Evolving Neural Network Ensembles for Reliable Time Series Prediction

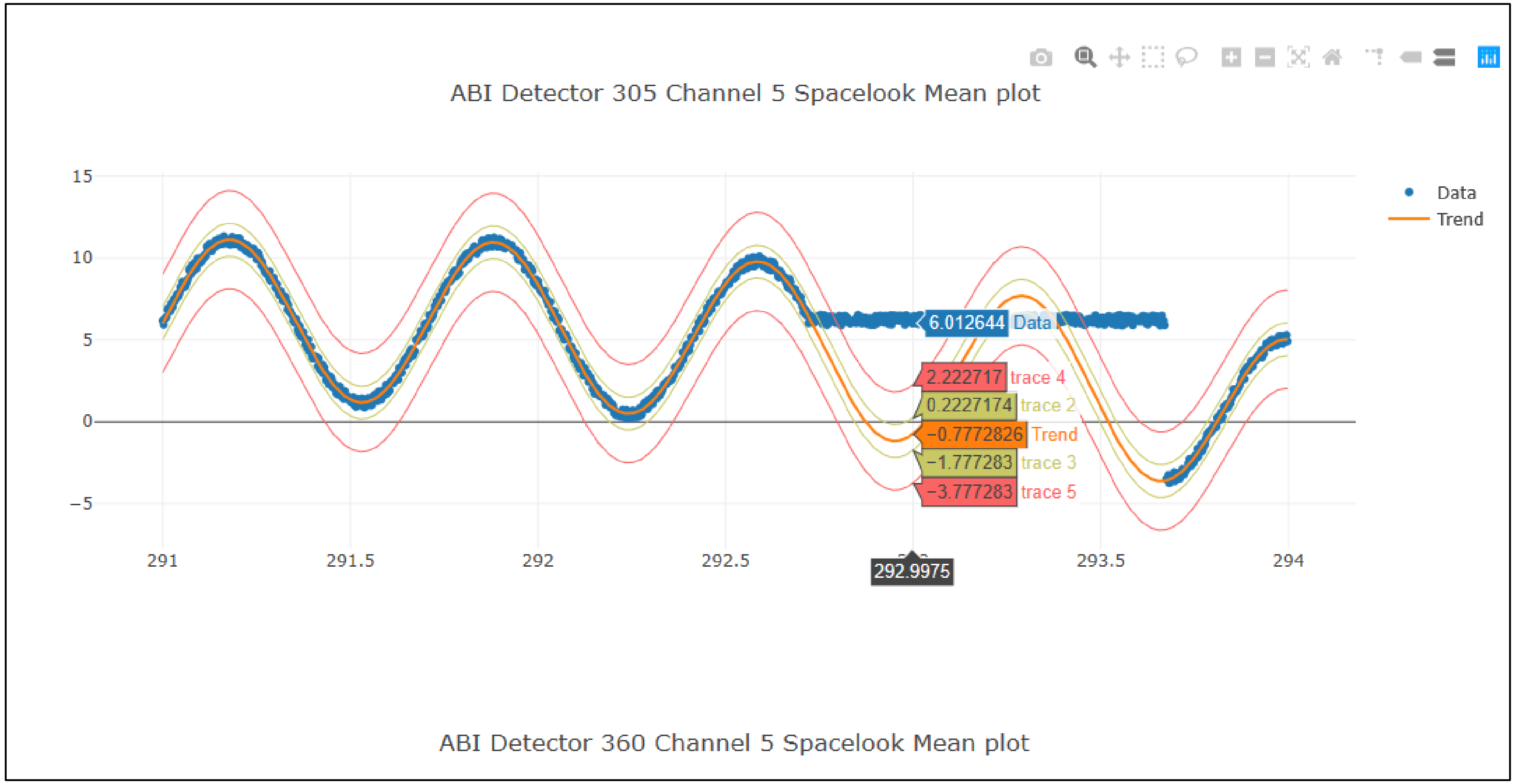
Philip Feldman, PhD

ASRC Federal

3.3.2020

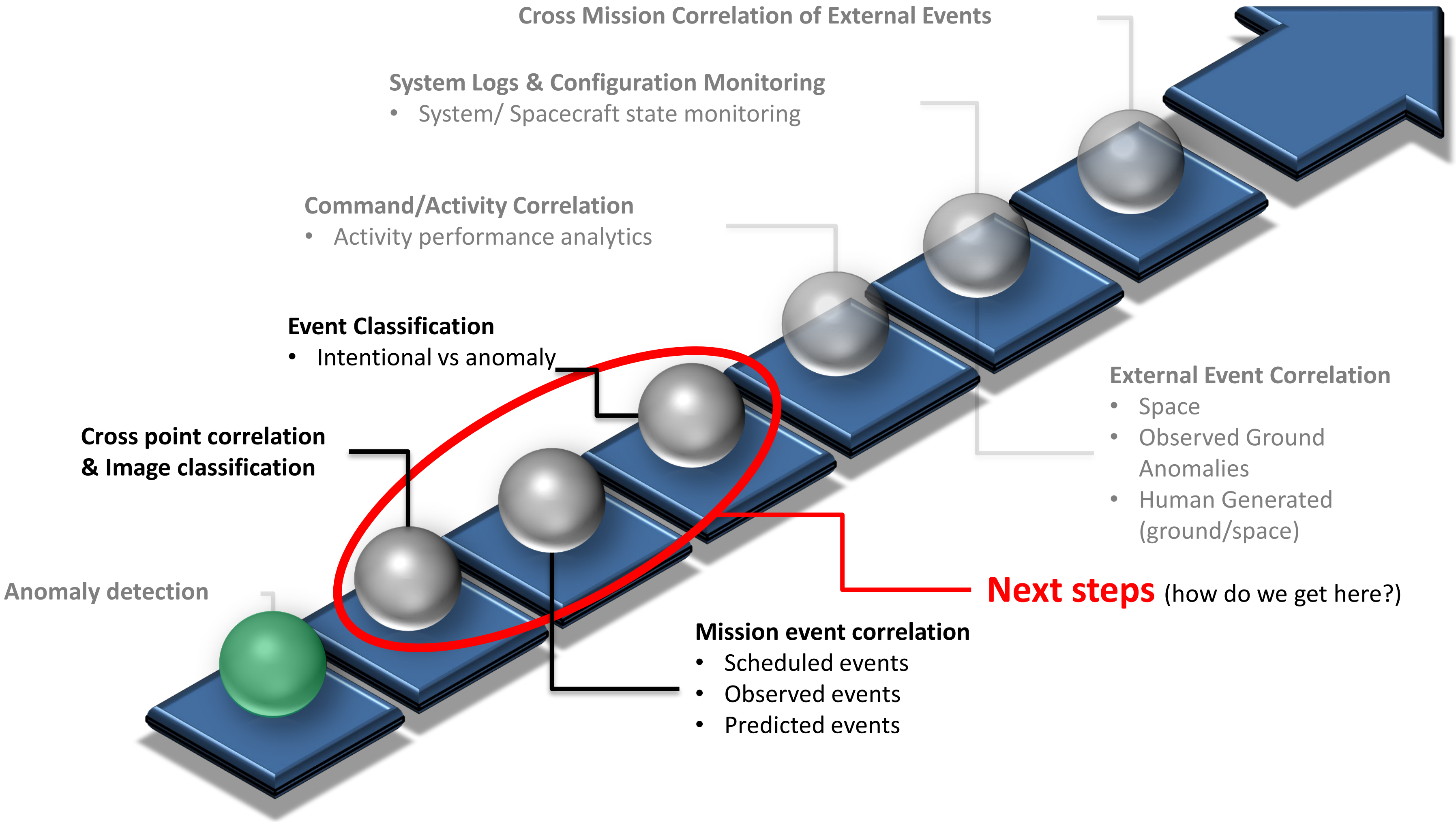
Background

- AIMS: Low-level monitoring and trending for single mnemonics using machine learning

