



# ML Noobs - Kaggle Competition Presentation

Jeffrey Feng, Yujian He, Tat-Hei Tsin, Dennis Wu

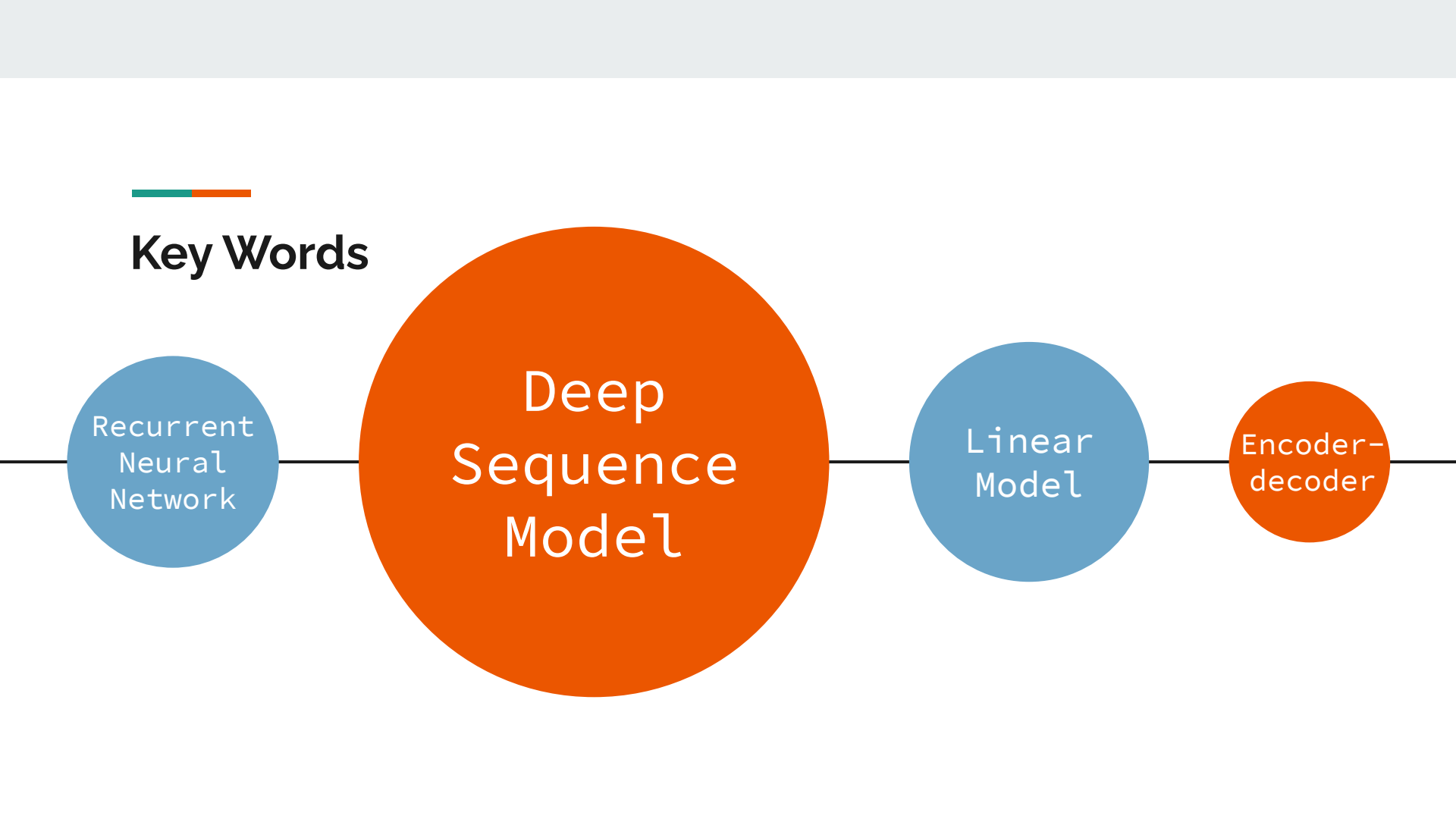


## Summary

- Team Member:
  - Jeffrey Feng(A15507377), Dennis Wu(A15537107), TaiHei Tsin(A15541127), Yujian He(A15386248)
- We use linear regression as our base model to find the most efficient features, then put those features into our deep learning model later. Accuracy did not improve.
- Time Series data training using simple RNN model with different layers
- One take away: “Keep things simple” (our Linear regression is still the best solution!)



