# Exploring Policy-based training and enforcement of Compositional Neural Networks

Sobhan Chatterjee

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

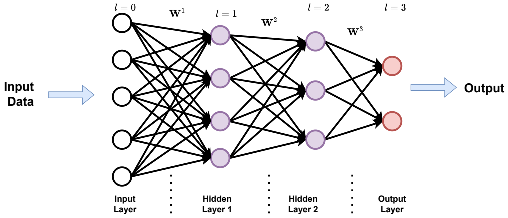
21-11-2024

# Outline

# Outline

# Data Driven Design - Artificial Neural Networks

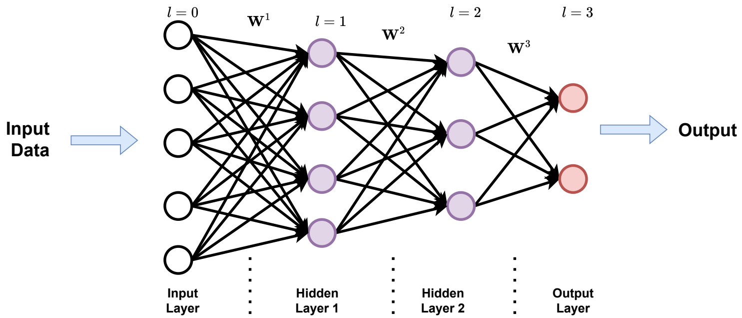
- Created to imitate their biological counterparts
- Two phases
  - *Training*
  - *Inference*
- Multiple layers
  - Input Layer
  - Hidden Layer(s)
  - Output Layer
- Neurons operate on “activation functions”




- Multiple types
  - **Multi-layer Perceptrons**
  - Convolutional Neural Networks
  - Recurrent Neural Networks

# ANN and Verification

Often designed as large monolithic (single) models that are difficult to verify for safety and timing requirements.



# Outline

Synchronous neural networks for cyber-physical systems

Partha S Roop
*University of Auckland*
Auckland, New Zealand
p.roop@auckland.ac.nz

Hammond Pearce
*University of Auckland*
Auckland, New Zealand
hammond.pearce@auckland.ac.nz

Keyan Monadjem
*University of Auckland*
Auckland, New Zealand
kmon173@aucklanduni.ac.nz

**A Compositional Approach for Real-Time Machine Learning**

Nathan Allen
nall426@aucklanduni.ac.nz
University of Auckland
Auckland, New Zealand

Yash Raje
yraj429@aucklanduni.ac.nz
University of Auckland
Auckland, New Zealand

Jin Woo Ro
jro002@aucklanduni.ac.nz
University of Auckland
Auckland, New Zealand

Partha Roop
p.roop@auckland.ac.nz
University of Auckland
Auckland, New Zealand

A compositional approach using Keras for neural networks in real-time systems

Xin Yang
*The University of Auckland*
Auckland, New Zealand
xyan510@aucklanduni.ac.nz

Partha Roop
*The University of Auckland*
Auckland, New Zealand
p.roop@auckland.ac.nz

Hammond Pearce
*The University of Auckland*
Auckland, New Zealand
hammond.pearce@auckland.ac.nz

Jin Woo Ro
*The University of Auckland*
Auckland, New Zealand
jro002@aucklanduni.ac.nz

# Outline

# Exploring Compositional NNs for Real-Time Systems

2024 22nd ACM-IEEE International Symposium on Formal Methods and Models for System Design (MEMOCODE)

## Exploring Compositional Neural Networks for Real-Time Systems

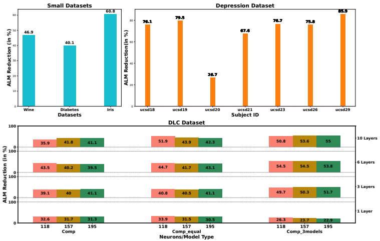
Sobhan Chatterjee* †, Nathan Allen*, Nitish Patel* and Partha Roop*
*Department of Electrical, Computer and Software Engineering, Faculty of Engineering
University of Auckland, Auckland, New Zealand
†Email: schb534@aucklanduni.ac.nz

