Earthquake Prediction: by Stacking of Deep Neural Networks and other ML Model

Chunlin Zhu, Alice Chen, Cindy Le

Introduction

Problem: Earthquake Prediction

- Use real-time acoustic seismic data to predict the duration of time.
- Use one single, continuous, and large segment of data to train prediction models.
- Generalize from experimental data to field data.
- Make better predictions in real life to save lives and money.

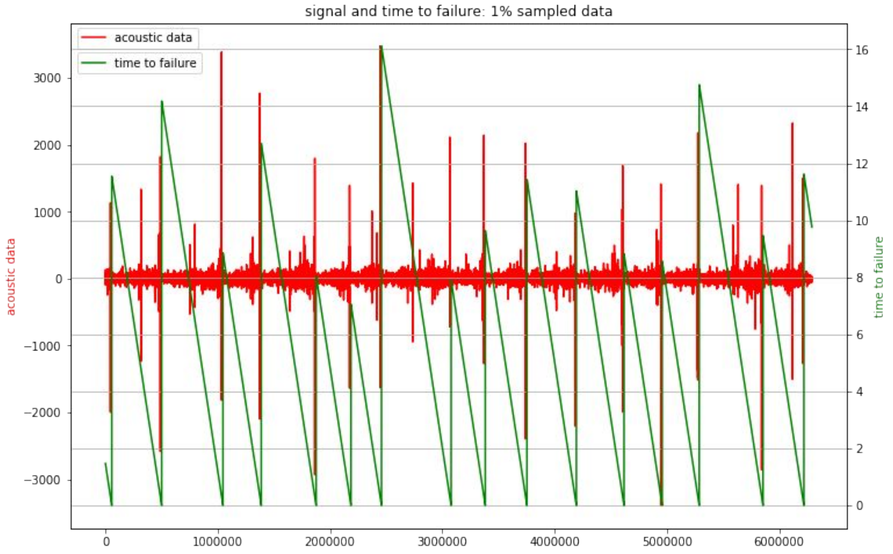
Related Work

- Current earthquake prediction methods mainly fall under one or several of the four categories
  - mathematical analysis
  - precursor signal investigation
  - traditional machine learning
  - deep learning
- Wang (2017) - an LSTM network to discover spatiotemporal correlations
- Ross (2019) - PhaseLink, a bidirected GRU model predicting based on seismic phase association

Wang, Q., Guo, Y., Yu, L., & Li, P. (2017). Earthquake prediction based on spatio-temporal data mining: an LSTM network approach. *IEEE Transactions on Emerging Topics in Computing*.
Ross, Z. E., Yue, Y., Meier, M. A., Hauksson, E., & Heaton, T. H. (2018, December). A Deep Learning Approach to Seismic Phase Association. In *AGU Fall Meeting Abstracts*.

Acoustic Data Set

A single sequence of the seismic signal that contains ~600 million data points of signal value and its corresponding time interval.



Feature Engineering

Experiments with Feature Generation

- We experiment with basic statistics features
- Compare 4 features with 36 features using the same model (LSTM, GRU)
- More features lead to better performance
- Sliding window of 1000: Considering the time of generating training features on fly.
- Extract features on fly vs. extract features beforehand

Feature Extraction:
Andrew’s Script (831 features)

```
# basic stats

# basic stats on absolute values

# geometric and harminic means

# k-statistic and moments

# aggregations on various slices of data

# calc_change_rate on slices of data

# percentiles on original and absolute values

# exponential rolling statistics

。。。
```

Sequence Models

Recurrent Neural Network: GRU & LSTM

Layer (type)	Output Shape	Param #
cu_dnngru_4 (CuDNNGRU)	(None, 48)	12384
dense_10 (Dense)	(None, 100)	4900
dense_11 (Dense)	(None, 10)	1010
dense_12 (Dense)	(None, 1)	11

Total params: 18,305
Trainable params: 18,305
Non-trainable params: 0

Layer (type)	Output Shape	Param #
cu_dnnlstm_1 (CuDNNLSTM)	(None, 48)	16512
dense_1 (Dense)	(None, 100)	4900
dense_2 (Dense)	(None, 10)	1010
dense_3 (Dense)	(None, 1)	11

Total params: 22,433
Trainable params: 22,433
Non-trainable params: 0

CNN + RNN

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_4 (Conv1D)            (None, None, 32)          3488
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, None, 32)          0
_________________________________________________________________
conv1d_5 (Conv1D)            (None, None, 64)          6208
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, None, 64)          0
_________________________________________________________________
cu_dnnlstm_4 (CuDNNLSTM)     (None, 48)                21888
_________________________________________________________________
dense_7 (Dense)              (None, 100)               4900
_________________________________________________________________
dense_8 (Dense)              (None, 10)                1010
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 11
=================================================================
Total params: 37,505
Trainable params: 37,505
Non-trainable params: 0
_________________________________________________________________
```

Stacked LSTM

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
cu_dnnlstm_12 (CuDNNLSTM)    (None, None, 48)          16512     
_________________________________________________________________
cu_dnnlstm_13 (CuDNNLSTM)    (None, None, 48)          18816     
_________________________________________________________________
cu_dnnlstm_14 (CuDNNLSTM)    (None, None, 48)          18816     
_________________________________________________________________
cu_dnnlstm_15 (CuDNNLSTM)    (None, 48)                18816     
_________________________________________________________________
dense_17 (Dense)             (None, 100)               4900      
_________________________________________________________________
dense_18 (Dense)             (None, 10)                1010      
_________________________________________________________________
dense_19 (Dense)             (None, 1)                 11        
=================================================================
Total params: 78,881
Trainable params: 78,881
Non-trainable params: 0
_________________________________________________________________
```

