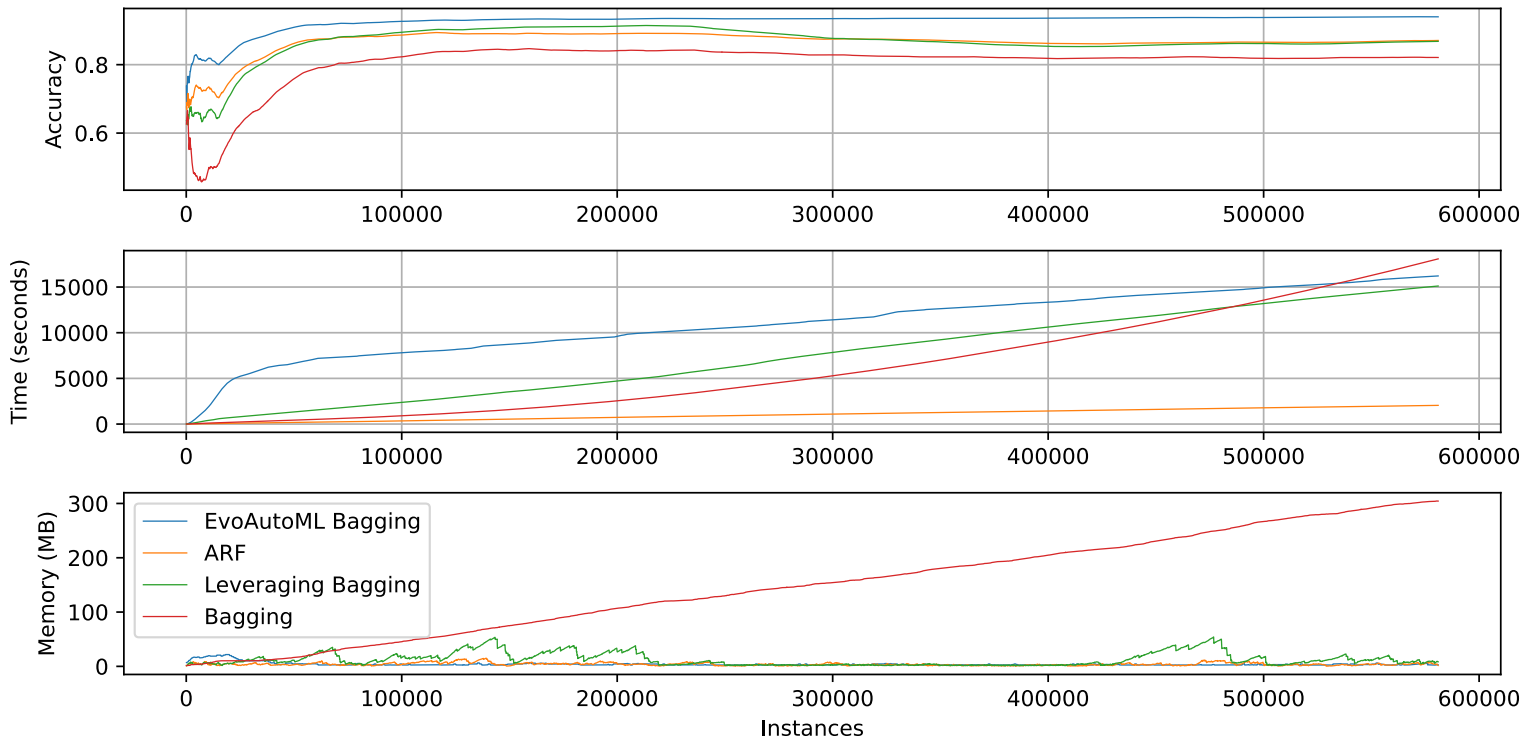
Dataset	EvoAutoML Bagging			Hoeffding Tree			GaussianNB			KNN		
	Time	Accuracy	Mem.	Time	Accuracy	Mem.	Time	Accuracy	Mem.	Time	Accuracy	Mem.
Agrawal(50)	12,647	**99.02**	17.61	459	98.09	0.60	318	62.31	0.01	892	55.73	0.46
Agrawal(50,000)	25,079	**94.43**	56.85	1,023	91.84	2.22	383	62.33	0.01	1,106	55.52	0.46
HYP(50,0.0001)	52,091	87.51	104.58	5,234	84.38	18.29	726	**91.61**	0.07	2,214	67.93	2.02
HYP(50,0.001)	60,162	**83.69**	127.88	5,117	81.79	18.52	533	80.83	0.07	1,907	68.01	2.02
LED()	29,553	**76.49**	35.95	1,930	75.95	2.10	1,258	76.48	0.05	1,359	66.6	0.38
RBF(10,0.0001)	43,412	99.82	24.53	5,017	89.32	13.36	1,788	65.86	0.13	2,831	**100**	2.02
RBF(10,0.001)	50,814	99.63	36.11	10,425	77.61	30.53	1,703	39.75	0.13	2,855	**99.99**	2.02
RBF(50,0.0001)	78,967	97.51	64.46	19,484	83.05	24.17	6,760	35.26	0.17	6,408	**99.83**	2.02
RBF(50,0.001)	57,469	96.99	29.29	9,797	48.15	9.17	5,248	25.32	0.17	6,186	**99.80**	2.02
SINE()	8,012	**99.87**	9.76	181	99.63	0.42	111	93.62	0.00	589	98.75	0.17
SEA(50)	9,793	**98.99**	17.83	228	97.78	0.72	114	95.65	0.01	520	97.23	0.21
Elec	731	**88.09**	12.70	16	79.61	0.21	12	72.87	0.01	37	79.53	0.42
Covtype	16,216	**91.09**	12.08	1,346	66.67	0.13	401	63.64	0.08	643	73.74	2.17
Avg. Acc.		**93.32**			82.61			66.58			81.74	
RAM-Hours		7.16			0.32			0			0.01	
Avg. rank		**1.38**			2.54			3.38			2.69	