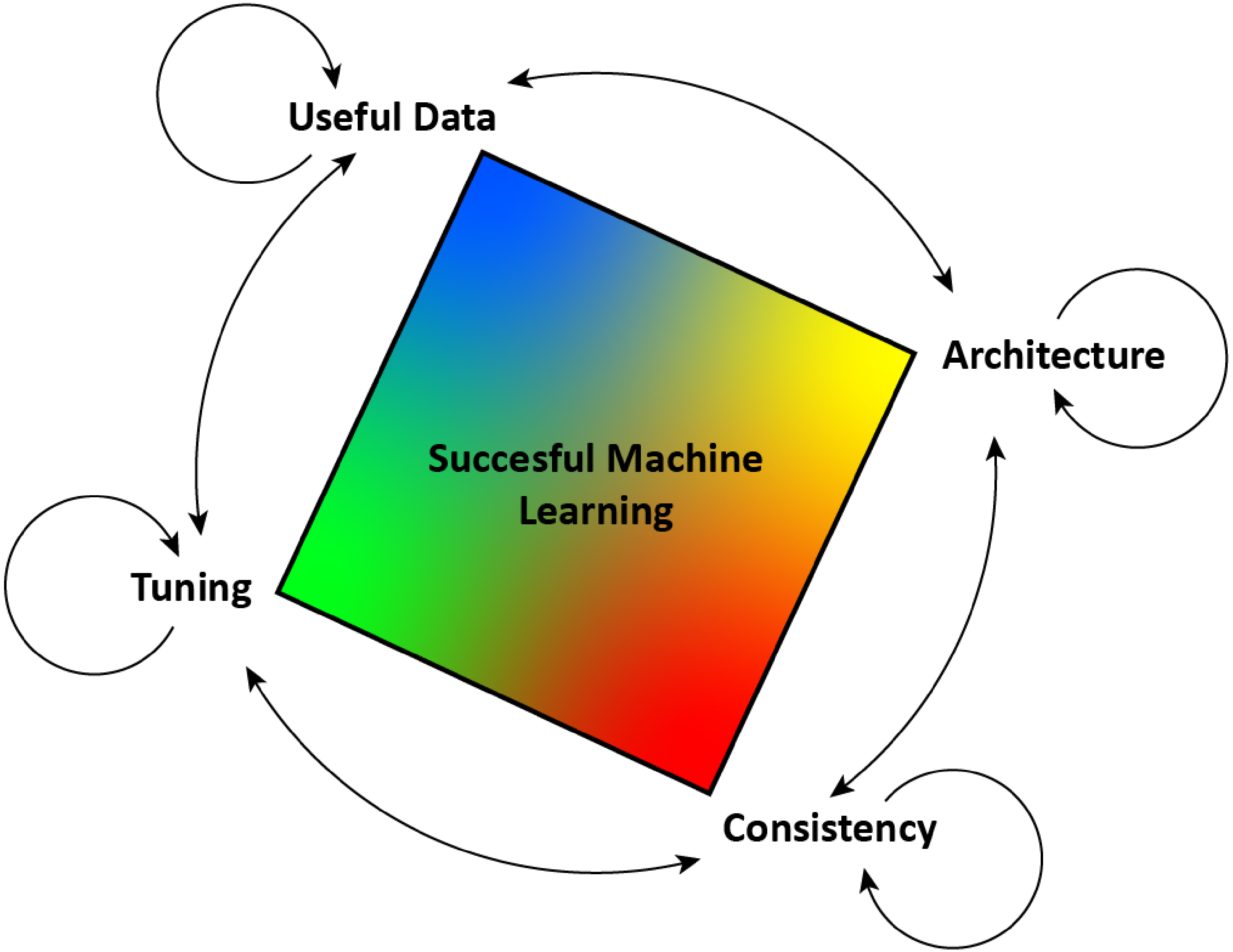


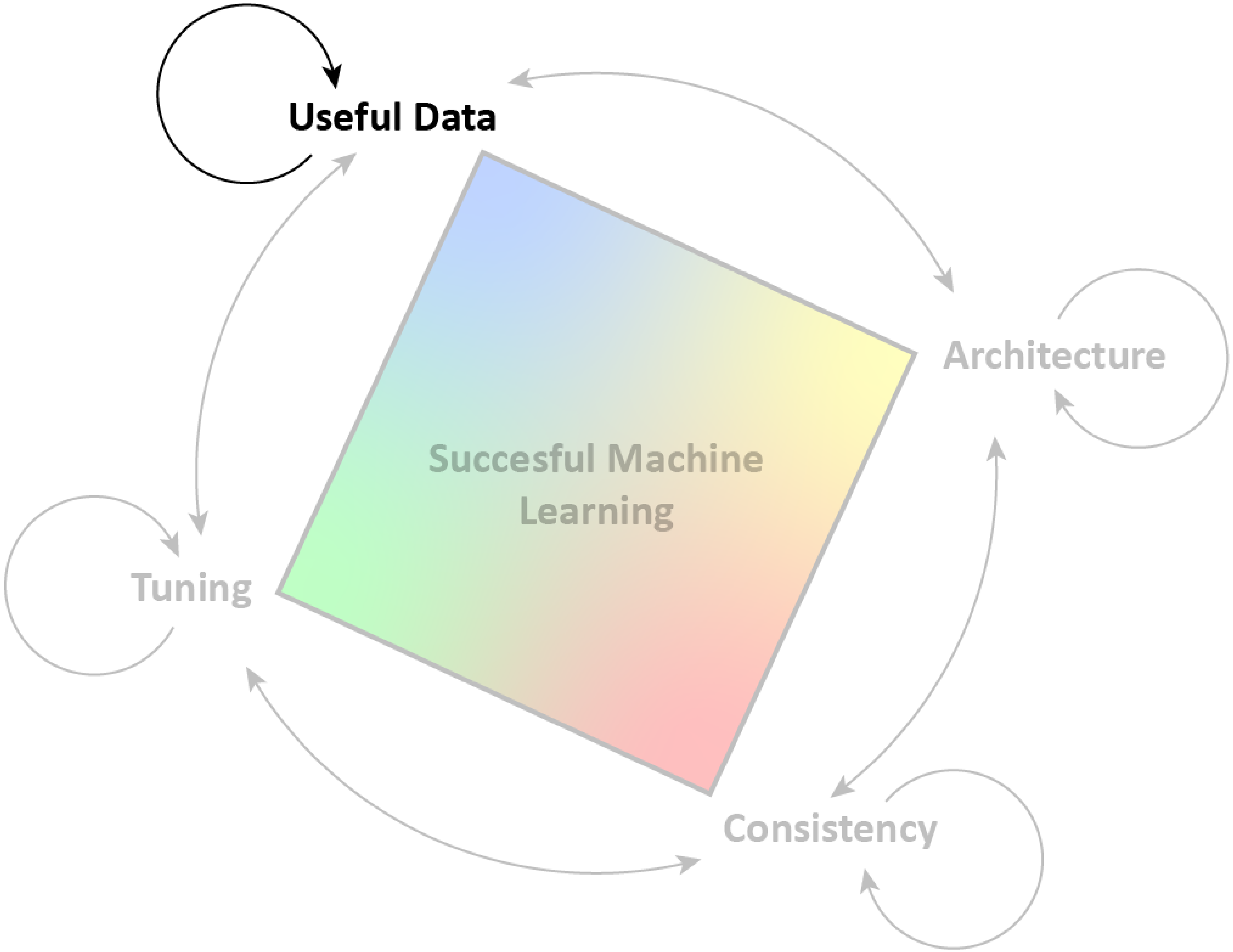
- Goal: High-level monitoring and trending across multiple mnemonics

