ML for Debugger

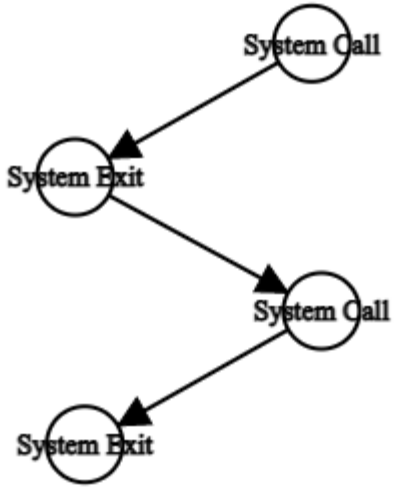
Heeyoung Jung | May 2nd, 2023

THE UNIVERSITY OF CHICAGO

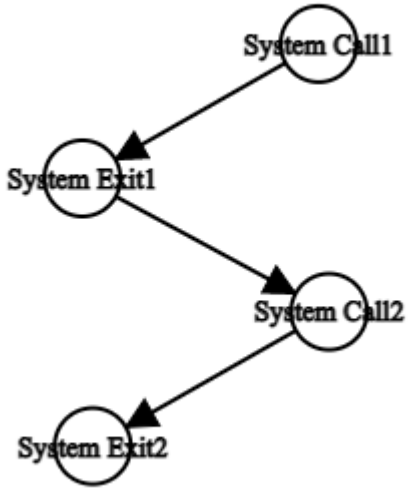
Research Objective

Large : ML for systems
Small : Mapping a Trace to a Graph

Imagine a system call..

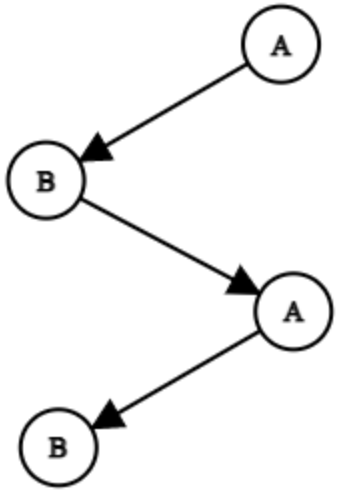


System Call Trace

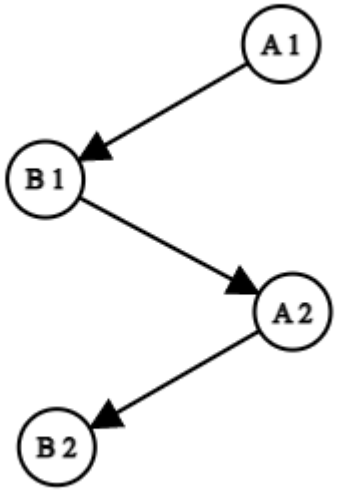


System Call Graph

Imagine a system call..

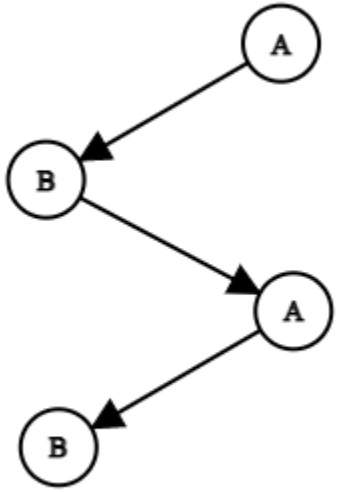


System Call Trace

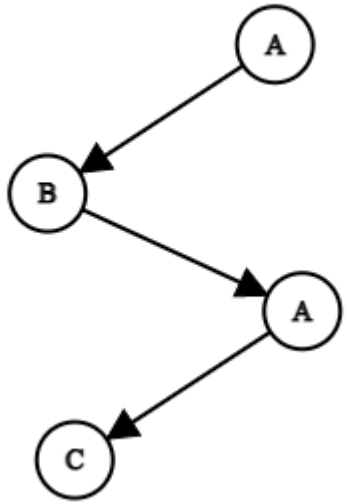


System Call Graph

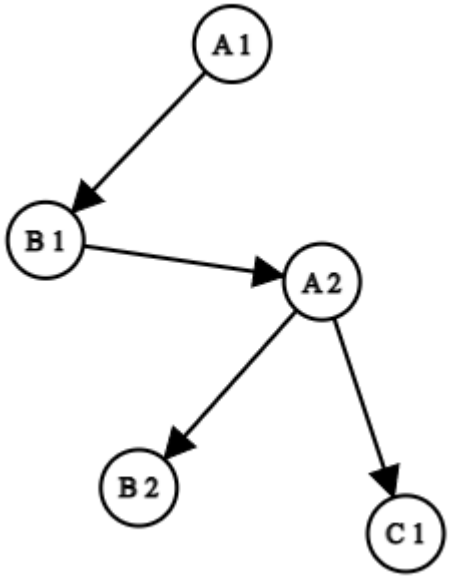
End Goal?