## Key Words



Recurrent  
Neural  
Network

Deep  
Sequence  
Model

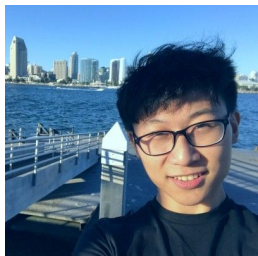
Linear  
Model

Encoder-  
decoder

---

# Introduction

# Team Introduction



Jeffrey Feng

- Third year student majoring in data science and cognitive science spec. ML



Yujian He

- Third year student majoring in data science and management science



Tat Hei Tsin (Dexter Tsin)

- Third year student majoring in Bioinformatics



Tat Hei Tsin

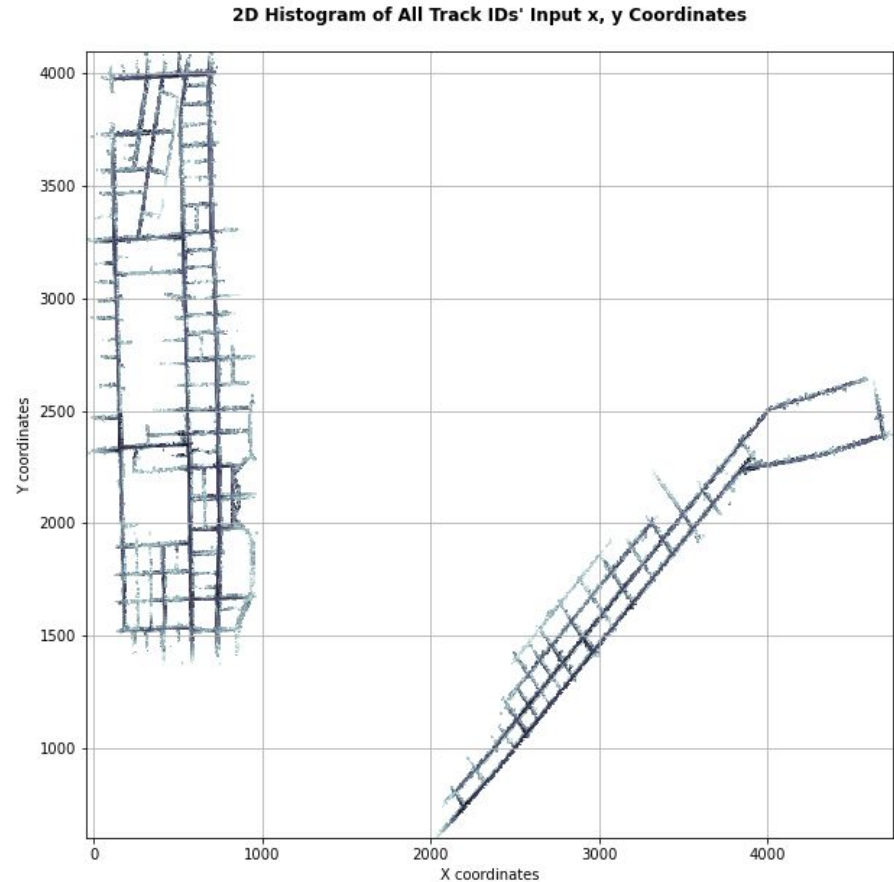
- Third year student majoring in data science

---

# Methodology

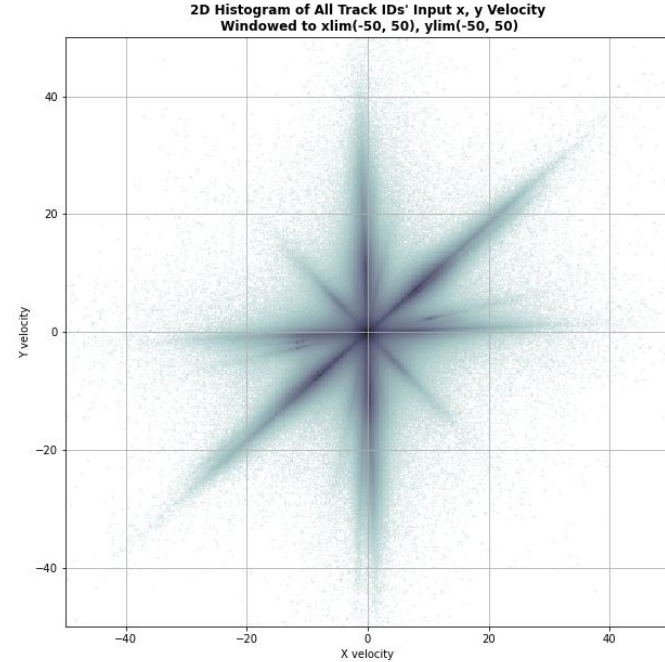
# Data Processing

- We visualize the 2D map graph using the x, y coordinates of all tracks
- From the map, we can see the distribution of all agents positions



# Data Processing

- This is the velocity of all agents with respect to x and y coordinates.