- **Evaluation:**

  •EvO AutoML:
  - 3 Scalers
  - 4 Predictors + configuration
  - 174 possible configurations

— Results

Ensemble Evaluation



- **Covertype Dataset:** Labeled instances of forest cover type (54 attributes, 7 classes)
- **ARF:**
  Adaptive Random Forest [3]
- **Leveraging Bagging** [4]
- **Bagging:** Online Bagging [5]

- **Evaluation:**

  • Ensemble Base Model:
  10 Hoeffding Trees [7]

  • EvO AutoML:
  - 3 Scalers
  - 4 Predictors + configuration
  - 174 possible configurations

— Results

Ensemble Evaluation

Dataset	EvoAutoML Bagging			ARF			Leveraging Bagging			Bagging		
	Time	Accuracy	Mem.	Time	Accuracy	Mem.	Time	Accuracy	Mem.	Time	Accuracy	Mem.
Agrawal(50)	12,647	99.02	17.61	5,146	94.98	11.09	9,182	**99.69**	12.21	3,432	98.46	6.01
Agrawal(50,000)	25,079	94.43	56.85	7,853	93.03	12.92	20,276	**97.52**	37.60	8,749	92.89	21.50
HYP(50,0.0001)	52,091	**87.51**	104.58	340,212	71.19	229.90	192,060	84.54	528.85	58,183	87.14	180.87
HYP(50,0.001)	60,162	83.69	127.88	282,634	71.67	356.60	126,480	83.95	395.20	71,293	**84.43**	187.15
LED()	29,553	**76.49**	35.95	7,829	76.47	10.13	34,276	**76.49**	39.72	8,934	76.42	18.57
RBF(10,0.0001)	43,412	99.82	24.53	18,703	**99.85**	25.80	78,631	99.64	22.90	59,496	98.07	134.99
RBF(10,0.001)	50,814	**99.63**	36.11	17,058	99.22	11.67	67,374	99.01	4.89	98,400	93.68	291.35
RBF(50,0.0001)	78,967	97.51	64.46	33,902	98.21	27.12	91,652	**98.71**	35.64	90,834	96.17	236.12
RBF(50,0.001)	57,469	**96.99**	29.29	16,711	94.31	25.45	124,927	93.56	8.02	53,210	71.87	98.34
SINE()	8,012	**99.87**	9.76	4,488	99.74	14.62	5,414	99.68	11.13	1,364	99.77	4.21
SEA(50)	9,793	98.99	17.83	3,036	99.64	8.41	5,532	**99.67**	14.07	3,217	98.34	7.45
Elec	731	**88.09**	12.70	296	87.79	6.85	448	87.32	1.73	110	81.74	1.94
Covtype	8,559	**91.09**	12.08	355	89.70	4.75	2,260	90.41	15.55	936	83.66	19.37
Avg. Acc.		**93.32**			90.45			93.09			89.43	
RAM-Hours		7.16			50.38			44.55			24.35	
Avg. rank		**1.69**			2.69			2.15			3.38	

- **ARF:**
  Adaptive Random Forest [3]
- **Leveraging Bagging** [4]
- **Bagging:** Online Bagging [5]

- **Evaluation:**

  • Ensemble Base Model:
  10 Hoeffding Trees [7]

  • EvO AutoML:
  - 3 Scalers
  - 4 Predictors + configuration
  - 174 possible configurations

02 Incremental Deep Learning

— Incremental Deep Learning

Implementation … at which cost?