ASRC Federal Machine Learning Evolution



The 4 parts of successful machine learning




Useful Data
Architecture
Succesful Machine Learning
Tuning
Consistency

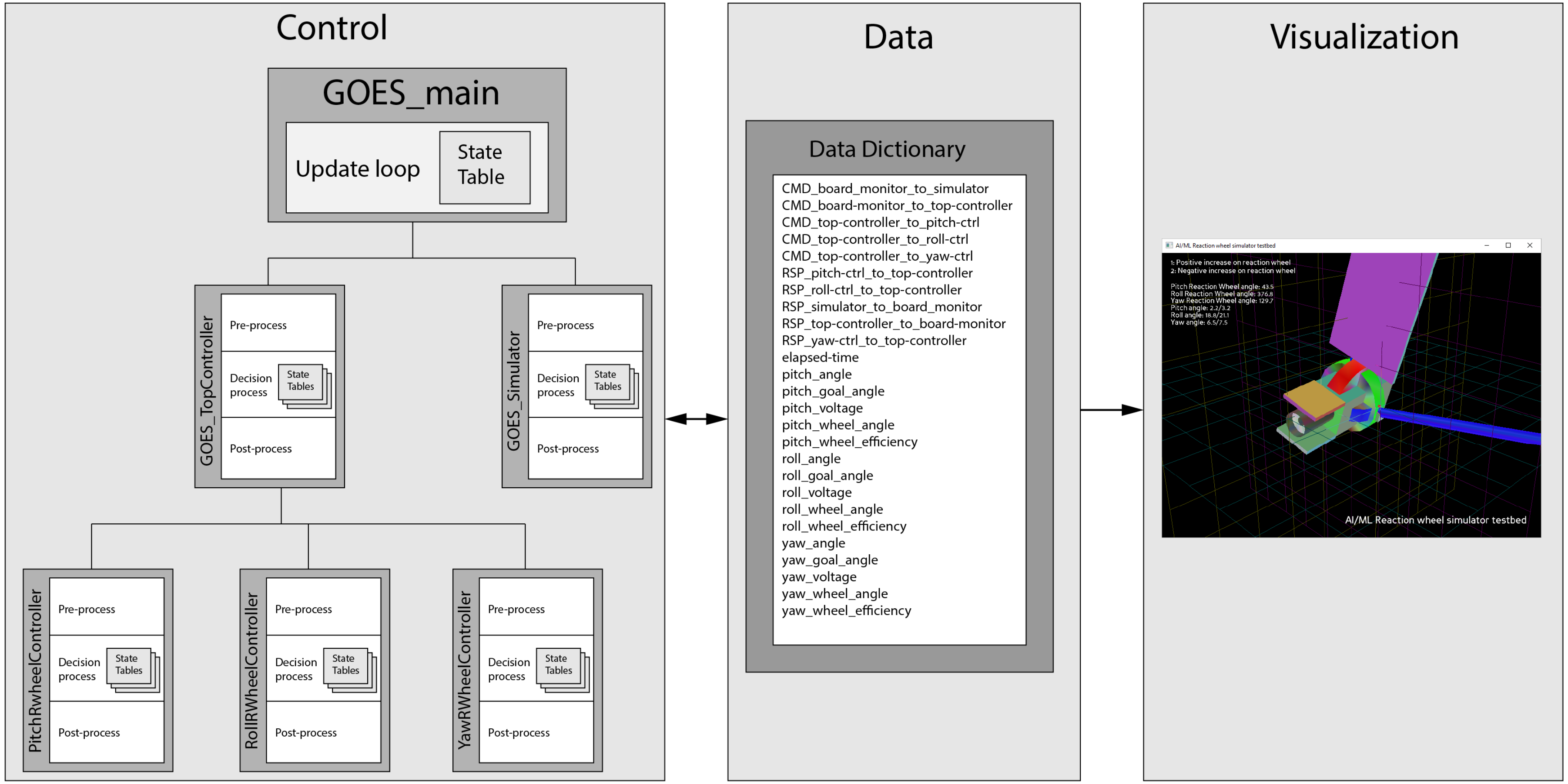
Getting useful data

1. Accidents in real life are rare.
    1. Your chance of a fatal car wreck is 0.00000125% per mile
    2. Your chance of a fatal plane crash is 0.0000000007% per mile
2. Machine learning is based on meeting an objective function, such as accuracy prediction.
    1. Assume no airplane fatalities *ever*: 99.99999999983% accurate
    2. Assume no auto fatalities *ever*: 99.99999875% accurate
3. Simulation lets us change those odds to 50-50, where machines learn best:

Accident simulator *par excellence* – Grand Theft Auto

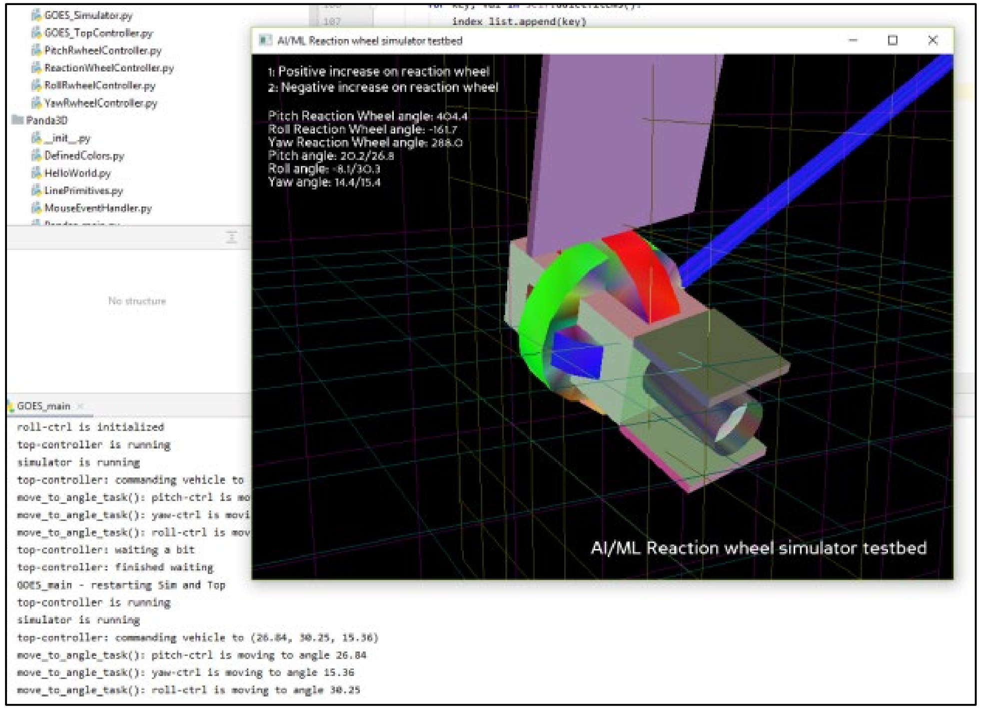
GOES-R Synthetic data generation

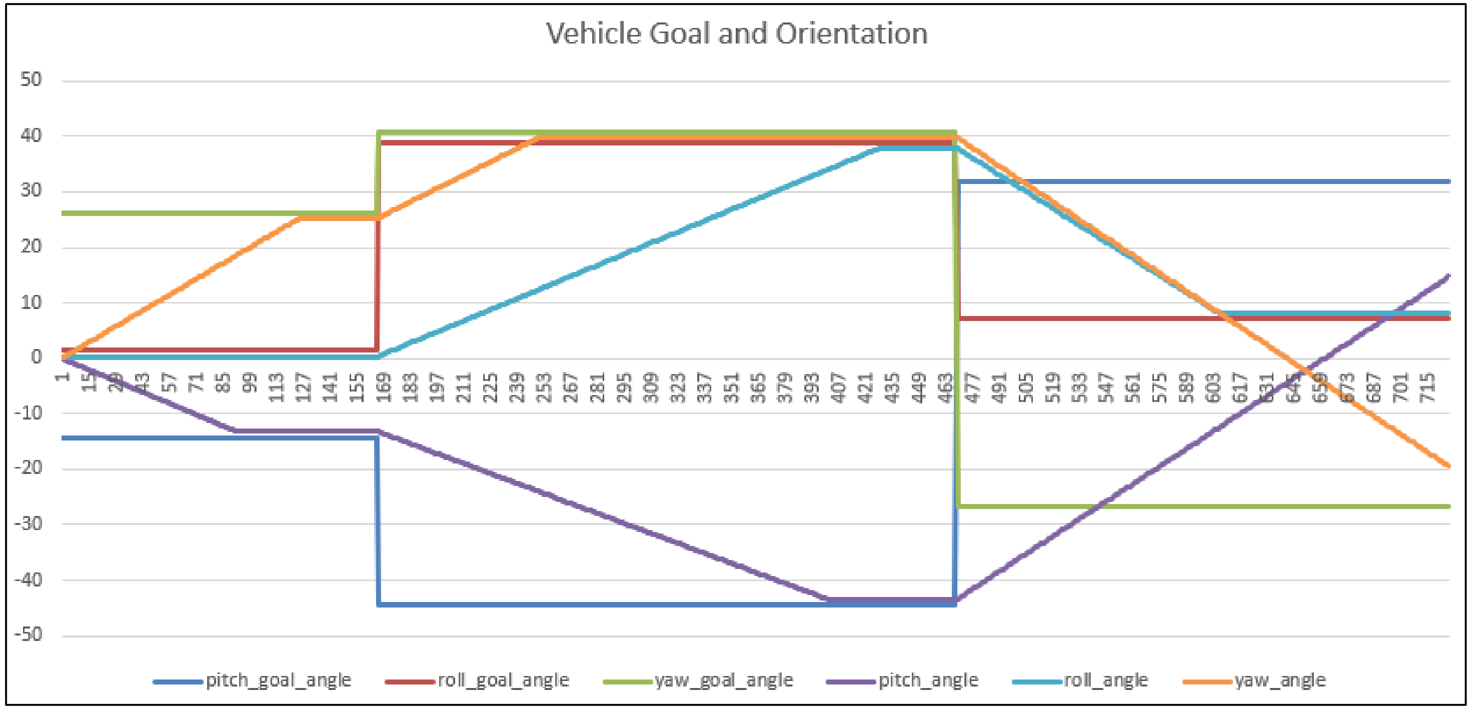
- Designed and implemented a proof-of-concept *control* and *simulation* API
- First use case: Reaction wheel (RW) degradation and failure