***Abstract*—Real-time CPSs using Artificial Neural Networks (ANNs) are traditionally developed as monolithic black-boxes. This results in designs that are often difficult to formally verify against safety specifications and implement on hardware for formal timing analysis. Consequently, their implementation as a composition of smaller ANNs has received recent interest. These are easier to implement, parallelise and validate. Despite**

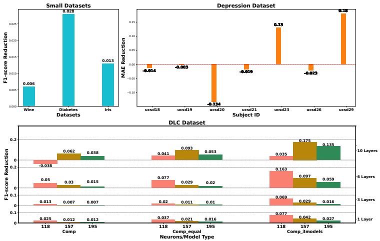
systems with strict safety (safety-critical) and response-time (time-critical) specifications often mandate formal guarantees on their functional [5] and timing correctness [6].

While several techniques have been proposed to handle formal verification [7] of ANNs, which is an NP-Complete problem [8], the use of such methods is usually limited to either small networks or require complex model abstraction

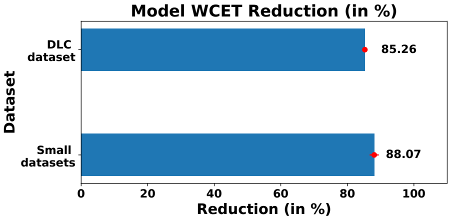
# Observations




(a)



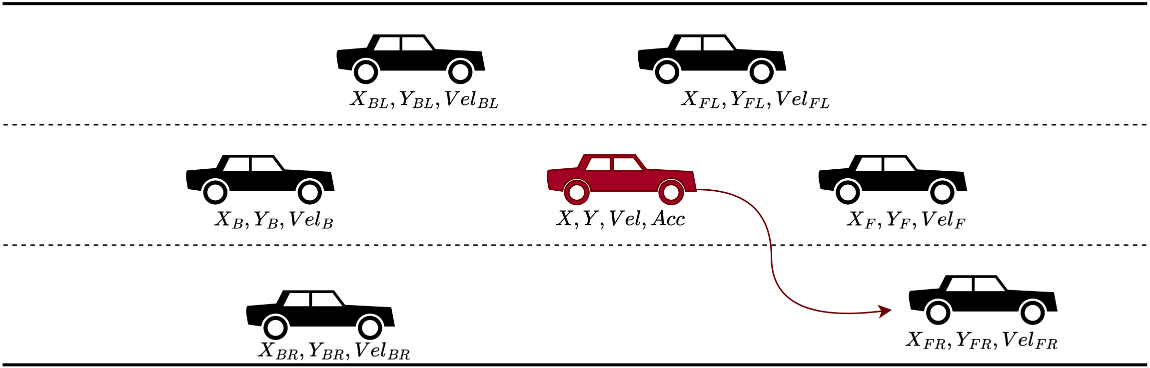

(b)



# Outline

# Outline

# Car system on Freeway



# Simple Autonomous Vehicle (AV) Case Study

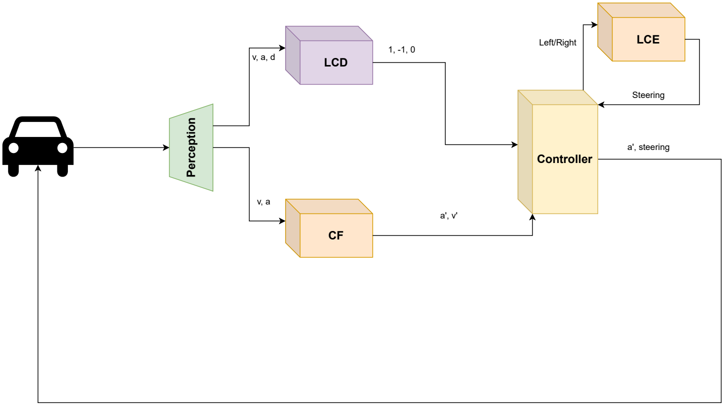



Figure: AV freeway/highway example. LCD: Lane change decision, LCE: Lane Change Execution, CF: Car Following