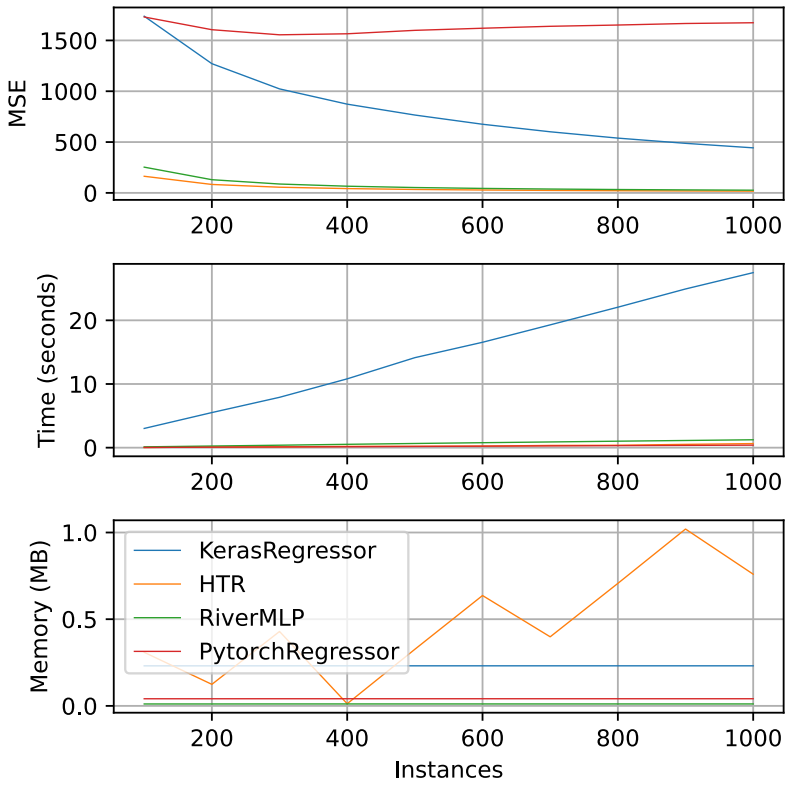
River [10]

PyTorch [11][12]

Keras [13]

TensorFlow [14]

