UMBC
AN HONORS UNIVERSITY IN MARYLAND

Learning Device Models with Recurrent Neural Networks

John Clemens
University Of Maryland, Baltimore County (UMBC)

CORAL LAB
cognition | robotics | learning

Motivation

Sometimes you have peripherals but no software:

- Different OS
- Proprietary
- Obsolete

Sometimes you can't assume hardware is correct:

- Testing and Verification
- Hardware trojans

Sometimes you want to adapt to unknown environments

- Evolutionary robotics

Goal

UMBC
AN HONORS UNIVERSITY IN MARYLAND

Can computers learn to interact with unknown peripheral devices?

- Devices act on sequences of commands that change internal state to produce a sequence of outputs
- Traditional black-box techniques are impractical for complex systems[1]
- Recurrent Neural Networks can approximate arbitrary complex systems[2]
  - But can they *learn* such models?

Hypothesis:

- ***We can train recurrent networks to create functionally accurate models of computer peripheral devices***

Empirical Study Overview

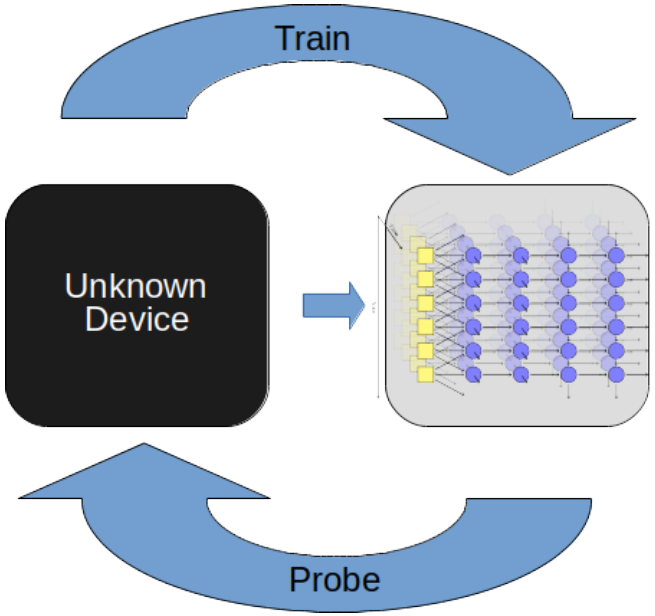
UMBC
AN HONORS UNIVERSITY IN MARYLAND

Process

- Fuzz the input space with random input sequences to the hardware
- Observe output
- Train RNN with observations
- Repeat

Caveats

- Not possible to learn every device through passive observation
  - NP-Complete in the general case[3]

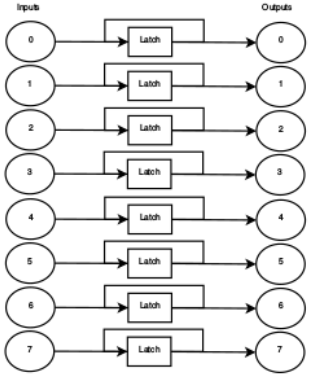


RNN image: http://www.hexahedria.com/2015/08/03/composing-music-with-recurrent-neural-networks/

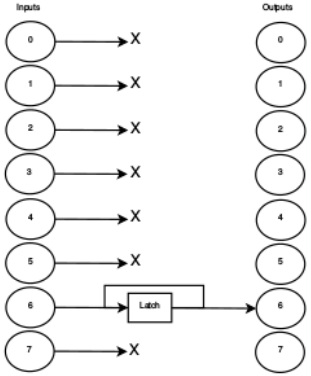
Emulated Test Devices

UMBC
AN HONORS UNIVERSITY IN MARYLAND

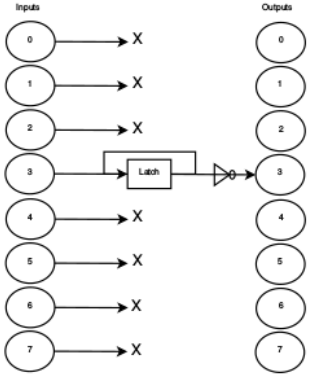
6 Example Devices
- 5 simple examples
  - All have memory
  - XOR
  - Parity Calculator
- A 16550 UART
- Binary (bit) inputs (0,1)
- Increasing complexity
  - UART is a deceptively complex device to learn



