Nanomaterial-Based Sensor Array Signal Processing and TB Classification Using Machine Learning

Presented by Chenxi Liu
Supervised by Prof. Israel Cohen
June 1st 2023

Andrew and Erna Viterbi Faculty of Electrical and Computer Engineering,
Technion — Israel Institute of Technology, Haifa, Israel.
Contents

1. Introduction and Background
2. Data Preprocessing
3. Proposed Models
   I. LSTM
   II. CNN
   III. GAF-CNN
   IV. MT-MinCutPool
4. Results and Discussion
5. Other Attempt: TL
6. Conclusion and Future Research
1. Introduction and Background
Introduction and Background

Tuberculosis (TB)

- An ancient, chronic disease worldwide
- Caused by: Bacillus Mycobacterium Tuberculosis
- Threatens ≈ 25% world’s population
- Spread through the air. e.g. cough or sneeze

What Is Tuberculosis (TB)?

Figure 1. Healthy lung (left) and TB infected lung (right).

Solutions?

Yes!

- Biomolecular tests
- Direct microscopy test
- Whole genome sequencing
- Clinical signs and symptoms

Problems:

- Time consumption.
- Poor clinical performance.
- Unsuitable for resource-limited settings.

Figure 2. TB spread from person to person through the air.
Introduction and Background

New way for TB diagnostic

- “Profiles of Volatile Biomarkers Detect Tuberculosis from Skin.”
- A novel TB diagnostic pathway:
  - noninvasive, reliable, rapid

- TB-specific volatile organic compounds (VOCs)
  - Detected and quantified from the skin headspace.
  - Infected samples and healthy samples have different VOC patterns.
  - A set of nanomaterial-based sensor array will translate the collected VOCs into resistance signals.
  - Use ML models for final discrimination between ± TB.

Figure 3. Study schematics.
- a) Skin headspace sampling procedure with polymer.
- b) GC-MS analysis of the collected samples.
- c) Nanomaterial-based sensors array in conjugation machine learning analysis of the collected samples.
- d) A wearable device applied directly on the skin.

“Profiles of Volatile Biomarkers Detect Tuberculosis from Skin.” By Rotem Vishkin, Hossam Haick, etal.
Introduction and Background

Small Dataset Challenges

- Success of neural networks. e.g. Computer Vision (CV), Speech, Natural Language Processing (NLP)
- 2 characteristics:
  1) Large annotated datasets
  2) Deep structure networks with large number of trainable parameters
- **Question: Can these requirements be fulfilled all the time?**

1) **Large datasets:**
   - Not common in medicine, biology, chemical engineering, etc domains.
   - The resource can be limited
   - Obtaining and labeling data are costly
   - Time consumption

2) **Deep structure networks:**
   - Unsuitable for small dataset settings
   - May cause overfitting problems
   - Lead to poor results
Recent Studies on Solving Small Dataset Problems

- **Transfer learning**: source domain: \((\mathcal{D}_S, \mathcal{F}_S)\) → related target domain: \((\mathcal{D}_T, \mathcal{F}_T)\).
  - e.g. CV, NLP tasks.

- **Surrogate data**: generate from random numbers to imitate the original dataset distribution.
  - e.g. time series analysis.

- **Our task**: Diagnosis of Tuberculosis, Machine Learning + Small Dataset Problem.
Introduction and Background

New way for TB diagnostic

- **Goal:** Develop several machine learning based networks to classify the multivariate time series sensor signals and predict their corresponding labels.

- **Key words:** Machine Learning, Small Dataset Problem, TB Classification, Multivariate Time Series Sensor Signals.
2. Data Preprocessing
Data Preprocessing

Dataset Description

- Sample Number: 928 subjects
- Labels: 0 → Active TB samples (467)
  1 → Healthy controls (461)
- Obtained Places: India, South Africa, and Latvia.
- Sensor Number: 40
- Duration: T time steps.

Figure 5. Full signals of one sensor, under the exposure of
(1) vacuum, (2) ambient air and (3) N\textsubscript{2} respectively.

Figure 6. Wearable sensor devices that applies directly to
the skin on the sample’s chest and anterior arm regions.

