Hybrid Deep Neural Networks to Infer State Models of Black-Box Systems

Mohammad Jafar Mashhadi

Hadi Hemmati

UNIVERSITY OF CALGARY

Natural Sciences and Engineering Research Council of Canada | Conseil de recherches en sciences naturelles et en génie du Canada

Canada

ASE 2020

Problem context: Black-box Automated Systems

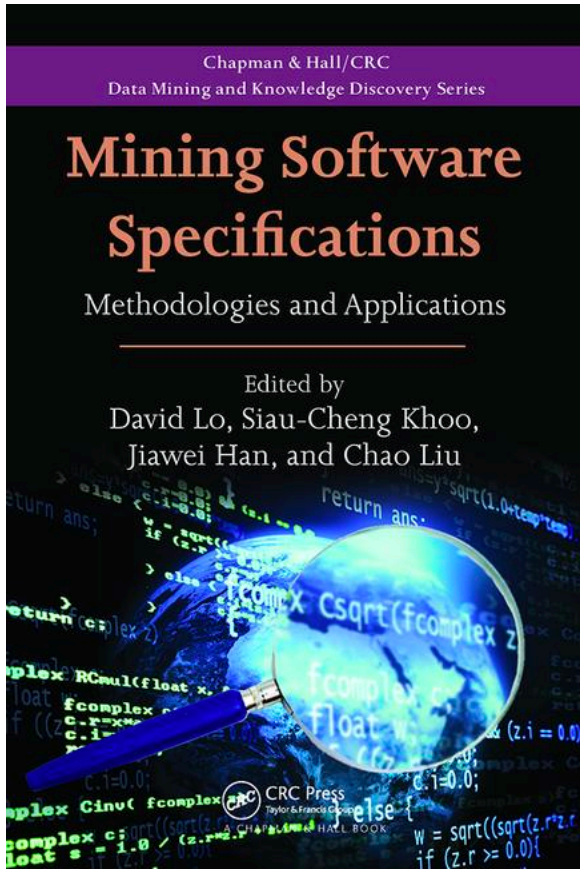
Automated systems are all around us.
At their core they use either:

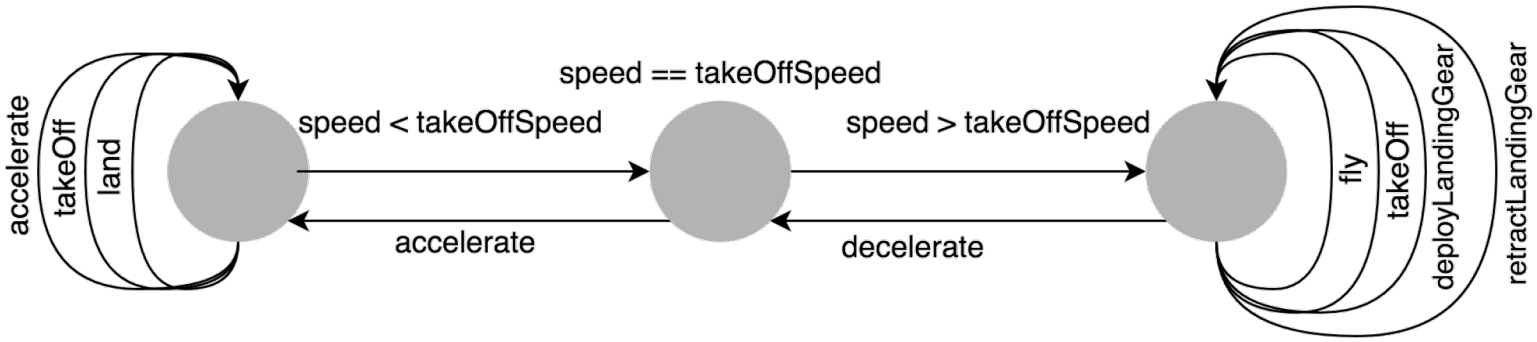
- AI: Self driving cars
- Feedback loop controllers: Autopilot, cruise control, plant controllers, etc.

Problem context: Understanding their behaviour

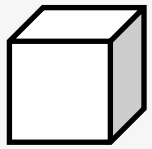
- Understanding their behaviour
  - Verification
  - Anomaly detection
  - Automated Testing
- They are often state based

Problem context: Specification Mining and Model Inference





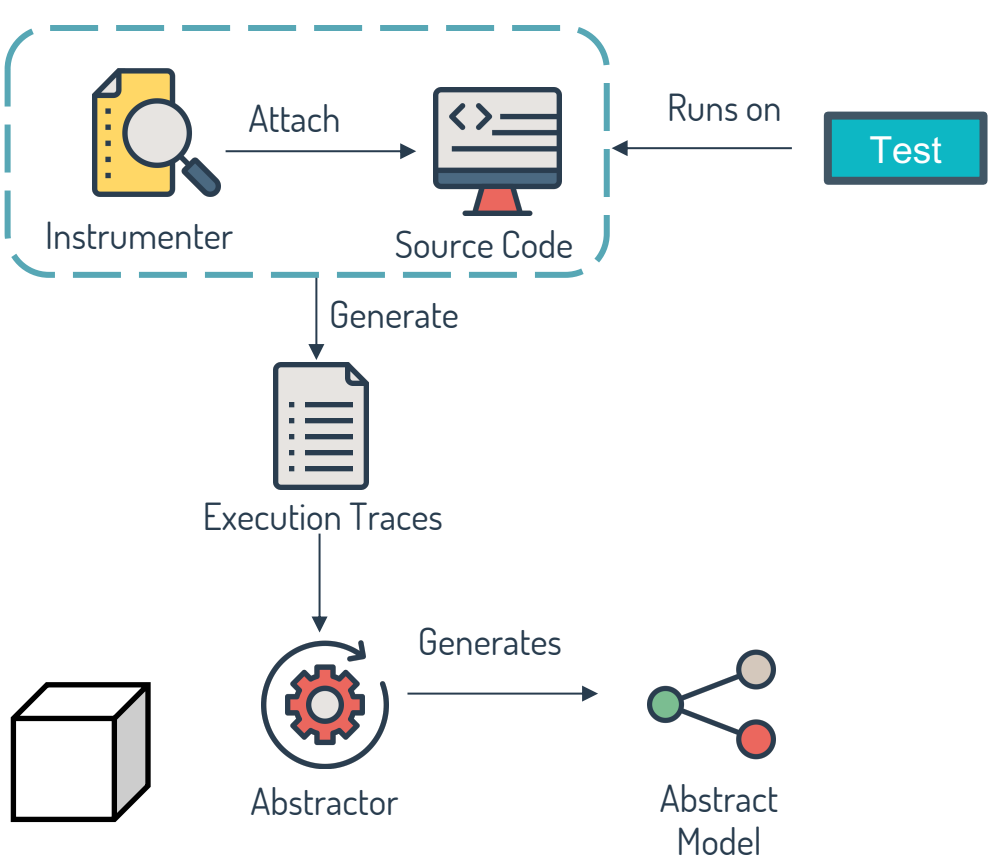
Problem context: white-box model inference

