**Problem**

Static analysis of Android-specific message-passing system:
- Link inference: Can component c communicate with component d? (c and d have a link?)

**Approach**

- A machine learning model to automatically learn from "must" links and to predict for "may" links.
- An end-to-end encoder and classifier architecture: link inference neural network (LINN) -- requires minimal domain knowledge.
- Type-directed encoder (TDE): schematically composing encoder from constituent data type definition -- general and flexible.

**Background: Android ICC**

- Android applications are conceptually collections of components.
- Applications can leverage the functionality of other applications through a sophisticated message-passing system: Inter-Component Communication (ICC).
- Any inter-component or inter-application program analysis must first begin by computing the ICC links.

**Interpretability**

- Kernel activations: single kernel detects useful patterns but also may not be so precise.
- Sensitivities to masking: model picks up right parts of information.
- Encoding visualizations: encoder is able to automatically learn semantic artifacts.
- Error inspection: some fails regarding the exact encoder out of constituent data type definition.

**Experiments & Results**

- Dataset of 10,500 Android applications from Google Play. See right for Training / testing details.
- Simulated ground truth for may links induced by empirical distribution of impressions.
- 4 instantiation of the architecture and TDE.
- Measuring inference time for efficiency; Spearman’s y for correlation; F1 score and area under ROC for accuracy and recall; entropy of predicted probabilities for the ability of triaging links; true positive rate (TPR) within y >= 0.95 for the efficiency of manual instantiation of highly ranked links.
- All instantiations are good at predicting linkages.
- Str-cnn is the smallest and the fastest; typed-tree is the best at predicting links given slightly more resources.

**Related works**

- Android analysis: Formal modeling for ICC process. Hard coded domain knowledge for triaging may links.
- Static analysis alarms: Counting based, more sophisticated statistics, probabilistic information-flow specifications. Manual labeling of small amount of data.
- Machine learning for programs: Learning from "big code". Tree structured neural nets for abstract syntax trees (AST) and expressions.

**Figure 5: Detailed results for the typed-tree instantiation**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>str-cnn</th>
<th>typed-simple</th>
<th>typed-tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>0.972</td>
<td>0.918</td>
<td>0.993</td>
</tr>
<tr>
<td>Precision</td>
<td>0.975</td>
<td>0.901</td>
<td>0.991</td>
</tr>
<tr>
<td>Recall</td>
<td>0.982</td>
<td>0.899</td>
<td>2.399</td>
</tr>
<tr>
<td>TPR</td>
<td>3.002</td>
<td>3.002</td>
<td>2.220</td>
</tr>
</tbody>
</table>

Table 4: Summary of model evaluation

**Table 5: Some CNN kernels and their top stimuli**

<table>
<thead>
<tr>
<th>Kernel Activation</th>
<th>Top Stimuli</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv2d_1x14</td>
<td>deep, &gt;0.5</td>
</tr>
<tr>
<td>conv2d_5x5</td>
<td>medium, &gt;0.5</td>
</tr>
<tr>
<td>conv2d_7x7</td>
<td>low, &gt;0.5</td>
</tr>
</tbody>
</table>

**Figure 6:** Explaining sensitivities to masking

**Figure 7:** Intent encodings visualized using t-SNE

(a) android.intent.* (b) Imprecise V29 actions

- Intent filter for a component

This work is partially supported by the National Science Foundation Grants CNS-156383 and CCF-1563140 and by the Google Faculty Research Award.