Working

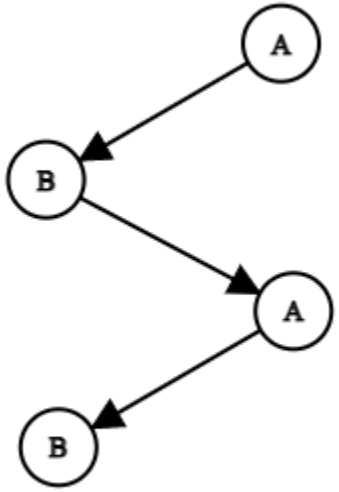


Not Working

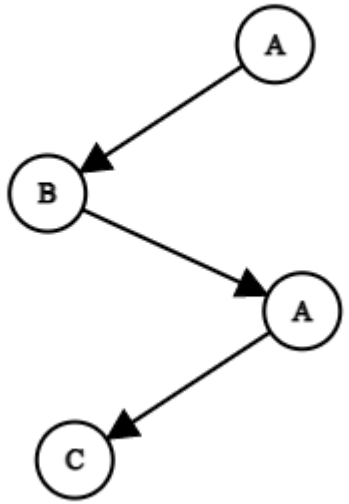


System Call Graph

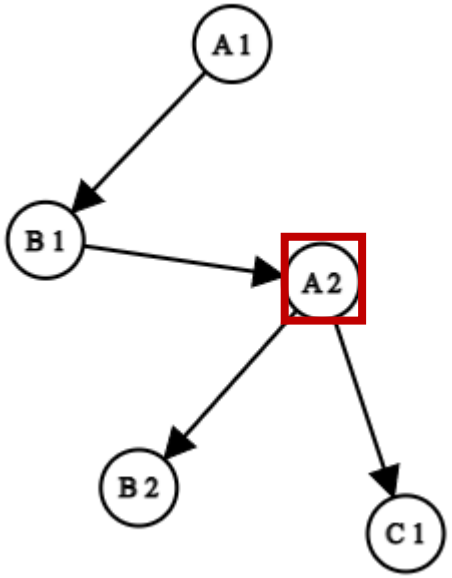
End Goal?



Working

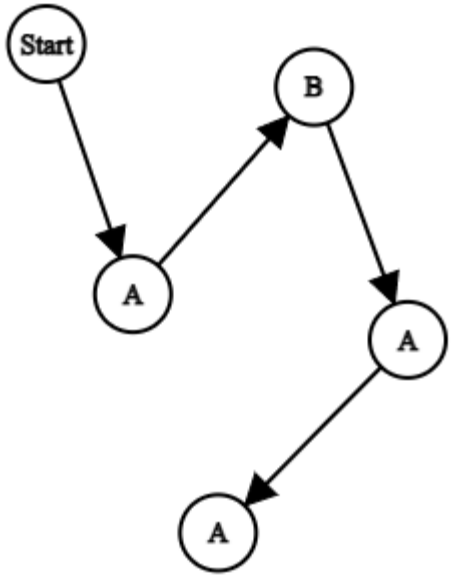


Not Working

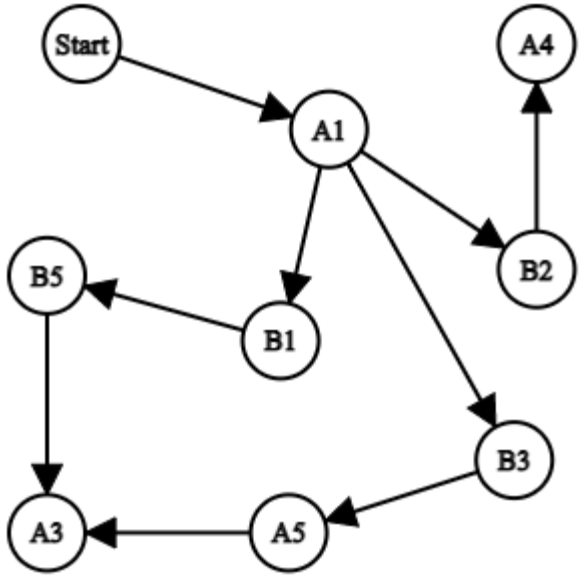


System Call Graph

Can be very complex..

