Master's Thesis: Time-Sequential Wind Dancing Signal Anomaly Detection

Xiaonan Wang
University of Electronic Science and Technology of China
2018.06
(Reorganized and Translated in 2023.04)

Table of Contents

1 Introduction

Research Objective Design and build methods for time-sequential anomaly detection

Traditional Approaches:

- Mechanics Methods
- Traditional data mining methods

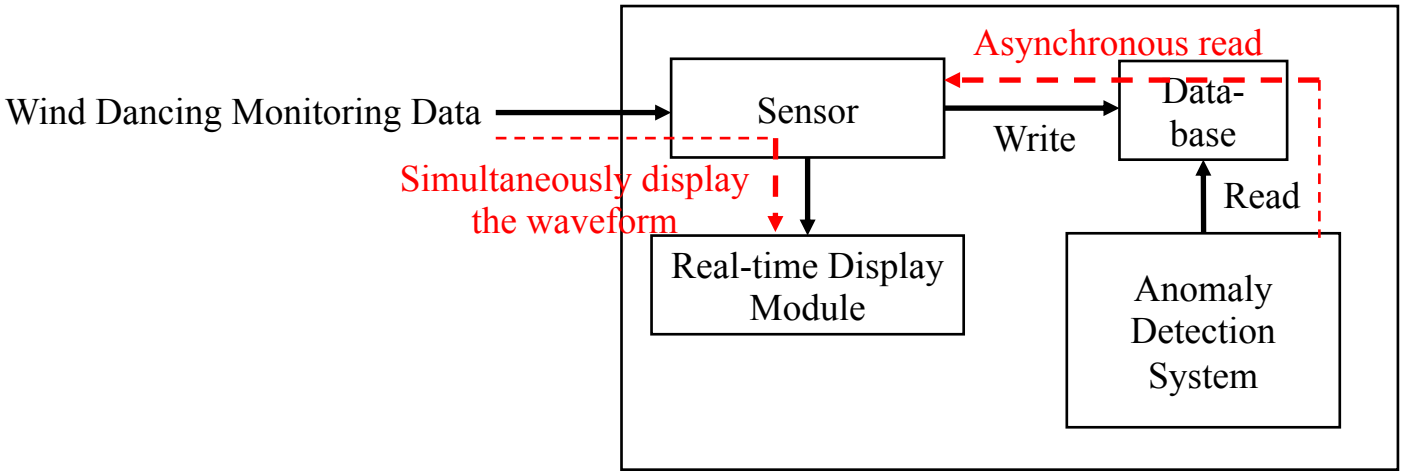
Research Objectives:

- Mining useful information from time-sequential wind dancing signal
- Anomaly Detection

Proposed:

- A Benchmark contains sequential token data
- Integrate time-domain and frequency-domain feature to 2D fusion feature
- Batch Normalization with Sliding-Window Preprocessing (BNSWP)
- CNN classifier and RNN predictor

2 Wind Dancing Monitoring System



3 Benchmark Dataset

Original wind dancing signal of a month:

$\{sample_1, sample_2, \dots, sample_i, \dots\}$

Framing

Framed time-sequential tokens:

$\{token_1: \{sample_1, \dots, sample_n\} \dots, token_i: \{sample_1, \dots, sample_n\}, \dots\}$

Consists of n samples within $frame_i$

$len(frame_i) = 50s$

$\frac{60s/min \times 60min/hour \times 24hour/day}{50s/frame}$ = 1728frames/day

Split

Remaining Part

Abnormal Labeling

1. Relatively large dancing amplitude
(empirical observation + box-plot statistics)
2. Has low frequency harmonics:

$corr_i > corr_{th}$

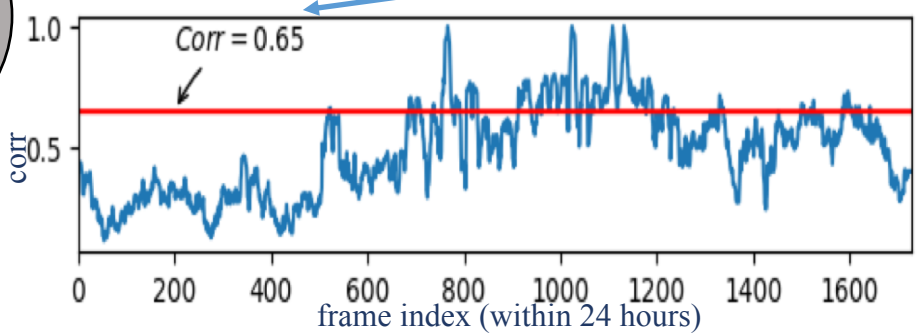
$corr_i = \max_{1 \le k \le K} PearsonCorrealtion(FFT(token_i), FFT(token\_anomaly_k))$

$corr_{th} = 0.65$

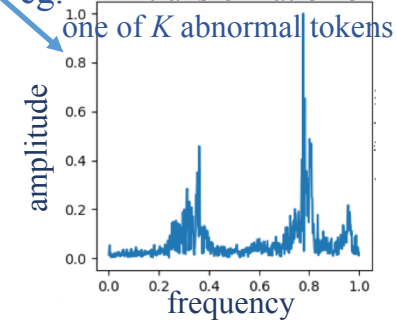
normal tokens *label*: 0 (933120 tokens)

abnormal tokens *label*: 1 (42660 tokens)