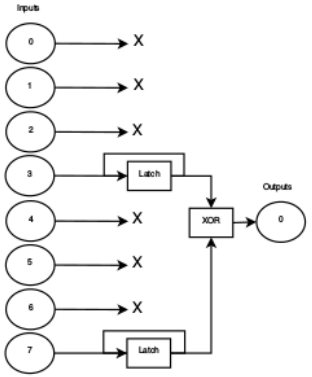
(a) EightBitMachine



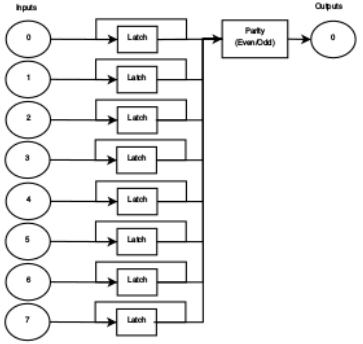
(b) SingleDirectMachine



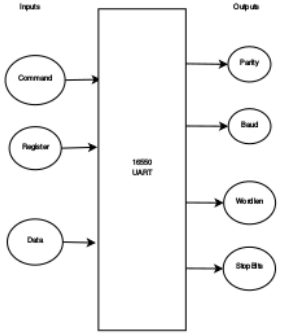
(c) SingleDirectInvertMachine



(d) SimpleXORMachine



(e) ParityMachine



(f) SerialPortMachine

RNN Structure

UMBC
AN HONORS UNIVERSITY IN MARYLAND

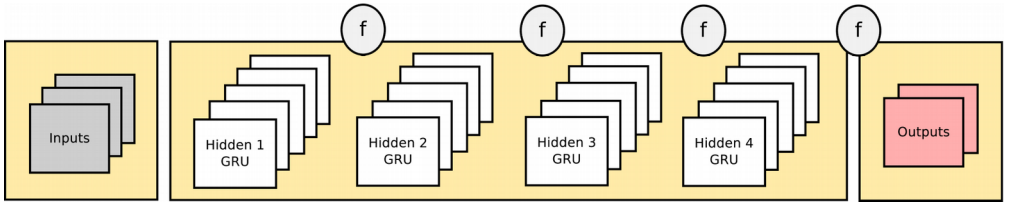
Not concerned with most optimal method
- Only that it learns

Keras + Tensorflow

Sequence learning
- GRU or LSTM cells
- Multiple Layers
  - Hidden size: max(I,O)+1
- Activation between each layer

Empirically determined
- Random sampling of hyperparameter space

Same structure used for each machine type!



Parameter	Value	Options
Layer Type	GRU	GRU,LSTM,SimpleRNN,Dense
Activation Function	tanh	tanh,linear,(s)elu,sigmoid,relu
Optimizer	nadam	adam,nadam,rmsprop,sgd
Loss Function	msle	msle,mse,mape
# Hidden Layers	4	1,2,4 or 8

TABLE VIII
KERAS HYPER-PARAMETERS USED FOR EXPERIMENTS

Result 1: Successful Learning

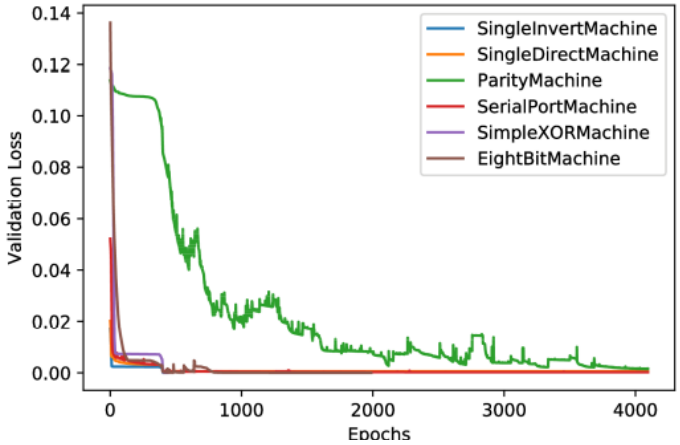
UMBC
AN HONORS UNIVERSITY IN MARYLAND

Training dataset

- 4096 training sequences
- 1024 validation sequences
- 128 evaluation sequences
- Sequences of 1024 commands

