

# Classifying Human Driving Behavior via Deep Neural Networks

Jae Hoon Kim

Drexel University

June 14, 2017

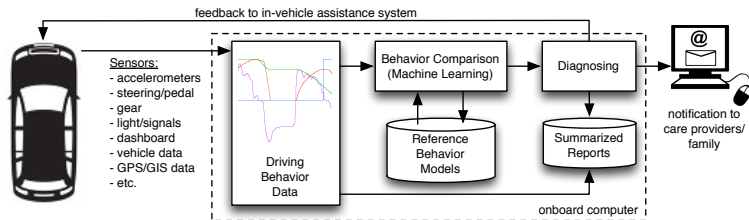
# Table of Contents

- 1 Introduction
- 2 Background
- 3 Data Set
- 4 Technical Approach
- 5 Experimental Result
- 6 Conclusion
- 7 Future Work

# Table of Contents

- 1 Introduction
- 2 Background
- 3 Data Set
- 4 Technical Approach
- 5 Experimental Result
- 6 Conclusion
- 7 Future Work

# Problem Statement



- Modern car systems collect real-time data of the car status and driving behavior of the driver
- Such data is very valuable for healthcare and research
- **Goal:** use driving behavior data to diagnose medical conditions

# Project Background

- Work in the context of the NSF SCH *Diagnostic Driving* project
- Partners:
  - Drexel University (lead)
  - Children's Hospital of Philadelphia (CHOP)
  - George Mason University
  - University of Central Florida
- CHOP and GM are in charge of data collection, and Drexel and UCF are in charge of machine learning

# Project Tasks

- Learn and understand LSTM and Auto-encoders neural networks
- Classify driving simulation data from novice and expert drivers using several different neural network models
- Compare result from three different neural network models created specifically for this task

# Table of Contents

- 1 Introduction
- 2 Background**
- 3 Data Set
- 4 Technical Approach
- 5 Experimental Result
- 6 Conclusion
- 7 Future Work

# Neural Networks - Introduction

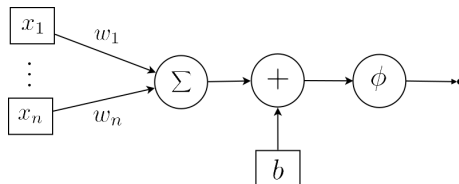


Figure: General Artificial Neuron

- The concept of artificial neurons or perceptrons was introduced by Warren McCullock and Walter Pitts [Ras15]
- An artificial neuron has input signals, weights, bias, and activation function



# Neural Networks - Introduction

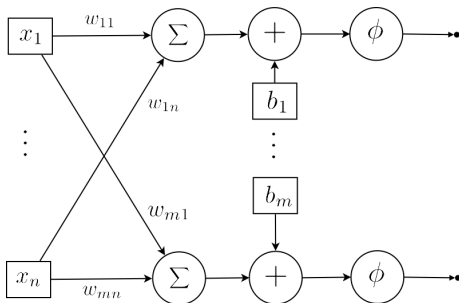


Figure: Example of a Neural Network

- Neural network is a network made up many artificial neurons

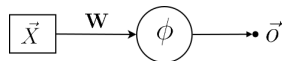


Figure: Simplified Neural Network

# Neural Networks - Forward Propagation

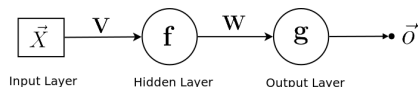


Figure: One Hidden Layer MLP

Pass inputs  $\vec{x}$  to outputs through hidden layers [Mur12]

$$\vec{h} = f(\vec{x}v + \vec{c}) = f(\vec{X}V)$$

$$\vec{o} = g(\vec{h}w + \vec{b}) = g(\vec{H}W)$$

$$\vec{o} = g(h(\vec{x}v + \vec{c})w + \vec{b}) = g(h(\vec{X}V)W)$$

# Neural Networks - Back Propagation

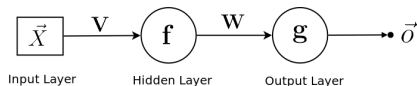


Figure: One Hidden Layer MLP

- Passes an error from the output layer to hidden layers [Mur12]
- Assumption:  
Activation Function:

$$f(z) = g(z) = \text{sigm}(z) = \frac{1}{1 + e^{-z}}$$

Error Function:

$$J(V, W) = \frac{1}{2} \sum_i (y_i - o_i)^2 = \frac{1}{2} \sum_i (y_i - g(h(\vec{X}V)W_i))^2$$

# Neural Networks - Back Propagation

Update  $W_{ij}$  -  $i$ th neuron for  $j$ th input:

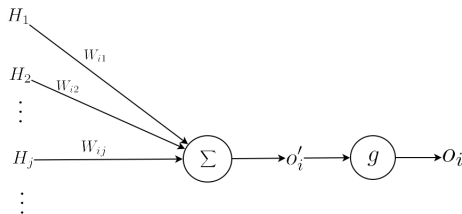


Figure: Back-propagation for  $W_{ij}$

$$W_{ij}^{next} = W_{ij} - \eta \frac{\partial J}{\partial W_{ij}} = W_{ij} - \eta \frac{\partial J}{\partial o_i} \frac{\partial o_i}{\partial o'_i} \frac{\partial o'_i}{\partial W_{ij}} = W_{ij} + \eta (y_i - o_i) o_i (1 - o_i) H_j$$