Figure 4. Schematic illustration of a sensor array.
Data Preprocessing

1. Middle Part Extraction

1. Find last 2 peak points: $P_2$, $P_3$.
2. Move $P_2$, $P_3$ to the left
3. Find the minimum length and intercept each signal into the same length.
   - The inputs have same feature dimension.

Figure 5. Full signals of one sensor, under the exposure of (1) vacuum, (2) ambient air and (3) $N_2$ respectively.

Figure 7. The change in the sensor resistivity (i.e. $\Delta R_{end}/R_0$) for different storage conditions at the starting point (M0) and after 9 months (M9).
Figure 8. Original 40 sensor signals correspond to 1 sample before middle part extraction.

Figure 9. 40 processed sensor signals after middle part extraction.
Data Preprocessing

2. Normalization and Calibration

- Why need normalization?
  - Different sampling times & external environments (pressure, temperature, humidity).
  - Various starting points & resistance ranges

- Why need calibration?
  - Supervise the sensor’s functionality
  - Overcome possible sensor response drift

- Median Normalization:
  - \( X' = \frac{X}{\text{med}(X_{[0,30]})} \)
  - \( \hat{X} = X' - \min(X') \in [0,1] \)

- Calibration:
  - \( \bar{X} = \hat{X} \cdot \frac{\text{Peak} \hat{X}}{\text{Peak}_{\text{sensor}}} \)

Figure 9. 40 processed sensor signals after middle part extraction.
2. Normalization and Calibration

- **Why need normalization?**
  - Different sampling times & external environments (pressure, temperature, humidity).
  - Various starting points & resistance ranges

- **Why need calibration?**
  - Supervise the sensor’s functionality
  - Overcome possible sensor response drift

- **Median Normalization:**
  - \( X' = \frac{X}{\text{med}(X_{[0,30]})} \)
  - \( \hat{X} = X' - \min(X') \in [0,1] \)

- **Calibration:**
  - \( \tilde{X} = \hat{X} \cdot \frac{\text{Peak} \hat{C}}{\text{Peak} \hat{X}} \)

Figure 10. 40 processed sensor signals after normalization and calibration.
3. Good sensors selection

Figure 11. Pearson correlation coefficient matrix heat maps for both (a) good sensors and (b) unstable sensors.

\[ C_{xy} = \frac{\sum_{i=1}^{T} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{T} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{T} (y_i - \bar{y})^2}} = \frac{\text{cov}(x, y)}{\sqrt{\text{var}(x) \cdot \text{var}(y)}} \]

- Unstable sensors → lower similarity coefficient
- Reliable sensors → higher similarity coefficient

Figure 10. 40 processed sensor signals after normalization and calibration.
Data Preprocessing

3. Good sensors selection

**Group A**: 467 Active TB Samples, **Group B**: 461 Healthy Controls

- Select 150 Random Samples / Group
- Compute 150 Pearson Correlation Coefficient / Sensor
- Compute Mean Coefficient Value / Sensor
- Set $\theta = 0.65$

Mean Value $> \theta \rightarrow$ Good Sensors (29)
Mean Value $< \theta \rightarrow$ Unstable Sensors (11)

Figure 11. 29 processed sensor signals after good sensor selection.
3. Proposed Models
1. Long Short Term Memory (LSTM)

- LSTM networks are a type of recurrent neural network (RNN).

- **LSTM Network Composition:**
  - Memory vector $m$
  - Hidden state vector $h$
  - Input $x$

- **Benefits of using LSTM:**
  1. Reduce vanishing gradient problem
  2. Capability to handle sequences of different lengths
  3. Ability to learn from past experiences
  4. Effective at detecting complex patterns

- **Common tasks to employ LSTM:**
  - Natural Language Processing
  - Speech recognition
  - Time series analysis

---

![Image of LSTM network](image)

Figure 12. The structure of LSTM network.
1. Long Short Term Memory (LSTM)

Figure 12. (a) ReLU and (b) Sigmoid activation function.

- Loss Function: Binary Cross Entropy (BCE)

\[ BCE = - \frac{1}{N} \sum_{i=1}^{N} [y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))] \]

Figure 13. Proposed LSTM model architecture.

