Appendix

A. Interpreting t-SNE Plots

By mapping PCs back to source code, we observe that the model has learned about program structure. We show examples from two of the most challenging SPEC CPU2006 applications to learn, mcf and omnetpp.
A.1. mcf

The following function from mcf appears in two different t-SNE clusters:

```c
while( node )
{
    if( node->orientation == UP )
        node->potential = node->
            basic_arc->cost + node->pred->potential;
    else /* == DOWN */
    {
        node->potential = node->pred->
            potential - node->basic_arc->cost;
        checksum++;}
    tmp = node;
    node = node->child;
    node = tmp;

    while( node->pred )
    {
        tmp = node->sibling;
        if( tmp )
            node = tmp;
        break;
        else
            node = node->pred;
    }
}
```

One cluster contains only different instances of line 4, unrolled into three different instructions at three different PCs. We show the line of code, followed by the assembly code in (PC: Instruction) format:

```
node->potential = node->basic_arc->cost
    + node->pred->potential;
401932: mov 0x18(%rdx),%rsi
401888: mov 0x18(%r10),%rsi
4018df: mov 0x18(%r11),%rsi
```

A second cluster identifies only the PCs responsible for the linked list traversal, at lines 11 and 16:

```
node = node->child;
401878: mov 0x10(%rdx),%r10
40187c: mov %rcx,(%rdx)

4019a2: mov 0x20(%r9),%rcx
```

A.2. omnetpp

We show the result of running t-SNE on the learned (∆, PC) embeddings of omnetpp in Figure 1.

![Figure 1. A t-SNE visualization of the concatenated (∆,PC) embeddings on omnetpp colored according to PC instruction.](image)
Examine some of the clusters closely, we find interesting patterns. The following code inserts and removes items into an owner’s list:

```c
// remove from owner’s child list
if (ownerp!=NULL) {
    if (nextp!=NULL)
        nextp->prevp = prevp;
    if (prevp!=NULL)
        prevp->nextp = nextp;
    if (ownerp->firstchildp==this)
        ownerp->firstchildp = nextp;
    ownerp = NULL;
}
```

The main insertion and removal path are both shown in the same t-SNE cluster:

```c
// insert into owner’s child list as first elem
if (newowner!=NULL) {
    ownerp = newowner;
    prevp = NULL;
    nextp = ownerp->firstchildp;
    if (nextp!=NULL)
        nextp->prevp = this;
    ownerp->firstchildp = this;
}
```

omnetpp’s t-SNE clusters also contain many examples of comparison code from very different source code files that are used as search statements being mapped to the same t-SNE cluster. Since these comparators are long, they get compiled to many different assembly instructions, so we only show the source code below. Lines 3 and 17 are both mapped to the same t-SNE cluster among other similar comparators:

```c
if (m>=0 && m<=last && vect[m])
    return vect[m];
else
    return NULL;
```

```c
void cMessageHeap::shiftup(int from){
    int i,j;
    cMessage *temp;
    i=from;
    while ((j=2*i) <= n)
    {
        if (j<n && (*h[j] > *h[j+1])) // direction
            j++;
        if (*h[i] > *h[j]) //is change necessary?
        {
            temp=h[j];
            (h[j]=h[i])->heapindex=j;
            (h[i]=temp)->heapindex=i;
            i=j;
        }
        else
            break;
    }
}
```

B. Experimental Results

The experimental results for precision/recall are given in Table 1/Table 2 respectively.

C. LSTM Hyperparameters

The hyperparameters for both LSTM models are given in Table 3

D. K-Means Clustering on an Address Trace

In Figure 2 we show the results of running k-means with 6 clusters on 10^6 addresses from omnetpp.
### Table 1. Experimental Results: Precision