Let

$$\delta_i = \frac{\partial J}{\partial o'_i} = \frac{\partial J}{\partial o_i} \frac{\partial o_i}{\partial o'_i}$$

# Neural Networks - Back Propagation

Update  $V_{ij}$  -  $i$ th neuron for  $j$ th input:

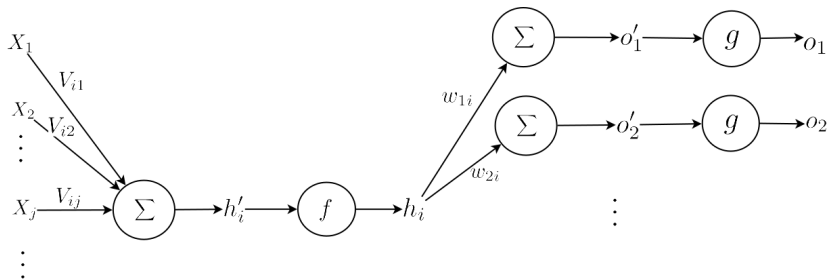


Figure: Back-propagation for  $V_{ij}$

$$V_{ij}^{next} = V_{ij} - \eta \frac{\partial J}{\partial V_{ij}} = V_{ij} - \eta \frac{\partial J}{\partial h_i} \frac{\partial h_i}{\partial h'_i} \frac{\partial h'_i}{\partial V_{ij}} = V_{ij} - \eta \sum_k (\delta_k w_{ki}) h_i (1 - h_i) X_j$$

# Neural Networks - Back Propagation

Update  $V_{ij}$  -  $i$ th neuron for  $j$ th input:

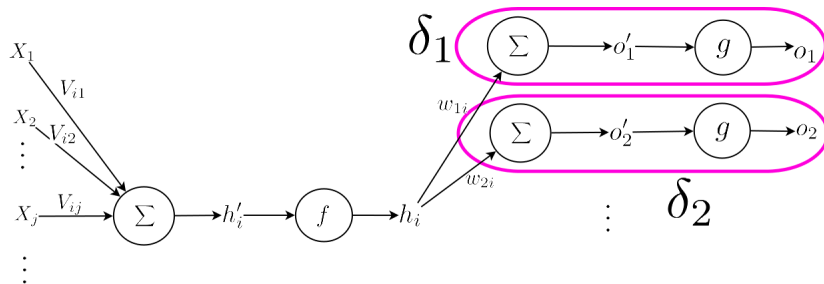


Figure: Back-propagation for  $V_{ij}$

$$V_{ij}^{next} = V_{ij} - \eta \frac{\partial J}{\partial V_{ij}} = V_{ij} - \eta \frac{\partial J}{\partial h_i} \frac{\partial h_i}{\partial h'_i} \frac{\partial h'_i}{\partial V_{ij}} = V_{ij} - \eta \sum_k (\delta_k w_{ki}) h_i (1 - h_i) X_j$$

# Long Short Term Memory Networks (LSTM)

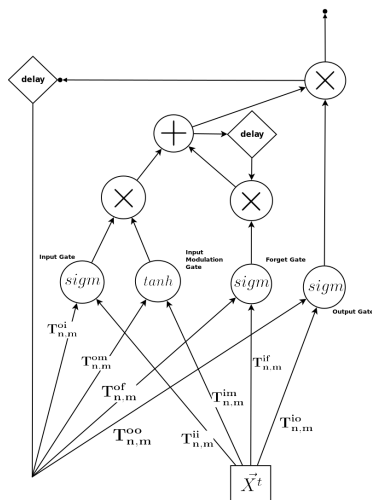
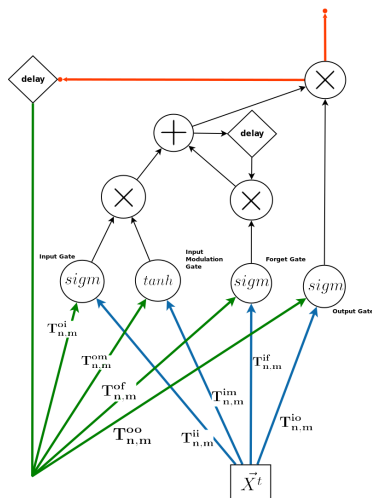


Figure: LSTM

- LSTM is a type of RNN introduced by Hochreiter and Schmidhuber [HS97]
- Four gates and Memory Cells handle long term dependencies