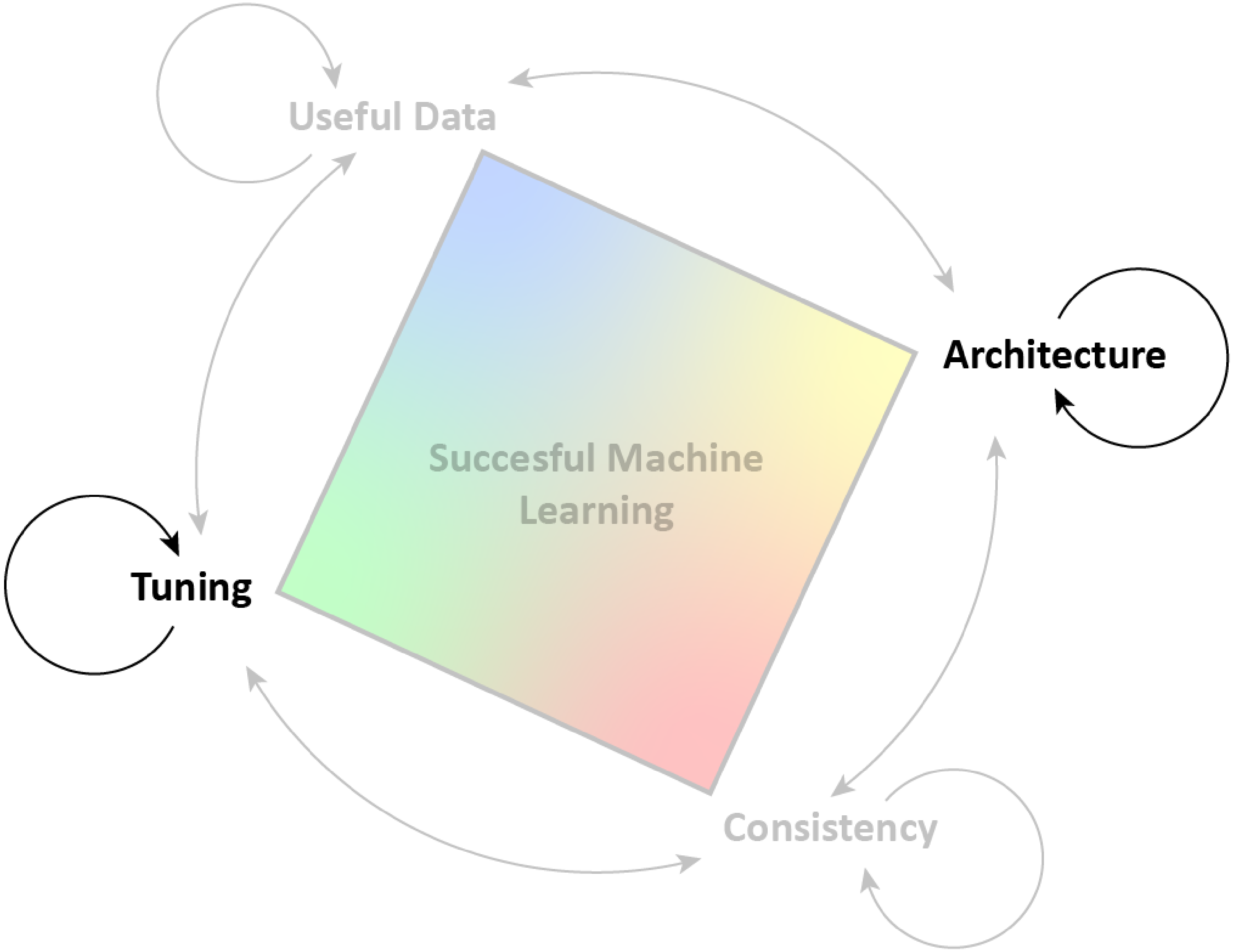


Creating tagged data

- Reaction wheel speed and vehicle slew rate is generated for random efficiencies (50% - 100%)
- The simulation creates an *input vector* of slew rates
- The simulation creates an *output vector* of RW efficiency
- Data is stored for train/test





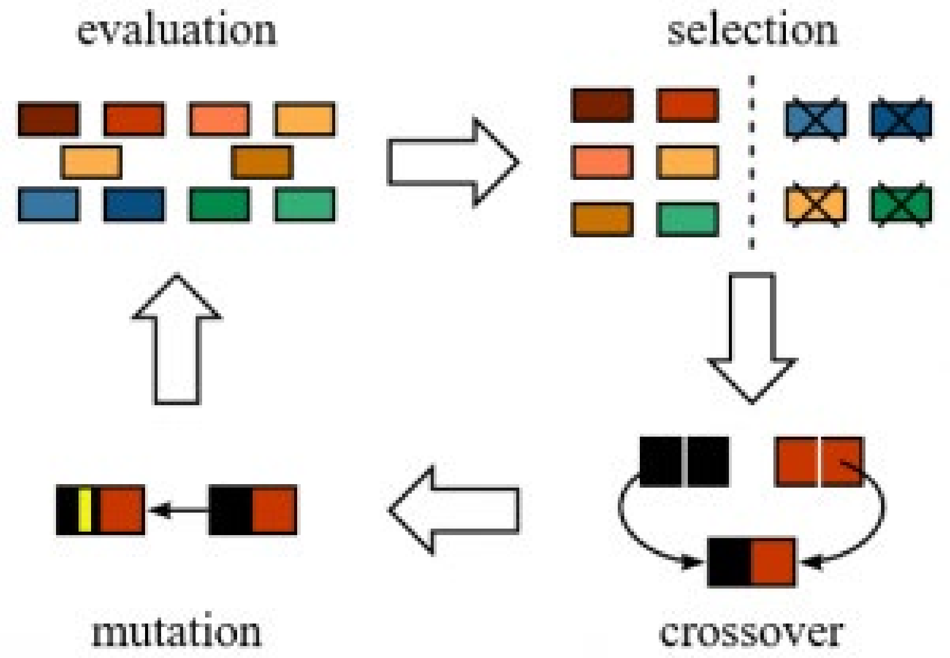

Useful Data
Architecture
Succesful Machine Learning
Tuning
Consistency

Architecture and tuning

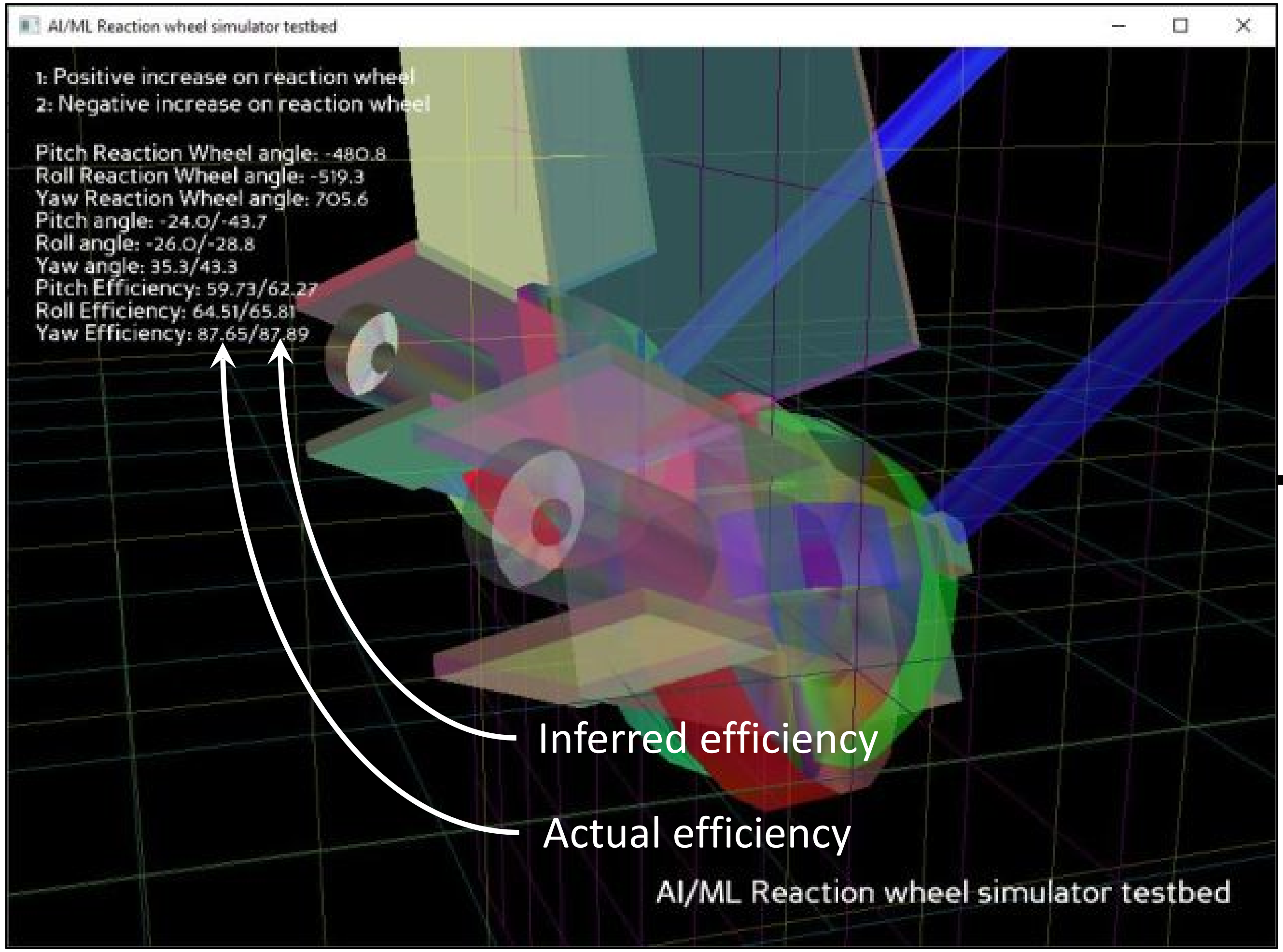
- Initial architecture guess: 1 layers of 100 neuron MLPs
- Initial hyperparameter guess:
  - Optimizer Adam, with learning rate of 0.01
  - Batch size = 10
  - Epochs = 40

