Self-Organizing Cellular Radio Access Network with Deep Learning

Wuyang Zhang\textsuperscript{1}, Russell Ford\textsuperscript{2}, Joonyoung Cho\textsuperscript{2}, Charlie Jianzhong Zhang\textsuperscript{2}, Yanyong Zhang\textsuperscript{1,3}, Dipankar Raychaudhuri\textsuperscript{1}

\textsuperscript{1} WINLAB, Rutgers University
\textsuperscript{2} Samsung Research America
\textsuperscript{3} University of Science and Technology of China
Problem Statement

• RAN Performance Problems Prevalent
  • *My phone shows 5 signal bars but the connection is so slow!*
  • *Cannot hear your voice!*
  • *This web page is not loading at all!*

• It is not straightforward to **diagnose the root cause** and **solve** the problems.

• Example root causes of RAN performance problems

  - Excessive uptilt/downtilt (EU/DU)
  - Coverage hole (CH)
  - Too late handover (TLHO)
  - Inter-cell Interference (II)
  - Excessive Cell Power Reduction (ERP)
  - Cell Overload (CO)

• Network Operators must diagnose RAN performance problems quickly!
Self-Healing Radio Access Network

Can cellular network operators automate the diagnosis and self-healing of RAN?

- **System challenge:**
  - How to **predict anomaly KPIs** before any faults really appear?
  - How to **figure out root causes** based on thousands of cell KPIs?
  - How can the system **self recover** from the faults?
  - How to deal with **~ TB level** data of cell KPIs?
System Overview: SORA

Real-time cell KPI Monitoring

- SINR DL
- Cell Thru DL
- SumHaX2Out

Time slot

S1

Anomaly KPI Prediction

- Anomaly points
- Surprise > thresh?

Time slot

S2

Close Loop

Self Healing

Agent
- State
- DNN
- policy parameter

Reward
- Take action

Environment
- Observe state

S4

Root Cause Analysis

- Hole area
- Excessive util/downtil (EU/DU)
- Coverage hole (CH)
- Too late handover (TLHO)

S3

Big Data Platform (Apache Spark + HDFS + Apache HBase)
Real-world KPI Dataset Overview

real-world data from a top-tier US cellular operator

- Aggregated Cell Dataset
  - KPIs & error code summary: ~100. e.g., mobile subscriber count, attach count, detach count, handover count; x2_attempt, x2_enb_to, x2_dns_fail, s1_intra_src_attempt, s1_intra_tar_sgw_chg, etc.
  - Overall size: ~335 GB
  - Collection date: 2017-06-30 – 2018-03-20
  - Collection interval: 1 hour

- Cell dataset
  - KPIs summary: ~4k.
  - Overall size: ~ 100 TB
  - Collection date: 2018-02-01 – 2018-07-31
  - Collection interval: 15 minutes

- Example KPIs in time series

- Partial example KPIs
**Anomaly Prediction: Objective & System Challenges**

**Objective:** based on the currently/historically-reported cell KPIs, to **predict the potential anomaly KPIs/events** in the future

**System Challenges:**
- Difficult to know in advance which of the **thousands** of KPIs (from the same or nearby cells) are **relevant and correlated** with the predictive KPIs.
- Some KPIs from neighboring cells may be related, like in the case of high inter-cell interference, but may not trigger an anomaly event at these neighbor cells. Needs to extract both **temporal** and **spatial** features in the multi-cell environment.
- The anomaly event labels **rarely** account for less than **0.1 percent** over all the reported KPIs. The model needs to focus on those anomaly points.
Anomaly Detection: Model Selection

- Select an appropriate deep learning models to extract both **spatial** and **temporal** features?