# Long Short Term Memory Networks (LSTM)

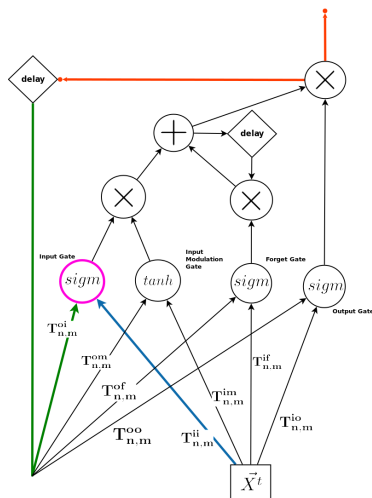


- Blue = Current Input
- Green = Previous Output

Figure: LSTM Input



# Long Short Term Memory Networks (LSTM)

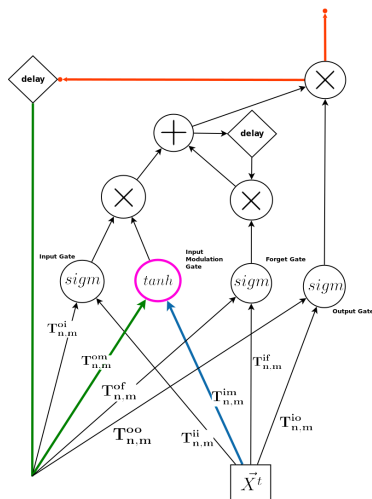


Input gate:

$$\vec{i}^t = \text{sigm}(\vec{X}^t T_{n,m}^{ii} + H^{t-1} T_{m,m}^{oi})$$

Figure: LSTM Input Gate

# Long Short Term Memory Networks (LSTM)

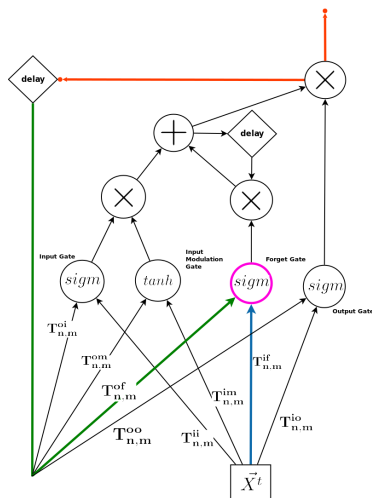


Input modulation gate:

$$\vec{m}^t = \tanh(\vec{X}^t T_{n,m}^{im} + H^{t-1} T_{m,m}^{om})$$

Figure: LSTM Input Module Gate

# Long Short Term Memory Networks (LSTM)

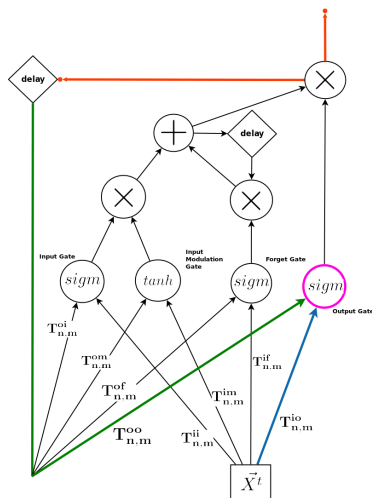


Forget gate:

$$\vec{f}^t = \text{sigm}(\vec{X}^t T_{n,m}^{if} + H^{t-1} T_{m,m}^{of})$$

Figure: LSTM Forget Gate

# Long Short Term Memory Networks (LSTM)

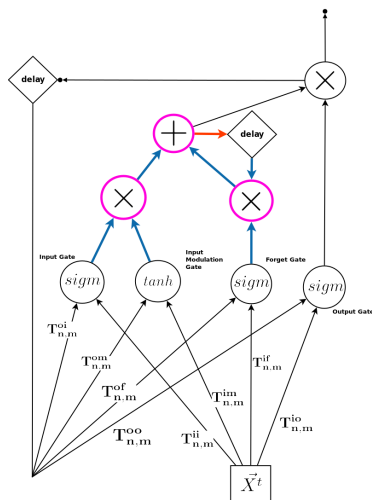


Output gate:

$$\vec{o}^t = \text{sigm}(\vec{X}^t T_{n,m}^{io} + H^{t-1} T_{m,m}^{oo})$$

Figure: LSTM Output Gate

# Long Short Term Memory Networks (LSTM)

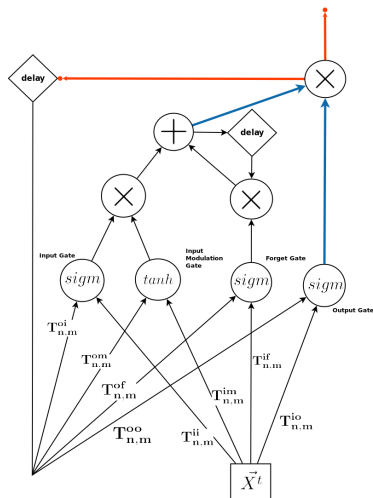


Memory cells:

$$\vec{c}^t = \vec{i}^t * \vec{m}^t + \vec{f}^t * \vec{c}^{t-1}$$

Figure: LSTM Memory Cells

# Long Short Term Memory Networks (LSTM)



Output:

$$\vec{h}^t = \vec{c}^t * \vec{o}^t$$

Figure: LSTM Output

# Long Short Term Memory Networks (LSTM)

Math model [ZSV14]