## Data Processing (continued):



- In order to remove redundant data for more efficient implementation, for each scene, we dropped all the car where car-mask == 0. This lead to 1812171 instances, where we can later split into smaller batches.
- According to physics, and our validation, the x, y coordinates along should cover the weight of the velocity, which should be able to estimated by using distance difference over time, so we did not use the velocity variables;
- Other variables, such as the lane information, were not included in the model;

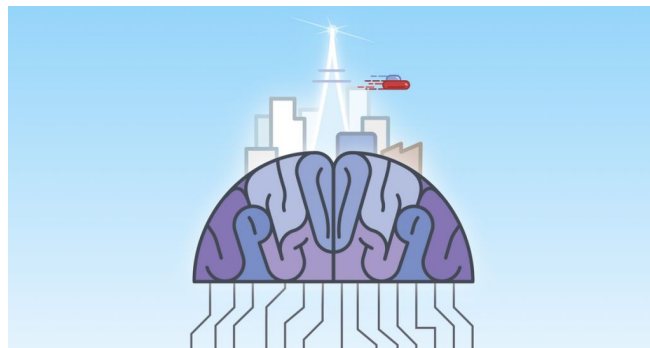
# Data Processing:



- The only variables we used are the x and y locations.
- However, instead of the normalized location value or the raw location values, we find the difference between the current location and the previous location, resulting in  $(19 - 1)$  difference in x values, and  $(19-1)$  difference in y values;
  - We saved the initial values for x, y location for the testing data.

# Deep Learning Model

- Multi-layer linear model
- LSTM
- Encoder-decoder





# Deep Learning Model: multiple linear layers

- The idea is inspired by the top team presentation on Tuesday, 6/1/2021, where they used 2 linear layers with a sigmoid layer.
  - Hidden units:  $36 + i \rightarrow 18 \rightarrow 6 \rightarrow 1$
- We confirmed that compared with other optimizer, [torch.optim.Adam](#) performed the best by obtaining the fastest convergence.
- Our batch size of 512 is large, because we have 1812171 instances.

```
[Epoch 1, 3540] loss: 0.12302844
[Epoch 2, 3540] loss: 0.12289178
[Epoch 3, 3540] loss: 0.12291648
[Epoch 4, 3540] loss: 0.12286522
[Epoch 5, 3540] loss: 0.12287612
[Epoch 6, 3540] loss: 0.12283727
[Epoch 7, 3540] loss: 0.12280002
[Epoch 8, 3540] loss: 0.12282911
[Epoch 9, 3540] loss: 0.12287485
[Epoch 10, 3540] loss: 0.12282326
[Epoch 11, 3540] loss: 0.12277096
[Epoch 12, 3540] loss: 0.12275954
[Epoch 13, 3540] loss: 0.12277921
[Epoch 14, 3540] loss: 0.12274584
[Epoch 15, 3540] loss: 0.12277067
[Epoch 16, 3540] loss: 0.12278083
[Epoch 17, 3540] loss: 0.12277364
[Epoch 18, 3540] loss: 0.12275211
[Epoch 19, 3540] loss: 0.12280717
[Epoch 20, 3540] loss: 0.12276342
```



## Deep Learning Model: multiple linear layers

- Auto-regression: used the 18 input x, 18 input y to predict the first x output, using the first linear regression classifier we have made from the training set.
- We used 19 input x (19 input from the real input, and 1 output from the prediction), 18 input y, in total 38 features to predict the first y output.
- We used 19 input x, and 19 input y, (38 features) to predict the second x output.