- Input: \((X, Y), X \in \mathbb{R}^{N \times n \times T}, Y \in \mathbb{R}^{N}\)
  - \(N\): number of samples
  - \(n\): number of sensors \(\rightarrow 29\)
  - \(T\): time steps \(\rightarrow 147\)
2. Convolution Neural Network (CNN)

- CNN is the most widely used structure for time series classification tasks: robustness, less training time.

- **Benefits of using CNN for MTSC:**
  1. Effective in learning complex temporal patterns.
  2. Robust to variability
  4. Reduce overfitting: Pooling operation

- 1D CNN: Convolutional and Pooling layers
- Loss function: BCE loss.
- Input: \((X, Y)\), where \(X \in \mathbb{R}^{N \times n \times T}\), \(Y \in \mathbb{R}^N\).

Figure 14. Proposed CNN model architecture.
3.1 Gramian Angular Field (GAF)

What is GAF?

- Encode univariate time series → 2D images.
  1. Represent time series in a polar coordinate system.
  2. Form the Gramian matrix and visualize it as 2D images.
  3. Use CNNs to identify visual patterns for prediction.

Why GAF?

- Easier to visualize and analyze the data.
- Capture cyclical patterns and correlations present in the original time series data.
- Take advantage of DL in computer vision.
- Use in a wide range of time series data.

![Diagram](image.png)

Figure 15. Known GAF values at different time steps.

- Financial data,
- Environmental data
- Speech data
- Electrocardiogram (ECG)
- Etc

Explanation in “reference” by Zhiguang Wang and Tim Oates.
3.1 Gramian Angular Field (GAF)

1. Convert \( \hat{X} \) into polar coordinates:
   - \( \phi_i = \arccos \hat{x}_i, \quad -1 \leq \hat{x}_i \leq 1, \quad \hat{x}_i \in \hat{X} \)
   - \( r_i = \frac{t_i}{N}, \quad t_i \in \mathbb{N} \)

2. The GAF method defines its own “special” inner product as:
   - \( \langle \hat{x}_1, \hat{x}_2 \rangle = \cos(\phi_1 + \phi_2) \)
     \[= \hat{x}_1 \cdot \hat{x}_2 - \sqrt{1 - \hat{x}_1^2} \cdot \sqrt{1 - \hat{x}_2^2} \]
   - \( G = \begin{pmatrix}
             \cos(\phi_1 + \phi_1) & \cos(\phi_1 + \phi_2) & \cdots & \cos(\phi_1 + \phi_n) \\
             \cos(\phi_2 + \phi_1) & \cos(\phi_2 + \phi_2) & \cdots & \cos(\phi_2 + \phi_n) \\
             \vdots & \vdots & \ddots & \vdots \\
             \cos(\phi_n + \phi_1) & \cos(\phi_n + \phi_2) & \cdots & \cos(\phi_n + \phi_n)
           \end{pmatrix} \)

   \[= \hat{X}' \cdot \hat{X} - \sqrt{I - \hat{X}'^2} \cdot \sqrt{I - \hat{X}^2} \]

Figure 16. Gramian Angular Field (GAF) Transition.

- Assume we have a time series: \( X = \{x_1, x_2, \cdots, x_N\} \).
- \( \hat{x}_i = \frac{(x_i - \text{max}(X)) + (x_i - \text{min}(X))}{\text{max}(X) - \text{min}(X)} \)
- The normalized values:
  \( \hat{X} = \{\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_N\} \in [-1, 1] \).

3.2 Gramian Angular Field-CNN (GAF-CNN)

- \( X \in \mathbb{R}^{N \times n \times T} \rightarrow X_{GAF} \in \mathbb{R}^{N \times T \times T \times n} \) : (batch size, height, width, channels).
- Difference: 2D convolutional layers and 2D max pooling layers.

Figure 16. GAF-CNN model architecture.
4. MT-MinCutPool

1) Graph representation: Use graph Laplacian matrix.
2) Graph coarsen: Use MinCutPool to cluster similar nodes into one cluster.

- In paper: the authors proposed a novel graph pooling based framework “Multivariate Time Series Classification with Variational Graph Pooling (MTPool)”.
  - Obtain an expressive global representation of MTS.
  - Adopt graph neural network (GNN) to solve MTSC problem.
- We propose MT-MinCutPool model, a modified MTPool.
- 2 Main differences.