Input gate:

$$\vec{i}^t = \text{sigm}(\vec{X}^t T_{n,m}^{ii} + H^{t-1} T_{m,m}^{oi})$$

Input modulation gate:

$$\vec{m}^t = \tanh(\vec{X}^t T_{n,m}^{im} + H^{t-1} T_{m,m}^{om})$$

Forget gate:

$$\vec{f}^t = \text{sigm}(\vec{X}^t T_{n,m}^{if} + H^{t-1} T_{m,m}^{of})$$

Output gate:

$$\vec{o}^t = \text{sigm}(\vec{X}^t T_{n,m}^{io} + H^{t-1} T_{m,m}^{oo})$$

Memory cells:

$$\vec{c}^t = \vec{i}^t * \vec{m}^t + \vec{f}^t * \vec{c}^{t-1}$$

Output:

$$\vec{h}^t = \vec{c}^t * \vec{o}^t$$

# Auto-encoder (AE) - Single layer AE

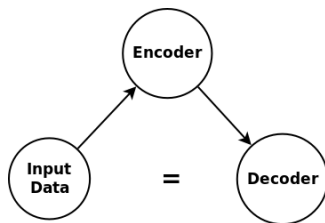


Figure: Abstract structure of Auto-encoder

Auto-encoder is a neural network to attempt to copy its inputs to its outputs [GBC16]



# Auto-encoder (AE) - Single layer AE

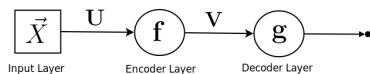


Figure: Basic Auto-encoder

$$\vec{e} = f(\vec{X}U)$$

$$\vec{d} = g(\vec{E}V)$$

$$E(\vec{x}, \vec{d}) = E(\vec{x}, g(\vec{E}V)) = E(\vec{x}, g(f(\vec{x}u + \vec{b}_e)v + \vec{b}_d))$$

# Auto-encoder (AE) - Multiple layer AE

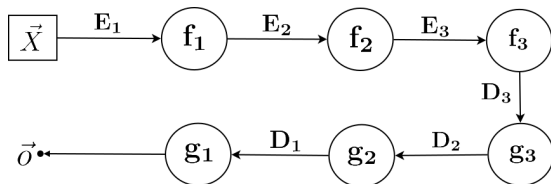


Figure: Multilayer Auto-encoder Example

- Multiple encoders and decoders
- Pretraining process [HS06]

# Auto-encoder (AE) - Multiple layer AE

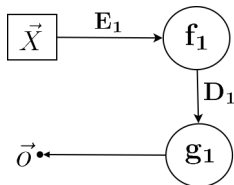


Figure: Pretraining First Step

$$\begin{aligned} E(\text{input}, \text{data\_from\_first\_decoder}) \\ = E(\vec{x}, g_1(f_1(\vec{X}E_1)D_1)) \end{aligned}$$

# Auto-encoder (AE) - Multiple layer AE

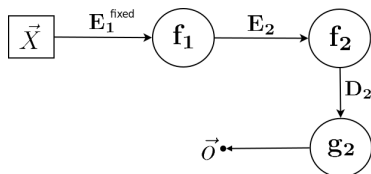


Figure: Pretraining Second Step

$$\begin{aligned} &E(\text{data\_from\_first\_encoder}, \text{data\_from\_second\_decoder}) \\ &= E(f_1(\vec{X}E_1^{\text{fixed}}), g_2(f_2(f_1(\vec{X}E_1^{\text{fixed}})E_2)D_2)) \end{aligned}$$

# Auto-encoder (AE) - Multiple layer AE

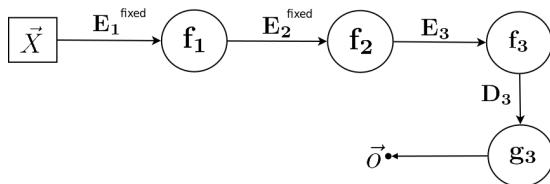


Figure: Pretraining Thrid Step

$$\begin{aligned} & E(\text{data\_from\_second\_encoder}, \text{data\_from\_third\_decoder}) \\ &= E(f_2(f_1(\vec{X}E_1^{\text{fixed}})E_2^{\text{fixed}}), g_3(f_3(f_2(f_1(\vec{X}E_1^{\text{fixed}})E_2^{\text{fixed}})E_3)D_3)) \end{aligned}$$

# Auto-encoder (AE) - Multiple layer AE

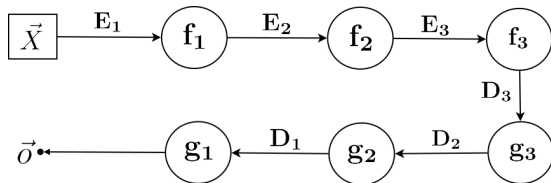


Figure: Multilayer Auto-encoder Example

$$E(input, output)$$
$$= E(\vec{x}, g_1(g_2(g_3(f_3(f_2(f_1(\vec{X}E_1)E_2)E_3)D_3)D_2)D_1))$$

# Table of Contents

- 1 Introduction
- 2 Background
- 3 Data Set**
- 4 Technical Approach
- 5 Experimental Result
- 6 Conclusion
- 7 Future Work