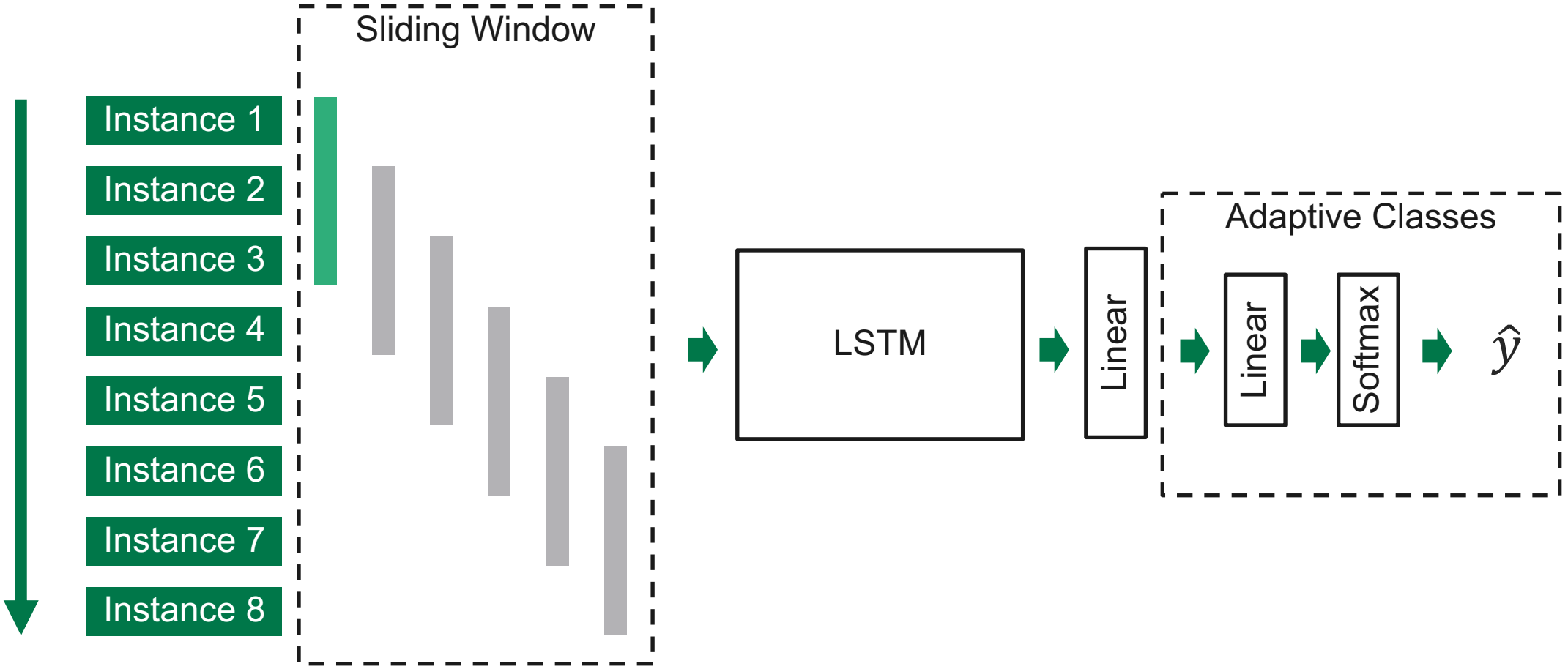
IT´S BEEN MERGED



Trump approval dataset [5]

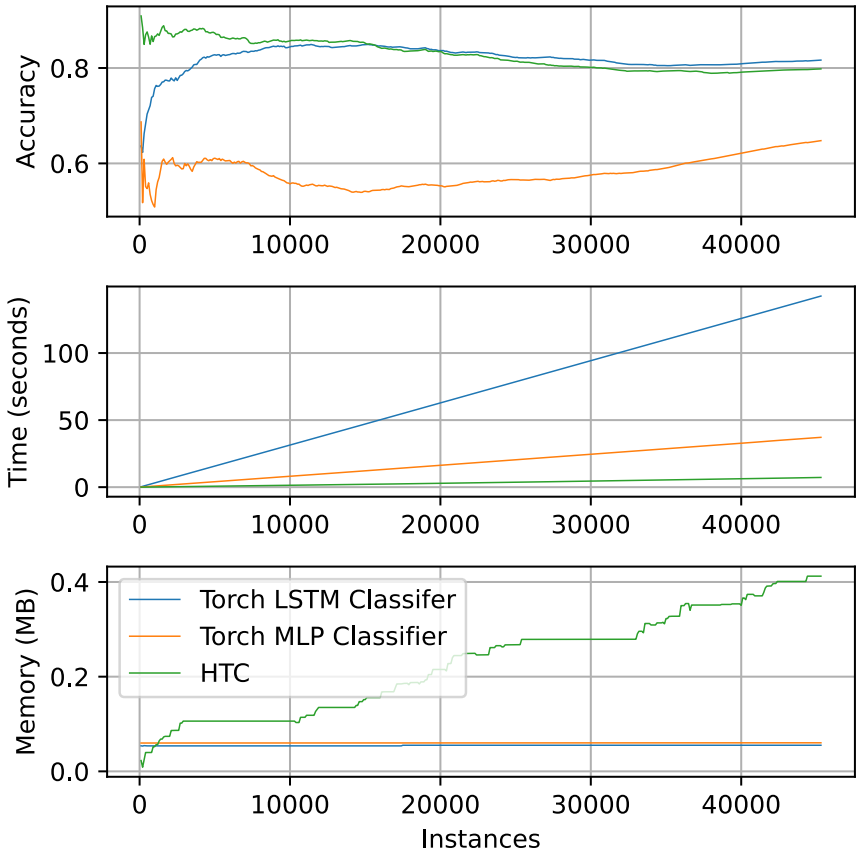
— And next?

Changing the problem definition – Incremental Multivariate Timeseries



— Incremental DL

First Results



- Electricity Datastream
  - 6 features
  - goal: identify the electricity price change

— Outlook

EvO AutoML

- Evaluation against more recent algorithms such as Streaming Random Patches [15]

Incremental DL

- What is the influence of the learning rate on adaptability?
- Further advantages in incremental deep learning?

Merci!