Benchmark Dataset

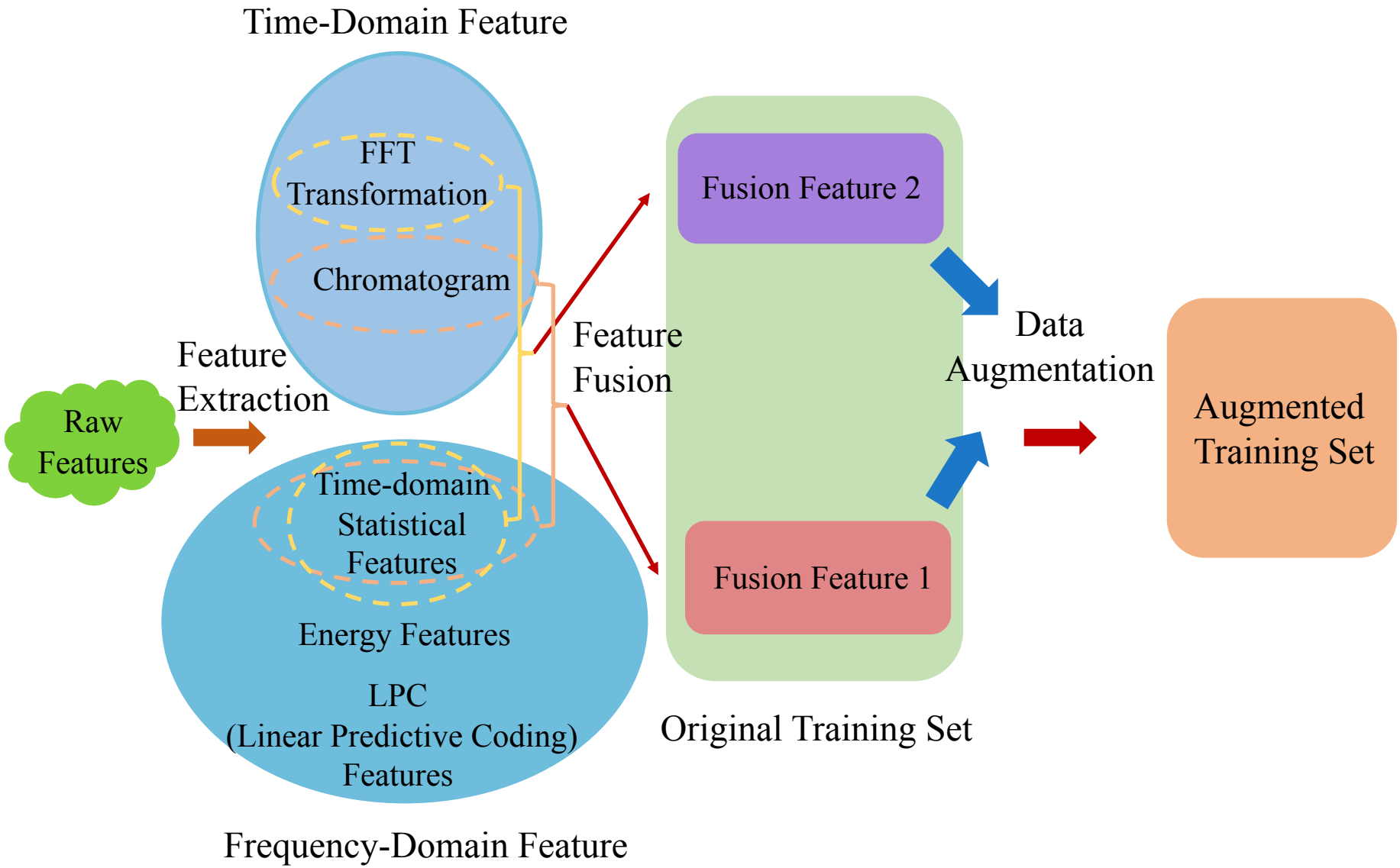


eg. FFT-transformation of one of K abnormal tokens



4 Feature Engineering

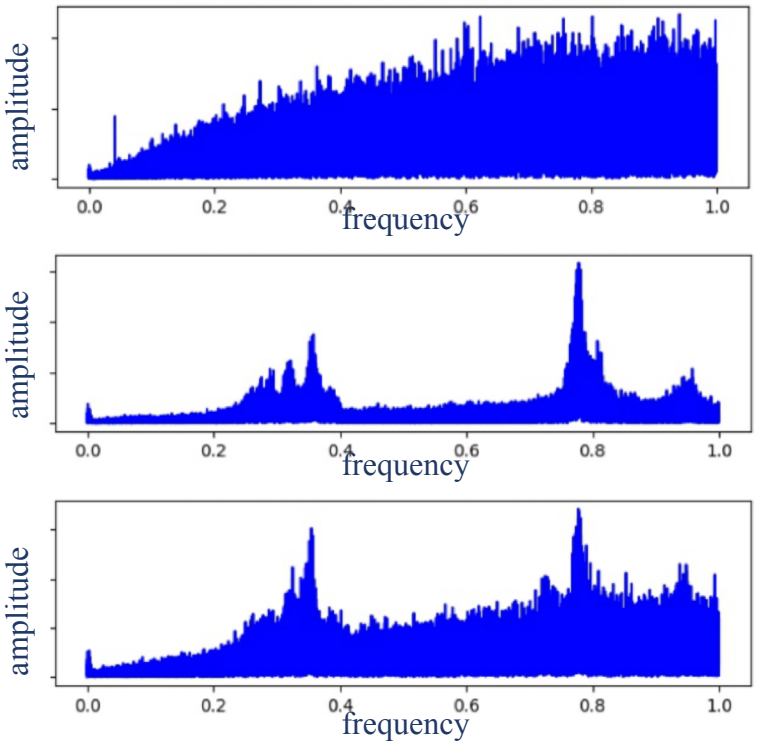
Overview of Feature Engineering



4 Feature Engineering

Frequency-Domain Feature

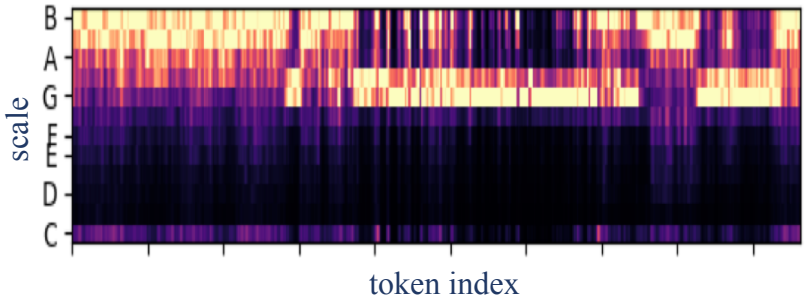
FFT Transform of Certain Token



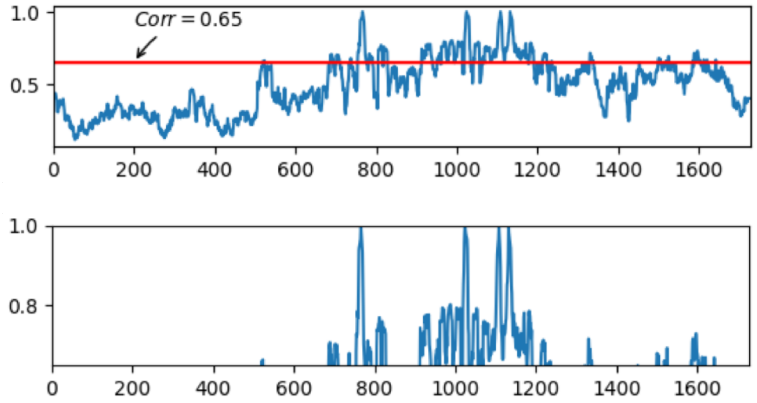
Normal

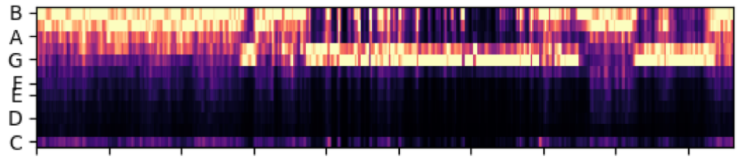
Abnormal

Chromatogram Changes as Token Varies



Corr vs Chromatogram





Chromatogram Reference:
D. P. W. Ellis, G. E. Poliner. Identifyingcover songs‘ with chroma features and dynamic programming beat tracking[C]. Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on. IEEE, 2007, 4: IV-1429-IV-1432

4 Feature Engineering

Time-Domain Feature

Step1: Calculate statistical values

$timeMax_i = \max\{abs(sample_j), sample_j \in token_i\}$

$timeMaxMin_i = \max\{sample_j \in token_i\} - \min\{sample_j \in token_i\}$

$timeMean_i = \text{mean}\{abs(sample_j), sample_j \in token_i\}$

$timeStd_i = \text{std}\{sample_j \in token_i\}$ std: standard deviation

$timeRMS_i = \text{RMS}\{sample_j \in token_i\} = \text{sqrt}(\text{mean}\{sample_j{}^2, sample_j \in token_i\})$

Step2: Batch Normalization with Sliding-Window Preprocessing (BNSWP)

Input: Values of x over a sliding-window: $\mathcal{SW} = \{x_{l\ldots r}\}$;
Size of sliding-window $m = r - l + 1$;
For $i < m - 1$: $l = 0,\ r = m - 1$;
Else: $\begin{cases} l = i - \frac{m-1}{2},\ r = i + \frac{m-1}{2}; & when\ m\ is\ odd \\ l = i - \frac{m}{2} + 1,\ r = i + \frac{m}{2}; & when\ m\ is\ even \end{cases}$
Parameters to be learned: γ_1, β_1

Output: $\{y_i = \text{SWP}_{\gamma_1,\beta_1}(x_i)\}$

$\widehat{x_i} \leftarrow \frac{x_i}{\max_{\mathcal{SW}}\{x_j\}} + 1$ // normalize