Figure: Driving Simulator at CHOP



- Collected in the high-fidelity simulator of the Center for Injury Research Prevention Studies at the *Children's Hospital of Philadelphia* (CHOP)
- 16 traces = 4 tracks  $\times$  (2 expert drivers + 2 inexperienced drivers)
- 100 features collected at 60Hz
  - Car status: velocity, steer, Brake, throttle and etc.
  - Environment status: current speed limit, and etc.
  - instruction: left turn, right turn, and etc.
- Two datasets
  - *raw dataset*: 98 features without time stamps
  - *filtered dataset*: 23 features - more important features from 100 features

**Table:** Length of each of the traces

<i>Trace</i>	<i>Driver</i>	<i>Track</i>	<i>Length</i>
Trace0	Expert0	Track0	50029
Trace1	Expert0	Track1	26375
Trace2	Expert0	Track2	29629
Trace3	Expert0	Track3	26298
Trace4	Expert1	Track0	51295
Trace5	Expert1	Track1	26674
Trace6	Expert1	Track2	29680
Trace7	Expert1	Track3	27075
Trace8	Inexpert0	Track0	49691
Trace9	Inexpert0	Track1	30058
Trace10	Inexpert0	Track2	26441
Trace11	Inexpert0	Track3	27373
Trace12	Inexpert1	Track0	47658
Trace13	Inexpert1	Track1	29380
Trace14	Inexpert1	Track2	26684
Trace15	Inexpert1	Track3	27255

- Min: 26298  
(about 7 minutes 18 seconds)
- Max: 51295  
(about 14 minutes 15 seconds)
- Average: 33224.6875  
(about 9 minutes 13 seconds)

# Table of Contents

- 1 Introduction
- 2 Background
- 3 Data Set
- 4 Technical Approach**
- 5 Experimental Result
- 6 Conclusion
- 7 Future Work

# Cross Validation & Normalization

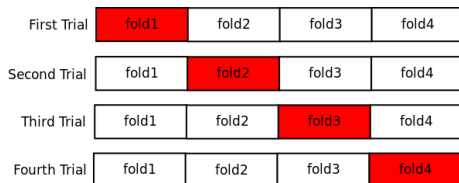


Figure: 4 Folds Cross Validation

- Cross Validation

- To validate experiment models when data is not large enough
- Divides a data set into  $K$  folds
- Uses the  $i$ th fold as the test set and other folds as the training set

- Normalization

- Normalize training set (mean 0, std 1)
- Normalize test set by mean and standard deviation of training set

# First Model

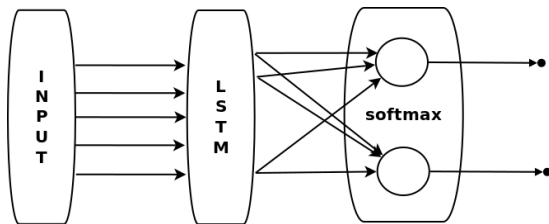


Figure: First experiment NN

- Hidden neurons in LSTM: 16, 32, 64, 128, and 256
- Without Auto-encoder on *filtered dataset*

## Second Model

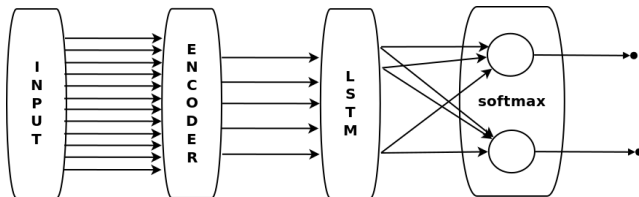


Figure: Second experiment NN

- Hidden neurons in LSTM: 16, 32, 64, 128, and 256
- With Single layer Auto-encoder on *raw dataset*:  
98 features  $\rightarrow$  25 features

# Third Model

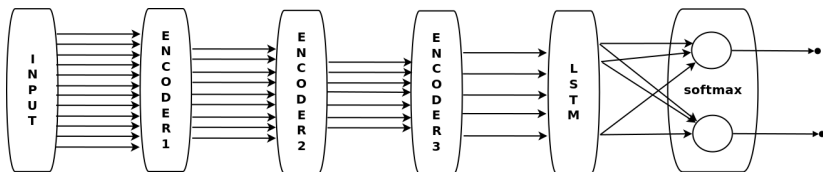


Figure: Third experiment NN

- Hidden neurons in LSTM: 16, 32, 64, 128, and 256
- With Three layer Auto-encoder on *raw dataset*:  
98 features  $\rightarrow$  75 features  $\rightarrow$  50 features  $\rightarrow$  25 features

- Period
  - 1 over 10
  - 1 over 20
  - 1 over 50
- Sampling Methods
  - Last sample from each period
  - Mean of each period
  - Gaussian filtered value of each period