- Final architecture : 2 layers of 230 neuron MLPs
- Initial hyperparameter guess:
  - Optimizer Adam, with learning rate of 0.01
  - Batch size = 13
  - Epochs = 70

```
    v1 = VA.EvolveAxis("X", VA.ValueAxisType.FLOAT, min=-5, max=5, step=0.25)
    v2 = VA.EvolveAxis("Y", VA.ValueAxisType.FLOAT, min=-5, max=5, step=0.25)

    vzvals = VA.EvolveAxis("Zvals1", VA.ValueAxisType.FLOAT, parent=vzfunc)
    vzvals = VA.EvolveAxis("Zvals2", VA.ValueAxisType.FLOAT, parent=vzfunc)

    eo = EvolutionaryOpimizer(keep_percent=.5, threads=0)
    eo.add_axis(v1)
    eo.add_axis(v2)

    eo.create_intital_genomes(10)

    evolve_list = []
    num_generations = 20
    for i in range(num_generations):
        fitness = eo.run_optimizer(example_evaluation_function,
example_save_function)
        evolve_list.append(fitness)
```



Real-time inference

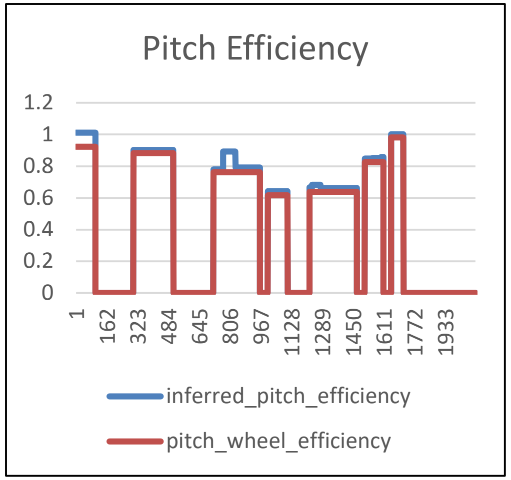


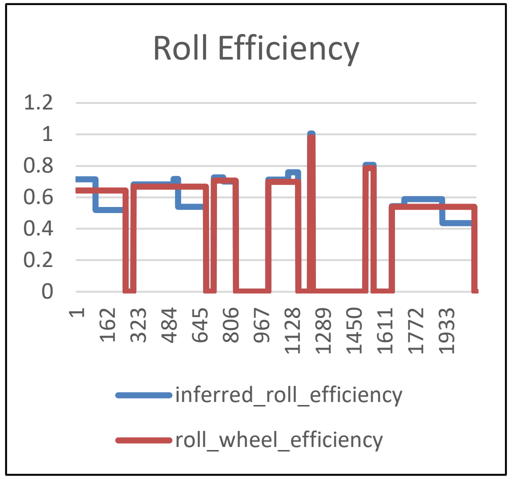
Simulation run

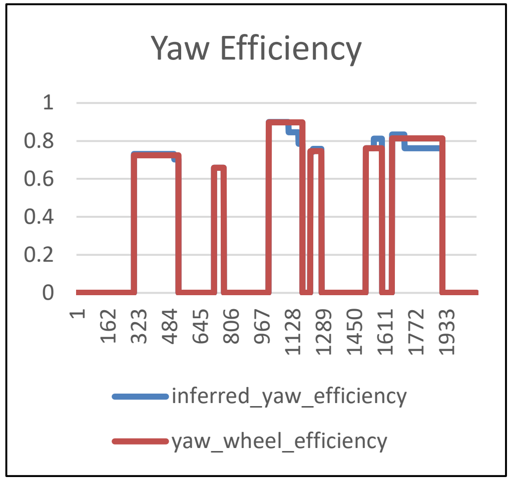
	A	B	C	D	E	F	G	H	I	J	K	L
1		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2	elapsed-time	0.1	0.1	0.2	0.2	0.3	0.3	0.4	0.4	0.5	0.5	0.6
3	inferred_pitch_efficiency	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4	inferred_roll_efficiency	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
5	inferred_yaw_efficiency	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	inferred_pitch	-0.1	-0.3	-0.6	-0.8	-1.1	-1.3	-1.6	-1.8	-2.1	-2.3	-2.6
7	inferred_roll	0.0	0.2	0.4	0.6	0.7	0.9	1.1	1.3	1.5	1.6	1.8
8	inferred_yaw	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	pitch_goal_angle	-24.5	-24.5	-24.5	-24.5	-24.5	-24.5	-24.5	-24.5	-24.5	-24.5	-24.5
10	roll_goal_angle	42.5	42.5	42.5	42.5	42.5	42.5	42.5	42.5	42.5	42.5	42.5
11	yaw_goal_angle	-0.3	-0.3	-0.3	-0.3	-0.3	-0.3	-0.3	-0.3	-0.3	-0.3	-0.3
12	CMD_board-monitor_to_top-controller	RUN	RUN	RUN	RUN	RUN	RUN	RUN	RUN	RUN	RUN	RUN
13	RSP_top-controller_to_board-monitor	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU
14	pitch_wheel_angle	-1.8	-6.5	-11.1	-15.7	-20.3	-25.0	-29.6	-34.2	-38.8	-43.4	-48.1
15	roll_wheel_angle	1.3	4.5	7.7	10.9	14.1	17.4	20.6	23.8	27.0	30.2	33.4
16	yaw_wheel_angle	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17	pitch_wheel_efficiency	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
18	roll_wheel_efficiency	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
19	yaw_wheel_efficiency	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
20	pitch_angle	-0.1	-0.3	-0.6	-0.8	-1.0	-1.2	-1.5	-1.7	-1.9	-2.2	-2.4
21	roll_angle	0.1	0.2	0.4	0.5	0.7	0.9	1.0	1.2	1.4	1.5	1.7
22	yaw_angle	0	0	0	0	0	0	0	0	0	0	0
23	rw_mass	1	1	1	1	1	1	1	1	1	1	1
24	vehicle_mass	20	20	20	20	20	20	20	20	20	20	20
25	CMD_board_monitor_to_simulator	RUN	RUN	RUN	RUN	RUN	RUN	RUN	RUN	RUN	RUN	RUN
26	RSP_simulator_to_board_monitor	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU
27	pitch_voltage	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100
28	yaw_voltage	0	0	0	0	0	0	0	0	0	0	0
29	roll_voltage	100	100	100	100	100	100	100	100	100	100	100
30	CMD_top-controller_to_pitch-ctrl	MOVI	MOVI	MOVI	MOVI	MOVI	MOVI	MOVI	MOVI	MOVI	MOVI	MOVI
31	RSP_pitch-ctrl_to_top-controller	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU
32	CMD_top-controller_to_yaw-ctrl	MOVI	MOVI	MOVI	MOVI	MOVI	MOVI	MOVI	MOVI	MOVI	MOVI	MOVI
33	RSP_yaw-ctrl_to_top-controller	DONE	DONE	DONE	DONE	DONE	DONE	DONE	DONE	DONE	DONE	DONE
34	CMD_top-controller_to_roll-ctrl	MOVI	MOVI	MOVI	MOVI	MOVI	MOVI	MOVI	MOVI	MOVI	MOVI	MOVI
35	RSP_roll-ctrl_to_top-controller	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU	EXECU