**CNN**

- Good at extracting spatial features from input: which KPIs are more correlated to the predictive target?
- Ignore temporal relations

**RNN**

- Good at extracting temporal relations between time-series inputs
- Detect “periodic” pattern
- Selectively remember “important” time slots
- Gradient vanishing & gradient explosion
- Cannot remember long-term information

**LSTM (Long Short Term Memory)**

- Resolve gradient vanishing & gradient explosion
- Enable long-term memory
- Cannot well extract spatial features
Anomaly Detection: ConvLSTM

- can extract both **temporal** and **spatial** features
- input: thousands of historical cell KPIs
- output: predictive values of target cell KPIs
- model structures (shares a lot with **LSTM**) 

\[
\begin{align*}
    i_t &= \sigma(W_{xi} \ast X_t + W_{hi} \ast H_{t-1} + W_{ci} \circ C_{t-1} + b_i) \\
    f_t &= \sigma(W_{xf} \ast X_t + W_{hf} \ast H_{t-1} + W_{cf} \circ C_{t-1} + b_f) \\
    C_t^* &= \tanh(W_{xc} \ast X_t + W_{ch} \ast H_{t-1} + W_{ch} \circ C_{t-1} + b_f) \\
    C_t &= f_t \circ C_{t-1} + C_t^* \\
    o_t &= \sigma(W_{xo} \ast X_t + W_{ho} \ast H_{t-1} + W_{co} \circ C_t + b_o) \\
    H_t &= o_t \circ \tanh(C_t) \\
    Y_t &= W_{hy} \ast H_t + b_{hy}
\end{align*}
\]

- the operator "\(\ast\)" is the convolution operation that is the key in this model
- the convolution operation enables to extract spatial features

How to handle extremely unbalanced dataset?

- **Data undersampling**
  - *discard the redundant dataset* that is far from the time when the anomaly points appear. The model can concentrate on the points surrounding the anomaly points and explore the pattern of how KPIs will distribute before an anomaly appear.

- **penalized classification**
  - *penalizing error anomaly classification* will introduce an extra cost to the model when it falsely classifies an anomaly point as a normal one. These penalties force the model to give greater emphasis to the minority class.

\[
\text{training_loss} = \alpha \times \text{normal_class} + \beta \times \text{anomaly_class}
\]
Root Cause Analysis: System Challenges

System Challenges

- root cause labels are *not available* for supervised training
  - network engineers did not deliberately attach the resulting fault to the associated logs
  - *too expensive* to collect the logs by purposely introducing the cell faults

Solutions

- Generate a *synthetic dataset* of cell faults with NS3
- employ unsupervised clustering by removing the fault labels, with which we are able to quantify how the model performs
- apply the model to a real-world dataset
Root Cause Analysis: NS3 simulation

NS3 simulation steps

- Generate “normal” topology
- Randomly select $x$ fault cells
- Randomly assign fault case $y$
- Run simulations and collect metrics (labeled by fault case)
- Train/test classifier

NS3 eNB topology configuration

power radiation of normal/anomaly eNBs
NS3 simulation setup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topology</td>
<td>3-sector hexagonal grid, 3 sites</td>
</tr>
<tr>
<td>Carrier Freq.</td>
<td>2.12 GHz</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>10 MHz</td>
</tr>
<tr>
<td>Channel model</td>
<td>UMi, shadow fading, no fast fading</td>
</tr>
<tr>
<td>TX power</td>
<td>46 dBm</td>
</tr>
<tr>
<td>Antenna</td>
<td>3D parabolic 70° azim., 10° vertical beamwidth 9° downtilt</td>
</tr>
<tr>
<td>Handover algorithm</td>
<td>A3 RSRP (default Hyst = 3 dB, TTT = 256 ms)</td>
</tr>
<tr>
<td>Scheduler</td>
<td>Proportional fair</td>
</tr>
<tr>
<td>Mobility model</td>
<td>Steady state random waypoint UE speeds ∈ U(1,20)m/s</td>
</tr>
<tr>
<td>Traffic model</td>
<td>Constant bit rate 800 kbps DL + UL flows</td>
</tr>
</tbody>
</table>

- **EU**: excessive uptilt
- **ED**: excessive downtilt
- **ERP**: excessive cell power reduction
- **CH**: coverage hole
- **TLHO**: too late handover
- **II**: inter-cell interference