$y_i \leftarrow \gamma_1 \widehat{x_i} + \beta_1 \equiv \text{SWP}_{\gamma_1,\beta_1}(x_i)$ // scale and shift

Algorithm 1: Sliding-Window Preprocess, applied to $x = fea_i$ (e.g.$timeMax_i$) over a sliding-window in time-ordered training set.

Note: m is decided by the length of sliding-window, we choose 24 hours as the sliding-window's length, so $m = 1728$ tokens,

4 Feature Engineering

Time-Domain Feature

Step2: Batch Normalization with Sliding-Window Preprocessing (BNSWP)

Input: Values of x over a mini-batch $\mathcal{B} = \{x_{1\ldots n}\}$;
Parameters to be learned: γ_2, β_2

Output: $\{z_i = \mathrm{BN}_{\gamma_2,\beta_2}(y_i) = \mathrm{BN}_{\gamma_2,\beta_2}(\mathrm{SWP}_{\gamma_1,\beta_1}(x_i))\}$

Algorithm 2: Batch Normalization with Sliding-Window Preprocess (BNSWP), applied to x over a mini-batch in shuffled training set.

Input: Network N with trainable parameters Θ;
subset of shuffled inputs $\{x\}$

Output: BNSWP network for inference, $N^{\mathrm{inf}}_{\mathrm{BNSWP}}$

1: $N^{\mathrm{tr}}_{\mathrm{BNSWP}} \leftarrow N$ // Training BNSWP network

2: Add $z = \mathrm{BNSWP}_{\gamma_1,\beta_1,\gamma_2,\beta_2}(x)$ to $N^{\mathrm{tr}}_{\mathrm{BNSWP}}$ (Alg. 2)

3: Train $N^{\mathrm{tr}}_{\mathrm{BNSWP}}$ to optimize $\Theta \cup \{\gamma_1, \beta_1, \gamma_2, \beta_2\}$

4: $N^{\mathrm{inf}}_{\mathrm{BNSWP}} \leftarrow N^{\mathrm{tr}}_{\mathrm{BNSWP}}$ // Inference BNSWP network
// with frozen parameters

5: Process multiple training sliding-window $\mathcal{SW}$, each of size m, and average over them:

$$\hat{x} \leftarrow \frac{x_i}{\mathrm{E}[\max_{\mathcal{SW}}\{x_j\}]} + 1$$

$$y \leftarrow \gamma_1 \hat{x} + \beta_1$$

6: Process multiple training mini-batches $\mathcal{B}$, each of size n, average over $\mathcal{B}$; replace… // see original BN