Figure 17. MT-MinCutPool model architecture.
Part 1. Graph Structure Learning

- Euclidean distance: prone to sensitivity and distortion in time axis.
- Dynamic Time Warping (DTW) distance:
  - Effective distance measurement for time series problems.
  - Non-linear alignment of similar time series that are locally out of phase.
- For subsequent computing:
  1) Construct distance matrix $DTW \in \mathbb{R}^{N \times n \times n}$
  2) Set a threshold $\theta$

Figure 18. An illustration between Euclidean matching and DTW matching from paper.
4. MT-MinCutPool

Graph Structure Learning

1. Build Adjacency Matrix $A$:
   
   \[ A_{ij} = \begin{cases} 
   1, & \text{if } \text{DTW}[i, j] < \theta, \\
   0, & \text{if } \text{DTW}[i, j] > \theta. 
\end{cases} \]

2. Build Degree Matrix $D$:
   
   - the row sum of $A$

3. Build Laplacian Matrix $L$:
   
   \[ L = D - A \]
1. Build Adjacency Matrix $A$:
   $$A_{ij} = \begin{cases} 
   1, & \text{if } DTW[i, j] < \theta, \\
   0, & \text{if } DTW[i, j] > \theta.
   \end{cases}$$

2. Build Degree Matrix $D$:
   - the row sum of $A$

3. Build Laplacian Matrix $L$:
   $$L = D - A$$
4. MT-MinCutPool

Part 1. Graph Structure Learning

Figure 18. MT-MinCutPool model graph structure.

Algorithm 3.1 Build Laplacian Adjacency matrix.

Input: $X \in \mathbb{R}^{N \times n \times T}$, $\theta$
Output: $L \in \mathbb{R}^{N \times n \times n}$

1: (1) Build Distance Matrix DTW
2: $DTW \leftarrow$ empty matrix with shape $\mathbb{R}^{N \times n \times T}$
3: for $i = 1$ to $N$ do
4: $x \leftarrow X[i]$
5: distance $\leftarrow$ empty matrix with shape $\mathbb{R}^{n \times n}$
6: for $j = 1$ to $n$ do
7: for $k = 1$ to $n$ do
8: $distance[j, k] \leftarrow$ dtw$(x[j], x[k])$ \{dtw$(\cdot)$ is the dynamic time warping distance function\}
9: end for
10: end for
11: (2) Build Degree Matrix $D$
12: $D \leftarrow$ empty array with shape $\mathbb{R}^{N \times n \times T}$
13: $A \leftarrow \text{int}(DTW < \theta)$
14: for $i = 1$ to $N$ do
15: $adj \leftarrow A[i]$
16: $D[i] \leftarrow \text{diagonal}(\sum_{k=1}^{n} adj[k,:])$ \{diagonal$(\cdot)$ is the diagonalized function\}
17: end for
18: (3) Build Laplacian Matrix $L$
19: $L = D - A$
4. MT-MinCutPool

Variable interaction

Time

Multivariate Time Series

Temporal Convolution

Graph Structure Learning

Part 1: Graph Structure Learning

Part 3: Spatial-Temporal Modeling

Graph Neural Network

Part 2: Temporal Feature Extraction

Part 4: MinCutPool Layer

Part 5: Graph-Level Embedding Classification

Output labels

Fully Connected Layer

XTC
Part 2: \( X_{TC} = |\bigoplus_{i=1}^{m} f_i| = |\bigoplus_{i=1}^{m} \sigma(W_i \ast X + b) \)
- \( |\bigoplus_{i=1}^{m} \) concatenation operation.
- \( f_i \) output feature maps of each convolution layer.