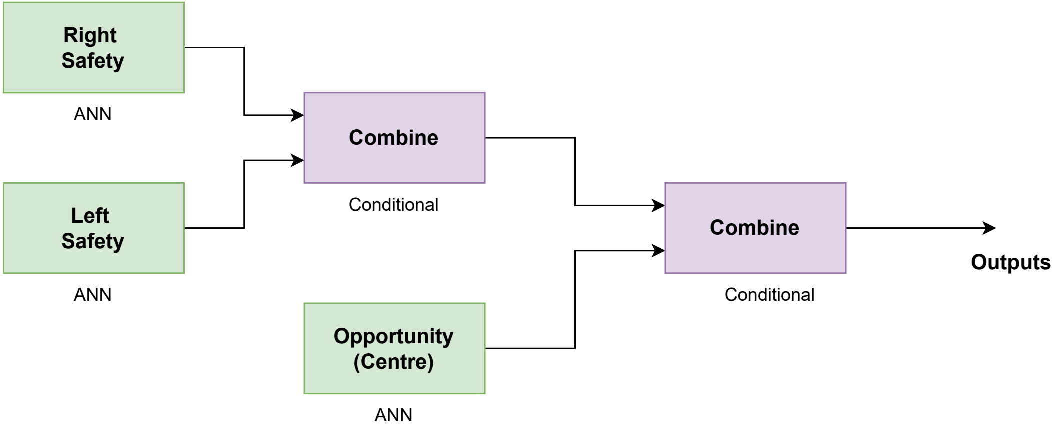



Figure: LCD Module

# CF Module

## Intelligent Driver Model (IDM)

$$
\begin{aligned}
\dot{v} &= a\left(1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*}{s}\right)^{2}\right) \\
\dot{s} &= v
\end{aligned} \tag{1}
$$

where $v$ is the velocity of the vehicle, $s$ is the distance between the vehicle and the vehicle in front, $a$ is the acceleration, $v_0$ is the desired velocity, $\delta$ is the acceleration exponent, and $s^*$ is the desired safe distance. We choose $v_0 = 30m/s$ and $s^* = 2m$.

# LCE Module

## Simple Polynomial Time-based model

$$y(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0$$

$$\dot{y}(t) = 3a_3 t^2 + 2a_2 t + a_1$$

$$a1 = 0, a2 = 0, a3 = 3\frac{lane\_width}{t_{LC}^2}, a4 = -2\frac{lane\_width}{t_{LC}^3} \quad (2)$$

where $y(t)$ is the lateral position of the vehicle at time $t$, $a_3$, $a_2$, $a_1$, and $a_0$ are the polynomial coefficients, and $t$ is the time. $t_{LC}$ is the desired time for lane change. Here, $t_{LC} = 5seconds$ and $lane\_width = 3.7m$.

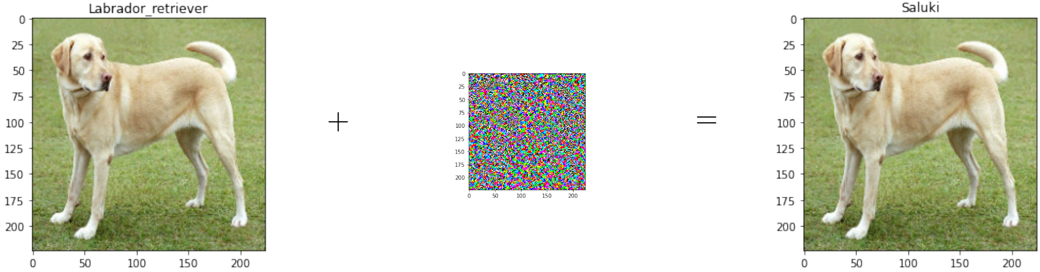
# A simple System-level policy

The car must not collide into nearby cars

# ANN-based CPS approaches are not dependable...

## Difficulties

- ANNs prone to errors and not reasonably robust to noisy inputs.
- E.g. Sensor noise can lead to misclassifications.