Ensemble of Simple Neural Networks

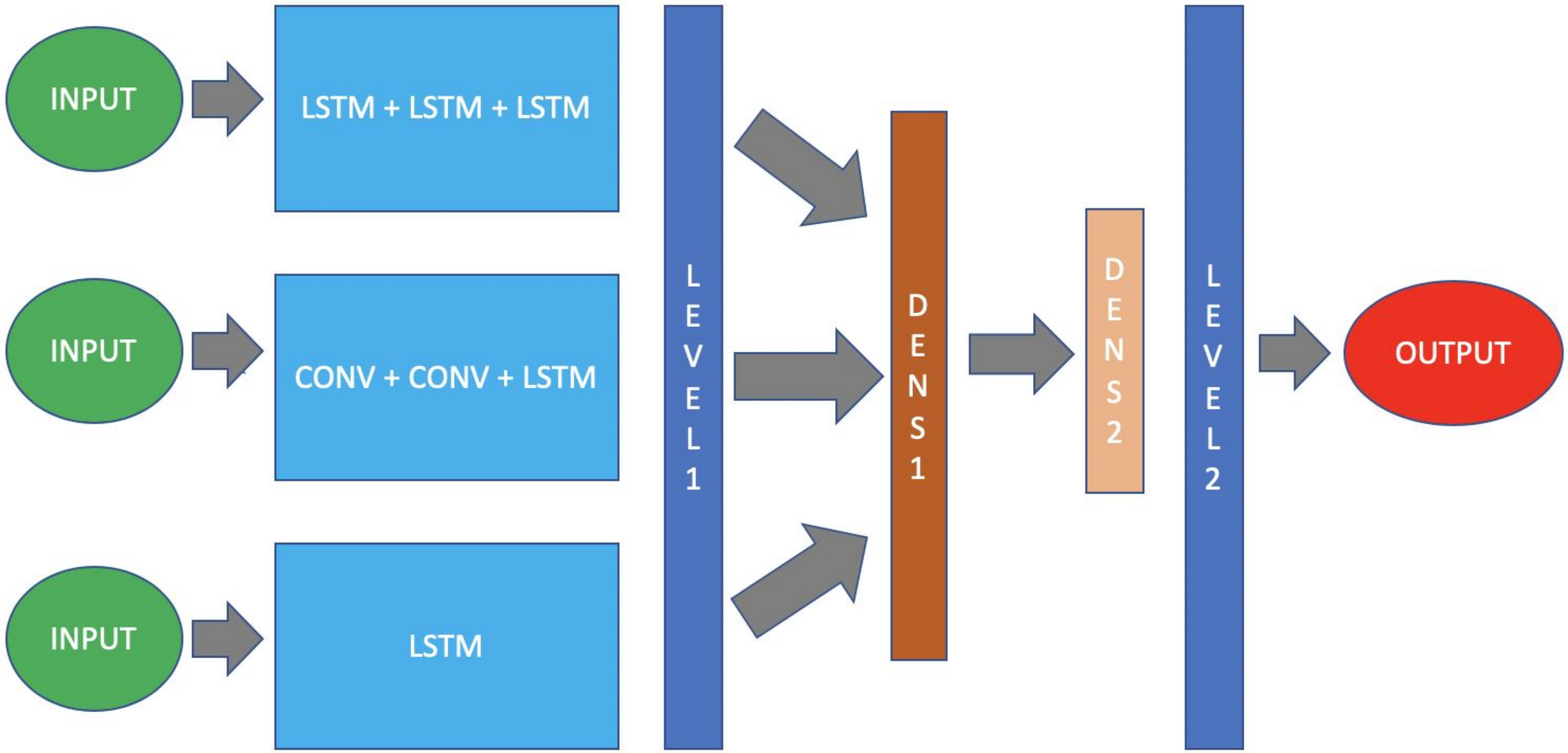
- We experiment with this model mainly to test the power of ensemble.
  - Build a simple neural network.
  - Train this model 10 times separately and save the 10 models.
  - Take average of the 10 simple models.
- This strategy improves the performance of a simple neural network significantly and we thus decide to further use the ensemble technique.

Stacking

Naive Weight-Average Ensemble

- Take a weighted average of the several models.
- Such a simple model is already able to generate fairly good results.
- No proper explanation for the weights we assign to each model.