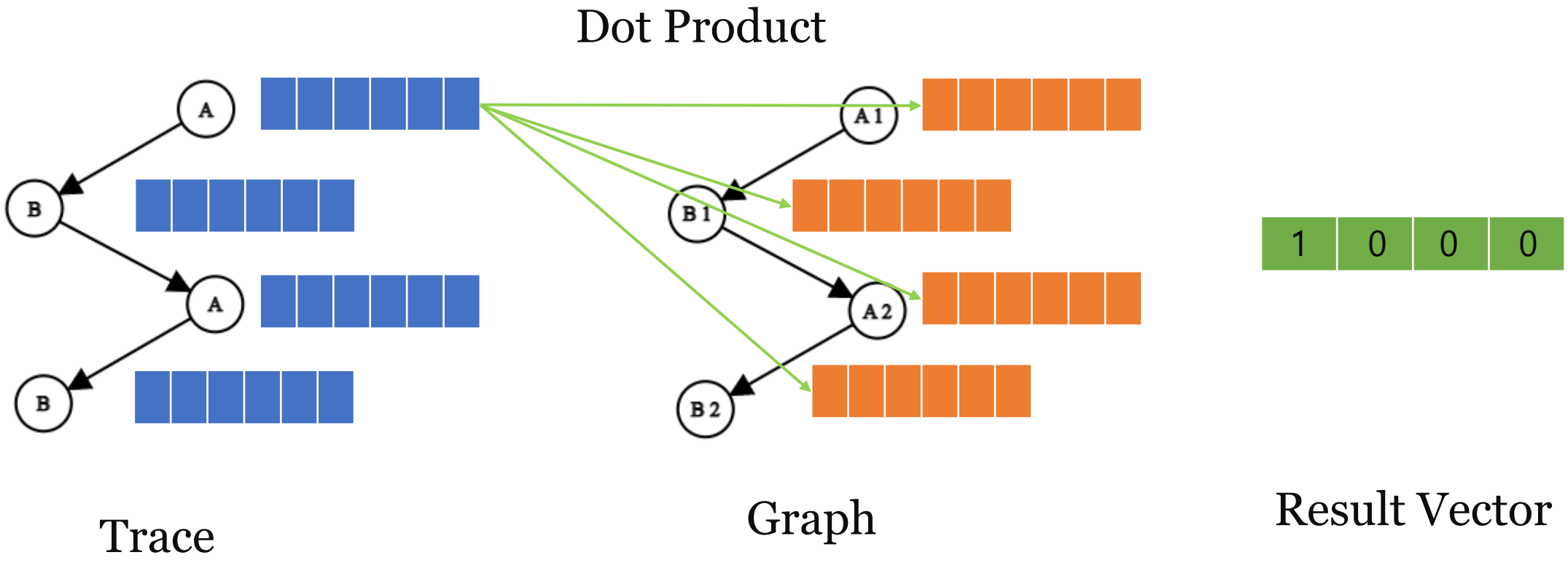


System Call Trace

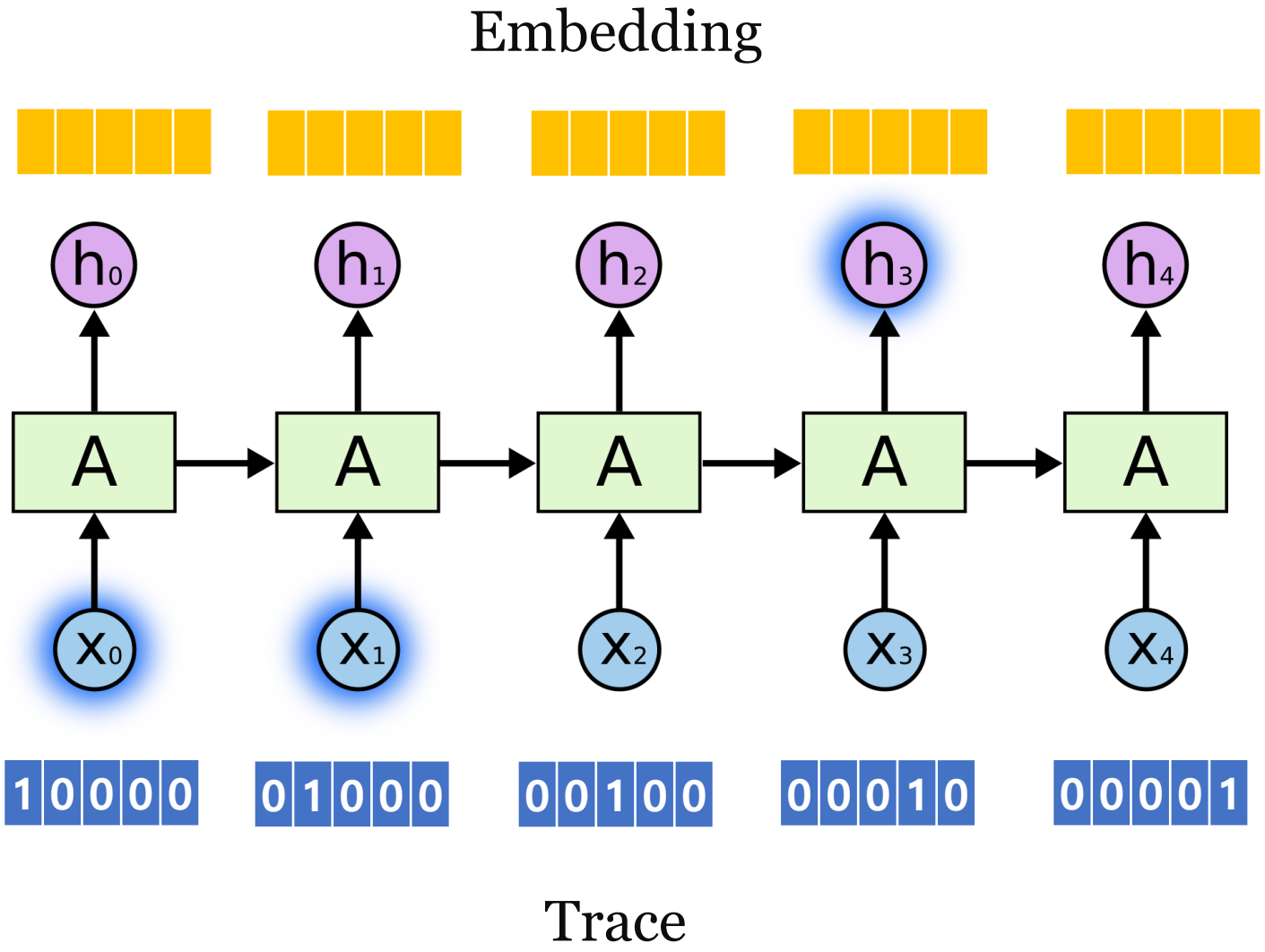