Algorithm 3: Training a BNSWP Network

4 Feature Engineering

Feature Fusion

Time-Domain Feature with BNSWP

Feature Fusion

Chromatogram

$$Q = \begin{bmatrix} timeMax \cdot \overrightarrow{Chroma} \\ timeMaxMin \cdot \overrightarrow{Chroma} \\ timeMean \cdot \overrightarrow{Chroma} \\ timeStd \cdot \overrightarrow{Chroma} \\ timeRMS \cdot \overrightarrow{Chroma} \end{bmatrix}$$

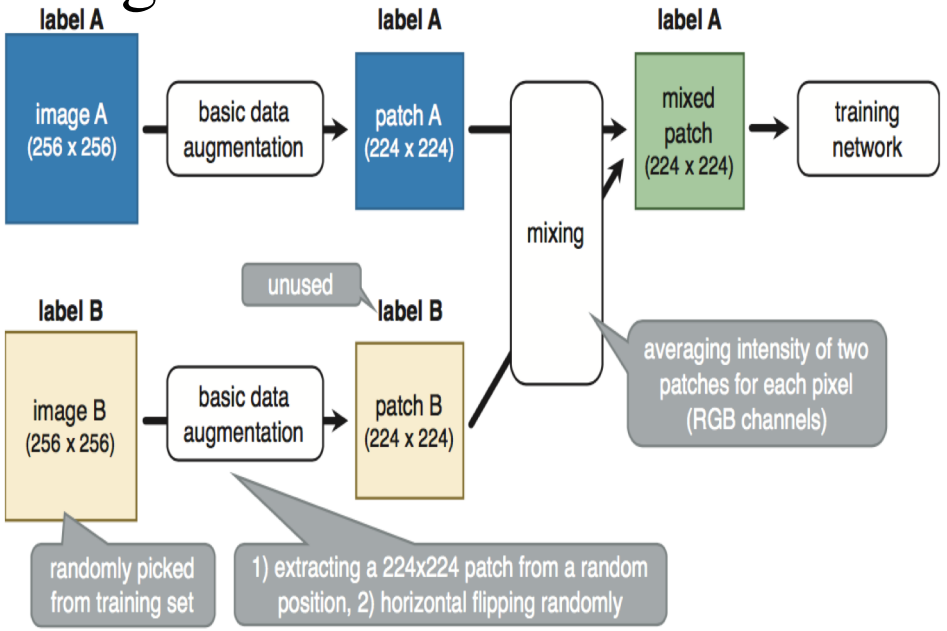
Fusion Feature 1
(5 rows 12 cols)

FFT Transformation

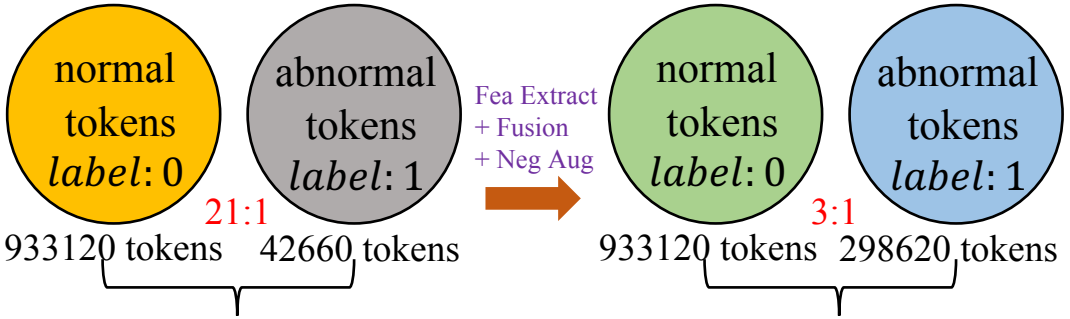
$$Q = \begin{bmatrix} timeMax \cdot FFT \\ timeMaxMin \cdot FFT \\ timeMean \cdot FFT \\ timeStd \cdot FFT \\ timeRMS \cdot FFT \end{bmatrix}$$

Fusion Feature 2
(5 rows 50 cols)

Data Augmentation



Reference： H. Inoue. Data Augmentation by Pairing Samples for Images Classification[J]. arXiv preprint arXiv:1801.02929, 2018



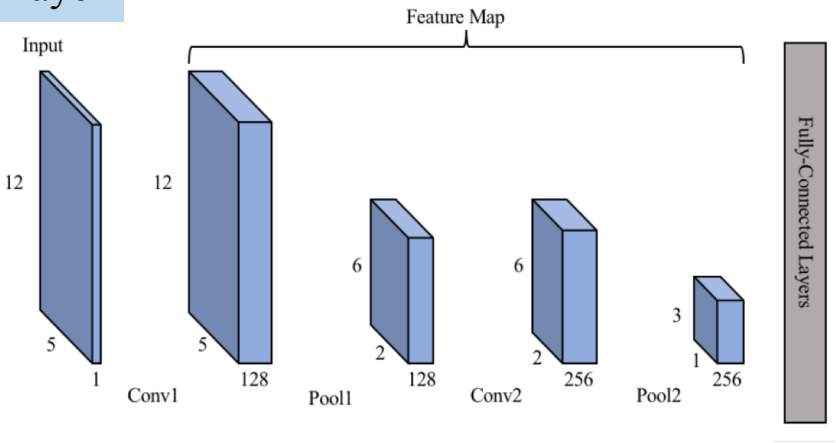
5 Time-Sequential Detection and Prediction

Detection: CNN Classifier

CNN Classifier for Feature-set 1 (Fusion 1)