Table: Sampling and Models

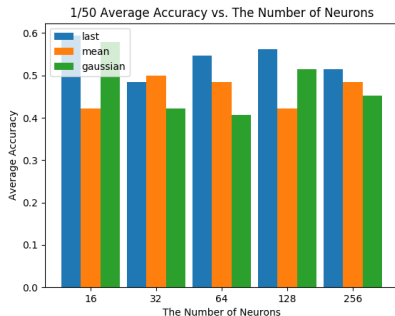
Period	Method	Model
1 over 10	last	<i>first, second, third</i>
	mean	<i>first</i>
	gaussian	<i>first</i>
1 over 20	last	<i>first</i>
	mean	<i>first</i>
	gaussian	<i>first</i>
1 over 50	last	<i>first</i>
	mean	<i>first</i>
	gaussian	<i>first</i>



# Table of Contents

- 1 Introduction
- 2 Background
- 3 Data Set
- 4 Technical Approach
- 5 Experimental Result**
- 6 Conclusion
- 7 Future Work

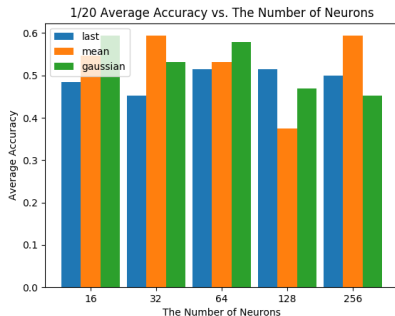
# First Model with 1 over 50



- Best: 0.59375 (59.375%) on *last* and 16 neurons
- Worst: 0.40625 (40.625%) on *gaussian* and 64 neurons
- Average: 0.492708 (49.2708%)

Figure: Result of first model with 1 over 50 *filtered dataset*

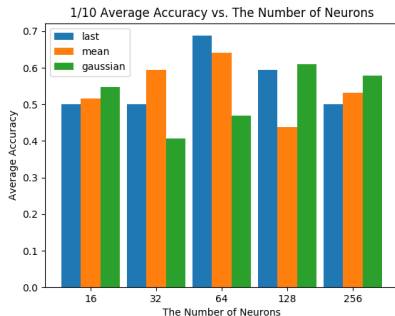
# First Model with 1 over 20



- Best: 0.59375 (59.375%) on *mean* and 32 and 256 neurons
- Worst: 0.375 (37.5%) on *mean* and 128 neurons
- Average: 0.515625 (51.5625%)

**Figure:** Result of first model with 1 over 20 *filtered dataset*

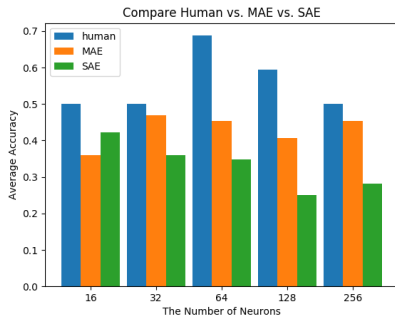
# First Model with 1 over 10



- Best: 0.6875 (68.75%) on *last* and 64 neurons
- Worst: 0.40625 (40.625%) on *gaussian* and 32 neurons
- Average: 0.540625 (54.00625%)