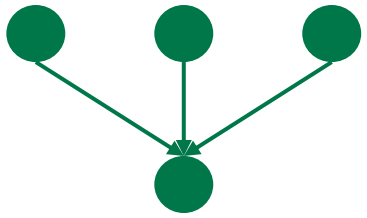
Cedric Kulbach
kulbach@fzi.de

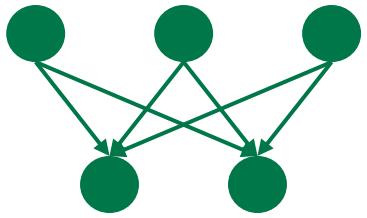
Sources

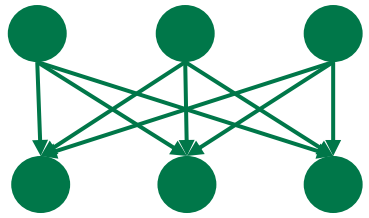
[0] https://xkcd.com/1838/, accessed on 26.10.2021
[1] https://www.it-production.com/produktionsmanagement/synchrone-prozesse-in-der-produktion/, accessed on 06.10.21
[2] Archivbilder Microsoft Powerpoint
[3] Gomes et al. "Adaptive random forests for evolving data stream classification". Mach. Learn. 106. 9-10(2017): 1469–1495.
[4] Bifet et al. "Leveraging Bagging for Evolving Data Streams." ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part I. Springer
[5] Oza, . "Online bagging and boosting." IEEE International Conference on Systems, Hawaii, USA, October 10-12, 2005. IEEE,
[6] https://projects.fivethirtyeight.com/trump-approval-ratings/, accessed on 26.10.2021
[7] Domingos et al. "Mining high-speed data streams." ACM SIGKDD, Boston, MA, USA, August 20-23, 2000. ACM,
[8] Feurer et al. „Efficient and robust automated machine learning“, NIPS 2015
[9] Rich, "AutoML“, ICML Workshop, 2015
[10] Montiel, et al. "River: machine learning for streaming data in Python." (2020).
[11] https://icon-icons.com/icon/pytorch-logo/169823, zuletzt aufgerufen am 06.10.21
[12] https://pytorch.org/, zuletzt aufgerufen am 06.10.21
[13] https://keras.io/, zuletzt aufgerufen am 06.10.21
[14] https://www.tensorflow.org/, zuletzt aufgerufen am 06.10.21
[15] Gomes et al. "Streaming Random Patches for Evolving Data Stream Classification." ICDM 2019, Beijing, China, November 8-11, 2019. IEEE,

— Incremental Deep Learning

What about classification with variable outputs?







Regression	
$\widehat{y_1}$	p

Multi Target Regression	
$\widehat{y_1}$	p
$\widehat{y_n}$	p_n

Binary classification	
$\widehat{y_1}$	p
$\widehat{y_2}$	$1-p$

Binary classification	
$\widehat{y_1}$	p_1
$\widehat{y_2}$	p_2

Multiclass classification	
$\widehat{y_1}$	p_1
$\widehat{y_n}$	p_n

— Incremental Deep Learning

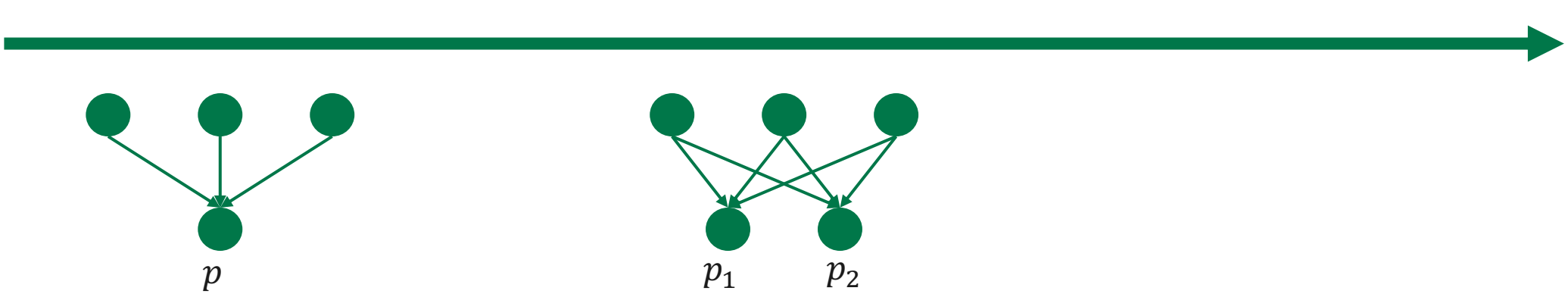
What about classification with variable outputs?