Repeat for 50 networks

- Stop at 4096 epochs or when validation loss < 0.1% for 20 consecutive epochs.
- Success if < 5% loss on evaluation data

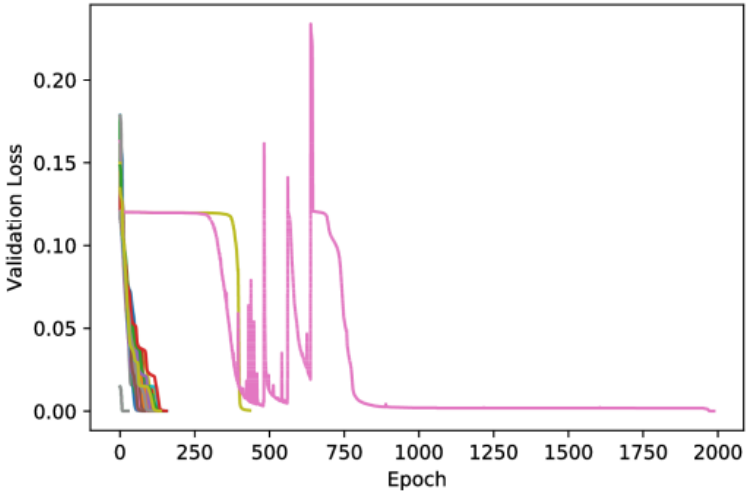


Machine	# Params	Epochs	% Success	Eval Loss
EightBitMachine	2,461	231	98%	0.002515
SingleDirectMachine	2,461	206	98%	0.003003
SingleInvertMachine	2,461	37	100%	0.000018
SimpleXORMachine	2,384	155	98%	0.002466
ParityMachine	2,384	1875	78%	0.023999
SerialPortMachine	10,398	773	98%	0.003326

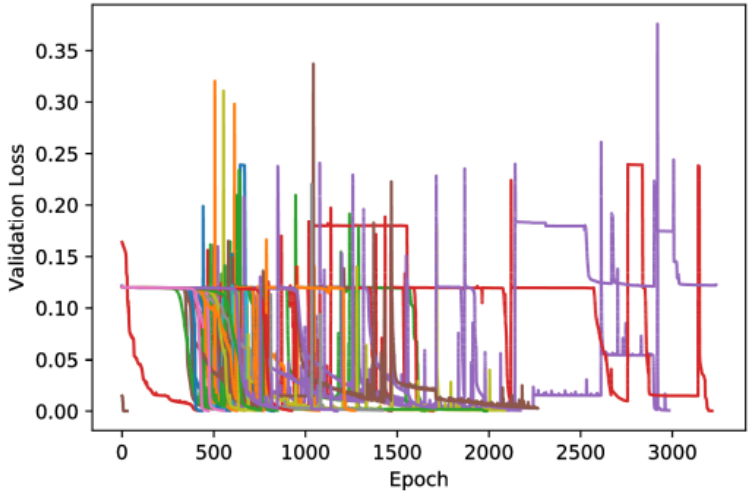
TABLE II
RESULTS OF TRAINING 50 NETWORKS FOR EACH MACHINE

Result 1 (cont)

Val Loss / Epoch: EightBitMachine



Val Loss / Epoch: ParityMachine



Result 2: Functional Equivalence

UMBC
AN HONORS UNIVERSITY IN MARYLAND

Different than validation loss

- Output vector decoded into corresponding real values (one-hot decoding, rounding, etc)

Every output at every sequence step must decode correctly

Results:

- Even simple machines required more epochs to achieve perfect accuracy at each step
- Parity remains difficult
- SerialPort failed to achieve 100% accuracy despite validation error approaching 10^{-5}

Machine	# Outputs	Epochs	Epochs+	Accuracy
EightBitMachine	1048576	210	57	100%
SingleDirectMachine	1048576	28	0	100%
SingleInvertMachine	1048576	69	40	100%
ParityMachine	131072	4026	1939	99.9992%
SimpleXORMachine	131072	83	21	100%
SerialPortMachine	2883584	N/A	33000+	N/A

TABLE III
MAXIMUM EVALUATION ACCURACY WITH OUTPUT MAPPING.