Figure: Result of first model with 1 over 10 *filtered dataset*

# Comparison of three models with 1 over 10, *last* method



Models with Auto-encoder show worse performance

**Figure:** Result of three models with 1 over 10, *last* method

# Comparison of three models with 1 over 10, *last* method

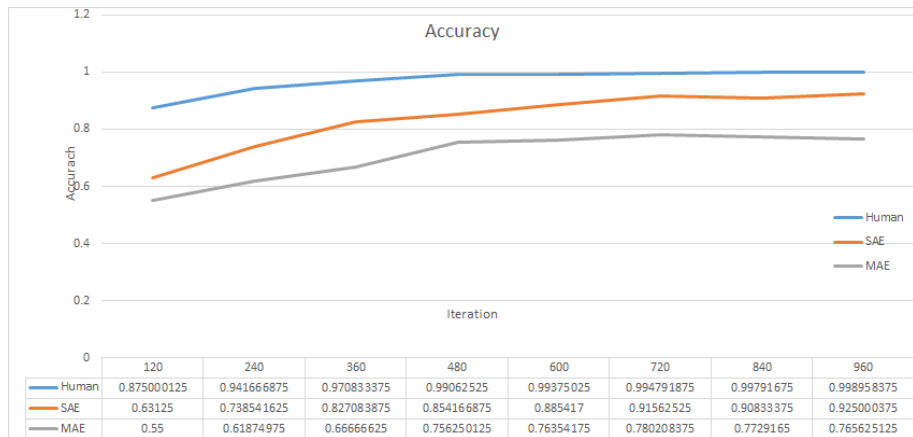


Figure: Three Models Training Accuracy

# Comparison of three models with 1 over 10, *last* method



Figure: Three Models Training Loss

# Problems with second and third models

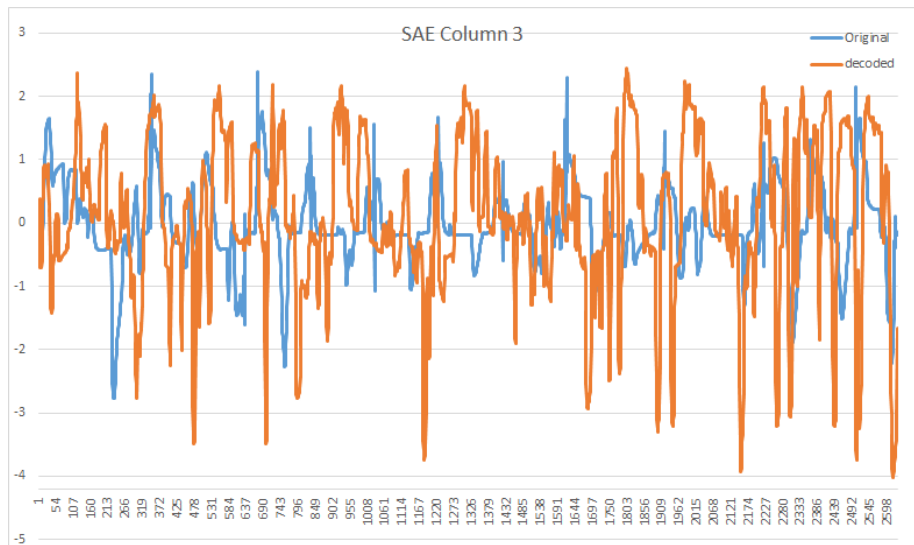


Figure: SAE, AE Error for Column 3



# Problems with second and third models

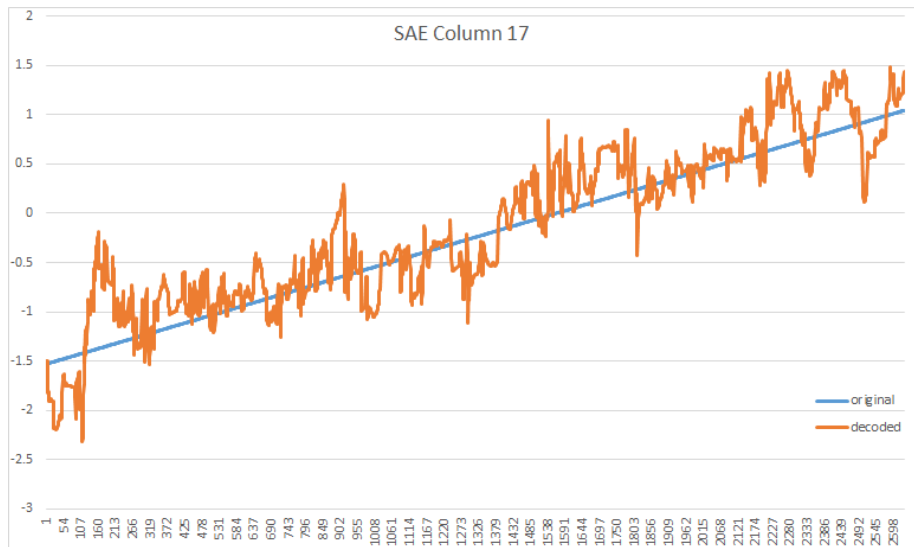


Figure: SAE, AE Error for Column 17

# Problems with second and third models

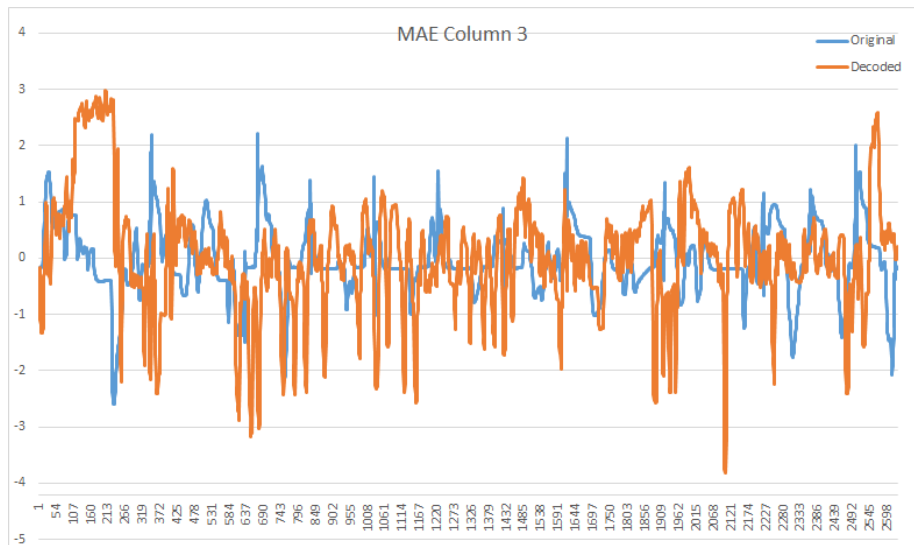


Figure: MAE, AE Error for Column 3

# Problems with second and third models

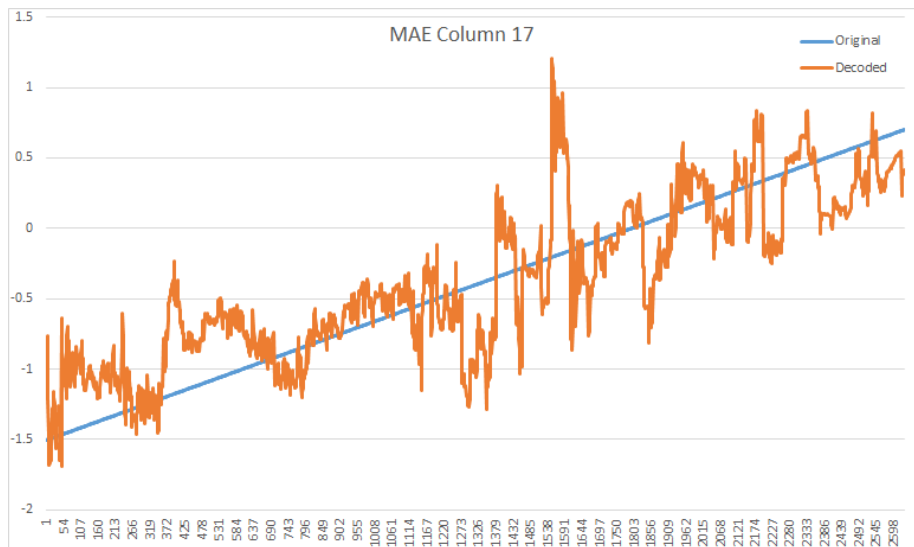


Figure: MAE, AE Error for Column 17

# Table of Contents

- 1 Introduction
- 2 Background
- 3 Data Set
- 4 Technical Approach
- 5 Experimental Result
- 6 Conclusion**
- 7 Future Work

# Conclusion - Data Size

- The number of samples in trace:  
From the result of experiments with different re-sampled periods, more densely re-sampled data gives better performance
- The number of traces:  
12 traces are not enough to train the proposed neural networks

# Conclusion - Limitation of Auto-encoders

- No optimize transfer matrices:  
When Auto-encoder tries to reduce dimensions, if encoder and decoder transfer matrices do not exist, encoded data has noise. It can make it hard to train subsequent neural networks.
- Always try to keep all information:  
Auto-encoder does not ignore or exclude unimportant features (it's unsupervised)

# Conclusion - Problem in training MAE

- Training time:  
Processing of pretraining takes a long time