Regression	
$\widehat{y_1}$	p

Binary classification	
$\widehat{y_1}$	p
$\widehat{y_2}$	$1-p$

— Incremental Deep Learning

What about classification with variable outputs?



Regression	
$\widehat{y_1}$	p

Multi Target Regression	
$\widehat{y_1}$	p
$\widehat{y_n}$	p_n

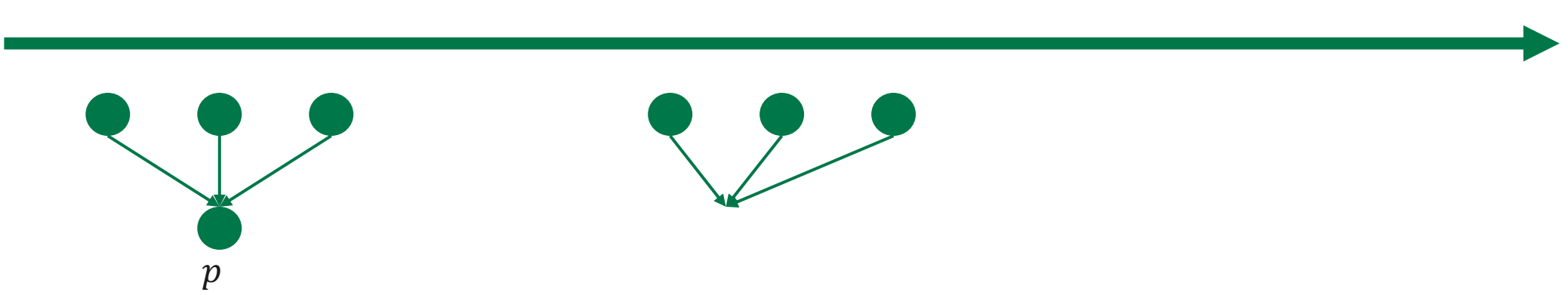
Binary classification	
$\widehat{y_1}$	p
$\widehat{y_2}$	$1-p$

Binary classification	
$\widehat{y_1}$	p_1
$\widehat{y_2}$	p_2

Multiclass classification	
$\widehat{y_1}$	p_1
$\widehat{y_n}$	p_n

— Incremental Deep Learning

What about classification with variable outputs?



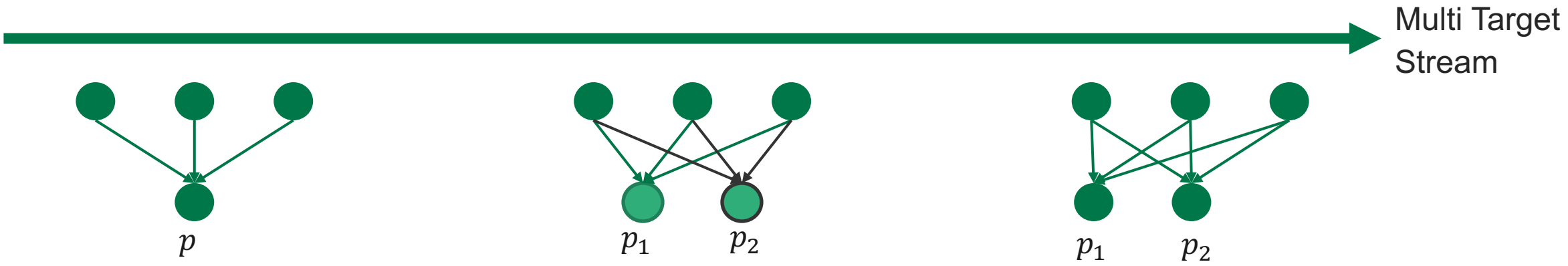
Binary classification	
$\widehat{y_1}$	p
$\widehat{y_2}$	$1-p$

Binary classification	
$\widehat{y_1}$	p_1
$\widehat{y_2}$	p_2

Multiclass classification	
$\widehat{y_1}$	p_1
$\widehat{y_n}$	p_n

— Incremental Deep Learning

What about classification with variable outputs?



