Automating test results analysis using neural networks

Carlos Hernandez, Texas Instruments
Jonah Willis, Texas Instruments
Agenda

1. What's complicated about test results
   - Embedded functional tests
   - Scaling

2. Proposed model for automated results analysis

3. Preliminary results

4. Future Work
What’s complicated about test results
● Even the most simple **embedded** boot test needs a lot:
  ○ Build machine
  ○ Host server (aka TEE, Controller, dispatcher, etc.)
  ○ DUT in good health, working power, Ethernet, no loose cables, etc.
  ○ TFTP server working
  ○ NFS server working
  ○ Bootloader
  ○ Linux kernel
  ○ Filesystem
  ○ Correct test logic
- Functional tests required more equipment
  1. Digital Multi-meter
  2. Camera tester 1
  3. Camera tester 2
  4. USB MSC
  5. Audio loopback
  6. Ethernet
  7. GPIO
  8. HDMI tester
- More potential points of failure
- Test equipment behavior variance, even on the same brand and model!
Some setups can get really complex

- Multiple DUTs connected together
- Multiple boards controlled by automation
  1. Auto Interface1
  2. DUT 1
  3. Auto Interface2
  4. DUT 2
Scaling

- Dozens of racks
- Hundreds of DUTs
- Thousands of functional/performance tests run every day
- Constantly growing list of supported devices, operating systems, and products.
- Multiple failure reasons
  - Known active bugs
  - Setup issues
  - Test issues
  - New bugs
Proposed model to automate results classification
Model Description

- Input data is gathered from our test results and bug databases
- Results that can be classified by a single criterion are excluded from the classification model
  - Passing test cases
  - Test cases with known active bugs
- Remaining classifications are determined by the model
  - Two categories of input to the model
    - Execution notes
      - Metadata, error strings, failure criteria, partial test logs.
    - Execution history
      - Previous ten execution results
  - Input data must be preprocessed before it can be used as input to the neural network
Model Description: Inputs

- **Execution notes**
  - **Step 1: strip metadata from the notes**
    - Timestamps
    - Machine names
    - Hyperlinks
  - **Step 2: extract textual features**
    - Use TF-IDF* to generate top 1024 weighted features
      - $TF(t) = \frac{\text{Number of times term } t \text{ appears in a document}}{\text{Total number of terms in the document}}$
      - $IDF(t) = \ln \left( \frac{\text{Total number of documents}}{\text{Number of documents with term } t \text{ in it}} \right)$
      - $TF-IDF(t) = TF(t) \times IDF(t)$

- **Execution history**
  - Represented as a binary string of ten executions where ‘0’ is a failure and ‘1’ is a pass.
    - Ex: “1010111000”
  - Use one-hot encoding to assign an input to all $2^{10} = 1024$ combinations

Model Description: Outputs

- The neural network classifies each failure into one of three classifications:
  - Potential software or hardware issues
    - Failures that appear to be real problems with the product
  - Test issues
    - Failures that appear to be an issue with the test scripts
    - Generally, this is triggered by execution histories that flip back-and-forth between passing and failing. Since code affecting a single test case does not usually change every night, this indicates a potential problem with the test.
  - Setup issues
    - Failures that appear to be a problem with the test setup
      - Bootloader issues
      - Missing UART or JTAG connection
      - Unresponsive test equipment
      - Host service (nfs, tftp, etc.) failures
Training the Model

- Optimal hyper-parameters for the neural network are chosen via grid search
- Thousands of test cases → hundreds of thousands of executions
  - Not feasible to manually label the executions for model training
  - Necessary to define heuristics to pre-classify past execution data
    - This means the initial version of the model can only be as good as the heuristics

- Heuristics

<table>
<thead>
<tr>
<th>Setup Issues</th>
<th>Failing results that match predefined setup issue strings.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Issues</td>
<td>“Obvious” patterns where execution has flipped between pass and fail repeatedly. E.g., ‘1010101010’</td>
</tr>
<tr>
<td>Potential SW/HW Issues</td>
<td>All remaining failures.</td>
</tr>
</tbody>
</table>
Retraining the Model