# Deep Learning Model: LSTM

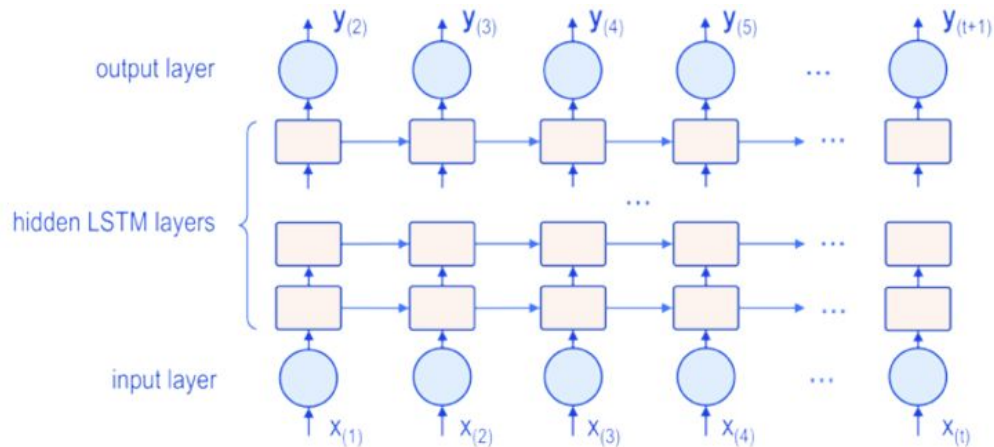
First try in recurrent neural network:

Multi layer LSTM

- 3 layer stacked LSTM
- 2048 hidden dimensions
- LSTM -> Dropout -> Linear -> output
- Predict velocity + distance

Second variation:

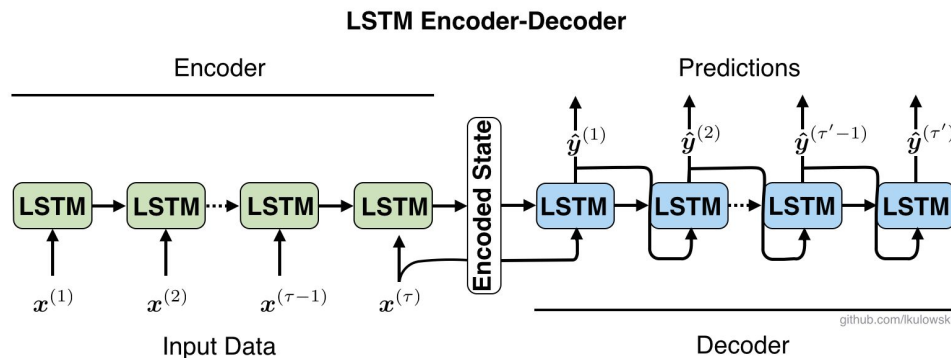
- LSTM -> Dropout -> Linear -> dropout -> linear -> prediction



# Deep Learning Model: Encoder-decoder

Encoder-decoder model:

- Two sequences of LSTM models: one for input and one for output
- Trained with input data on the encoder
- Extract the last hidden state of the encoder model
- Fit the hidden state to be the initial state of the decoder
- Use methods like teacher forcing to increase accuracy of the decoder model



# Engineering Tricks



- The very simple linear regression model (we will introduce that in experiment 1) takes <15 mins to predict 60 columns , while the DL models take much longer than that.
  - We can then use linear regression performance to evaluate which variables/features are important for the prediction task. We used this trick to confirmed that difference in location work better than location, and itself is better than location and velocity combined.
- Teacher forcing
  - When coding for our encoder-decoder model and iterating through the decoder, we would have a portion of data from real world data and another portion of data from our prediction in order to increase the training rate/ accuracy of the model

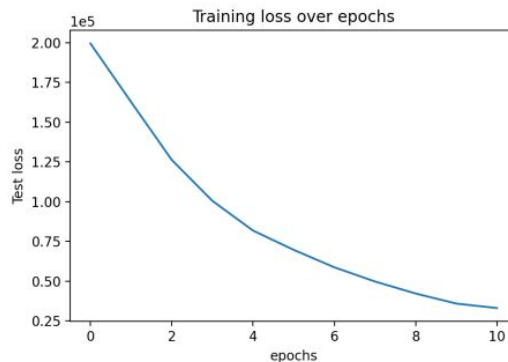
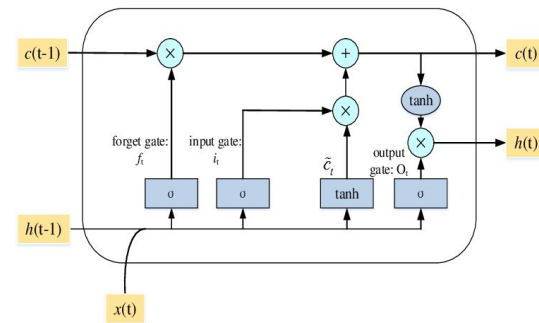


---

# Experiments

# Experiment 1: basic LSTM model

- The basic LSTM model does not work well!
  - Tried dropout rate tuning
  - More hidden states
  - More stacked LSTM
- Results are unsatisfactory:
  - RMSE around 800