System Call Graph

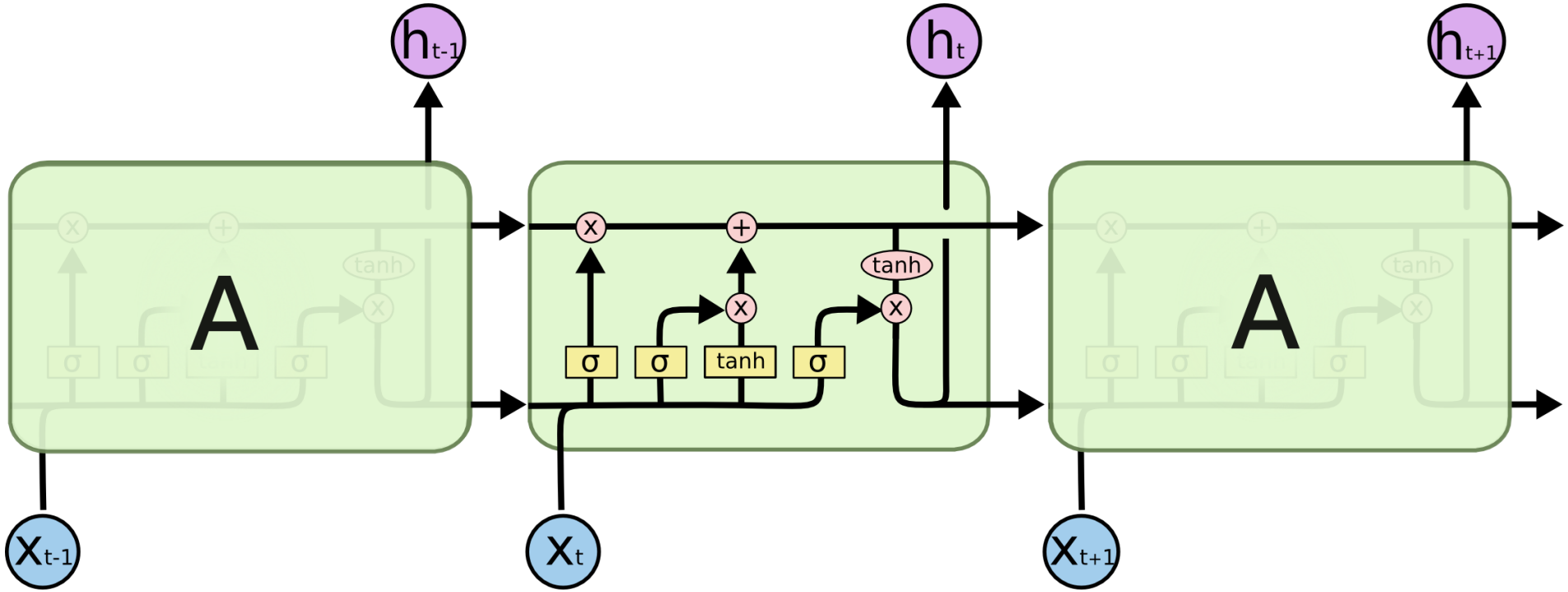
Structure



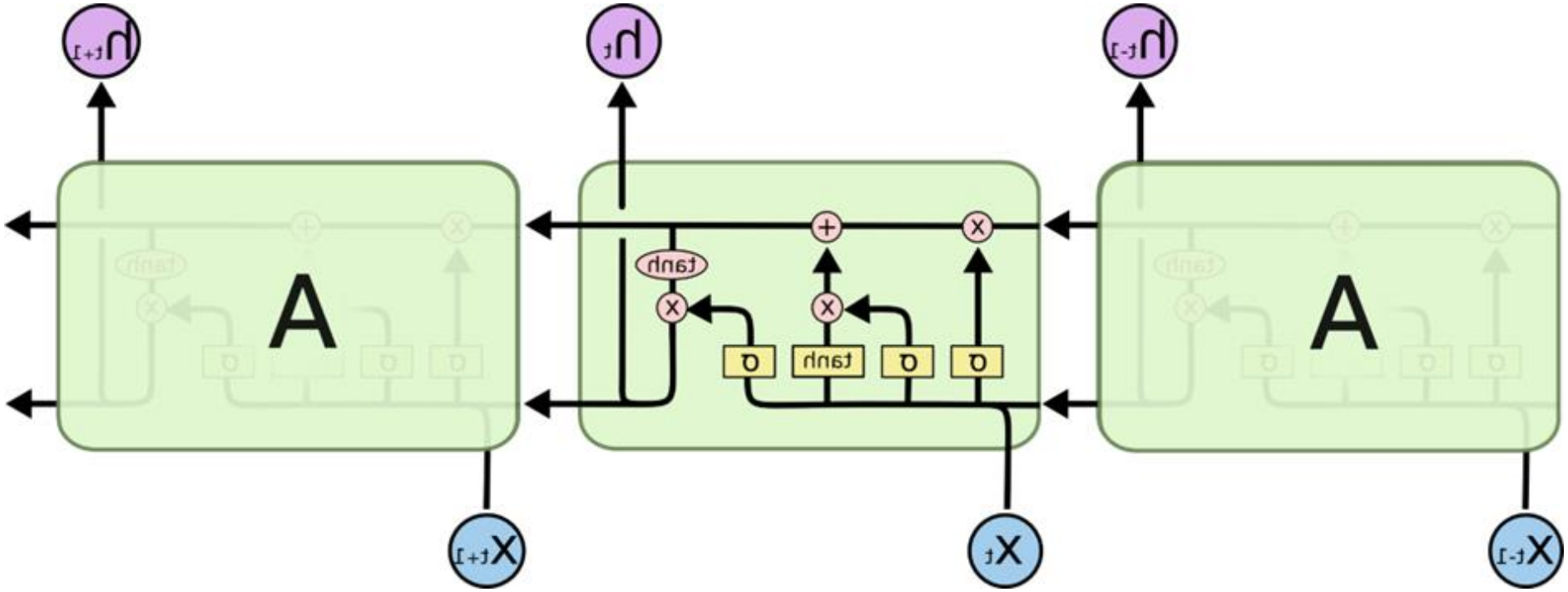
Bi-LSTM


Embedding
h0
h1
h2
h3
h4
A
A
A
A
A
x0
x1
x2
x3
x4
10000
01000
00100
00010
00001
Trace