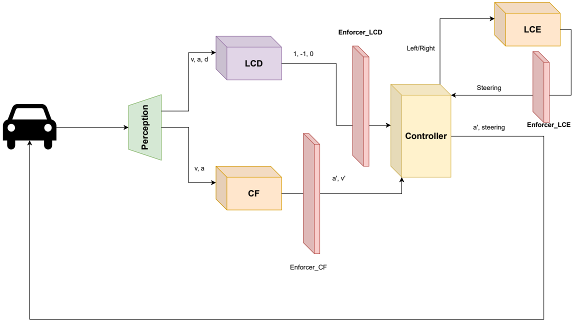
# ANN-based CPS approaches are not dependable...

## Difficulties

- ANNs susceptible to adversarial attacks that lead to wrong predictions.
- E.g. A misclassified obstacle can have horrible consequences.

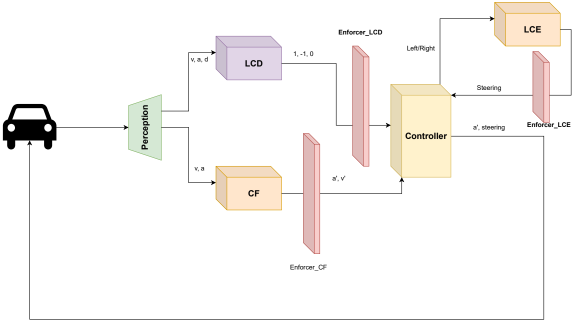
# Outline

# Add Runtime Enforcers after modules




- **Unidirectional *Runtime Enforcers* at module output to modify unsafe outputs from entering the controller.**

# Train ANN models on policies




- Unidirectional *Runtime Enforcers* at module output to modify unsafe outputs from entering the controller.
- **Train ANN models based on policies to increase adherence to polcies.**

# What policies should the Enforcers enforce?

## System Level Policy

No Collision with surrounding cars.

## Decompose System Level Policy

- No Collision Due to Lane Change
  - No collision due to unsafe lane change decision
  - No collision due to unsafe lane change execution
- No collision due to Car following

# What policies should the Enforcers enforce? Safety Policies

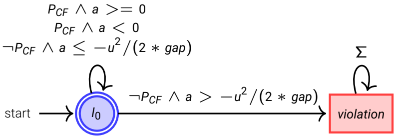



Figure: Safety property for car following module

## No collision due to car following

- Car should brake and stop if it comes too close (determined by policy$P_{CF}$) to car in front.
- $P_{CF}$ : $gap >= 2m$
- Recover by EDITING $a = -u^2/(2 * gap)$ or braking.

# What policies should the Enforcers enforce? Safety Policies

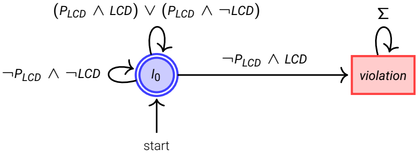



Figure: Safety property for LCD

## No collision due to decision to lane change

- Car should only lane change when there is enough gap to lane change
- When the gap determined by both policy $P_{LCD}$ and the ANN output ($LCD$) is safe, the car can lane change, otherwise not.
- Recover by EDITING $LCD = 0$ or providing output for No lane change.

# What policies should the Enforcers enforce? Safety Policies

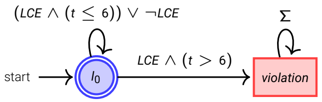



Figure: Safety property for LCE

## No collision due to lane change

- It should not take too long to lane change ($LCE = 1$). If it takes longer than 6 seconds for the lane change process, we enter violation state.
- Recover by EDITING $LCE = 0$ or stopping lane change.

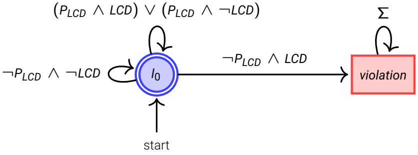
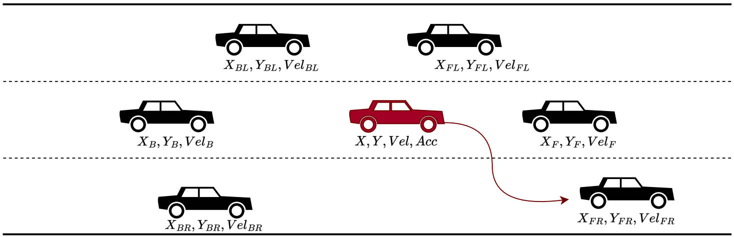