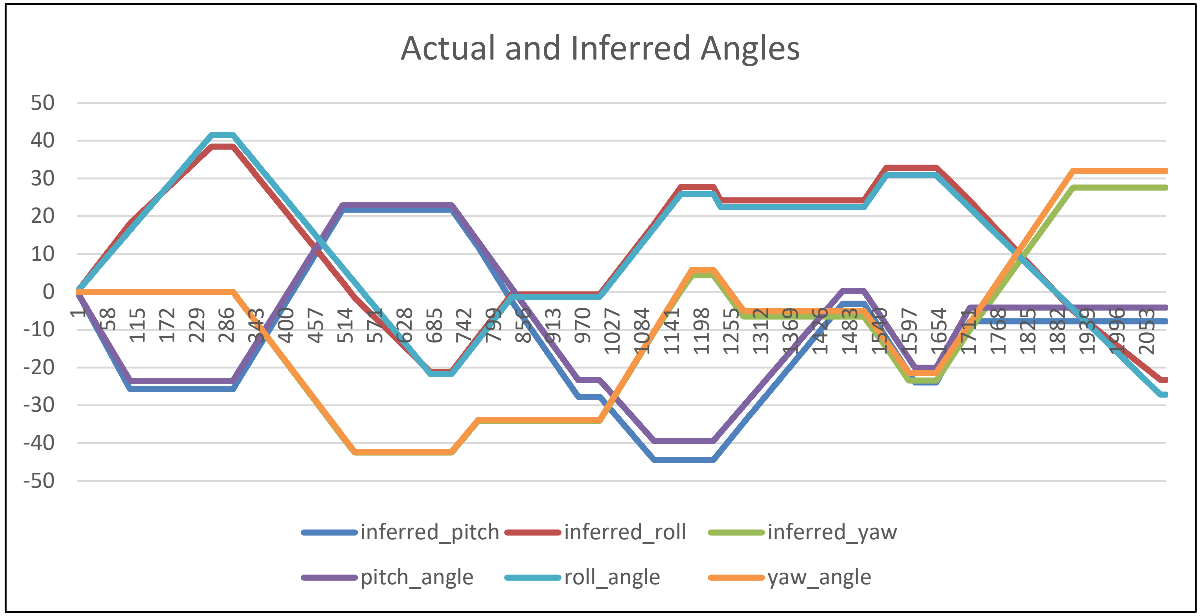
Saved simulation data

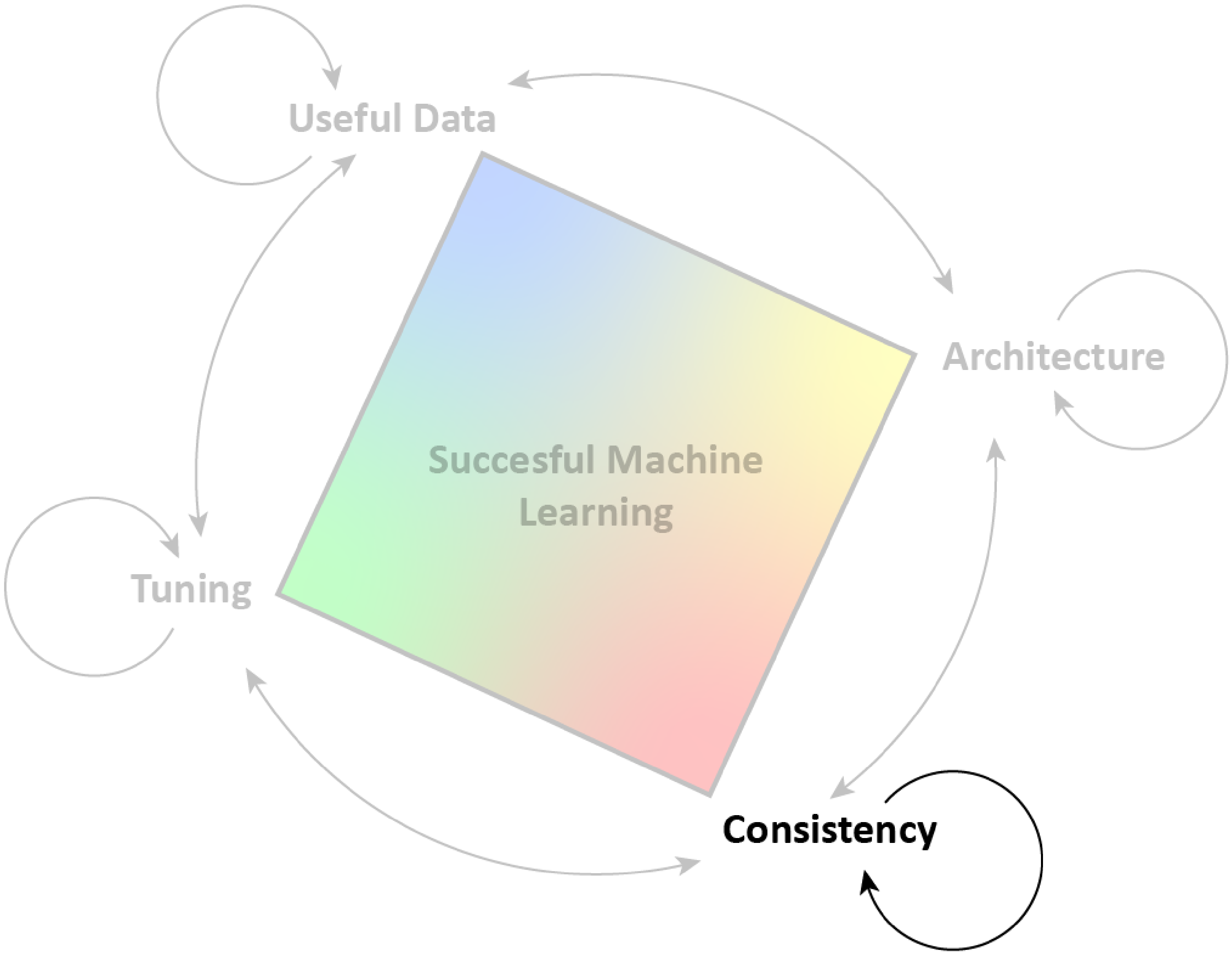
Validation against new synthetic data









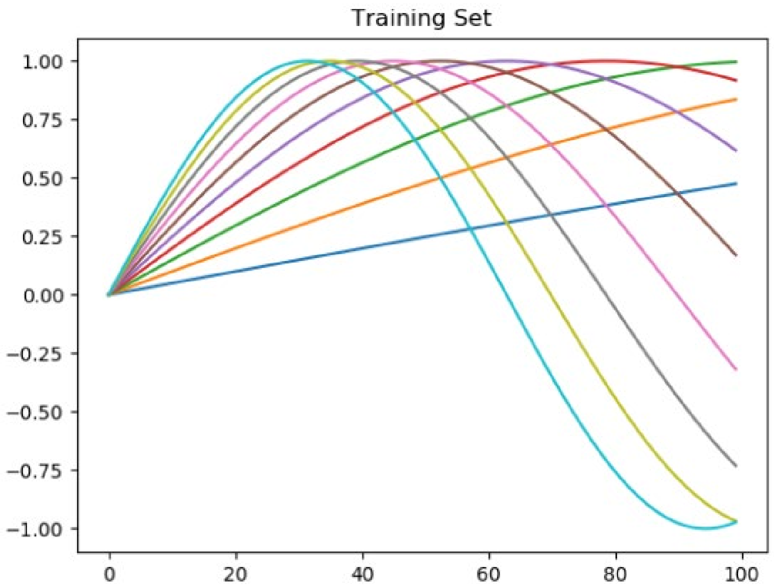


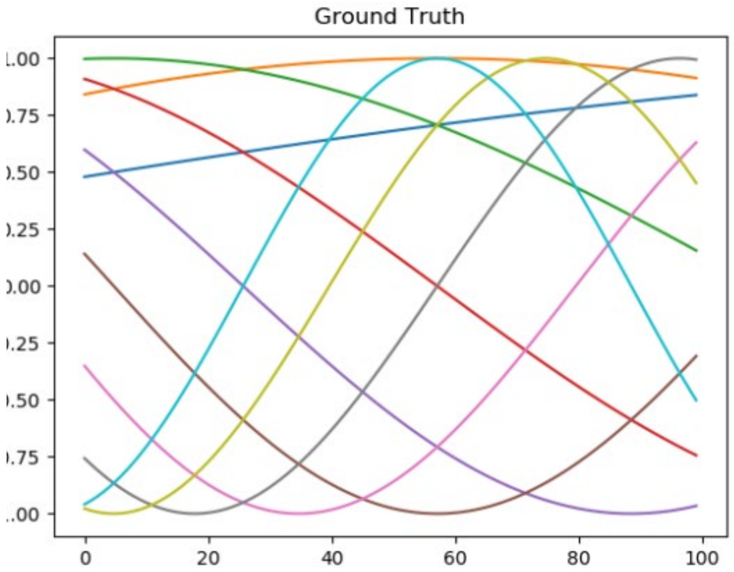
Useful Data
Architecture
Succesful Machine Learning
Tuning
Consistency