Bi-LSTM



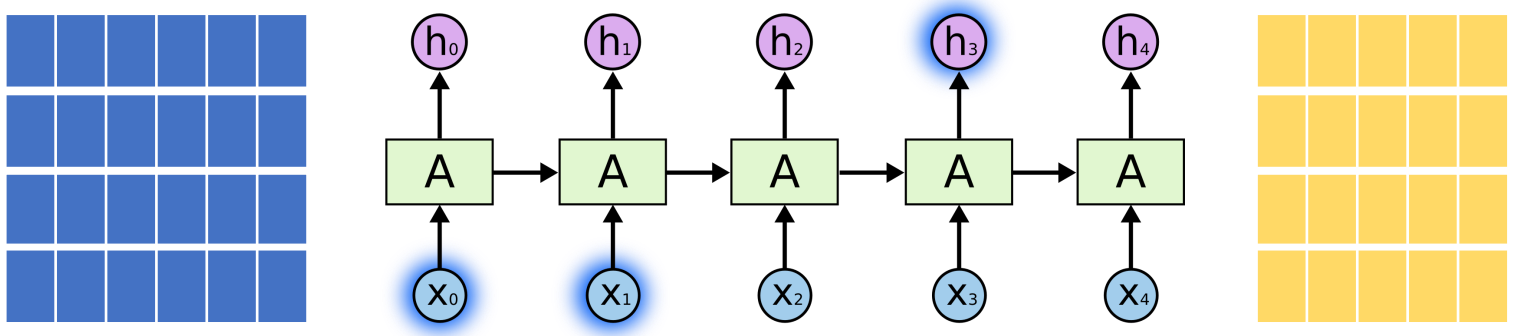
Bi-LSTM



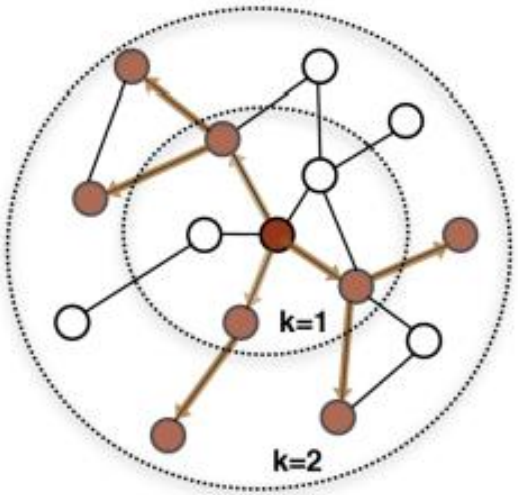
Bi-LSTM

```
class BiLSTM(torch.nn.Module):
    def __init__(self, in_channels, hidden_channels, num_layers=1):
        super(BiLSTM, self).__init__()
        self.lstm = nn.LSTM(input_size=in_channels,
                            hidden_size=hidden_channels,
                            num_layers=num_layers,
                            bidirectional=True,
                            batch_first=True)

    def forward(self, x):
        x = x.unsqueeze(0)
        output, _ = self.lstm(x)
        return output.squeeze(0)
```