### Normal cell configuration

### Fault cell configuration

<table>
<thead>
<tr>
<th>Fault Cause</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>EU</td>
<td>Down tilt = [0.1]°</td>
</tr>
<tr>
<td>ED</td>
<td>Down tilt = [16,15,14]°</td>
</tr>
<tr>
<td>ERP</td>
<td>$\Delta P_{TX} = [7,8,9,10]$ dB</td>
</tr>
<tr>
<td>CH</td>
<td>$\Delta hole = [49,50,52,53]$ dB</td>
</tr>
<tr>
<td>TLHO</td>
<td>$P_{TX_{max}} = 33$ dB</td>
</tr>
<tr>
<td>TLHO</td>
<td>Down tilt = 15°</td>
</tr>
<tr>
<td>TLHO</td>
<td>AB = [30, 60]°</td>
</tr>
<tr>
<td>TLHO</td>
<td>EB = 10°</td>
</tr>
<tr>
<td>No fault</td>
<td>Normal</td>
</tr>
</tbody>
</table>
Root Cause Analysis: NS3 simulation

- 6 possible faults:
  - EU (excessive upilt),
  - ED (excessive downtilt),
  - ERP (excessive power reduction),
  - II (inter-cell interference)
  - TLHO (too late handover)
  - CH (coverage hole)

- Randomly select 6 out of 30 cells as the faulty ones

- Randomly assign 1 possible fault to the faulty cell

- 40 KPIs
  - 'ul_delay_max', 'ul_PduSize_avg', 'dlrx_size', 'dl_TxBytes',
  - 'ulmac_mcs', 'dl_PduSize_std', 'fault', 'dl_delay_max',
  - 'ul_delay_avg', 'ul_PduSize_min', 'ul_TxBytes', 'dltx_size',
  - 'dl_nRxPDUs', 'ultx_mcs', 'ulmac_sframe', 'dlrsrp',
  - 'ul_delay_std', 'ul_PduSize_std', 'ul_nTxPDUs', 'dist',
  - 'dl_PdSize_max', 'ultx_size', 'dl_delay_std', 'ul_TxBytes',
  - 'dl_PduSize_min', 'dl_RxBytes', 'ul_PduSize_max', 'ul_nRxPDUs',
  - 'dlrx_mcs', 'dlsinr', 'dl_delay_avg', 'ulmac_frame',
  - 'dlrx_mode', 'dl_delay_min', 'ulmac_size', 'dl_PduSize_avg',
  - 'dl_nTxPDUs', 'dltx_mcs', 'ul_delay_min', 'UE location'

- 1 hour duration
Root Cause Analysis: Unsupervised Learning

- feature selections with an auto-encoder
  - a critical preprocessing step that selects a subset from the high-dimension input to decrease the overfitting probability and to reduce the training/inference time
- Auto-encoder is an unsupervised data coding approach that can extract both linear and nonlinear relations from high-dimensional input
  - the similar feed-forward network structure with CNN and consists of two symmetrical components: encoder and decoder
    - The encoder takes the high-dimensional data and outputs the low-dimensional one, while the decoder will learn to fully recover the initial input from the compressed output with little loss.
Root Cause Analysis: Unsupervised Learning

- Agglomerative Clustering
  - a bottom-up algorithm.
  - flow: starts by regarding each feature input as an independent cluster and repeats to merge two nearest clusters (measured by Euclidean distance or Pearson correlation distance) iteratively until the total remaining cluster number equals to a predefined number.
  - limitation: cannot naturally map each cluster to a particular fault class. A network expert may further need to empirically infer the physical representation of each cluster, e.g., intercell interference, based on the distributions of significant KPIs.
Evaluations: Anomaly Prediction

- **Prediction Objective**: used the last 5 hours data to predict the value in the next hour of "X2 handover failure rate" (only an example)

- **Deep Learning Models** (implemented with Tensorflow/Keras):
  - CNN (resnet50)
  - LSTM
  - convLSTM
  - CNN + convLSTM