Part 3: \( \tilde{X} = G(A, X_{TC}, W, b) = \sigma(A \ast X_{TC} \ast W + b) \)
- \( G(\cdot) \) graph convolution function.
4. MT-MinCutPool
4. MT-MinCutPool

Part 4. MinCutPool Layer + Part 5. Graph Level Embedding Classification

Part 4: Graph coarsen
1) Cluster assignment matrix \( S \)
   - \( S = MLP(\tilde{X}, W_{MLP}) \)

2) Pooling by MinCutPool:
   - Normalized adjacency matrix:
     \[ \tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \in \mathbb{R}^{N \times n \times n} \]
   - Coarsened adjacency matrix:
     \[ A_{pool} = S^T \tilde{A} S \]
   - Coarsened feature matrix:
     \[ X_{pool} = S^T X \]
# Model Summary

<table>
<thead>
<tr>
<th>Layer</th>
<th>Stride</th>
<th>Activation</th>
<th>Kernel Size</th>
<th>Input Shape</th>
<th>Output Shape</th>
<th>Parameter Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv1D</td>
<td>1</td>
<td>ReLU</td>
<td>20</td>
<td>(bs, 147, 29)</td>
<td>(bs, 128, 16)</td>
<td>9296</td>
</tr>
<tr>
<td>MaxPooling1D</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>(bs, 128, 16)</td>
<td>(bs, 64, 16)</td>
<td>0</td>
</tr>
<tr>
<td>Conv1D</td>
<td>2</td>
<td>ReLU</td>
<td>2</td>
<td>(bs, 64, 16)</td>
<td>(bs, 32, 4)</td>
<td>2116</td>
</tr>
<tr>
<td>MaxPooling1D</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>(bs, 32, 4)</td>
<td>(bs, 16, 4)</td>
<td>0</td>
</tr>
<tr>
<td>Flatten</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>(bs, 16, 4)</td>
<td>(bs, 64)</td>
<td>0</td>
</tr>
<tr>
<td>Dense</td>
<td>-</td>
<td>ReLU</td>
<td>-</td>
<td>(bs, 64)</td>
<td>(bs, 16)</td>
<td>1040</td>
</tr>
<tr>
<td>Dense</td>
<td>-</td>
<td>Sigmoid</td>
<td>-</td>
<td>(bs, 16)</td>
<td>(bs, 1)</td>
<td>17</td>
</tr>
</tbody>
</table>

Table 2. CNN model layers and parameters.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Output Shape</th>
<th>Activation</th>
<th>Parameter Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>(Batch size, 32)</td>
<td>-</td>
<td>7936</td>
</tr>
<tr>
<td>Dropout</td>
<td>(Batch size, 32)</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>Dense</td>
<td>(Batch size, 16)</td>
<td>ReLU</td>
<td>528</td>
</tr>
<tr>
<td>Dense</td>
<td>(Batch size, 1)</td>
<td>Sigmoid</td>
<td>17</td>
</tr>
</tbody>
</table>

Table 1. LSTM model layers and parameters.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Stride</th>
<th>Activation</th>
<th>Kernel Size</th>
<th>Input Shape</th>
<th>Output Shape</th>
<th>Parameter Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv2D</td>
<td>1</td>
<td>ReLU</td>
<td>(5, 5)</td>
<td>(bs, 147, 147, 29)</td>
<td>(bs, 143, 143, 12)</td>
<td>8712</td>
</tr>
<tr>
<td>MaxPooling2D</td>
<td>2</td>
<td>-</td>
<td>(2, 2)</td>
<td>(bs, 143, 143, 12)</td>
<td>(bs, 71, 71, 12)</td>
<td>0</td>
</tr>
<tr>
<td>Conv2D</td>
<td>1</td>
<td>ReLU</td>
<td>(5, 5)</td>
<td>(bs, 71, 71, 12)</td>
<td>(bs, 67, 67, 6)</td>
<td>1806</td>
</tr>
<tr>
<td>MaxPooling2D</td>
<td>2</td>
<td>-</td>
<td>(2, 2)</td>
<td>(bs, 67, 67, 6)</td>
<td>(bs, 33, 33, 6)</td>
<td>0</td>
</tr>
<tr>
<td>Flatten</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>(bs, 33, 33, 6)</td>
<td>(bs, 6534)</td>
<td>0</td>
</tr>
<tr>
<td>Dense</td>
<td>-</td>
<td>Sigmoid</td>
<td>-</td>
<td>(bs, 6534)</td>
<td>(bs, 1)</td>
<td>6535</td>
</tr>
</tbody>
</table>

Table 3. GAF-CNN model layers and parameters.