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Stream</th>
<th>GHB</th>
<th>Embedding</th>
<th>Kmeans</th>
<th>Only PCs</th>
<th>Only Deltas</th>
</tr>
</thead>
<tbody>
<tr>
<td>bwaves</td>
<td>0.65</td>
<td>0.07</td>
<td>0.89</td>
<td>0.93</td>
<td>0.89</td>
<td>0.89</td>
</tr>
<tr>
<td>gems</td>
<td>0.61</td>
<td>0.05</td>
<td>0.76</td>
<td>0.82</td>
<td>0.76</td>
<td>0.59</td>
</tr>
<tr>
<td>leslie3d</td>
<td>0.72</td>
<td>0.21</td>
<td>0.99</td>
<td>0.80</td>
<td>0.99</td>
<td>0.93</td>
</tr>
<tr>
<td>libquantum</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>solplex</td>
<td>0.68</td>
<td>0.18</td>
<td>0.73</td>
<td>0.83</td>
<td>0.73</td>
<td>0.70</td>
</tr>
<tr>
<td>sphinx3</td>
<td>0.72</td>
<td>0.08</td>
<td>0.97</td>
<td>0.81</td>
<td>0.96</td>
<td>0.86</td>
</tr>
<tr>
<td>astart</td>
<td>0.34</td>
<td>0.25</td>
<td>0.60</td>
<td>0.51</td>
<td>0.60</td>
<td>0.32</td>
</tr>
<tr>
<td>lbm</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.99</td>
<td>0.59</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>mcf</td>
<td>0.0001</td>
<td>0.18</td>
<td>0.33</td>
<td>0.45</td>
<td>0.33</td>
<td>0.28</td>
</tr>
<tr>
<td>milc</td>
<td>0.0001</td>
<td>0.02</td>
<td>0.56</td>
<td>0.82</td>
<td>0.56</td>
<td>0.56</td>
</tr>
<tr>
<td>omnetpp</td>
<td>0.08</td>
<td>0.06</td>
<td>0.63</td>
<td>0.53</td>
<td>0.62</td>
<td>0.51</td>
</tr>
<tr>
<td>websearch</td>
<td>0.1</td>
<td>0.12</td>
<td>0.43</td>
<td>0.55</td>
<td>0.41</td>
<td>0.41</td>
</tr>
<tr>
<td>Geometric Mean</td>
<td>0.11</td>
<td>0.06</td>
<td>0.70</td>
<td>0.69</td>
<td>0.70</td>
<td>0.61</td>
</tr>
</tbody>
</table>

### Table 2. Experimental Results: Recall

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Stream</th>
<th>GHB</th>
<th>Embedding</th>
<th>Kmeans</th>
<th>Only PCs</th>
<th>Only Deltas</th>
</tr>
</thead>
<tbody>
<tr>
<td>bwaves</td>
<td>0.86</td>
<td>0.38</td>
<td>0.10</td>
<td>0.93</td>
<td>0.05</td>
<td>0.06</td>
</tr>
<tr>
<td>gems</td>
<td>0.83</td>
<td>0.36</td>
<td>0.20</td>
<td>0.85</td>
<td>0.04</td>
<td>0.20</td>
</tr>
<tr>
<td>leslie3d</td>
<td>0.87</td>
<td>0.41</td>
<td>0.99</td>
<td>0.80</td>
<td>0.38</td>
<td>0.98</td>
</tr>
<tr>
<td>libquantum</td>
<td>0.99</td>
<td>0.99</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>solplex</td>
<td>0.95</td>
<td>0.41</td>
<td>0.14</td>
<td>0.83</td>
<td>0.14</td>
<td>0.14</td>
</tr>
<tr>
<td>sphinx3</td>
<td>0.89</td>
<td>0.30</td>
<td>0.57</td>
<td>0.81</td>
<td>0.46</td>
<td>0.58</td>
</tr>
<tr>
<td>astart</td>
<td>0.55</td>
<td>0.51</td>
<td>0.15</td>
<td>0.59</td>
<td>0.03</td>
<td>0.15</td>
</tr>
<tr>
<td>lbm</td>
<td>0.98</td>
<td>0.61</td>
<td>1.00</td>
<td>0.82</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>mcf</td>
<td>0.21</td>
<td>0.31</td>
<td>0.13</td>
<td>0.50</td>
<td>0.12</td>
<td>0.13</td>
</tr>
<tr>
<td>milc</td>
<td>0.21</td>
<td>0.05</td>
<td>0.10</td>
<td>0.82</td>
<td>0.001</td>
<td>0.04</td>
</tr>
<tr>
<td>omnetpp</td>
<td>0.64</td>
<td>0.22</td>
<td>0.19</td>
<td>0.59</td>
<td>0.18</td>
<td>0.19</td>
</tr>
<tr>
<td>websearch</td>
<td>0.57</td>
<td>0.20</td>
<td>0.32</td>
<td>0.59</td>
<td>0.23</td>
<td>0.27</td>
</tr>
<tr>
<td>Geometric Mean</td>
<td>0.72</td>
<td>0.39</td>
<td>0.27</td>
<td>0.75</td>
<td>0.12</td>
<td>0.24</td>
</tr>
</tbody>
</table>

### Table 3. Training hyperparameters for each model.

<table>
<thead>
<tr>
<th>Embedding</th>
<th>Network Size</th>
<th>Learning Rate</th>
<th>Number of Train Steps</th>
<th>Sequence Length</th>
<th>Embedding Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>128x2 LSTM</td>
<td>.001</td>
<td>500k</td>
<td>64</td>
<td>128</td>
</tr>
<tr>
<td>Clustering</td>
<td>Network Size</td>
<td>Learning Rate</td>
<td>Number of Train Steps</td>
<td>Sequence Length</td>
<td>Number of Centroids</td>
</tr>
<tr>
<td></td>
<td>128x2 LSTM</td>
<td>.1</td>
<td>250k</td>
<td>64</td>
<td>12</td>
</tr>
</tbody>
</table>
Figure 2. One million memory accesses from `omnetpp` after running k-means clustering on the address space.