Deep Dive: 16550 UART

UMBC
AN HONORS UNIVERSITY IN MARYLAND

Deceptively Complex

- 12 internal registers mapped to 8 addresses
  - Must set bits in one register to change mapping
- Hidden math formula
  - 115200 / 16-bit value spread over two hidden registers(!)
- Output vector of 22 values
  - Mix of one-hot, binary, and float outputs
- Data only output when certain command triggered

Not perfectly accurate despite low global validation loss

Setting	Values	Description
Word Length	5,6,7,8 bits	Size of the data to send
Baud Rate	Value between 0-115200	Base wire clock rate
Stop Bits	1, 1.5, 2 bits	End-of-frame bits
Parity	None,Odd,Even,High,Low	Meaning of parity bits
TX Data	Value 0-255	Data to transmit

TABLE I
RELEVANT RS-232 SETTINGS WITH DESCRIPTIONS

Target	Baudrate	Wordlen	Parity	Sbits	Output
115200,8n1	115285	8	None	1	Haxxo Wofxp!
9600,7e1	1936	7	Even	1	Haxxo Wofxp!
2400,7o2	853	7	Odd	2	Haxxo Wofxp!

TABLE IV
EXAMPLE "HELLO WORLD!" OUTPUT AFTER 33,000+ EPOCHS, SINGLE NETWORK

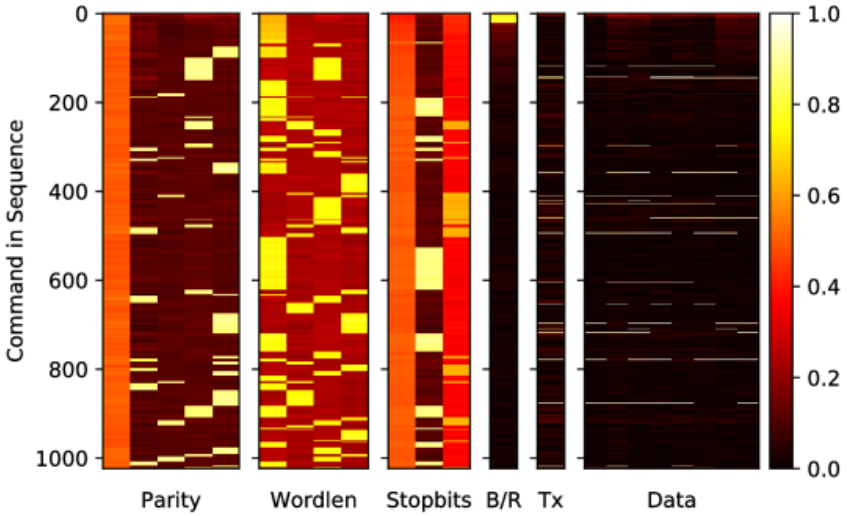
Deep Dive (cont)

Where is the error coming from?

- Heatmap

Insights

- Global error is not sufficient to drive optimization
- Floating point encoding for baud rate should be revisited