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameter Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>8481</td>
</tr>
<tr>
<td>CNN</td>
<td>12469</td>
</tr>
<tr>
<td>GAF-CNN</td>
<td>17053</td>
</tr>
</tbody>
</table>

Table 4. Total number of parameters of the first 3 proposed models.
Model Summary

Common Characteristics:
1. Limited number of parameters.
2. Lightweight models with limited layers.
3. Employed regularization techniques:
   • LSTM: Dropout layer
   • CNN, GAF-CNN: Max pooling layer
   • MT-MinCutPool: MinCutPool pooling layer

Technique Benefits:
▶ Improve generalization.
▶ Make the model to learn more robust features.
▶ Reduce spatial size of the feature map.
▶ Reduce the number of model parameters.
▶ Avoid overfitting problems.
4. Results and Discussion
Experiment Setup

Model Training

- TB samples:
  - 80% for training and validation
  - 20% for testing
- 5 fold stratified cross-validation
- 10 runs for each model
- Results:
  1) Choose the best model with the highest validation accuracy
  2) Evaluate on the test set / run
  3) Compute mean test accuracy from 10 runs as the final result.

Evaluation Metrics

- \( \text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \)
- \( \text{Sensitivity} = \frac{TP}{TP + FN} \)
- \( \text{Specificity} = \frac{TN}{TN + FP} \)
- Area Under the ROC Curve (AUC)

- \( TP \): True Positive
- \( TN \): True Negative
- \( FP \): False Positive
- \( FN \): False Negative
Results and Discussion

Table 5. Results for the proposed models.

<table>
<thead>
<tr>
<th></th>
<th>LSTM</th>
<th>CNN</th>
<th>GAF-CNN</th>
<th>MT-MinCutPool</th>
<th>Model Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accuracy</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>0.646</td>
<td>0.916</td>
<td>0.659</td>
<td>0.688</td>
<td>--</td>
</tr>
<tr>
<td>valid</td>
<td>0.69</td>
<td>0.687</td>
<td>0.692</td>
<td>0.69</td>
<td>--</td>
</tr>
<tr>
<td>test</td>
<td>0.611</td>
<td>0.606</td>
<td><strong>0.639</strong></td>
<td>0.604</td>
<td>0.615</td>
</tr>
<tr>
<td><strong>Sensitivity</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>0.618</td>
<td>0.931</td>
<td>0.766</td>
<td>0.76</td>
<td>--</td>
</tr>
<tr>
<td>valid</td>
<td>0.675</td>
<td>0.715</td>
<td>0.78</td>
<td>0.756</td>
<td>--</td>
</tr>
<tr>
<td>test</td>
<td>0.631</td>
<td>0.694</td>
<td><strong>0.777</strong></td>
<td>0.728</td>
<td>0.71</td>
</tr>
<tr>
<td><strong>Specificity</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>0.676</td>
<td>0.901</td>
<td>0.544</td>
<td>0.612</td>
<td>--</td>
</tr>
<tr>
<td>valid</td>
<td>0.704</td>
<td>0.658</td>
<td>0.597</td>
<td>0.619</td>
<td>--</td>
</tr>
<tr>
<td>test</td>
<td><strong>0.594</strong></td>
<td>0.533</td>
<td>0.532</td>
<td>0.5</td>
<td>0.54</td>
</tr>
<tr>
<td><strong>AUC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>test</td>
<td>0.634</td>
<td>0.657</td>
<td><strong>0.692</strong></td>
<td>0.661</td>
<td>0.661</td>
</tr>
</tbody>
</table>

Table 6. Results for the baseline models.