Figure: Safety property for LCD



# More on LCD policy: Compositional Division of Policies

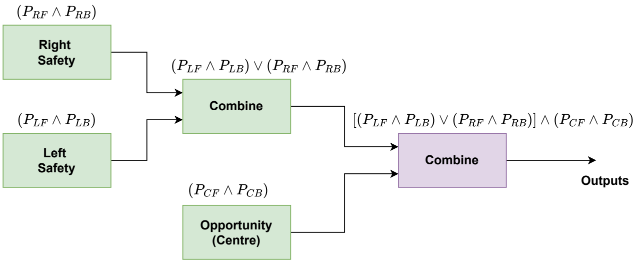



Figure: Compositional Properties

## Compositional

- Divide the model properties to models.
- Train each block using the properties
- Check overall satisfaction after training

# Policy Mining Optimisation: LCD Module

- Prepare linear template of predicates of form $A >= x1 * B + x2 * C$, where x1, and x2 are coefficients and A, B, C are features.
- Example: $P_{LF}$ : $lfx > x1 * lfxVelocity + x2 * lfxAcceleration$
- Linear Regression to find initial values of x1 and x2.
- Run optimiser to optimise the policy plane using Mean Squared Error.
- Convex optimisation. We stop when 90% datapoints satisfy the learnt policy.
- BFGS (Broyden–Fletcher–Goldfarb–Shanno) as optimiser.

# Example Policy Plot

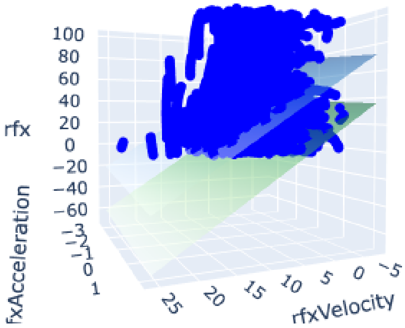
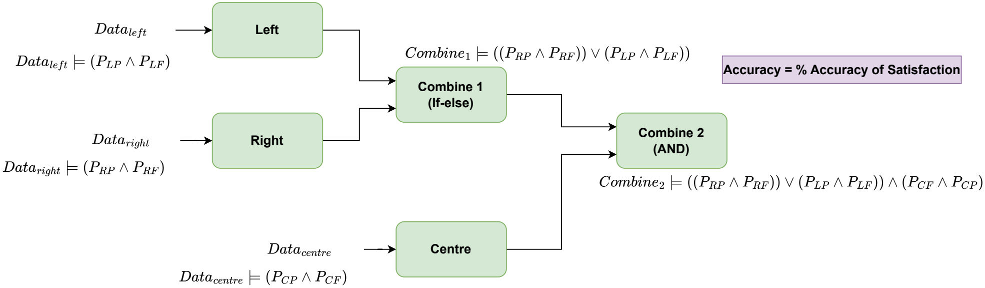


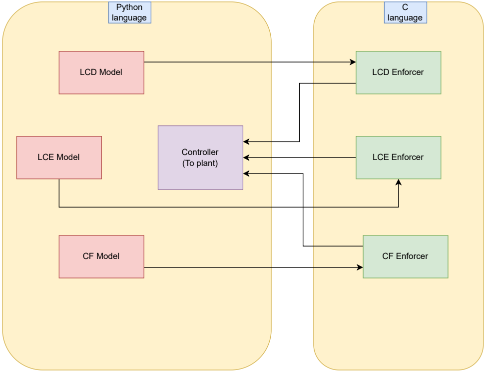

Figure: Blue plane is the plane from Linear Regression and the Green plane is after optimisation

# Policy Data Satisfaction




- Obtain the policies.
- Obtain datapoints satisfying the combined policies.
- Left LC: 69422, Right LC: 61342 and No LC: 43667

# Implementation




- easyRTE tool for Runtime Enforcement. `https://github.com/PRETgroup/easy-rte`h
- Python for synchronous execution.