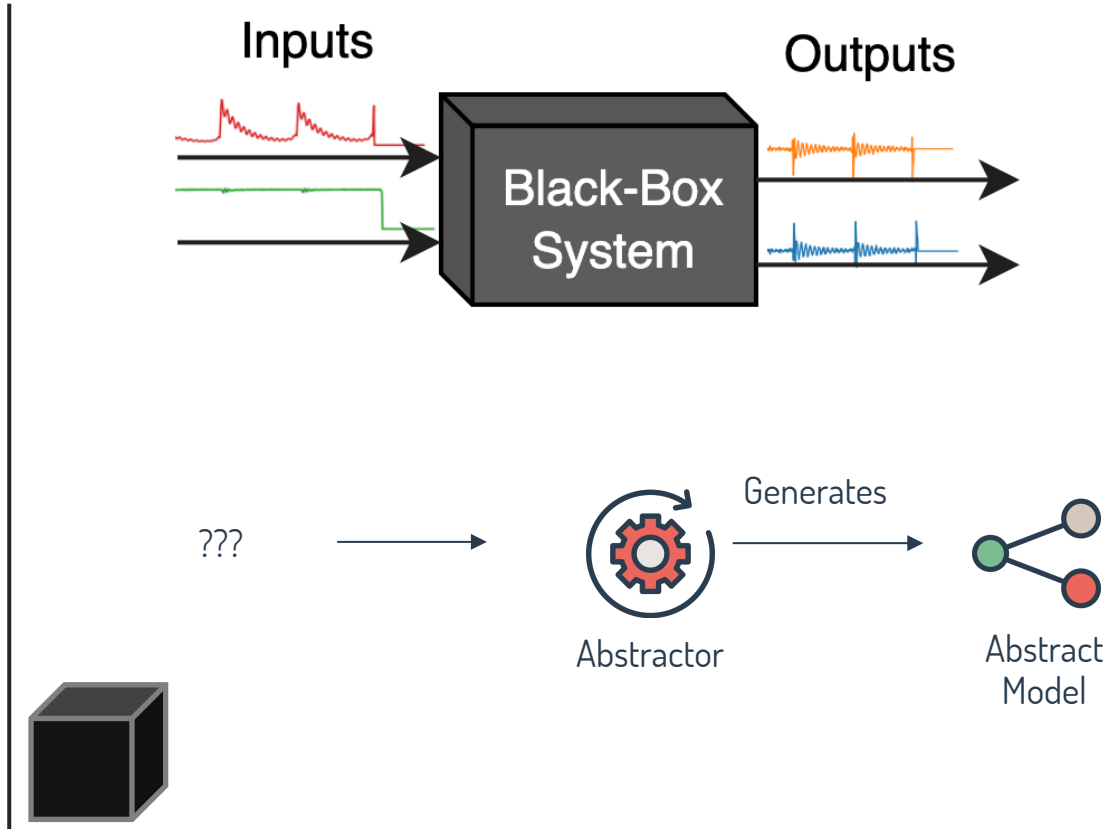


- Most of the prior work focuses on white-box analysis
- But, black-box systems are quite often used in practice

Problem context: white-box vs. black-box analysis

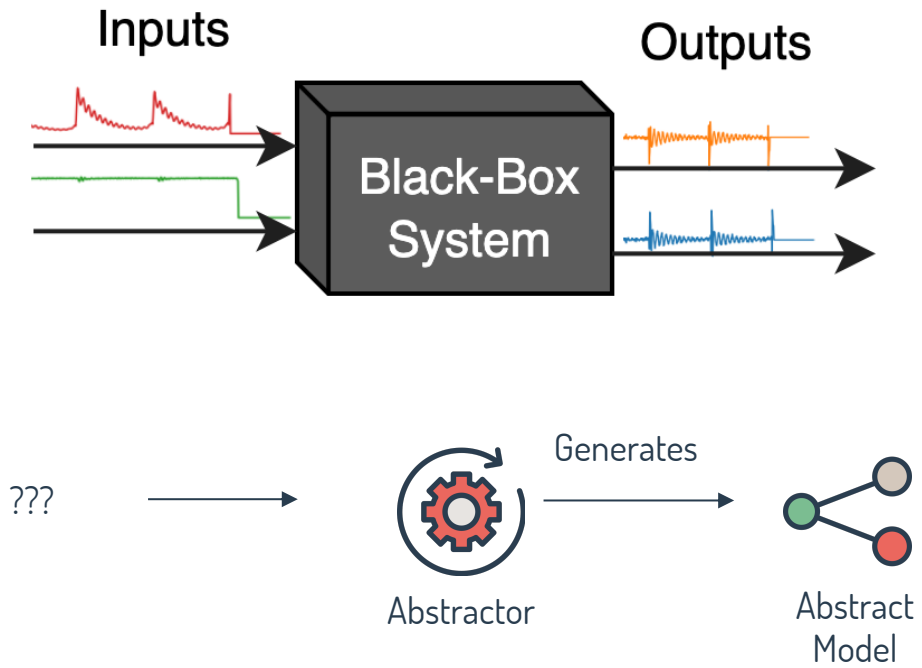
- Static analysis is out of question
- Dynamic analysis has limitations