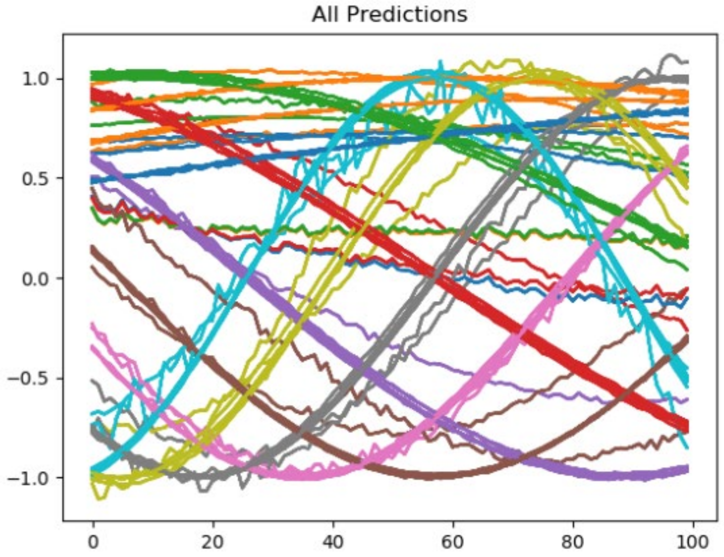
Consistency

1. Neural Networks are initialized with random numbers
2. The same network will train differently *every time*
3. Accuracy differences for the same model can be as high as 50%
4. This makes it very hard to *find* the right hyperparameters!

Let’s use some simple data to see this clearly

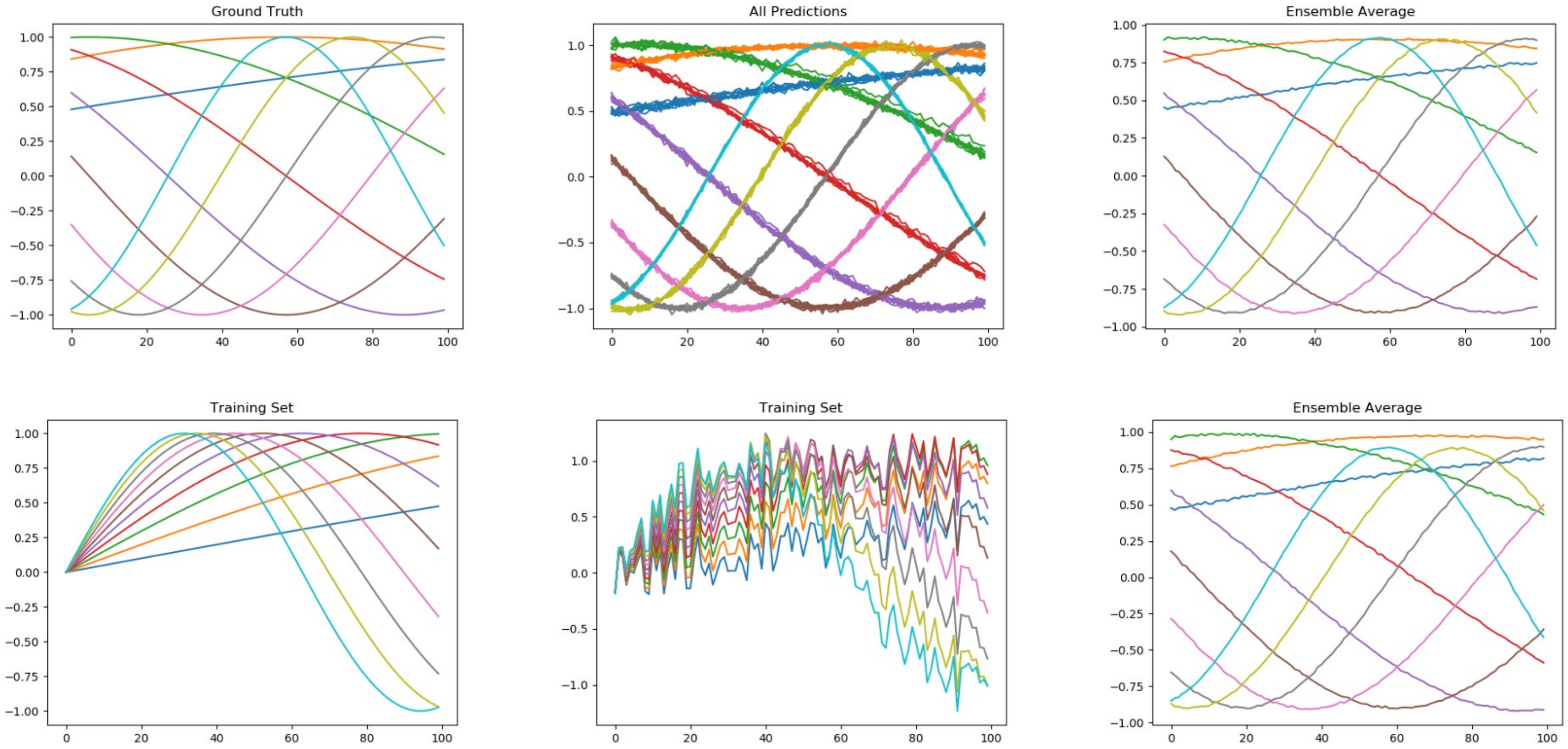






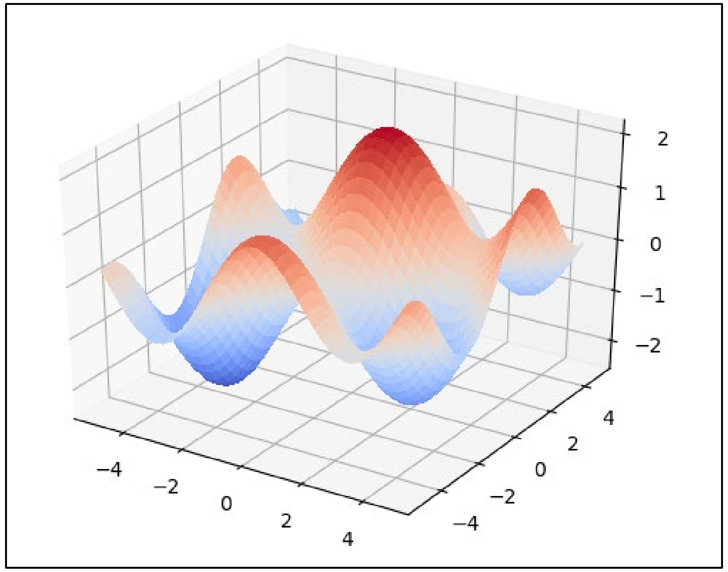
Ensembles

- The best parameters are based on an *Ensemble average* of multiple models
- Multiple models *for the same parameters* are stored
- In *inference*, the average of all model *predictions* is taken as the best value
- Ensembles produce *consistent* and *repeatable* results