Loss at Epoch 16

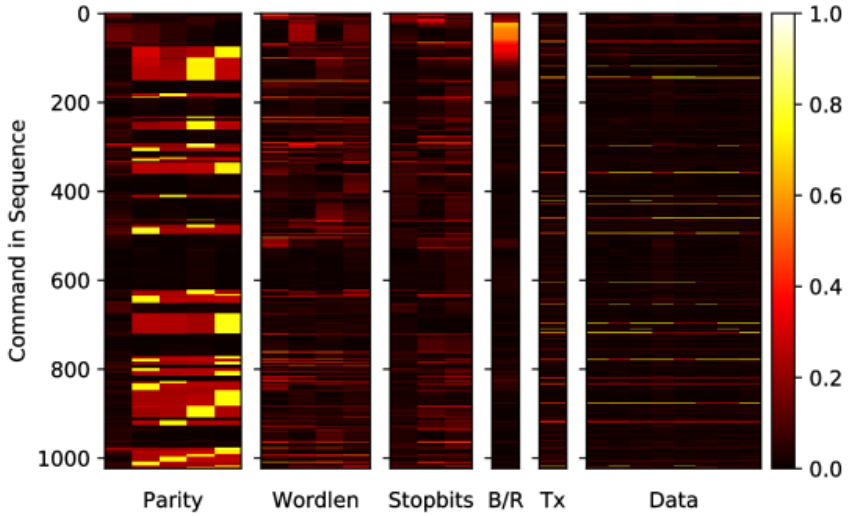
Deep Dive (cont)

Where is the error coming from?

- Heatmap

Insights

- Global error is not sufficient to drive optimization
- Floating point encoding for baud rate should be revisited



Loss at Epoch 176

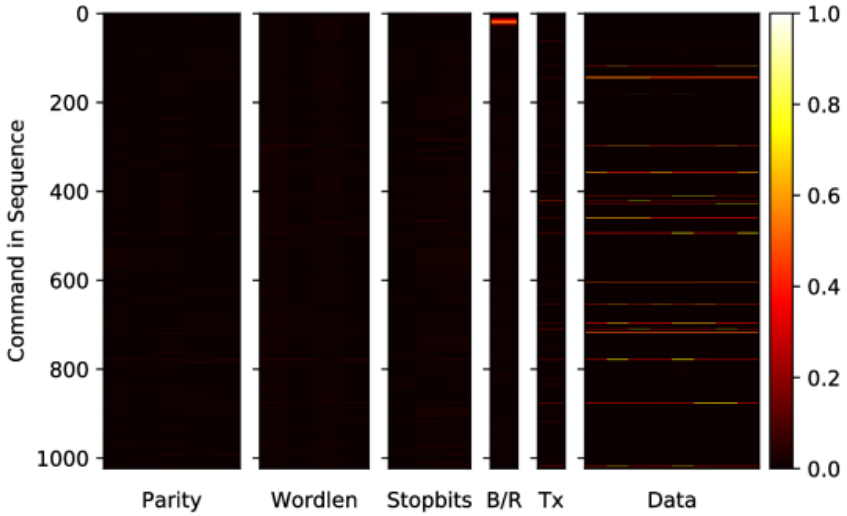
Deep Dive (cont)

Where is the error coming from?

- Heatmap

Insights

- Global error is not sufficient to drive optimization
- Floating point encoding for baud rate should be revisited



Loss at Epoch 2640

Results Summary

UMBC
AN HONORS UNIVERSITY IN MARYLAND

Learning observed in all test machines

Same general network architecture

Most achieve 100% accuracy

- Parity is the most difficult to learn

UART model achieves low error

- ...but not complete accuracy
- Better results achieved after paper published
  - Baudrate remains an issue