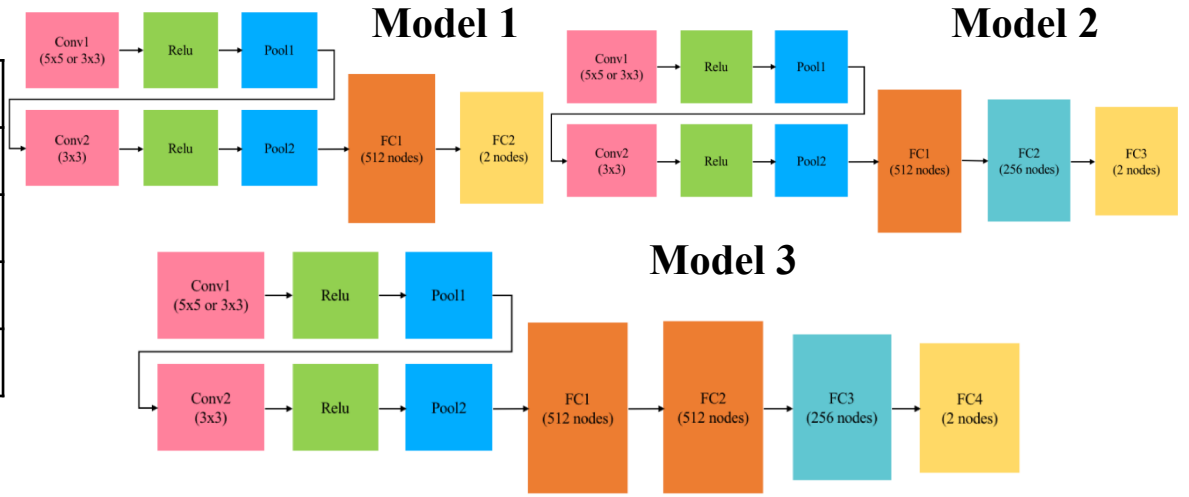
Convolutional Layer

	Conv-Kernel Size	Stride	Padding	Output Channels
Conv-Layer 1	5 × 5	1	2	128
	3 × 3	1	1	128
Pool-Layer 1	2 × 2	2	0	--
Conv-Layer 2	3 × 3	1	1	256
Pool-Layer 2	2 × 2	2	0	--



Fully Connected Layer

	Model 1	Model 2	Model 3
FC-Layer 1	512 units	512 units	512 units
FC-Layer 2	2 units	256 units	512 units
FC-Layer 3	--	2 units	256 units
FC-Layer 4	--	--	2 units

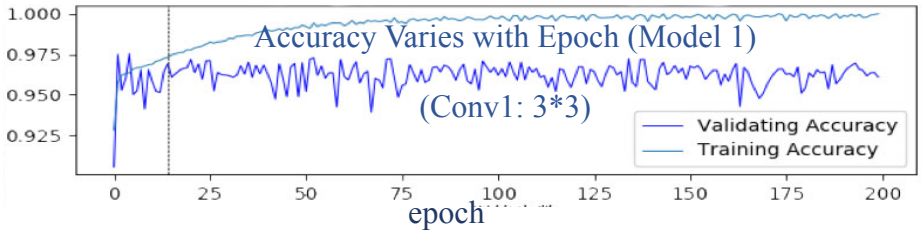


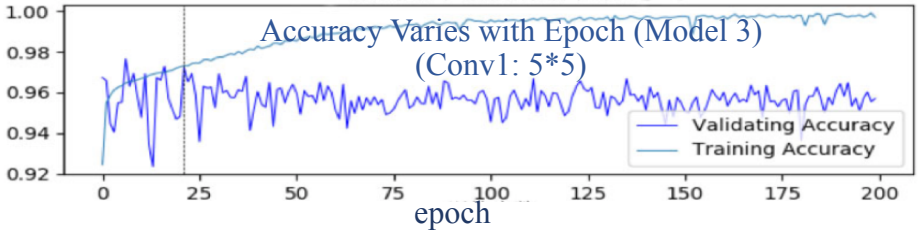
5 Time-Sequential Detection and Prediction

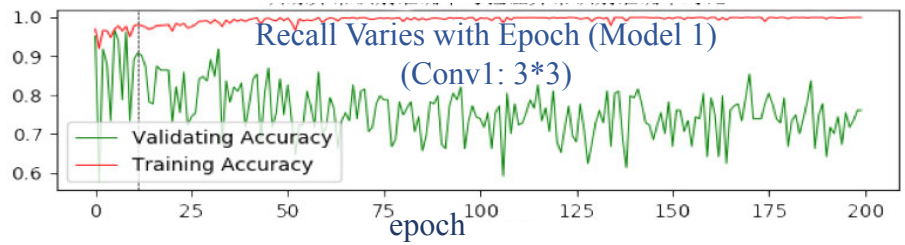
Detection: CNN Classifier

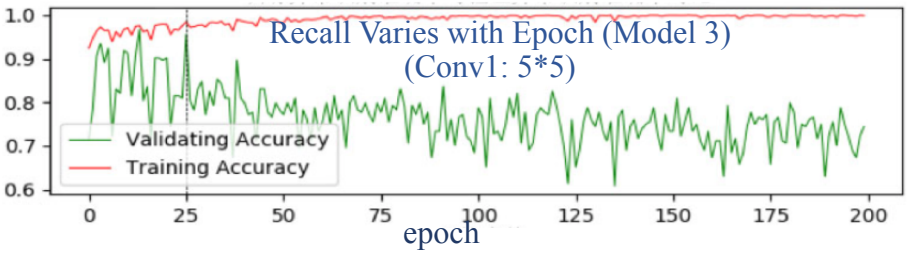
CNN Classifier for Feature-set 1 (Fusion 1) ——Experiments

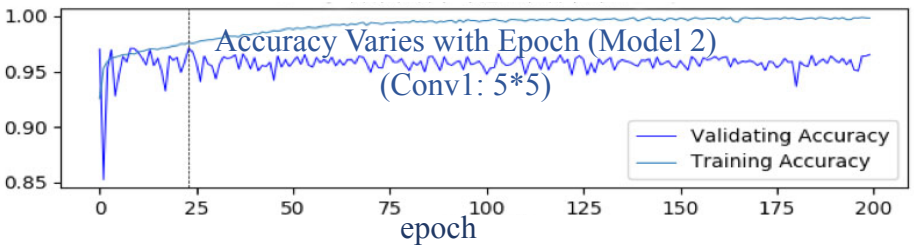
Train 1. Regularization: L2 Norm + Early Stop + Dropout; 2.Threshold: 0.5