Q

Search

Overview

Data

Code

Discussion





















Leaderboard

Rules

Team














My Submissions

Submit Predictions

1	ASharp	  	2.19798	11	7d
2	Two Three Three Three	   	2.19916	52	1h
3	Justin Allen		2.37659	50	4d
4	Fire Team Assemble	   	2.42332	17	5h
5	Leetcoders	   	2.52569	24	2d
6	ML noobs	   	2.52598	7	1h

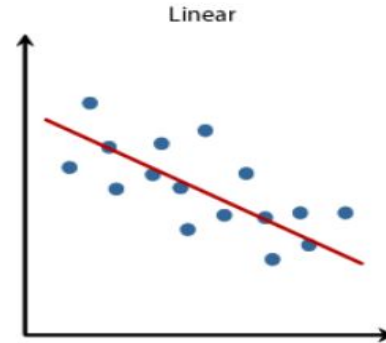
Your Best Entry ↑

Your submission scored 1169.96855, which is not an improvement of your best score. Keep trying!

7	Natural Dai	  	2.54703	6	4d
8	Elggak	  	2.56974	48	7d
9	SuperCats.punch()	  	2.61882	6	2h
10	Higher Brothers	   	2.93025	12	1h

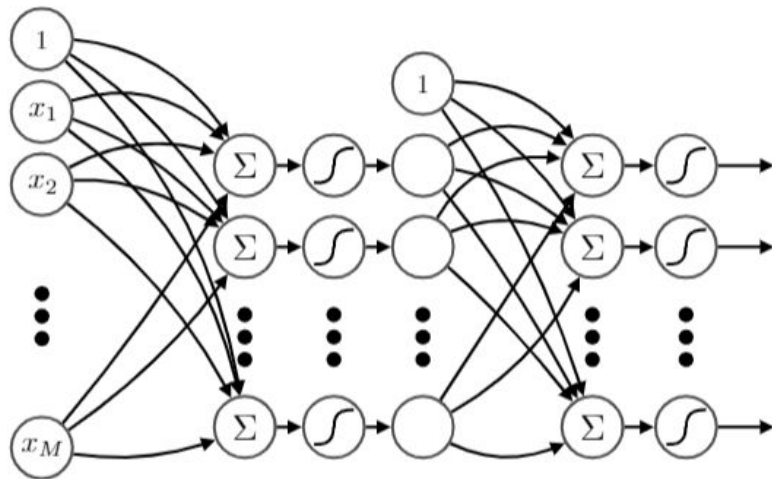
## Experiment 2: Closed form Linear Regression

- Also used auto-regression
- Very simple implementation: utilized `sklearn.linear_model.LinearRegression()`
- The accuracy of the model is surprisingly well on the test set, achieved leaderboard position #6 as of 5/16 with RMSE of 2.52598.



## Experiment 2 vs. Multiple-Layer DL model

- The closed form solution beats the MLDL mode by a lot!
  - The best MSRE with closed form solution is 2.52 on the kaggle competition, while the number increased to 10 for the MLDL model;
  - This is potentially due to the design of our multiple-layer deep learning model and if we have train it properly to match the power of closed form linear regression solution.

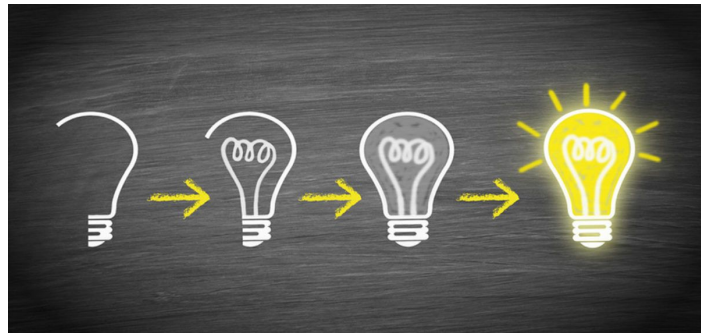


---

# Discussion

# What have you learned

- Keep Everything Simple
  - Sometimes complex model would worsen your prediction
  - Linear Regression and Linear Model Master
- Data Visualization is Important
  - Getting to know the data is much more important than training a model
- Always Learn from your Mistakes
  - We can correct our mistakes through the error and improve the model in a good manner
- Be Creative
  - We used the difference between the current location and the previous location to train in our model



# Future Work

- Multi-layer linear model
  - Improve the model design by adding more validations and convergence check (since we are creating 60 models, using same number of epochs for them might not work well)
- LSTM
- Attention based seq2seq model

$$\mathbf{o} = \sum_{i=1}^n b_i \mathbf{v}_i.$$