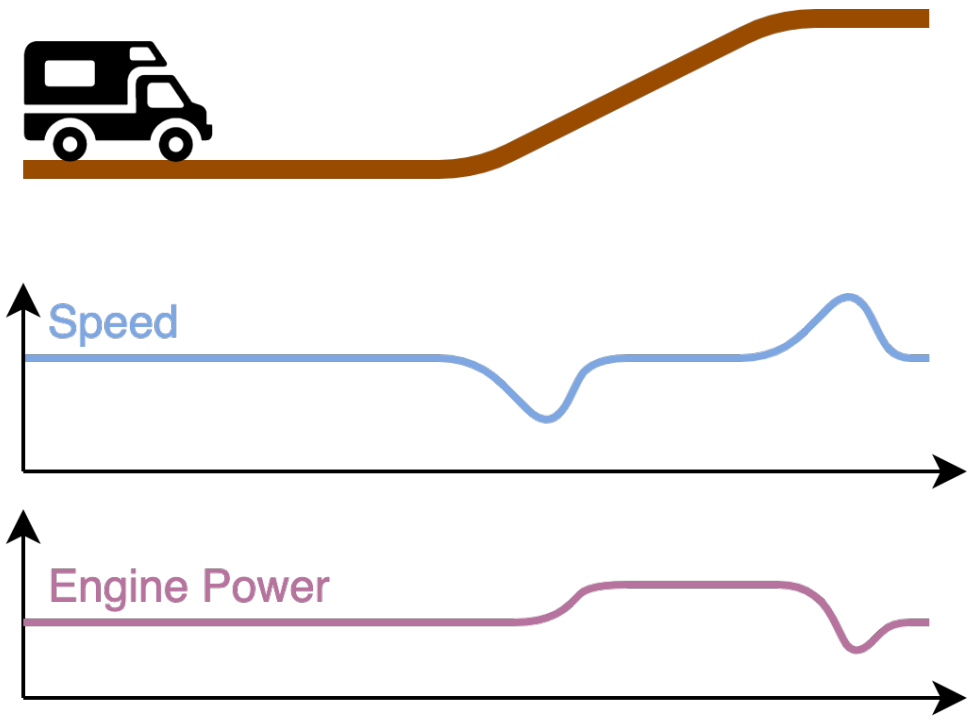
Problem: Inferring state models from black-box systems

- How to go about inferring states in a black-box controller?



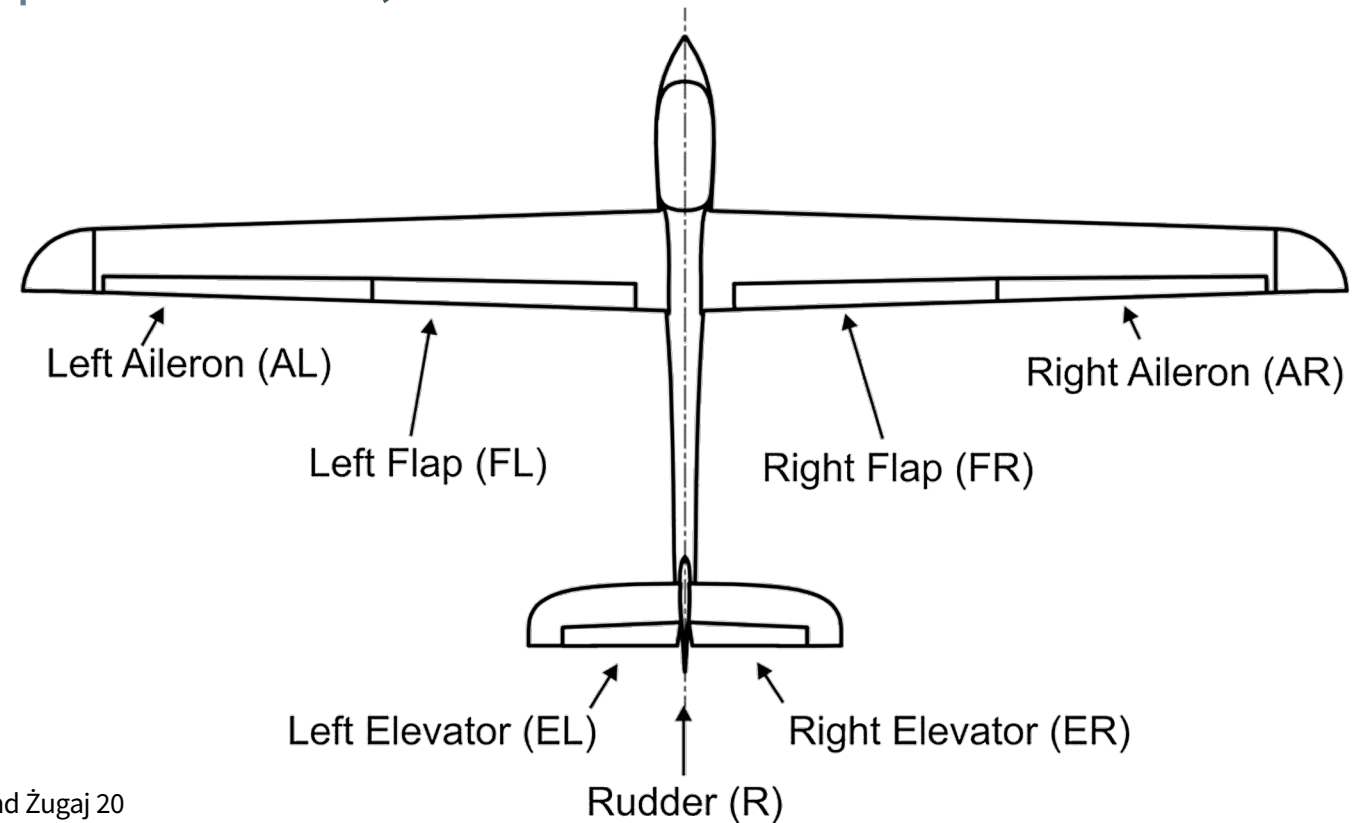
Solution Context: Feedback loop controllers

- Car cruise control
  - Goal: Keep current speed ≈ desired speed
  - Input: Speed
  - Output: Throttle increase/decrease



Solution Context: Feedback loop controllers

- Autopilot
  - Goal: Depends on the internal state
  - Input: Airspeed, AOA, attitude, MSL altitude, etc.
  - Outputs: Throttle, control surfaces' deflections



Aircraft illustration from Kopyt and Żugaj 20

Solution Context: Control loops in Autopilot (PID)