<table>
<thead>
<tr>
<th></th>
<th>MTPool</th>
<th>GAF-Attention</th>
<th>MLSTM FCN</th>
<th>Model Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accuracy</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>0.563</td>
<td>0.755</td>
<td>0.733</td>
<td>--</td>
</tr>
<tr>
<td>valid</td>
<td>0.577</td>
<td>0.663</td>
<td>0.693</td>
<td>--</td>
</tr>
<tr>
<td>test</td>
<td>0.517</td>
<td><strong>0.631</strong></td>
<td>0.586</td>
<td>0.578</td>
</tr>
<tr>
<td><strong>Sensitivity</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>0.682</td>
<td>0.78</td>
<td>0.757</td>
<td>--</td>
</tr>
<tr>
<td>valid</td>
<td>0.683</td>
<td>0.688</td>
<td>0.733</td>
<td>--</td>
</tr>
<tr>
<td>test</td>
<td>0.655</td>
<td>0.622</td>
<td><strong>0.664</strong></td>
<td>0.647</td>
</tr>
<tr>
<td><strong>Specificity</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>0.436</td>
<td>0.728</td>
<td>0.709</td>
<td>--</td>
</tr>
<tr>
<td>valid</td>
<td>0.464</td>
<td>0.636</td>
<td>0.651</td>
<td>--</td>
</tr>
<tr>
<td>test</td>
<td>0.4</td>
<td><strong>0.637</strong></td>
<td>0.521</td>
<td>0.519</td>
</tr>
<tr>
<td><strong>AUC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>test</td>
<td>0.538</td>
<td><strong>0.695</strong></td>
<td>0.648</td>
<td>0.627</td>
</tr>
</tbody>
</table>

- The mean performance of the proposed models > baseline models
- **For proposed models:**
  - 🏆 GAF-CNN: Highest Accuracy, Sensitivity, AUC
  - 🍀 LSTM: Highest Specificity
- **For baseline models:**
  - 🏆 GAF-Attention: Highest Accuracy, Specificity, AUC
  - 🍀 MLSTM-FCN: Highest Sensitivity
- **For all models:**
<table>
<thead>
<tr>
<th></th>
<th>0.639</th>
<th>GAF-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensitivity</td>
<td>0.777</td>
<td>GAF-CNN</td>
</tr>
<tr>
<td>Specificity</td>
<td>0.637</td>
<td>GAF-Attention</td>
</tr>
<tr>
<td>AUC</td>
<td>0.695</td>
<td>GAF-Attention</td>
</tr>
</tbody>
</table>
Results and Discussion

<table>
<thead>
<tr>
<th></th>
<th>MT-MinCutPool</th>
<th>MTPool</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accuracy</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>0.688</td>
<td>0.563</td>
</tr>
<tr>
<td>valid</td>
<td>0.69</td>
<td>0.577</td>
</tr>
<tr>
<td>test</td>
<td><strong>0.604</strong></td>
<td><strong>0.517</strong></td>
</tr>
<tr>
<td><strong>Sensitivity</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>0.76</td>
<td>0.682</td>
</tr>
<tr>
<td>valid</td>
<td>0.756</td>
<td>0.683</td>
</tr>
<tr>
<td>test</td>
<td><strong>0.728</strong></td>
<td><strong>0.655</strong></td>
</tr>
<tr>
<td><strong>Specificity</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>0.612</td>
<td>0.436</td>
</tr>
<tr>
<td>valid</td>
<td>0.619</td>
<td>0.464</td>
</tr>
<tr>
<td>test</td>
<td><strong>0.5</strong></td>
<td><strong>0.4</strong></td>
</tr>
<tr>
<td><strong>AUC</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>test</td>
<td><strong>0.661</strong></td>
<td><strong>0.538</strong></td>
</tr>
</tbody>
</table>

Table 7. Results for MT-MinCutPool and MTPool.

<table>
<thead>
<tr>
<th></th>
<th>MT-MinCutPool</th>
<th>MTPool</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graph Structure</strong></td>
<td>Laplacian matrix</td>
<td>Correlation Coefficient matrix</td>
</tr>
<tr>
<td><strong>Graph Pooling</strong></td>
<td>MinCutPool</td>
<td>Adaptive Pooling</td>
</tr>
</tbody>
</table>

Table 8. Differences for MT-MinCutPool and MTPool.

- The performance of the MT-MinCutPool > MTPool for all metrics.
  1. **Laplacian matrix**: Contain more graph information.
  2. **MinCutPool**:
    - Directly identify strongly connected nodes, aggregate into one cluster.
    - Preserve important graph information during pooling process.
Results and Discussion

Figure 18. ROC curves for each model.