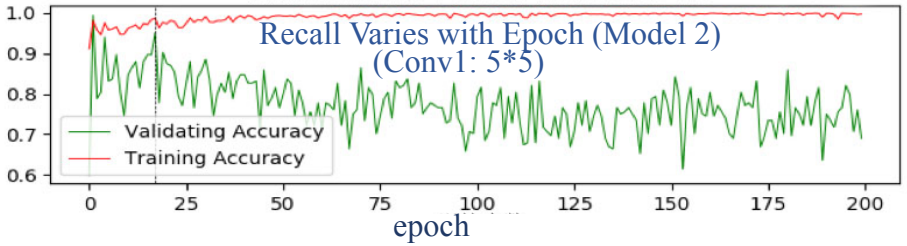












Test

	Conv1 Size	Epoch	Accuracy	Recall	Precision
Model 1	3 × 3	17	94.39%	82.3%	94.11%
		18	92.63%	85.9%	92.36%
	5 × 5	20	94.31%	82.8%	94.04%
		21	94.17%	83.3%	93.91%
Model 2	5 × 5	22	91.65%	88.5%	91.49%
		25	92.94%	85.4%	92.66%
Model 3	5 × 5	16	90.26%	94.3%	90.5%
		26	94.27%	84.9%	94.05%

Note: 1. Recall and Precision for Anomaly; 2. Threshold: 0.5

5 Time-Sequential Detection and Prediction

Detection: CNN Classifier

CNN Classifier for Feature-set 2 (Fusion 2)

Much Weaker than Using Feature-set 1

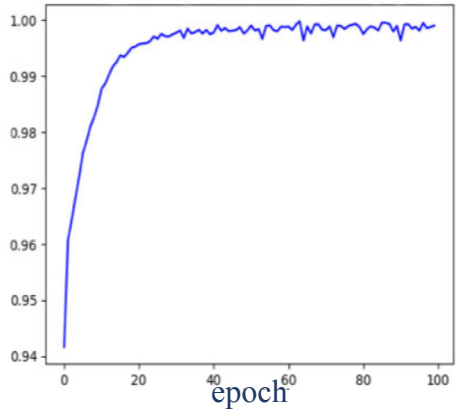
Model

	Conv-Kernel Size	Stride	Padding	Output Channels
Conv-Layer 1	5 × 5	1	2	128
Pool-Layer 1	2 × 2	2	0	--
Conv-Layer 2	5 × 5	1	2	256
Pool-Layer 2	2 × 2	2	0	--
Conv-Layer 3	3 × 3	1	1	512
Pool-Layer 3	2 × 2	2	0	--
FC-Layer 1	1024 units			
FC-Layer 2	512 units			
FC-Layer 3	2 units			

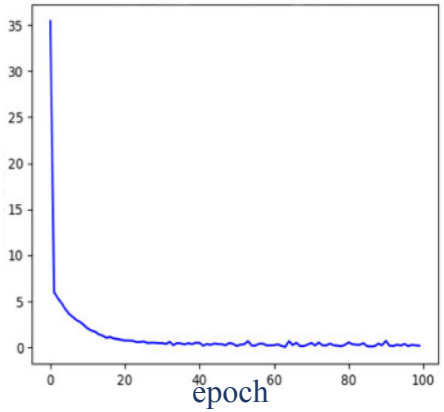
Experiment

1. Regularization in Training: L2 Norm + Dropout
2. Threshold: 0.5

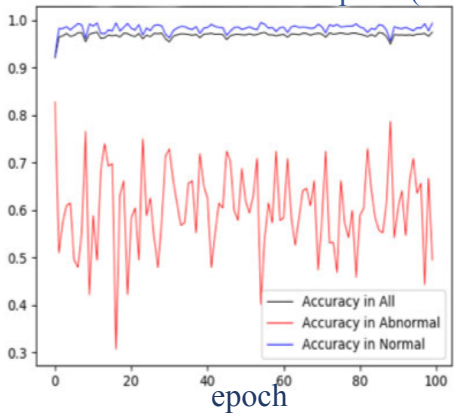
Accuracy Varies with Epoch (Train)



Loss Varies with Epoch (Train)

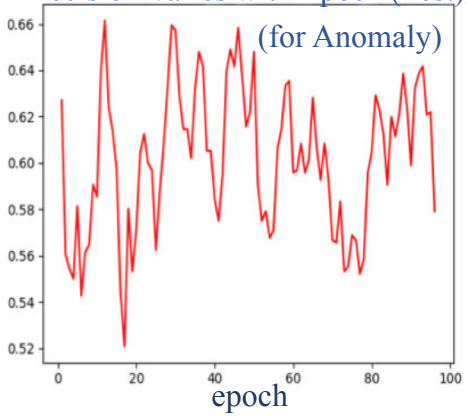


Acc/Recall Varies with Epoch (Test)



Precision Varies with Epoch (Test)

(for Anomaly)