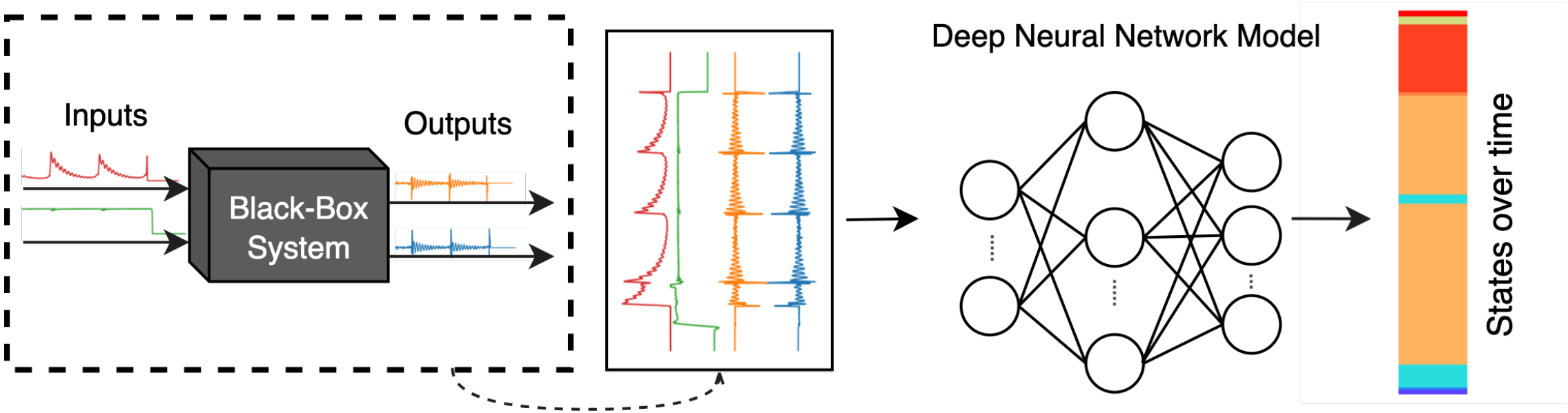
- "Goal: depends on the internal state":
  - During takeoff:
    - Heading: keep constant, align with runway
    - Throttle: keep constant, maximum
  - In cruise:
    - Heading: constant, based on flight plan
    - Throttle: reacts to altitude changes
    - Pitch: reacts to airspeed changes

- Hypothesis: The reverse should be possible
  - Observing which inputs/outputs are coupled at each moment to infer what the internal state is

Solution Context: Prior work

- Change point detection in time series is promising
- But they have limitations:
  - Assumptions on the data:
    - Univariate
    - One change point
    - Known number of change points
    - Specific distributions
  - Being too general:
    - Not exploiting the knowledge about the system

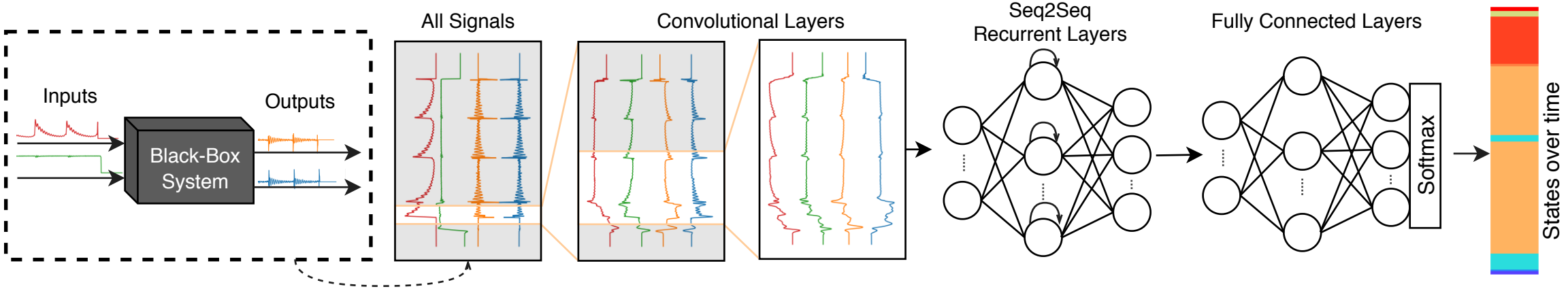
The Solution: A hybrid deep neural network



Training a neural network model that can infer the state from the stream of input and outputs

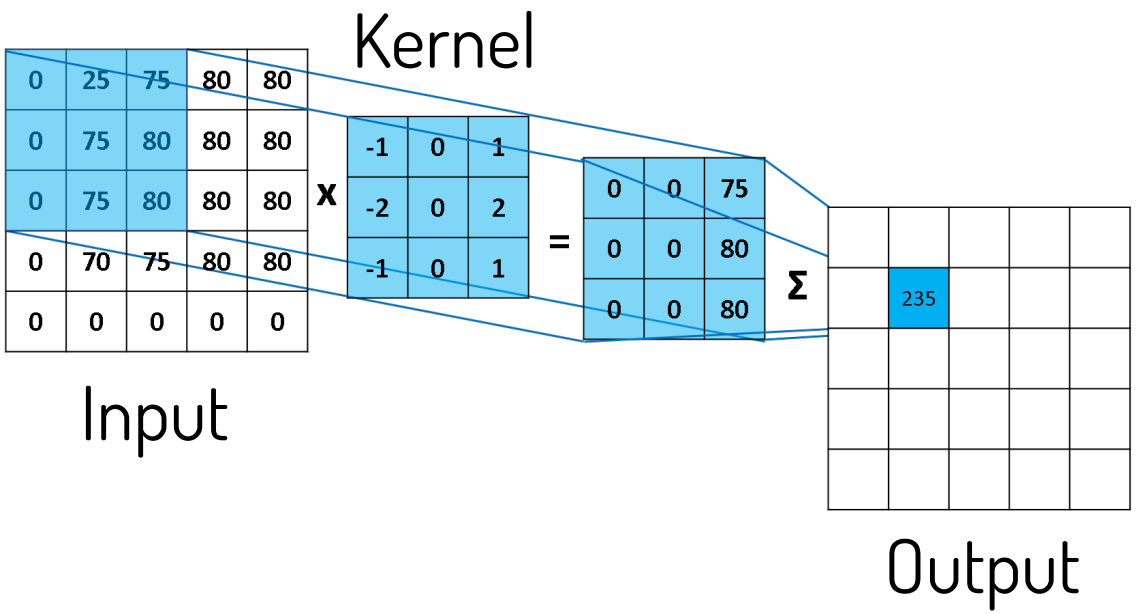
More on Hybrid Deep Neural Network Models