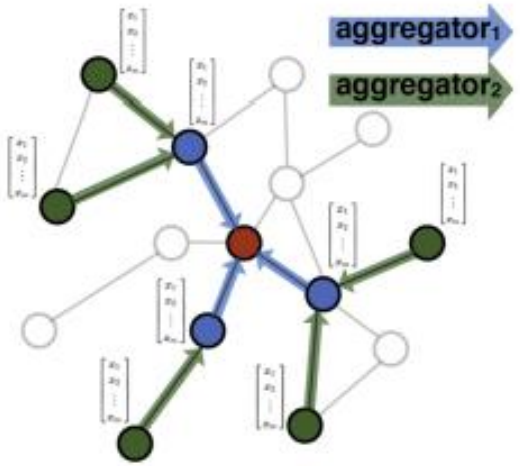
Bi-LSTM



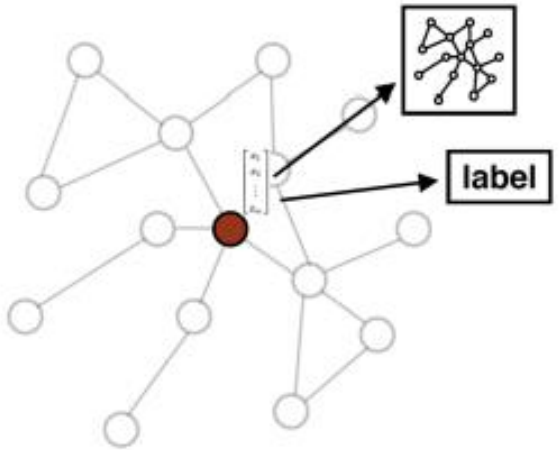
GraphSAGE



1. Sample neighborhood



2. Aggregate feature information from neighbors



3. Predict graph context and label using aggregated information

GraphSAGE

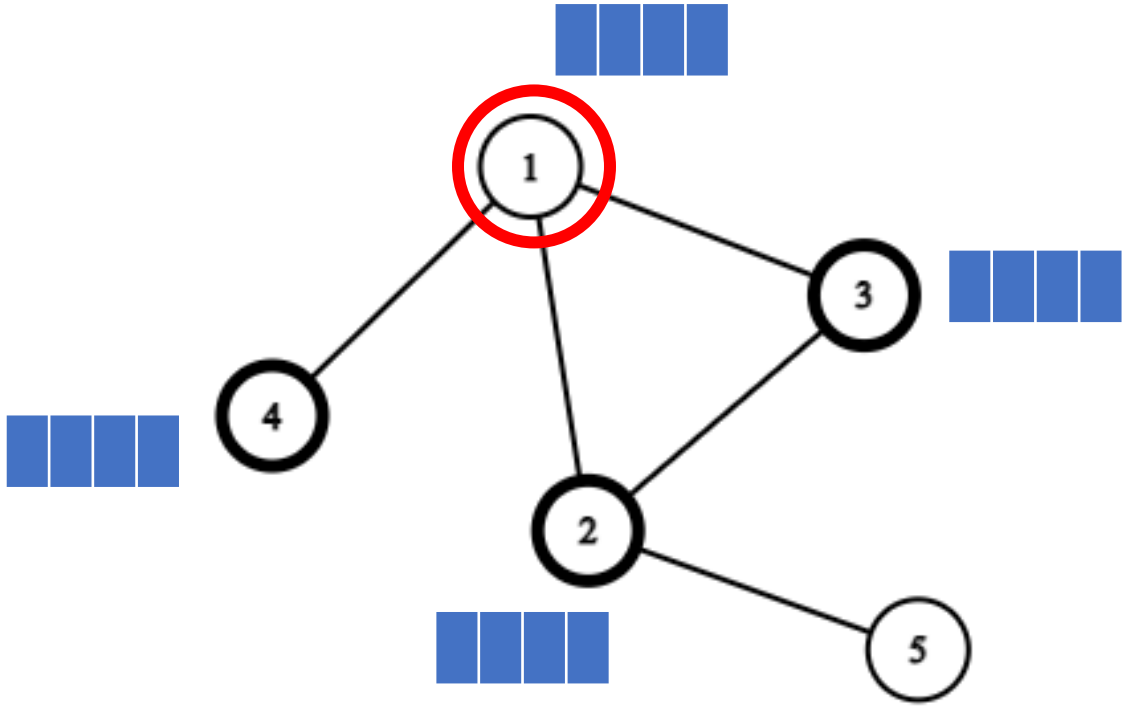
Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