5 Time-Sequential Detection and Prediction

Detection: CNN Classifier

Comparison with other models

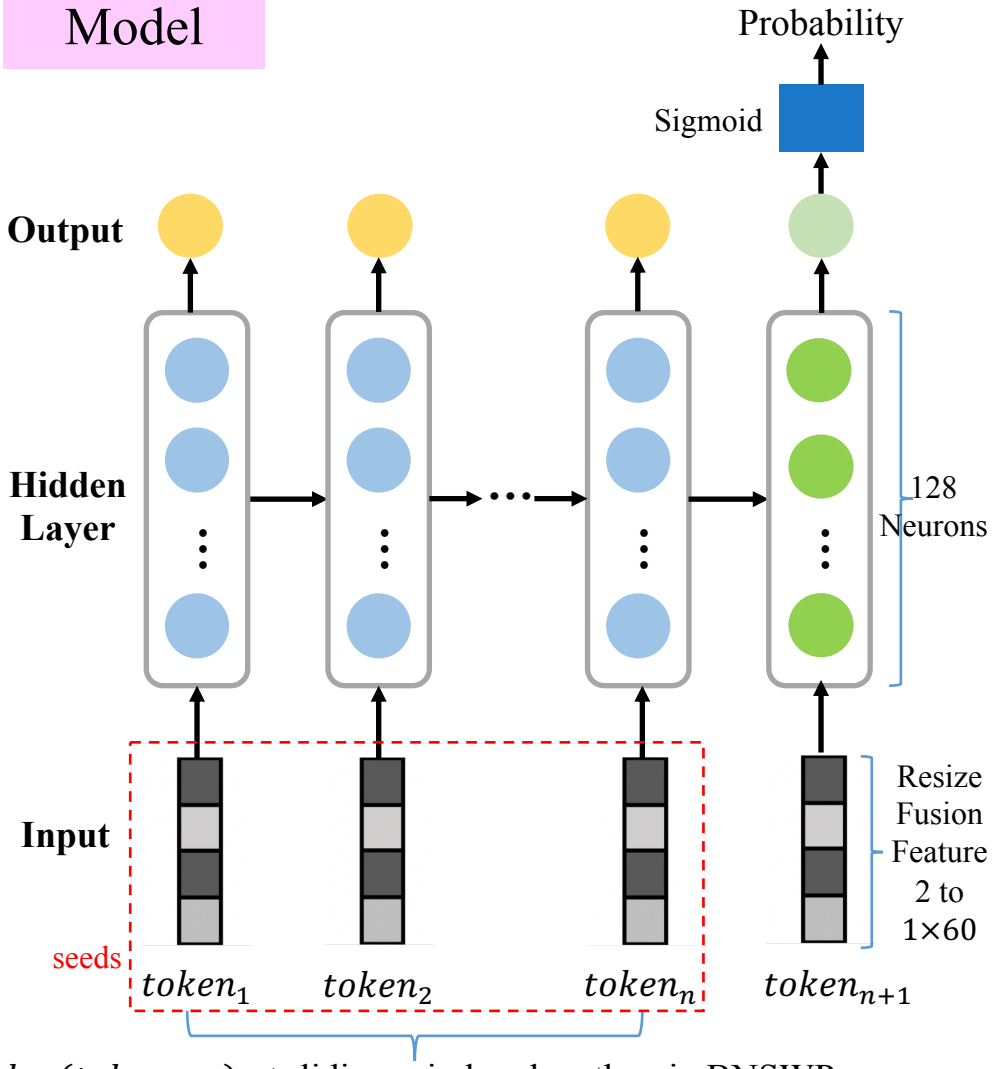
Model	kNN	LR	SVM	CNN
Training Set	Feature–set 2	Feature–set 2	Feature–set 2	Feature–set 1
Test Accuracy	90.28%	93.98%	91.67%	96%～98%
Test Recall (Anomaly)	84.90%	66.67%	70.31%	85%～90%
Test Precision (Anomaly)	89.92%	93.06%	90.35%	90%～95%
Other Instructions		SAGA Optimizer	SVM Kernel: 2nd Order Polynomial Kernel	1. Training epochs about 10 to 30 2. 3 or 4 fully connected layers

CNN Classifier with Feature-set 2 as inputs has a better classification effect comparing with other models

5 Time-Sequential Detection and Prediction

Prediction: RNN Predictor

Model



$len(token_{1\ldots n})$ < sliding-window length m in BNSWP
Usually equals to 72 (frames), equivalent to 1 hour

Experiment

	Training Data	Stride	Test Accuracy	Test Recall	Test Precision
Model 1	1 month data (origin)	1	92.36%	73.2%	64.11%
		3	92.63%	76.9%	62.46%
Model 2	1.5 month data (augmented)	1	91.50%	78.8%	60.49%
		3	91.94%	73.4%	62.67%
Model 3	2 month data (augmented)	1	89.26%	74.3%	61.52%
		3	92.42%	77.9%	64.05%

Note:

1. For training data: *Origin* as original feature-set 2, and *Augmented* as augmented (both for positive and negative samples) feature-set 2
2. $len(seeds) = seg\_len = n = 72$
3. Threshold: 0.5
4. Test Recall and Precision for Anomaly

5 Time-Sequential Detection and Prediction

Combine Classification and Prediction in Online Inference

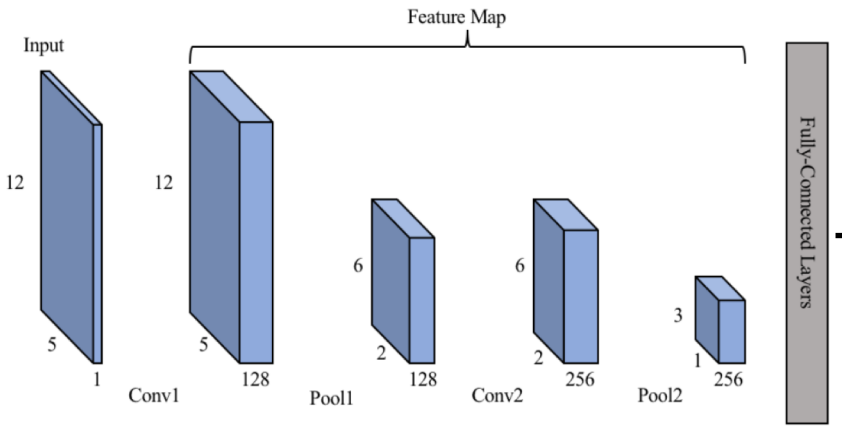
Combining result of CNN classifier and RNN predictor