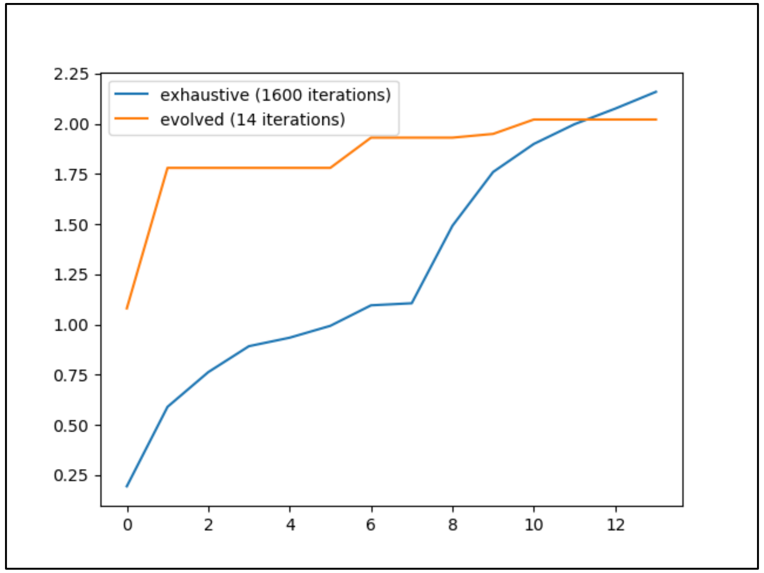


Evolving ensembles

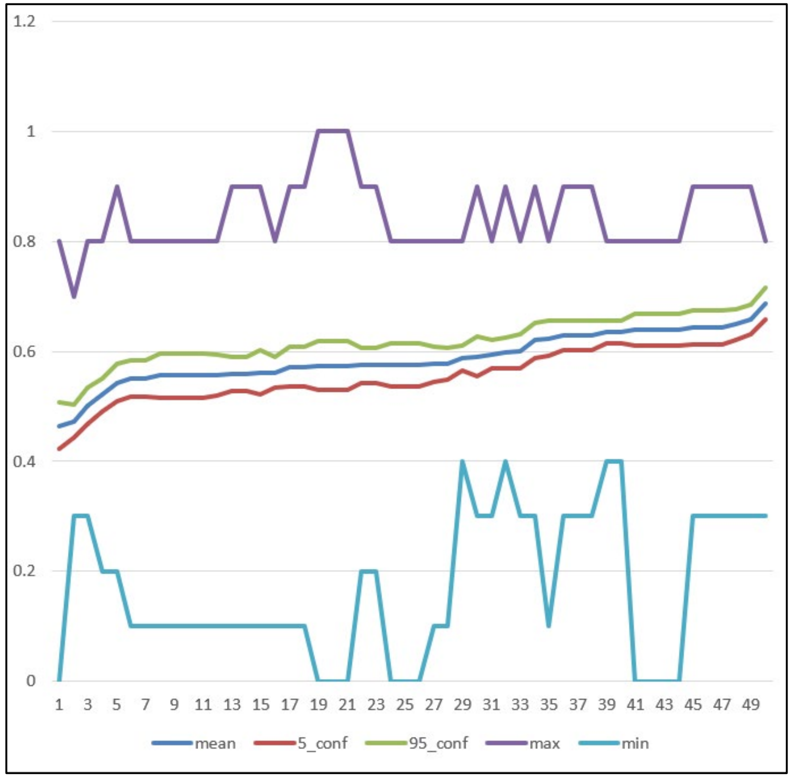
1. We created `Optevolver` – an Ensemble hyperparameter/architecture using GAs
2. Uses a set of models to find average, standard deviation and min/max
3. Much faster, though not quite as good as grid search



Known Fitness landscape



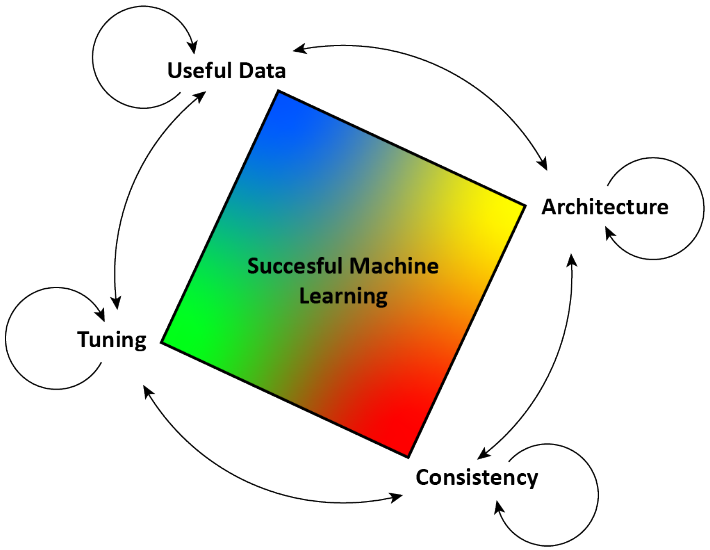
Exhaustive vs. genetic search



Improvement avg 47% - 64% accuracy

Conclusions

Simulation	Genetic Algorithms	Ensemble prediction
• Understandable, explainable models • Creates balanced data sets • Data is clean and perfectly tagged	• Evolutionary hyper-parameter tuning and architecture search • Reduces time spent in model development • Reduced compute cost for model training • Pip install optevolver	• Creates resilient, reliable models • Ensemble models are repeatable and reproducible



Questions?

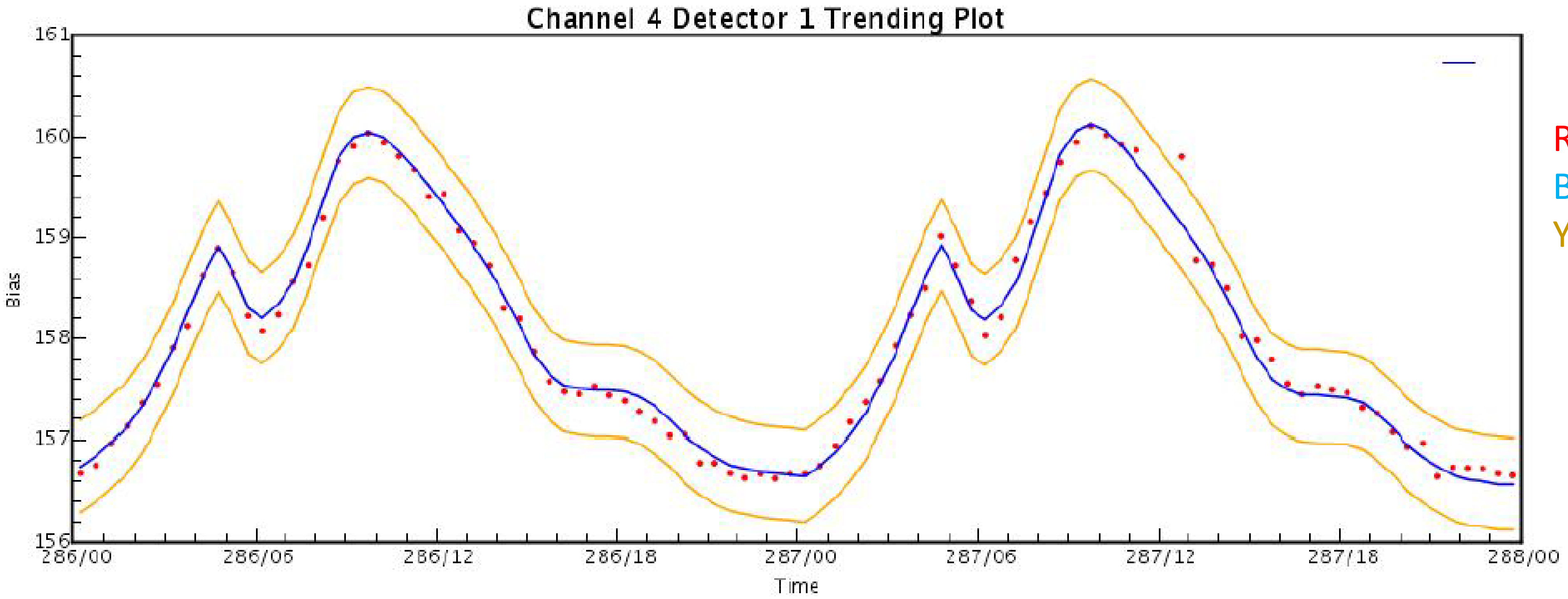
If you can read this, we've gone too far…

Questions?

- Where can I get Optevolver?
  - `pip -install optevolver`
- What does it work on?
  - Just Tensorflow 2.0 right now, but PyTorch soon
- Where can I get the source?
  - https://github.com/pgfeldman/optevolver
- Does it do Bayesian/particle swarm/<other cool algo>
  - Not yet…
  - But feel free to contribute!

Background

- AIMS: Low-level monitoring and trending for single mnemonics using machine learning



Red dots = data
Blue line = ML prediction
Yellow lines = Error bands

- Goal: High-level monitoring and trending across multiple mnemonics