- **Performance Metrics**:
  - true positive (TP): the number that anomaly points are correctly predicted (key indicator)
  - false negative (FN): the number that anomaly points are missing
  - false positive (FP): the number that we give a false alarm over a normal case
  - true negative (TN): the number that we correctly predict a normal case
  - MSE: mean square error over the anomaly points and the whole dataset
**Evaluations: Anomaly Prediction**

### Prediction Performance with Different ML Models

<table>
<thead>
<tr>
<th>Model</th>
<th>TP</th>
<th>FP</th>
<th>ANOM_MSE</th>
<th>ALL_MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>1</td>
<td>5</td>
<td>0.0185</td>
<td>0.0041</td>
</tr>
<tr>
<td>CNN</td>
<td>3</td>
<td>11</td>
<td>0.032</td>
<td>0.0083</td>
</tr>
<tr>
<td>ConvLSTM</td>
<td>15</td>
<td>17</td>
<td>0.0117</td>
<td>0.0032</td>
</tr>
<tr>
<td>CNNConvLSTM</td>
<td>18</td>
<td>23</td>
<td>0.00096</td>
<td>0.0022</td>
</tr>
</tbody>
</table>

- **convLSTM**, and **CNN+convLSTM** perform much better than **LSTM** and **CNN**
- **important to extract spatial and temporal features at the same time**

### Prediction Performance with Different Anomaly Class Weights

<table>
<thead>
<tr>
<th>Weight</th>
<th>TP</th>
<th>TN</th>
<th>FP</th>
<th>FN</th>
<th>recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01/1</td>
<td>16</td>
<td>5854</td>
<td>391</td>
<td>7</td>
<td>69.5%</td>
</tr>
<tr>
<td>0.001/1</td>
<td>20</td>
<td>4442</td>
<td>1802</td>
<td>3</td>
<td>86.9%</td>
</tr>
<tr>
<td>0.0001/1</td>
<td>23</td>
<td>3022</td>
<td>3223</td>
<td>0</td>
<td>100%</td>
</tr>
</tbody>
</table>

- an insufficiently high weight => low recall
- excessively increase the weight => blindly classify any input as anomaly KPIs
- needs to explore the trade-off between the anomaly prediction accuracy and the tolerance of false alarms to reach an optimal point.

**class weight:** normal/anomaly

**recall** = \(\frac{TP}{TP+FN}\)
Evaluations: Root Cause Analysis

Clustering accuracy: **99.5%** by comparing the fault labels in the dataset. (Auto-encoder + agglomerative clustering)

<table>
<thead>
<tr>
<th>Clusters</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 1</td>
<td>Normal</td>
</tr>
<tr>
<td>Cluster 2</td>
<td>Coverage Hole</td>
</tr>
<tr>
<td>Cluster 3</td>
<td>Too late HO</td>
</tr>
<tr>
<td>Cluster 4</td>
<td>Excessive reduction of cell power</td>
</tr>
<tr>
<td>Cluster 5</td>
<td>Excessive Upshift</td>
</tr>
<tr>
<td>Cluster 6</td>
<td>Inter-system interference</td>
</tr>
<tr>
<td>Cluster 7</td>
<td>Excessive downshift</td>
</tr>
</tbody>
</table>

*KPI distributions over 6 faulty cases + 1 normal case*

- Although the network cluster might be unknown, we can take it as the input to the deep reinforcement learning for the self-healing.
Conclusions & Future Work

- propose a self-organizing cellular radio access network system with deep learning
- design and implement the anomaly prediction and root cause analysis components with deep learning and the evaluation of the system performance with real world data from a top-tier US cellular network operator
- demonstrate that the proposed methods can achieve 86.9% accuracy for anomaly prediction and 99.5% accuracy for root cause analysis

Future Work

- continue to design and implement the last component, "self-healing functions" with deep reinforcement learning and make RAN as an integrated, close-loop, self-organizing system.
- investigate the root cause analysis with supervised learning with real-world fault labels.
- better understand how KPI sampling granularity will effect the anomaly prediction accuracy.