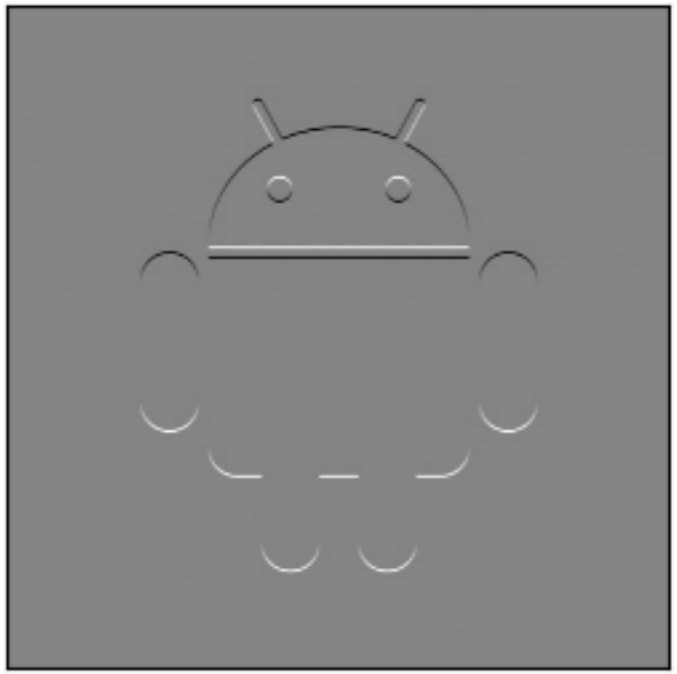
- Hybrid in this context: CNN + RNN
- Showed to be effective in similar problems



More on Hybrid Deep Neural Network Models

- Convolutional layers
  - Can detect local features (lines, circles, patterns, etc.)
  - Can learn preprocessing (rolling average, discreate derivative, etc.)





Images credit: mlnotebook

More on Hybrid Deep Neural Network Models

- Recurrent layers
  - Output depends on current input, and history: It has memory
  - Can discover long-term relations
  - Perfect for processing sequential data (text, time series, etc.)

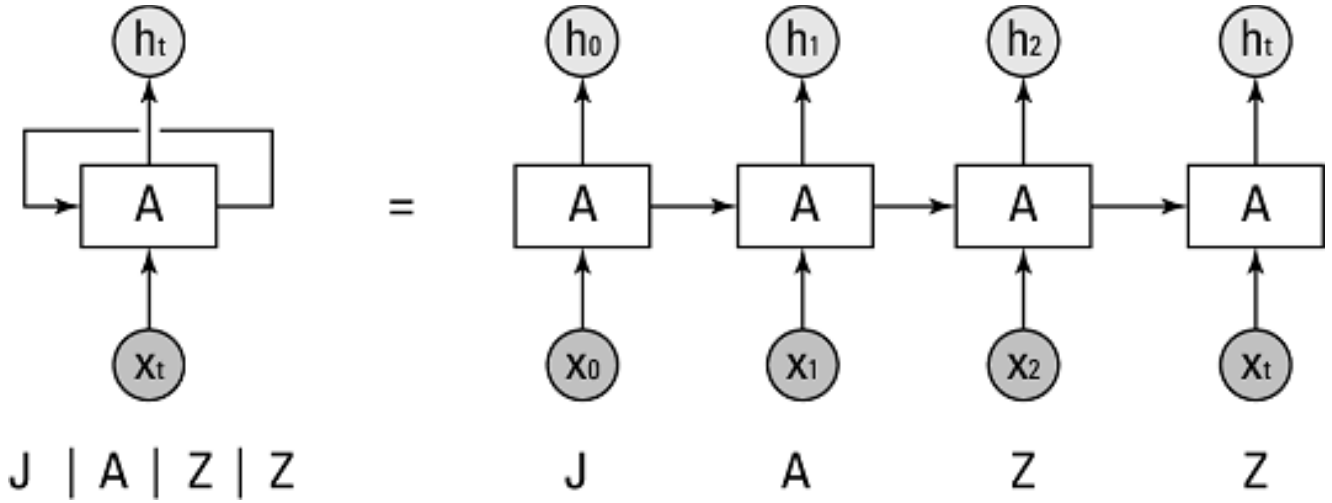
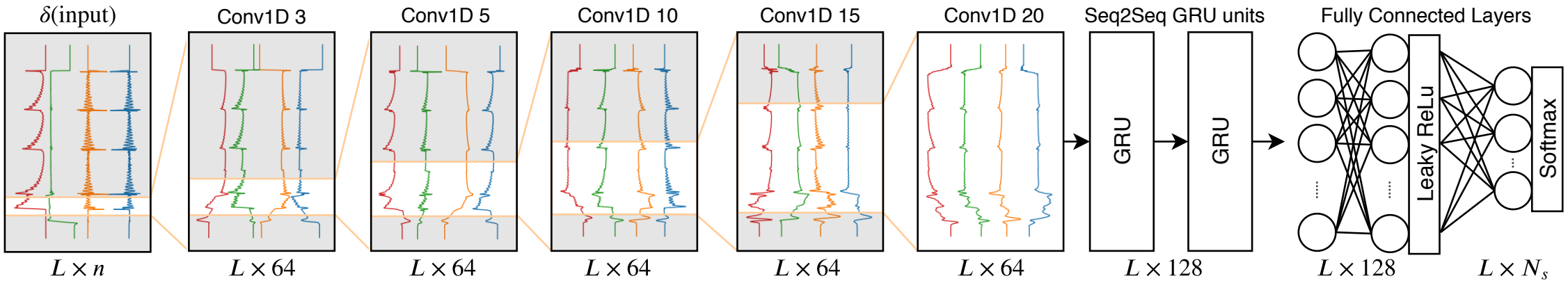


Image credit: Wiley

The final Solution



- Proposed architecture:
  - 5 Convolutions with increasing kernel size
  - 2 large Recurrent layers
- based on:
  - Similar architectures in literature
  - Experiments and tuning

Experiments and Results

State Change Detection

Internal State Detection

Non-Hybrid Models

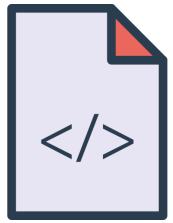
Case Study: Autopilot software

MicroPilot®

UAVs

1000+

Customers
In 85 Countries



500K

LoC in C

Icons made by DinosoftLabs from Flaticon is licensed by CC 3.0 BY

Experiment setup: Dataset

- 888 MicroPilot system tests
- In-house simulator
- 5 inputs + 5 outputs
- Test lengths: 200 – 20,000 samples (5Hz sampling rate)