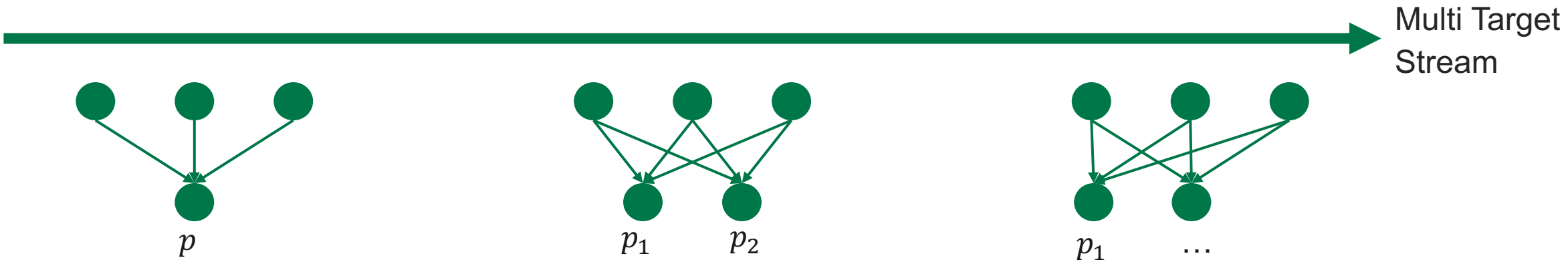
Binary classification	
$\widehat{y_1}$	p
$\widehat{y_2}$	$1-p$

Binary classification	
$\widehat{y_1}$	p_1
$\widehat{y_2}$	p_2

Multiclass classification	
$\widehat{y_1}$	p_1
$\widehat{y_n}$	p_n

— Incremental Deep Learning

What about classification with variable outputs?



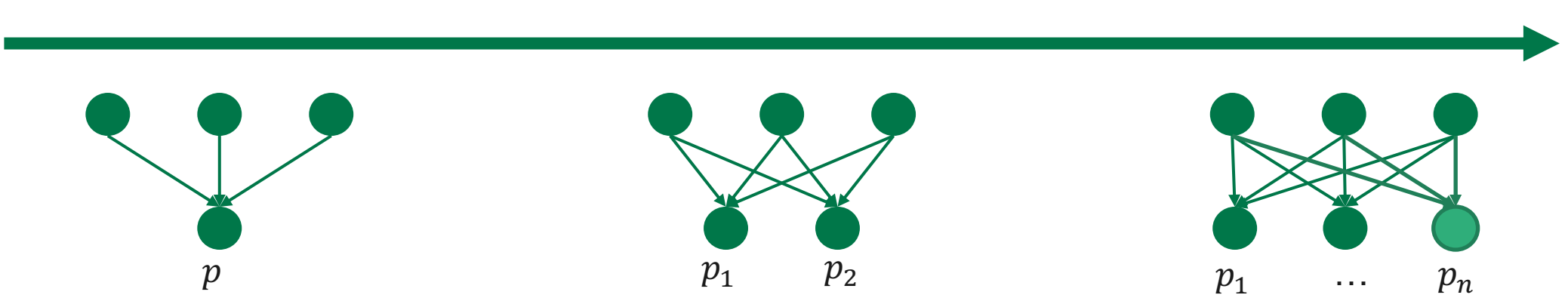
Binary classification	
$\widehat{y_1}$	p
$\widehat{y_2}$	$1-p$

Binary classification	
$\widehat{y_1}$	p_1
$\widehat{y_2}$	p_2

Multiclass classification	
$\widehat{y_1}$	p_1
$\widehat{y_n}$	p_n

— Incremental Deep Learning

What about classification with variable outputs?