- Learn error from previous layer during pretraining:  
Fix previous layer while the following auto-encoder layer is trained

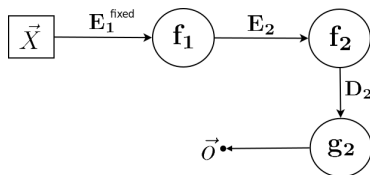


Figure: Pretraining Second Step

# Table of Contents

- 1 Introduction
- 2 Background
- 3 Data Set
- 4 Technical Approach
- 5 Experimental Result
- 6 Conclusion
- 7 Future Work**



- Experiment with more traces
- Research auto-encoder that accept some form of supervision and reduce dimensions with the ability to ignore unimportant features and keep information from the important features (and that can be used for sequential data with LSTMs)
- Try different ways to train MAE

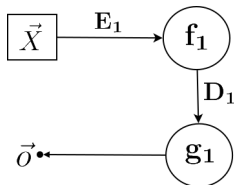


Figure: First Step of New Way to train MAE

$$\begin{aligned} & E(input, output) \\ &= E(\vec{x}, g_1(f_1(\vec{X}E_1)D_1)) \end{aligned}$$

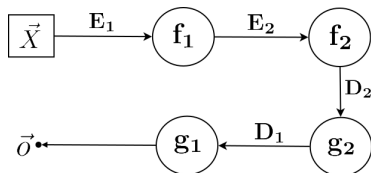


Figure: Second Step of New Way to train MAE

$$\begin{aligned} & E(input, output) \\ &= E(\vec{x}, g_1(g_2(f_2(f_1(\vec{X}E_1)E_2)D_2)D_1)) \end{aligned}$$

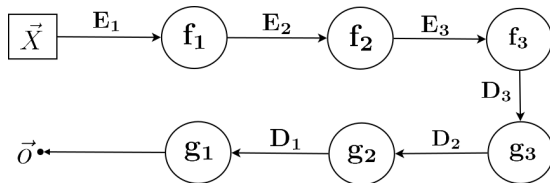








Figure: Thrid Step of New Way to train MAE

$$E(input, output)$$
$$= E(\vec{x}, g_1(g_2(g_3(f_3(f_2(f_1(\vec{X}E_1)E_2)E_3)D_3)D_2)D_1))$$

# Thank You

# References I

-  Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT Press, 2016, <http://www.deeplearningbook.org>.
-  Sepp Hochreiter and Jürgen Schmidhuber, *Long short-term memory*, Neural computation 9 (1997), no. 8, 1735–1780.
-  Geoffrey E Hinton and Ruslan R Salakhutdinov, *Reducing the dimensionality of data with neural networks*, science 313 (2006), no. 5786, 504–507.
-  Kevin P Murphy, *Machine learning: a probabilistic perspective*, MIT press, 2012.
-  Sebastian Raschka, *Python machine learning*, Packt Publishing Ltd, 2015.
-  Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals, *Recurrent neural network regularization*, arXiv preprint arXiv:1409.2329 (2014).