# Related work

## Deep Specification Mining

Tien-Duy B. Le
School of Information Systems
Singapore Management University, Singapore
btdle.2012@smu.edu.sg

David Lo
School of Information Systems
Singapore Management University, Singapore
davidlo@smu.edu.sg

## Policy-Based Diabetes Detection using Formal Runtime Verification Monitors

1st Abhinandan Panda
*School of Electrical Sciences*
*IIT Bhubaneswar*
Bhubaneswar, India
Email: ap53@iitbbs.ac.in

2nd Srinivas Pinisetty
*School of Electrical Sciences*
*IIT Bhubaneswar*
Bhubaneswar, India
Email: spinisetty@iitbbs.ac.in

3rd Partha Roop
*Dept. of Electr. & Comput. Eng.*
*University of Auckland*
Auckland, New Zealand
Email: p.roop@aucklanduni.ac.nz

## Assumption Generation for Learning-Enabled Autonomous Systems

**Authors**: Corina S. Păsăreanu, Ravi Mangal, Divya Gopinath, Huafeng Yu | Authors Info & Claims

Runtime Verification: 23rd International Conference, RV 2023, Thessaloniki, Greece, October 3–6, 2023, Proceedings • Pages 3 - 22
https://doi.org/10.1007/978-3-031-44267-4_1

# Outline

# Results: LCD Policies

$$
\begin{aligned}
lfx &> 16.07 + 6.62 * lfxVelocity + 10.74 * lfxAcceleration \\
lpx &< -18.15 + -2.82 * lpxVelocity + -2.52 * lpxAcceleration \\
rfx &> 13.19 + -3.11 * rfxVelocity + -1.7 * rfxAcceleration \\
rpx &< -15.75 + 3.82 * rpxVelocity + -2.01 * rpxAcceleration \\
fx &> 17.76 + 3.79 * fxVelocity + -11.99 * fxAcceleration \\
px &< -15.84 + -1.12 * pxVelocity + 4.48 * pxAcceleration
\end{aligned} \tag{3}
$$

- 71% adherence to $P_{LCD}$ for models not trained with policy on test data.
- **83% adherence to $P_{LCD}$ for models not trained with policy on test data.**

# Enforcement Results

## Enforcement of LCD and CF

```
Inputs: LCD=1, P=0 -> New LCD=0
Inputs: X=7, LCE=1 -> New LCE=0
Inputs: Acceleration=0, Velocity=201.00502512562815, and Gap=1.99 -> New Acceleration=-100.502510
```

## Enforcement of LCE

```
Inputs: LCD=0, P=1 -> New LCD=0
Inputs: X=7, LCE=1 -> New LCE=0
Inputs: Acceleration=0.7732927017091369, Velocity=21.67380790043201, and Gap=53.833012477972495 -> New Acceleration=0.773293
Step: 50, Time: 2.0s, LCD: 0, LCE: 0, Lateral Speed: 0, Accel: 0.77, Vel: 21.67, Gap: 53.83
```

## No Enforcement: Transparency

```
Inputs: LCD=0, P=1 -> New LCD=0
Inputs: X=0, LCE=False -> New LCE=0
Inputs: Acceleration=0.8911531126722197, Velocity=20.143163804952586, and Gap=50.31879548446812 -> New Acceleration=0.891153
Step: 4, Time: 0.2s, LCD: 0, LCE: 0, Lateral Speed: 0, Accel: 0.89, Vel: 20.14, Gap: 50.32
```

# Conclusion

## Summary

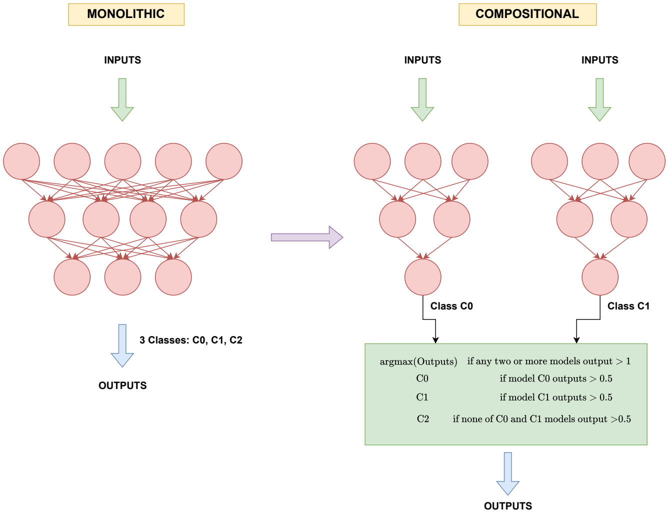
- Compositional models offer better timing performance
- Data-based compositional policy mining based for linear predicates.
- Policy trained compositional models are better than ones not trained on policies.