- GAF-Attention: 0.695
- GAF-CNN: 0.692
- MT-MinCutPool: 0.661
- CNN: 0.657
- MLSTM-FCN: 0.648
- LSTM: 0.634
- MTPool: 0.538

Figure 20. Accuracy and loss curves for GAF-CNN

Figure 21. Accuracy and loss curves for CNN

Figure 19. Accuracy and loss curves for LSTM model.

Figure 22. Accuracy and loss curves for MT-MinCutPool
Results and Discussion

Figure 23. Confusion matrix for each model.
5. Other Attempt: Transfer Learning
Transfer Learning

Figure 24. An example of an extracted ECG beat.

- Source data: ECG signals (PTB datasets)
- Source task: ECG heartbeat classification

Source and target data similarities:
- Time series data
- Binary classification tasks
- Normalized value ranges: (0, 1)

Figure 25. Different ECG signals.

Source and target data differences:
- Source: Univariate; Target: Multivariate
- Signal characteristics
- Number of source data (14552) >> Number of target data (928)

Reference: ECG Heartbeat Classification: A Deep Transferable Representation
Transfer Learning

- Train source model on source dataset.
- Freeze the first few layers of the pre-trained model.
- Train the pre-trained model on target dataset, update the last few layers.

**Freeze** the trainable parameters and weights from pre-trained model.

**Update** trainable parameters of new layers.
Results for Transfer Learning

<table>
<thead>
<tr>
<th></th>
<th>TL target</th>
<th>CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accuracy</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train</td>
<td>0.528</td>
<td>0.916</td>
</tr>
<tr>
<td>Valid</td>
<td>0.582</td>
<td>0.687</td>
</tr>
<tr>
<td>Test</td>
<td>0.514</td>
<td>0.606</td>
</tr>
<tr>
<td><strong>Sensitivity</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train</td>
<td>0.75</td>
<td>0.931</td>
</tr>
<tr>
<td>Valid</td>
<td>0.817</td>
<td>0.715</td>
</tr>
<tr>
<td>Test</td>
<td>0.804</td>
<td>0.694</td>
</tr>
<tr>
<td><strong>Specificity</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train</td>
<td>0.293</td>
<td>0.901</td>
</tr>
<tr>
<td>Valid</td>
<td>0.333</td>
<td>0.658</td>
</tr>
<tr>
<td>Test</td>
<td>0.27</td>
<td>0.533</td>
</tr>
<tr>
<td><strong>AUC</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>0.539</td>
<td>0.657</td>
</tr>
</tbody>
</table>

Table 9. Evaluation metrics for TL target and CNN.

- High sensitivity  = \( \frac{TP}{TP + FN} \)
- Low accuracy, specificity and AUC score
- **Domain shift**

Figure 26. Accuracy and loss curves of source data (left), and target data (right).
6. Conclusion and Future Research
Conclusion and Future Research

Summary

1. Data Preprocessing
   - Middle Part Extraction
   - Data Normalization
   - Data Calibration
   - Good Sensor Selection

2.1 Proposed ML Models
   - LSTM
   - CNN
   - GAF-CNN
   - MT-MinCutPool

2.2 Baseline ML Models
   - MLSTM-FCN
   - GAF-Attention
   - MTPool

3. Model Training
   - 20% test
   - 80% train+validation
   - 5 fold cross validation
   - 10 runs

4. Model Evaluation
   - Accuracy
   - Sensitivity
   - Specificity
   - ROC-AUC

5. Result Discussion
Conclusion and Future Research

Conclusion

- Proposed models exhibit overall better performance than baseline models.
  - MT-MinCutPool outperformed MTPool in all aspects.
  - GAF-CNN has the highest accuracy and sensitivity among all models.
- Lightweight models performed better than complex models for small dataset problems.

Limitations

- The focus on a single TB dataset
- The small size of the TB dataset
Conclusion and Future Research

Future Research

- Explore the efficacy of the proposed models on other types of TB datasets / similar MTSC problems/ larger datasets.
- Explore the potential of transfer learning (TL) if a more extensive and similar source dataset could be found.
- Explore surrogate data approaches, such as using Variational Auto Encoder (VAE).
- Explore the interpretability of the proposed models in the context of TB diagnosis.
Questions?
Thank You! 😊