- Assuming we did a good job defining our heuristics, the initial model should be pretty accurate, but probably not perfect.
- We can start using the test results report with the initial model. The report contains buttons that allow for a classification to be manually overridden so that developers or test engineers can reclassify any executions that have been incorrectly classified. These updated results are then used to retrain the model.
- The more we use the report, the better the classifications get!
Putting it all Together

Test database
- Passing test cases
- Failing test cases

Bug database
- Active bugs

Step 1: Gather Input Data
- Compare

Step 2: Process Input Data
- Test cases with active bugs
  - Execution notes
  - Strip metadata
  - TF-IDF
  - 1024 text features
  - Neural Network
  - 1024 execution patterns

Step 3: Classify Results
- Test cases without active bugs
  - Execution history
  - One-hot encoding

Key
- Database
- Data subset
- Data field
- Processing
- Classification result
- Report

PASS

ACTIVE BUGS

POTENTIAL SW/HW ISSUES

TEST ISSUES

SETUP ISSUES

Fix incorrect classifications
Corrected classifications
Retrain NN

Test results report
Preliminary results
As expected NN model learned well to identify results from heuristic classification

### Linux/Android

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.99</td>
<td>1.00</td>
<td>1.00</td>
<td>1148</td>
</tr>
<tr>
<td>1.00</td>
<td>0.99</td>
<td>1.00</td>
<td>420</td>
</tr>
<tr>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>2104</td>
</tr>
</tbody>
</table>

### TI RTOS

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>3979</td>
</tr>
<tr>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1298</td>
</tr>
<tr>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>9866</td>
</tr>
</tbody>
</table>

### Confusion Matrix

#### Linux/Android

<table>
<thead>
<tr>
<th></th>
<th>1148</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0</td>
<td>416</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2102</td>
</tr>
</tbody>
</table>

#### TI RTOS

<table>
<thead>
<tr>
<th></th>
<th>3979</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1298</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>9864</td>
<td>0</td>
</tr>
</tbody>
</table>
## Results Analysis: Summary Page

### Passing Results

<table>
<thead>
<tr>
<th>Classification</th>
<th>Area</th>
<th>Test</th>
<th>Platform</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- Passed (4934 test cases)

### Active Bugs

<table>
<thead>
<tr>
<th>Classification</th>
<th>Bugs</th>
<th>Test</th>
<th>Platform</th>
<th>Notes</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- Active Bug (9)

### Potential SW/HW Issues

<table>
<thead>
<tr>
<th>Classification</th>
<th>Area</th>
<th>Test</th>
<th>Platform</th>
<th>Notes</th>
<th>Reclassify</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- Potential SW/HW Problem (717 test cases)

### Test Issues

<table>
<thead>
<tr>
<th>Classification</th>
<th>Area</th>
<th>Test</th>
<th>Platform</th>
<th>Notes</th>
<th>Reclassify</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- Test Issue (143 test cases)

### Setup Issues

<table>
<thead>
<tr>
<th>Classification</th>
<th>Cluster</th>
<th>Machine</th>
<th>Test</th>
<th>Platform</th>
<th>Notes</th>
<th>Reclassify</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- Setup Issue (5)
### Results Analysis: Bugs

#### Active Bugs

<table>
<thead>
<tr>
<th>Classification</th>
<th>Bugs</th>
<th>Test</th>
<th>Platform</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Active Bug (9)</td>
<td>LCPD-7256: Board sometimes hangs after suspend/resume cycle</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LCPD-12443: SD boot time with coreSDK rootfs increases ~30% on</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kernel boot time test when bootloader, kernel and sdk-rootfs are in mmc-sd (1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LCPD-12777: PCIe link is not up for Inateck PCIe-USB card</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LCPD-9366: PCIe USB drive sometimes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Log Path:**

- Boot Time are collected. Total is [44.302221009, 44.655286311, 44.54302915, 44.50369042, 44.605928268], Boottime-InitFS out of expected range: 38.85323618459999 > 22.9180055239344262 + 4.965278326771514, Boottime-Total out of expected range: 44.522020755999996 > 28.889905845901642 + 4.96596216827945