Integrated Stacking Model



Results & Conclusions

Model Scores

Model Name	Average Absolute Error
LSTM	1.509
GRU	1.536
CNN + LSTM	1.510
Stacked LSTM	1.495
Simple NN Ensemble	1.501
Weighted Average Ensemble	1.477
Integrated Stacking	1.443
Stacking with other Non Deep Learning Models	1.419

Our best result currently rank top 2% on kaggle

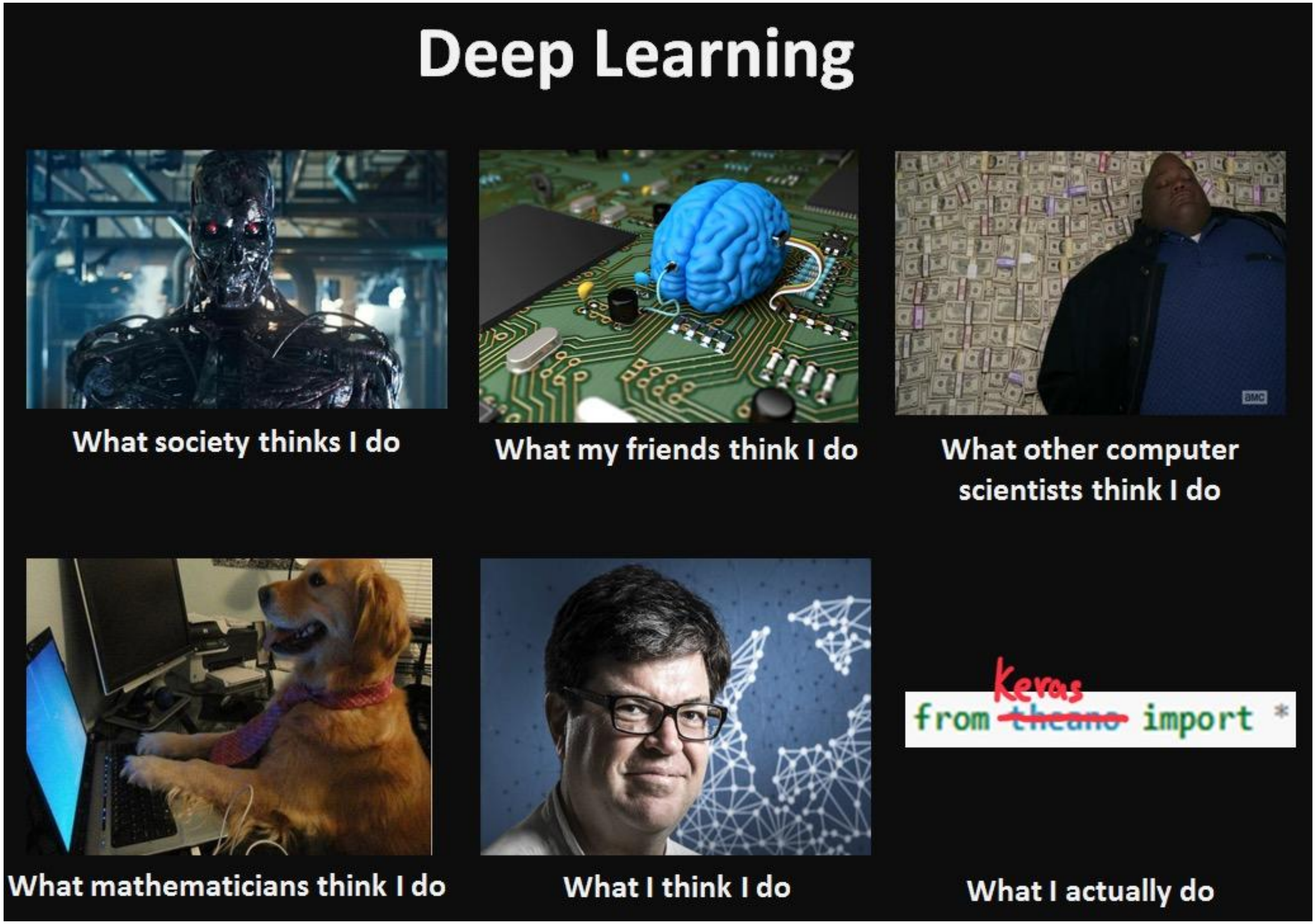
Conclusions

- Using the stacking technique in earthquake prediction, the model performs better than any single sub-level model.
- Linearly combining the prediction results from different models in general performs better since the prediction task is evaluated by average absolute error of the time difference.

Future Works

- Adjust and fine-tune the stacking part of the model.
- Separate training data for distinct levels of the stacked model to eliminate over-fitting.
- Generalize prediction methods from lab settings to natural ones.

What's More?


Deep Learning
What society thinks I do
What my friends think I do
What other computer scientists think I do
What mathematicians think I do
What I think I do
Keras
from ~~theano~~ import *
What I actually do

Q&As?