Experiment

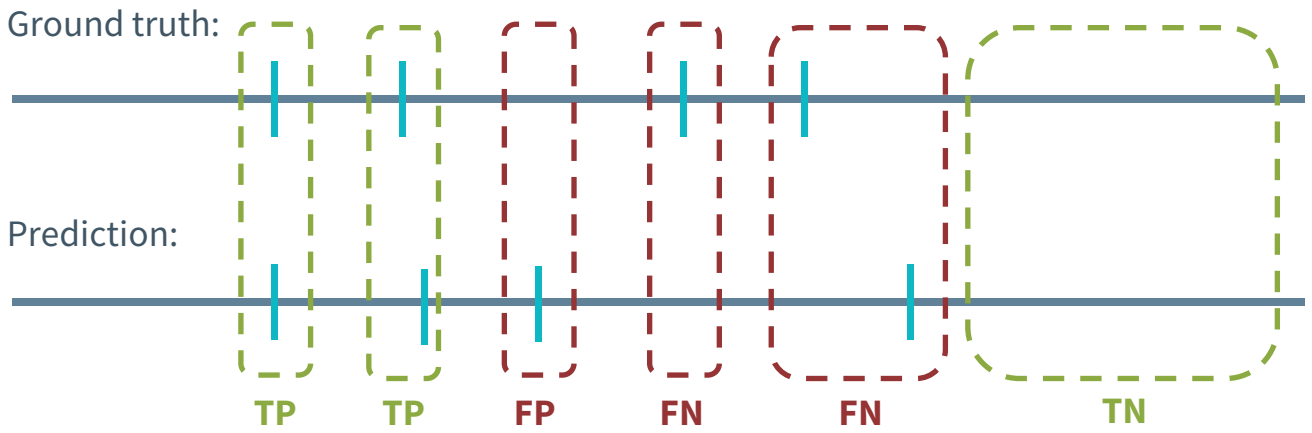
RQ1) How does the proposed technique perform in detecting the state changes?

- When the state change happens
- What the source and target states are

Experiment (RQ1)

- Metrics
- Baselines
- Design

Precision, Recall, and F1 Score with a tolerance



$$\text{Precision} = \frac{\text{TP}}{\text{P}} \qquad \text{Recall} = \frac{\text{TP}}{\text{T}} \qquad \text{F1} = 2\frac{\text{Prec} \times \text{Recall}}{\text{Prec} + \text{Recall}}$$

Experiment (RQ1)

State Change Detection

Internal State Detection

Non-Hybrid Models

- Metrics
- Baselines
- Design

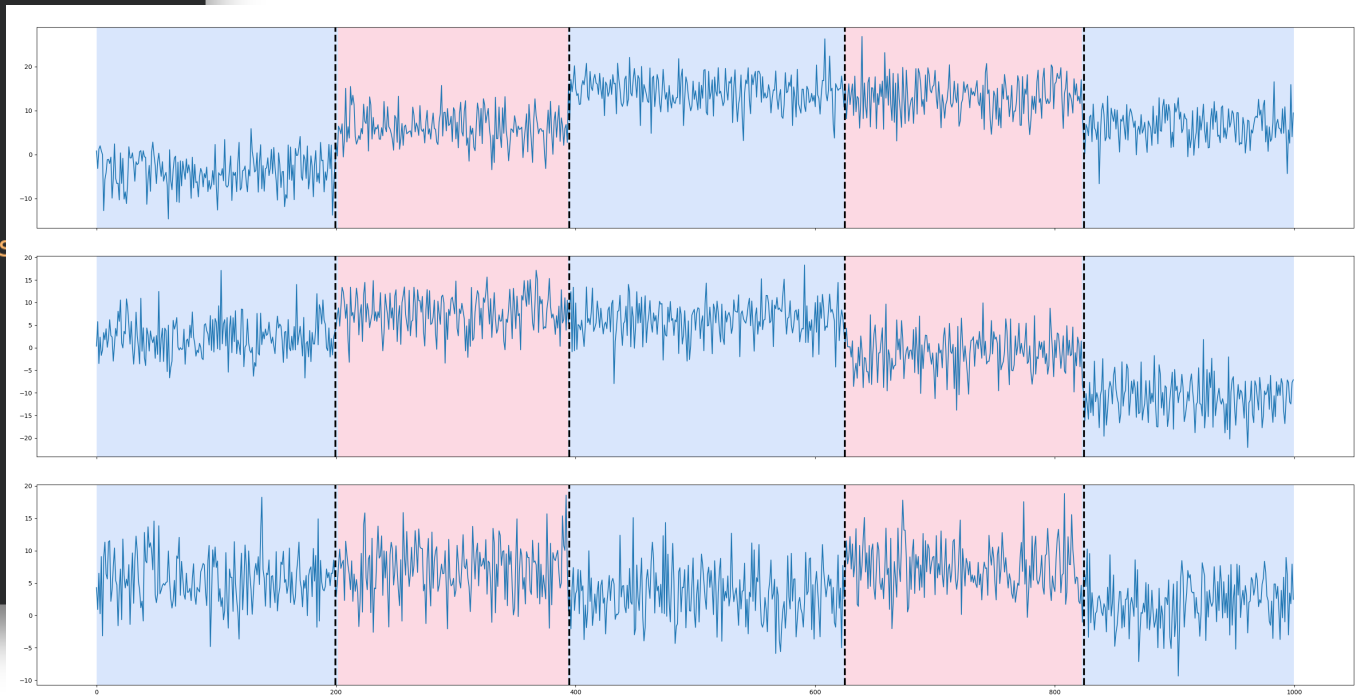
Ruptures library [Truong et al. ‘18]

```python
import matplotlib.pyplot as plt
import ruptures as rpt

# generate signal
n_samples, dim, sigma = 1000, 3, 4
n_bkps = 4  # number of breakpoints
signal, bkps = rpt.pw_constant(n_samples, dim, n_bkps, noise_s

# detection
algo = rpt.Pelt(model="rbf").fit(signal)
result = algo.predict(pen=10)

# display
rpt.display(signal, bkps, result)
plt.show()
```



Experiment (RQ1)

- Metrics
- Baselines
- Design

Different configurations of baseline algos

- Search Method:
  - ~~Pelt (Exact method)~~
  - Bottom-up Segmentation
  - Window based
- Penalty: 3 values
- Cost functions: 7 values