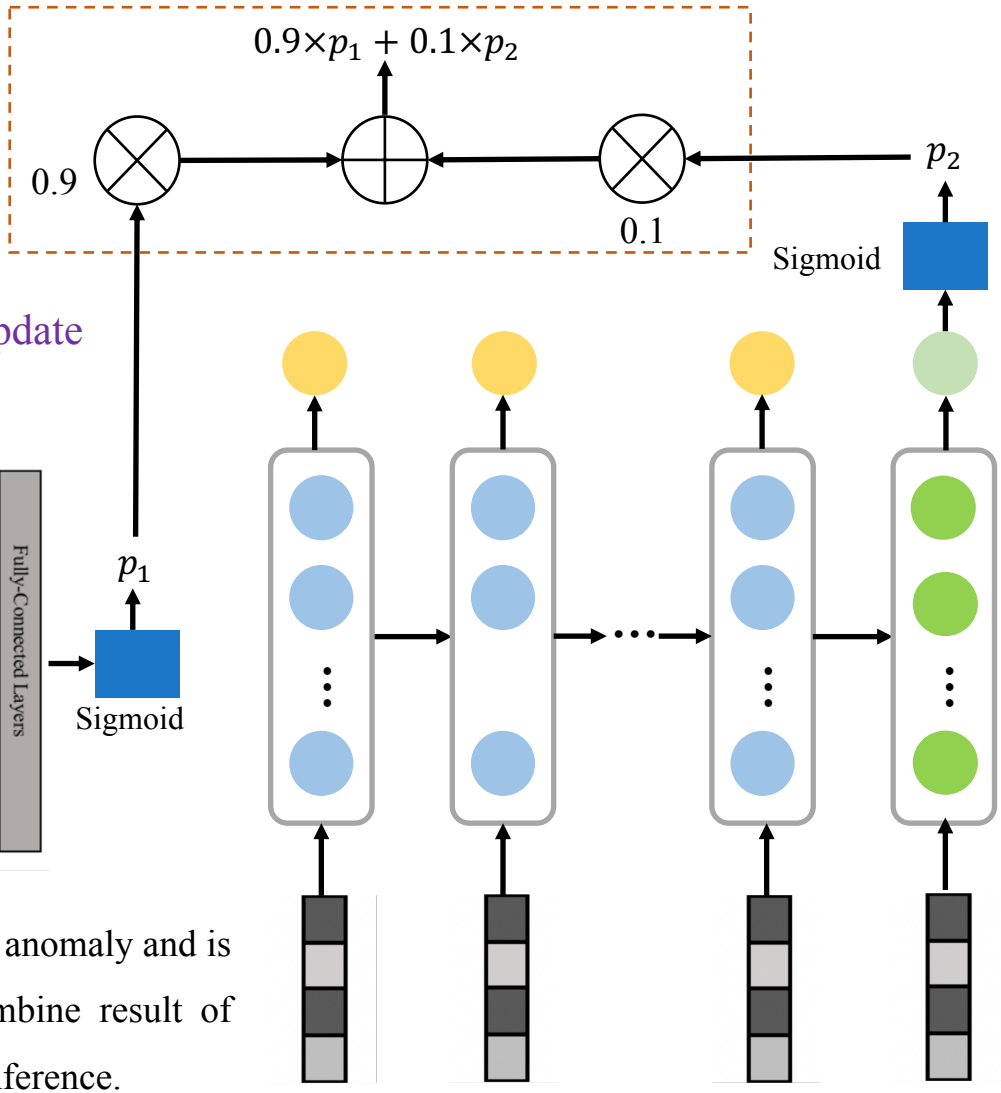
Future Work: Ensemble Learning:

$w1 \times p_1 + w2 \times p_2$

w_1, w_2 to be learned, and joint training to update

$w_1 \cup w_2 \cup \Theta_{CNN} \cup \Theta_{RNN}$





Since RNN model perform well in test recall for anomaly and is suitable for sequential inputs, we decided to combine result of CNN classification and RNN prediction in online inference.

6 Conclusion and Future Work

Conclusion

- Build a benchmark which contains sequential token data for training and evaluating models for wind dancing signal anomaly detection.
- Combine time-domain and frequency-domain feature to 2D-feature so that models such as CNN can be used. Propose Batch Normalization with Sliding-Window Preprocessing (BNSWP) method for normalization.
- Design and train CNN model with 2D fusion feature as inputs for anomaly classification and RNN model for prediction. Experiments show that CNN model performed well both in test recall and precision comparing with other models and RNN performed well in test recall. Combine result of CNN and RNN together when in inference.

Innovation

- BNSWP: Add time-sequential information when normalizing.
- Data Fusion (1D->2D): Combine time-domain and frequency-domain information, build 2D time-sequential feature.
- Combine Classification and Prediction: Use both CNN and RNN model for anomaly detection and prediction, take advantages of both CNN and RNN.

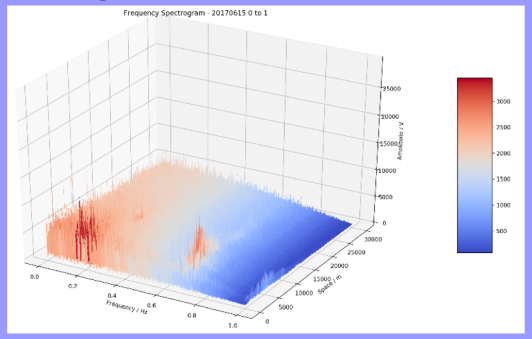
6 Conclusion and Future Work

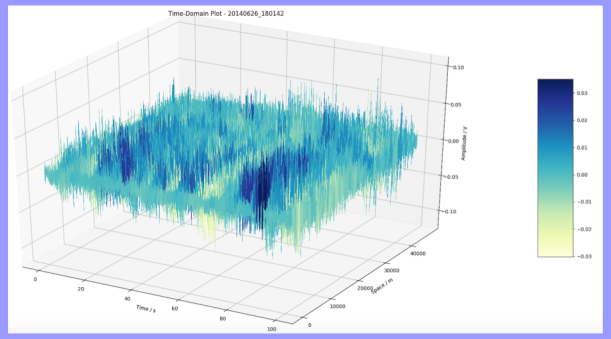
Discussion

- For CNN classifier, using fusion feature 1 get a much better result than using fusion feature 2: We suspect that it introduced more noises than information when applying fusion feature 2 to convolutional situation.

Future Work

- Dealing with problems about noise and sampling, utilize prior/posterior information when building models.
- Try to use continuous time-space/frequency-space data for further more complex modeling.





- Ensemble Learning: Assemble CNN and RNN as a whole model and joint training.
- Consider about clustering algorithms.

Thanks!