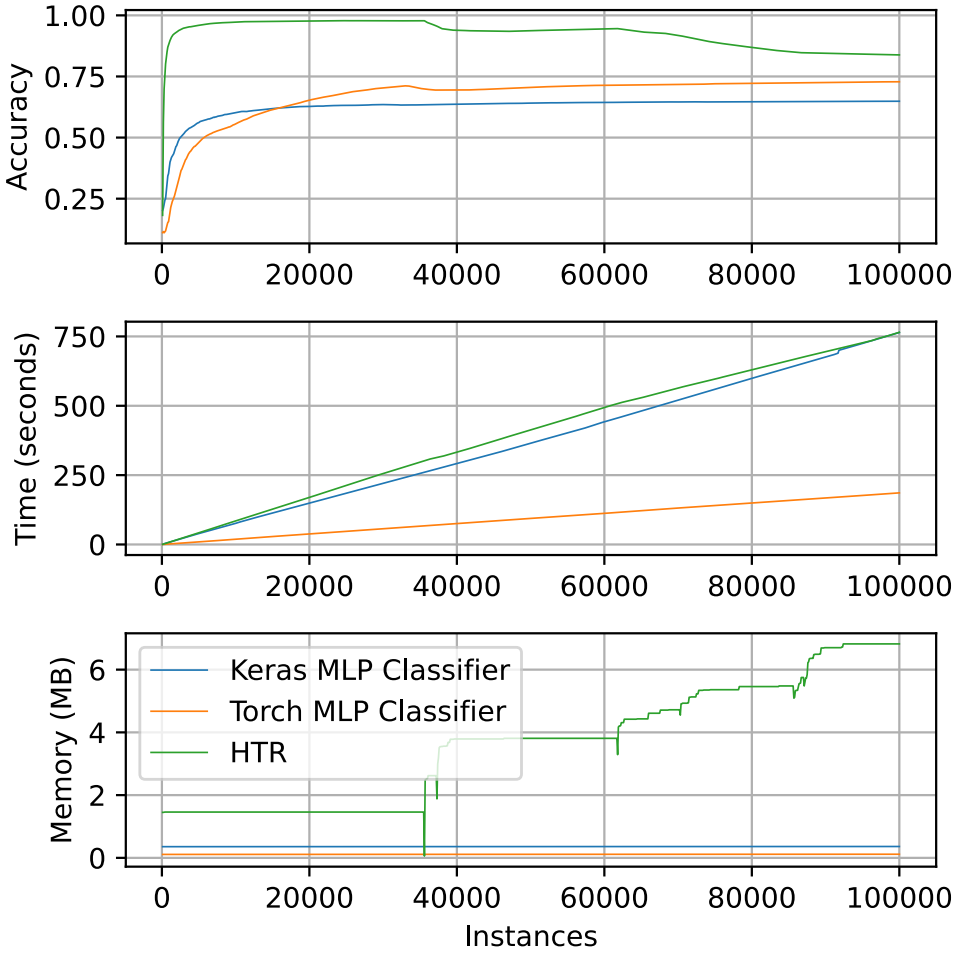
Binary classification	
$\widehat{y_1}$	p
$\widehat{y_2}$	$1-p$

Binary classification	
$\widehat{y_1}$	p_1
$\widehat{y_2}$	p_2

Multiclass classification	
$\widehat{y_1}$	p_1
$\widehat{y_n}$	p_n

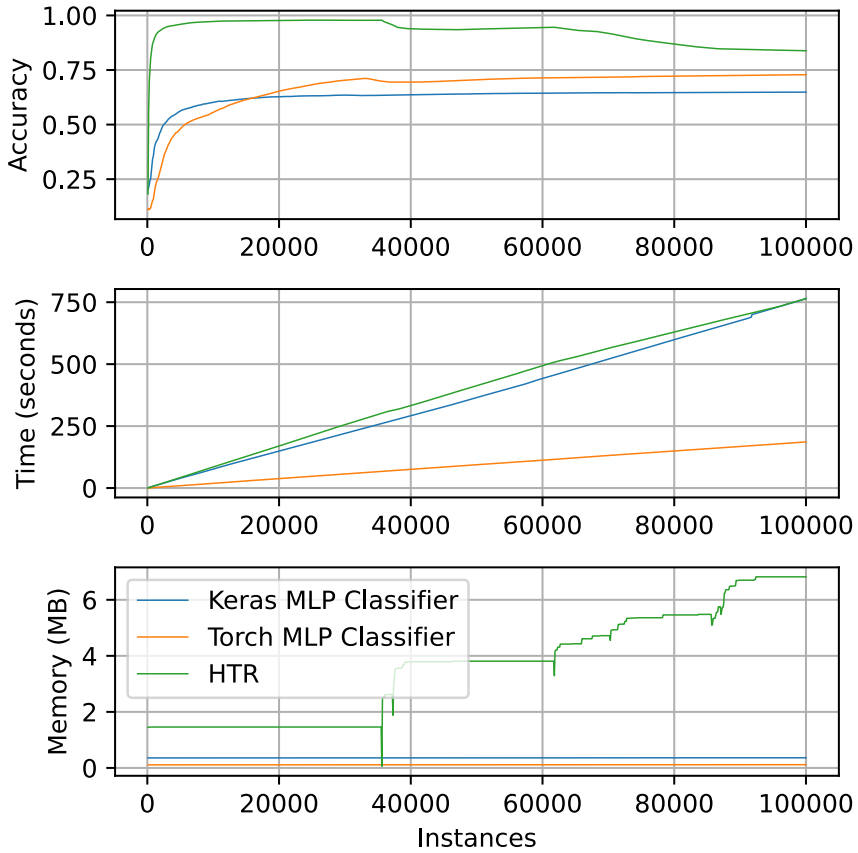
— Incremental Deep Learning

First Results



— Incremental Deep Learning

First Results



- RBF Generator
  - 200 features
  - 10 classes