Results (RQ1)

State Change Detection | Internal State Detection | Non-Hybrid Models

Cost Function	Search Method	Penalty	Prec.	Recall $\tau = 1s$	F1	Prec.	Recall $\tau = 3s$	F1	Prec.	Recall $\tau = 5s$	F1
Autoregressive Model	Bottom Up	1000	10.43%	75.44%	18.33%	21.21%	80.32%	33.55%	28.94%	81.22%	42.68%
	Window Based	100	2.94%	3.98%	3.38%	8.53%	11.41%	9.76%	12.89%	17.54%	14.86%
Least Absolute Deviation	Bottom Up	500	7.32%	52.54%	12.85%	17.52%	87.73%	29.20%	25.02%	88.95%	39.05%
	Window Based	500	5.24%	8.31%	6.42%	15.20%	24.03%	18.62%	21.79%	38.25%	27.76%
Least Squared Deviation	Bottom Up	1000	7.44%	85.09%	13.68%	16.40%	89.81%	27.74%	24.16%	90.47%	38.14%
	Window Based	500	3.59%	6.79%	4.70%	10.27%	16.51%	12.66%	16.18%	26.84%	20.19%
Linear Model Change	Bottom Up	100	**37.59%**	28.98%	**32.73%**	**45.20%**	38.39%	**41.52%**	**48.07%**	41.36%	**44.46%**
	Window Based	500	6.70%	4.14%	5.12%	20.50%	13.05%	15.95%	38.78%	26.77%	31.67%
Gaussian Process Change	Bottom Up	100	3.77%	**92.23%**	7.25%	8.99%	**92.23%**	16.39%	13.53%	**92.23%**	23.60%
	Window Based	100	2.94%	3.95%	3.37%	8.69%	11.50%	9.90%	13.64%	18.30%	15.63%
Rank-based Cost Function	Bottom Up	100	13.45%	60.19%	21.98%	19.49%	80.10%	31.35%	22.98%	87.23%	36.38%
	Window Based	100	8.10%	13.70%	10.18%	15.72%	30.73%	20.80%	21.38%	46.64%	29.32%
Kernelized Mean Change	Bottom Up	100	4.13%	3.24%	3.63%	12.22%	8.14%	9.77%	15.38%	10.58%	12.54%
	Window Based	100	2.82%	3.00%	2.91%	10.14%	8.40%	9.19%	13.64%	12.61%	13.10%

Results (RQ1)

State Change Detection | Internal State Detection | Non-Hybrid Models

τ	Prec.	Recall	F1 score		Baseline F1
1s	56.77%	79.32%	66.18%	+102%	32.73%
3s	69.58%	88.88%	78.06%	+88%	41.52%
5s	79.82%	91.87%	85.42%	+92%	44.46%

Linear Model Change	Bottom Up	100	**37.59%**	28.98%	**32.73%**	**45.20%**	38.39%	**41.52%**	**48.07%**	41.36%	**44.46%**
	Window Based	500	6.70%	4.14%	5.12%	20.50%	13.05%	15.95%	38.78%	26.77%	31.67%
Gaussian Process Change	Bottom Up	100	3.77%	**92.23%**	7.25%	8.99%	**92.23%**	16.39%	13.53%	**92.23%**	23.60%
	Window Based	100	2.94%	3.95%	3.37%	8.69%	11.50%	9.90%	13.64%	18.30%	15.63%
Rank-based Cost Function	Bottom Up	100	13.45%	60.19%	21.98%	19.49%	80.10%	31.35%	22.98%	87.23%	36.38%
	Window Based	100	8.10%	13.70%	10.18%	15.72%	30.73%	20.80%	21.38%	46.64%	29.32%
Kernelized Mean Change	Bottom Up	100	4.13%	3.24%	3.63%	12.22%	8.14%	9.77%	15.38%	10.58%	12.54%
	Window Based	100	2.82%	3.00%	2.91%	10.14%	8.40%	9.19%	13.64%	12.61%	13.10%

Results (RQ1)

Baselines	Our Method
16 CPUs	1 GPU
64 GB Memory	8 GB Memory
12H+	1H training (once) + <1m prediction

Results (RQ1)

Compared to traditional CPD algorithms the proposed model is both **faster** and **less resource** intensive; All while it showed an improvement of **88 to 102%**, measured by F1 score, compared to the best baseline.

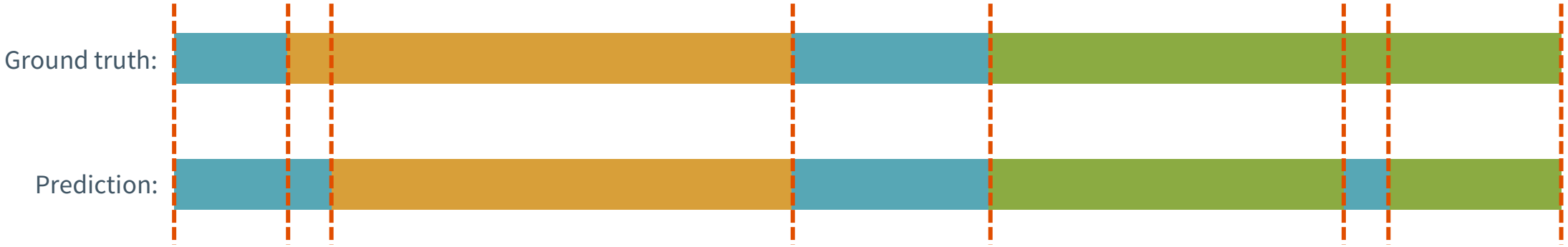
RQ2) How well does the proposed technique predict the internal state of the system?