- Add more policies: Liveness, timed policies.
- Restricted to Linear predicates.
- Proper simulation in SUMO.
- Python based enforcement.
- Series and parallel execution of modules.

THANKS!

# Outline

# Compositional Premise: Example



# Datasets

## NGSIM dataset: Classification

- US Highway 101 dataset: 7.50 - 8.35 a.m : 4824 cars.
- I80 Emeryville dataset: 4.00 - 4.15 & 5.00 - 5.30 p.m : 4383 cars.
- Divide based on intuition
- Left LC Samples: 4380
- Right LC Samples: 1290
- No LC samples: 11006

## Small datasets: Classification

- Iris: Classify iris flowers
- Wine: Classify wine type
- Diabetes: Classify severity of diabetes
- Divide based on confusion matrix

## Depression Dataset: Regression

- Dataset from UCSD, USA
- 14 mild-moderately depressed participants
- Predict Mood Score: 1 (happy) - 7 (depressed)
- Divide features into clusters and build models for each
- 6 clusters: Diet, Activity, Heart, Psychological, Sleep and Neurocognitve.

# Compositional Premise

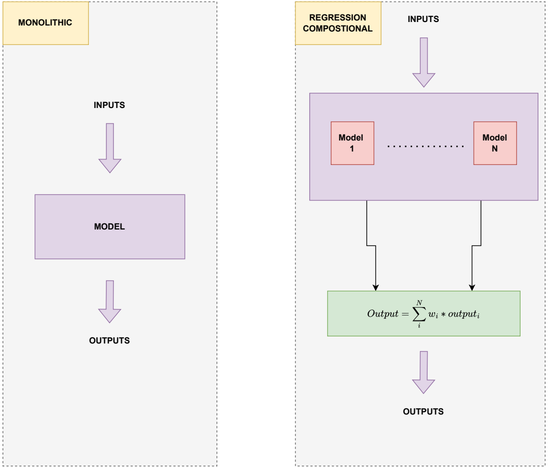



Figure: Caption

# Compositional Premise

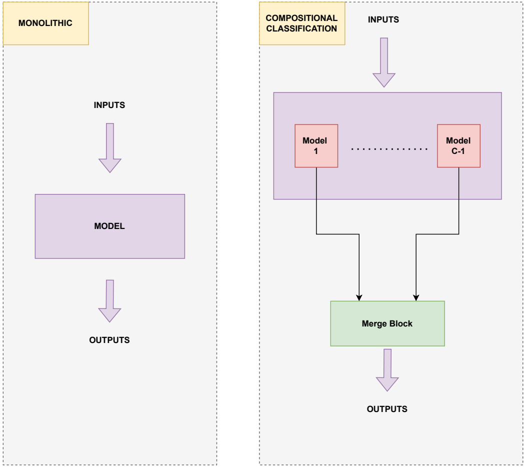



Figure: Caption

# Implementation Pipeline


- Linear Approximation of activations
- 32-bit Fixed-point numbers: 16 bit integer and 16 bit decimal
- Multi-cycle execution of neuron instances
- Pipelined execution of neurons

# References I


[1] P. S. Roop, H. Pearce, and K. Monadjem, "Synchronous neural networks for cyber-physical systems," in *2018 16th ACM/IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*, 2018, pp. 1–10.

[2] N. Allen, Y. Raje, J. W. Ro, and P. Roop, "A Compositional Approach for Real-Time Machine Learning," in *Proceedings of the 17th ACM-IEEE International Conference on Formal Methods and Models for System Design*, ser. MEMOCODE '19, event-place: La Jolla, California, New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3359986.3361204.

[3] X. Yang, P. Roop, H. Pearce, and J. W. Ro, "A compositional approach using Keras for neural networks in real-time systems," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2020, pp. 1109–1114.