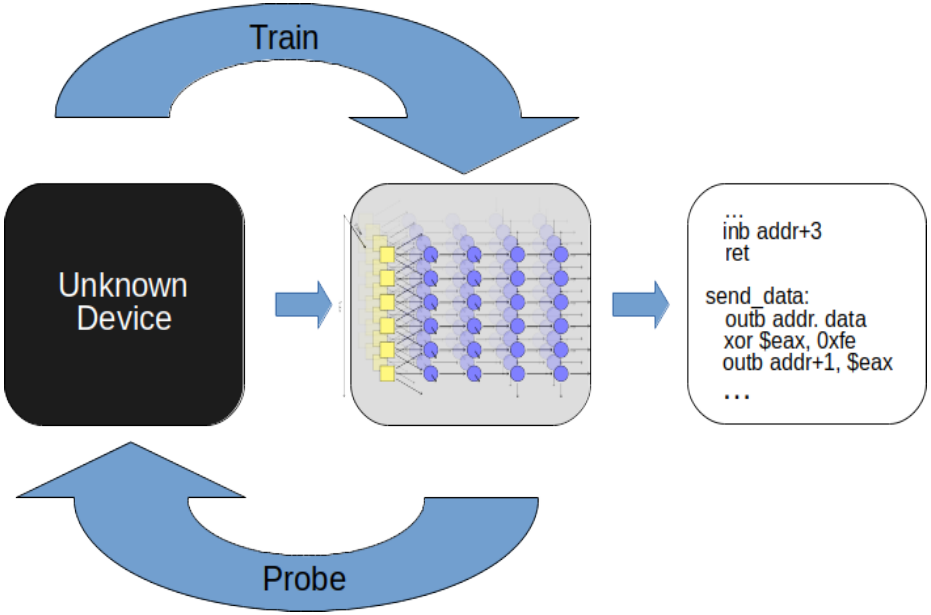
Next Steps

UMBC
AN HONORS UNIVERSITY IN MARYLAND

Refine technique
- Revisit activation functions
- Interrupts / DMA

Expand dataset
- VGA Text Mode

Rule Extraction
- Replaced “black box” device with “opaque box” trained RNN
- Automatically generate commands for original device based on knowledge extracted from learned model



Key Takeaways

UMBC
AN HONORS UNIVERSITY IN MARYLAND

RNNs can be used to accurately model computer peripherals!

- Novel application
- Dataset / code available

Successful learning of complex systems using existing tools

- All systems learned some aspects of the underlying machine
- Many learned perfectly accurate representations

Same basic RNN architecture works for all tested systems

- No domain specific knowledge imparted in network structure
- Must be careful not to impart knowledge in the encoding

Potential new avenue for black-box learning of unknown systems

Questions?

[1] F. Vaandrager, “Model learning,” Communications of the ACM, vol. 60, no. 2, pp. 86–95, 2017.

[2] H. T. Siegelmann and E. D. Sontag, “Turing computability with neural nets,” vol. 4, no. 6, pp. 77–80, 00299.

[3] E. M. Gold, “Complexity of automaton identification from givendata,” vol. 37, no. 3, pp. 302–320.

John Clemens
clemej1@umbc.edu
http://www.deater.net/john/
http://coral-lab.umbc.edu/

Non-custom image credits: Wikipedia, unless otherwise stated.

Backup Slides

Result 3: Decomposed UART

UMBC
AN HONORS UNIVERSITY IN MARYLAND

Try separate networks for each output class

- Attempt to focus error reduction on each output independently
- Not ideal solution
- Still requires full-size network

Results:

- Slight improvement
- Needs more study

Output	Encoding	Output Size	Epochs	Val. Loss
Parity	one-hot	5	579	$5.4 * 10^{-6}$
Word Length	one-hot	4	340	$1.2 * 10^{-5}$
Stop Bits	one-hot	3	251	$2.7 * 10^{-5}$
Baud Rate	float	1	1260	$1.2 * 10^{-4}$
Tx	true/false	1	322	$2.3 * 10^{-5}$
Data	binary	8	628	$1.1 * 10^{-5}$

TABLE V
CHARACTERISTICS OF DECOMPOSED MODEL NETWORKS FOR UART

Target	Baudrate	Wordlen	Parity	Sbits	Output
115200,8n1	115199	8	None	1	H!llo wop,d!
9600,7e1	2210	7	Even	1	Hello w/rld!
2400,7o2	146	7	Odd	2	Hello w/rld!