RQ1 was a binary classification, here we have a multiclass classification problem

Experiment (RQ2)

- Metrics
- Baselines
- Design

- Multi-class classification
  - Classes = The Possible States
  - Average of precision and recalls per class



Experiment (RQ2)

- Metrics
- Baselines
- Design

Traditional ML Classifiers

- Ridge Classifier
- 3 Decision Trees

Experiment (RQ2)

State Change Detection | Internal State Detection | Non-Hybrid Models

- Metrics
- Baselines
- Design

Input: A rolling window of size w over the 10 inputs: $10w$ features

$$w \in \{3, 5, 10, 15, 20\}$$

48 Configurations overall

Results (RQ2)

- Larger window size is better, because there are many long-term relations to discover
- Training complexity increases dramatically with growing window size

w	Classifier	Max Depth	Max Features	Prec.	Recall	F1
3	Ridge	-	-	71.39%	20.73%	32.13%
3	DT	-	-	69.21%	82.36%	75.21%
5	Ridge	-	-	69.15%	21.89%	33.26%
5	DT	100	-	68.37%	**83.16%**	75.04%
10	Ridge	-	-	71.97%	24.02%	36.02%
10	DT	260	-	67.94%	79.14%	73.12%
15	Ridge	-	-	76.87	25.90%	38.75%
15	DT	-	$\sqrt{10w}$	69.06%	80.76%	74.45%
20	Ridge	-	-	**80.38%**	26.50%	39.86%
20	DT	175	$\sqrt{10w}$	73.21%	82.16%	**77.42%**
	My Approach			**86.29%**	**95.04%**	**90.45%**
				+18%	+16%	+17%

Results (RQ2)

- Larger window size is better, because there are many long-

w	Classifier	Max Depth	Max Features	Prec.	Recall	F1
3	Ridge	-	-	71.39%	20.73%	32.13%
20	Ridge	-	-	**80.38%**	26.50%	39.86%
20	DT	175	$\sqrt{10w}$	73.21%	82.16%	**77.42%**
	My Approach			**86.29%**	**95.04%**	**90.45%**
				+18%	+16%	+17%

effective in discovering local features

The proposed model, improved classical ML models' best performance by up to **17%**, measured by F1 score, in a **shorter execution time** and using **less computational resources**.

RQ3) How much does the proposed model owe its performance to being a hybrid model?

Experiment (RQ3)

- Fully convolutional
- Fully recurrent

- Baselines
- Metrics

Experiment (RQ3)

- Metrics: the same as RQ1 and RQ2
  - CPD Precision and recall with 3 tolerances
  - Classification precision and recall

Results (RQ3)

- Larger recurrent layers is more effective than larger convolutions
  - Discovering long-term relations
  - Training larger convolutions requires more data

	τ	RNN only	CNN only	Full Model
Precision	1s	45.31%	38.12%	53.84%
Recall	1s	60.56%	58.50%	67.94%
F1	1s	51.84%	46.16%	60.06%
Precision	3s	56.06%	50.97%	72.00%
Recall	3s	78.00%	69.12%	88.56%
F1	3s	65.25%	58.69%	79.44%
Precision	5s	68.75%	67.38%	78.56%
Recall	5s	81.81%	73.75%	93.75%
F1	5s	74.69%	70.44%	85.50%
Classification Prec.		81.56%	71.56%	88.44%
Classification Recall		91.88%	86.88%	94.50%
Classification F1		86.44%	78.44%	91.38%

Results (RQ3)

- Larger recurrent layers is more effective than

The hybrid architecture performs **better than** a comparable **RNN** model or fully **convolutional** model, however the **recurrent section plays a more important** role in the model's performance.

	τ	RNN only	CNN only	Full Model
Precision	1s	45.31%	38.12%	53.84%
F1	5s	74.69%	70.44%	85.50%
Classification Prec.		81.56%	71.56%	88.44%
Classification Recall		91.88%	86.88%	94.50%
Classification F1		86.44%	78.44%	91.38%

Potential future extensions

- More case studies
- Applying it to other types of controller systems
- Using the inferred model in a downstream task
- Systematic hyper-parameter tuning

Summary

18 **Case Study:** Autopilot software

MicroPilot®

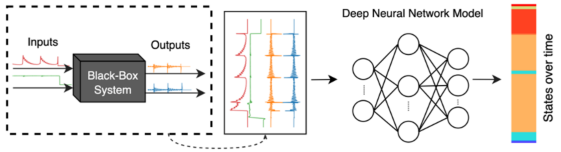
UAVs

1000+ Customers In 85 Countries

500K LoC in C

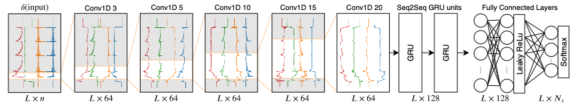
Icons made by DinosoftLabs from Flaticon is licensed by CC 3.0 BY

12 **The Solution:** A hybrid deep neural network



Training a neural network model that can infer the state from the stream of input and outputs

16 The final Solution



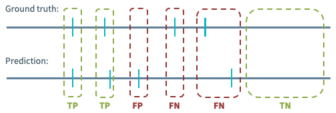
- Proposed architecture:
  - 5 Convolutions with increasing kernel size
  - 2 large Recurrent layers
- based on:
  - Similar architectures in literature
  - Experiments and tuning

21 Experiment (RQ1)

State Change Detection | Internal State Detection | Non-Hybrid Models

- Metrics
- Baselines
- Design

Precision, Recall, and F1 Score with a tolerance



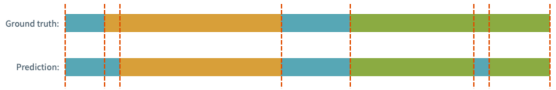
$$\text{Precision} = \frac{\text{TP}}{\text{P}} \qquad \text{Recall} = \frac{\text{TP}}{\text{T}} \qquad \text{F1} = 2\frac{\text{Prec} \times \text{Recall}}{\text{Prec} + \text{Recall}}$$

29 Experiment (RQ2)

State Change Detection | Internal State Detection | Non-Hybrid Models

- Metrics
- Baselines
- Design

- Multi-class classification
  - Classes = The Possible States
  - Average of precision and recalls per class



37 Results (RQ3)

State Change Detection | Internal State Detection | Non-Hybrid Models

- Larger recurrent layers is more effective than larger convolutions
  - Discovering long-term relations
  - Training larger convolutions requires more data

	τ	RNN only	CNN only	Full Model
Precision	1s	45.31%	38.12%	53.84%
Recall	1s	60.56%	58.50%	67.94%
F1	1s	51.84%	46.16%	60.06%
Precision	3s	56.06%	50.97%	72.00%
Recall	3s	78.00%	69.12%	88.56%
F1	3s	65.25%	58.69%	79.44%
Precision	5s	68.75%	67.38%	78.56%
Recall	5s	81.81%	73.75%	93.75%
F1	5s	74.69%	70.44%	85.50%
Classification Prec.		81.56%	71.56%	88.44%
Classification Recall		91.88%	86.88%	94.50%
Classification F1		86.44%	78.44%	91.38%