---

**Log Path:**

- Boot Time are collected. Total is [5.4907777075, 5.386413151, 5.389170008, 5.391028343, 5.488722454], Boottime-InitKernel contains outlier samples outside 10 stdev window. measured value=4.127195979080001, historical mean=11.3708232, std=0.12462420682680478. Please check your setup. Performance data won’t be saved

---

**Logo:**

- Texas Instruments
- Linaro Connect 2019
# Results Analysis: Test Issues

<table>
<thead>
<tr>
<th>Classification</th>
<th>Area</th>
<th>Test</th>
<th>Platform</th>
<th>Notes</th>
<th>Reclassify</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▼ Test Issue (132 test cases)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▼ ALSA (4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▼ DRM (26)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▼ DRM mode test AB15 (RGB16) (3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note:
- undefined method `each` for nil

```
```
## Setup Issues

<table>
<thead>
<tr>
<th>Classification</th>
<th>Cluster</th>
<th>Machine</th>
<th>Test</th>
<th>Platform</th>
<th>Notes</th>
<th>Reclassify</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup Issue (6)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 (1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SystemloadException (9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Failed to load bootloader (7)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tigtfarm25 (1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tigtfarm23 (3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tigtfarm08 (2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tigtfarm11 (16)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tigtfarm16 (1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tigtfarm24 (47)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No such file or directory (1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tigtfarm01 (1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ADC basic functionality evm

No such file or directory - /dev/hp_wavegen
"/usr/lib/ruby/vendor_ruby/serialport.rb:25:in "create"",
"/usr/lib/ruby/vendor_ruby/serialport.rb:25:in "new",
"/usr/local/vatf/source/target/base_listener.rb:45:in "new",
"/usr/local/vatf/source/target/equipment_connection.rb:38:in "connect",
"/usr/local/vatf/source/target/equipment_driver.rb:30:in "connect",
"/home/berryfols/openTest/vatf-scripts/vatflib/parago-
# Results Analysis: Potential HW/SW Problems

## Potential SW/HW Issues

<table>
<thead>
<tr>
<th>Classification</th>
<th>Area</th>
<th>Test</th>
<th>Platform</th>
<th>Notes</th>
<th>Reclassify</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Potential SW/HW Problem (732 test cases)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ALSA (3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARM Benchmarks (4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>boot (2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SoC Detection (2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kernel Long-term Stress Boot Test. Verify kernel boots 1000 times successfully (3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| | | | | | | |
| | [kvm] Kernel failed to boot 10 times out of 1000 LOG PATH | Setup Issue | Yes | |
| | | Test Issue | | | |
| | | Potential SW/HW Problem | | | |

| | | | | | | |
| | [kvm] Kernel failed to boot 1 times out of 1000 LOG PATH | Setup Issue | No | |
| | | Test Issue | | | |
| | | Potential SW/HW Problem | | | |

| | | | | | | |
| | [kvm] Kernel failed to boot 3 times out of 1000 LOG PATH | Setup Issue | No | |
| | | Test Issue | | | |
| | | Potential SW/HW Problem | | | |
Regressions Detection

1. **Latest Test Inputs** -> **Neural Network**
2. Neural Network outputs **New Classification**
3. **New Classification** is compared with **Previous Classification**
4. If the results are removed, the process stops.
5. If the results are updated, existing tests are adjusted.
6. New results are recorded for new tests.
7. If the regression is fixed, it is reported.
8. If a new regression is detected, it is also reported.
Results Analysis: Conclusions

- Still in early deployment phase and model may change in the future to accommodate missing important information
- Have already started saving significant validation time
  - Failures are classified automatically into appropriate buckets and different people/teams can focus in parallel on multiple areas (potential new bugs, setup issues, and test issues)
  - Easier to determine patterns when new bugs are submitted (e.g. affected platforms) since results are aggregated by test areas and test cases.
  - Extremely fast sorting, and filtering by multiple criteria from a single web page
- Have saved developers time by reducing false positive regressions notifications
- Have allowed validation team to increase test coverage on ‘secondary’ boards because analysis heavy-lifting is done by the model.
Future Work
Future Work

● Retrain weights
● Tf-Idf tweaks
● UI enhancements
● Apply model to other products and operating systems
Thank you

Join Linaro to accelerate deployment of your Arm-based solutions through collaboration

contactus@linaro.org