TABLE VI
EXAMPLE "HELLO WORLD!" OUTPUT, MULTIPLE NETWORKS

I/O Encodings

UMBC
AN HONORS UNIVERSITY IN MARYLAND

Bit-machine encoding:

Command Sequence		Input Encoding		Output Encoding
	cmd bit	cmd	bit	bit
1.	(set, 7)	[1,	0, 0, 0, 0, 0, 0, 0, 1]	[0, 0, 0, 0, 0, 0, 0, 1]
2.	(clear, 4)	[0,	0, 0, 0, 0, 1, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 1]
3.	(set, 0)	[1,	1, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0, 1]
4.	(set, 7)	[1,	0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0, 1]
5.	(set, 5)	[1,	0, 0, 0, 0, 0, 1, 0, 0]	[0, 0, 0, 0, 0, 1, 0, 1]
....		...		...
1022.	(clear, 5)	[0,	0, 0, 0, 0, 0, 1, 0, 0]	[1, 1, 0, 0, 1, 0, 0, 1]
1023.	(set, 3)	[1,	0, 0, 0, 1, 0, 0, 0, 0]	[1, 1, 0, 1, 1, 0, 0, 1]
1024.	(clear, 0)	[0,	1, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 1, 1, 0, 0, 1]

UART encoding:

Command Sequence		Input Encoding			Output Encoding					
	cmd, reg, data	cmd	reg	data	parity	wordlen	stopbits	baud	tx	data
1.	(write , 3, 0x25)	[1,	0, 1, 1,	0, 0, 1, 0, 0, 1, 0, 1]	[1, 0, 0, 0, 0,	0, 1, 0, 0,	0, 0, 1,	1,	0,	0, 0, 0, 0, 0, 0, 0, 0]
2.	(read , 0, 0x66)	[0,	0, 0, 0,	0, 1, 1, 0, 0, 1, 1, 0]	[1, 0, 0, 0, 0,	0, 1, 0, 0,	0, 0, 1,	1,	0,	0, 0, 0, 0, 0, 0, 0, 0]
3.	(write , 0, 0x45)	[1,	0, 0, 0,	0, 1, 0, 0, 0, 1, 0, 1]	[1, 0, 0, 0, 0,	0, 1, 0, 0,	0, 0, 1,	1,	1,	0, 1, 0, 0, 0, 1, 0, 1]
4.	(write , 3, 0x8F)	[1,	0, 0, 0,	0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0,	0, 0, 0, 1,	0, 0, 1,	1,	0,	0, 0, 0, 0, 0, 0, 0, 0]
5.	(write , 1, 0x33)	[1,	0, 0, 0,	0, 0, 1, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0,	0, 0, 0, 1,	0, 0, 1,	0.185,	0,	0, 0, 0, 0, 0, 0, 0, 0]
....		...			...					
1022.	(read , 7, 0x9a)	[0,	1, 1, 1,	1, 0, 0, 1, 1, 0, 1, 0]	[0, 0, 1, 0, 0,	0, 0, 1, 0,	1, 0, 0,	0.083,	0,	0, 0, 0, 0, 0, 0, 0, 0]
1023.	(write , 0, 0x81)	[1,	1, 1, 1,	1, 0, 0, 0, 0, 0, 0, 1]	[0, 0, 1, 0, 0,	0, 0, 1, 0,	1, 0, 0,	0.083,	1,	1, 0, 0, 0, 0, 0, 0, 1]
1024.	(write , 1. 0x02)	[1,	1, 1, 1,	1, 0, 0, 0, 0, 0, 0, 1]	[0, 0, 1, 0, 0,	0, 0, 1, 0,	1, 0, 0,	0.083,	0,	0, 0, 0, 0, 0, 0, 0, 0]

State Space Sizes

UMBC
AN HONORS UNIVERSITY IN MARYLAND

Machine	Input	Output	Internal
EightBitMachine	2^9	2^8	2^8
SingleDirectMachine	2^9	2^1	2^1
SingleInvertMachine	2^9	2^1	2^1
SimpleXORMachine	2^9	2^1	2^2
ParityMachine	2^9	2^1	2^8
SerialPortMachine	2^{12}	2^{37}	2^{37}

TABLE VII

APPROXIMATE MAGNITUDE OF STATE SPACES FOR EACH MACHINE.

Discussion

UMBC
AN HONORS UNIVERSITY IN MARYLAND

Same general network structure works well in many cases

- No domain specific knowledge imparted in network structure
- Must be careful not to impart knowledge in the encoding

Feature Encoding (input/output) is important

- Network needs “expressibility” to learn input/output encoding as well as device model
- Keep it simple, so more of network dedicated to model