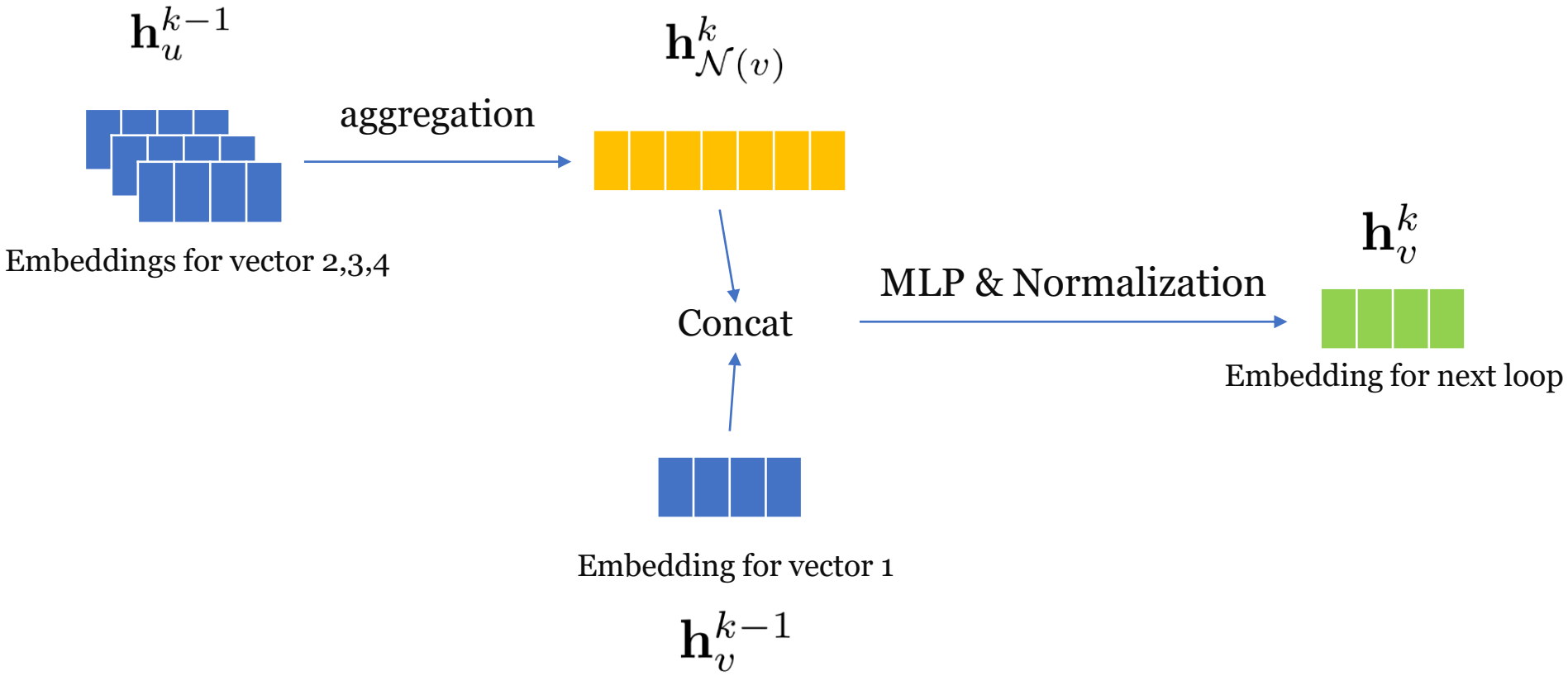
$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

GraphSAGE

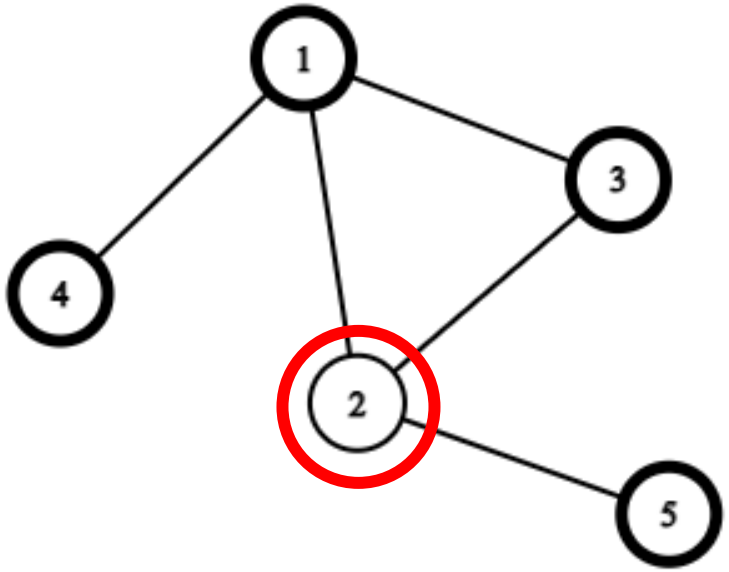


$k = 1, v = 1$

GraphSAGE


$\mathbf{h}_u^{k-1}$
aggregation
$\mathbf{h}_{\mathcal{N}(v)}^{k}$
Embeddings for vector 2,3,4
Concat
MLP & Normalization
$\mathbf{h}_v^{k}$
Embedding for next loop
Embedding for vector 1
$\mathbf{h}_v^{k-1}$

GraphSAGE

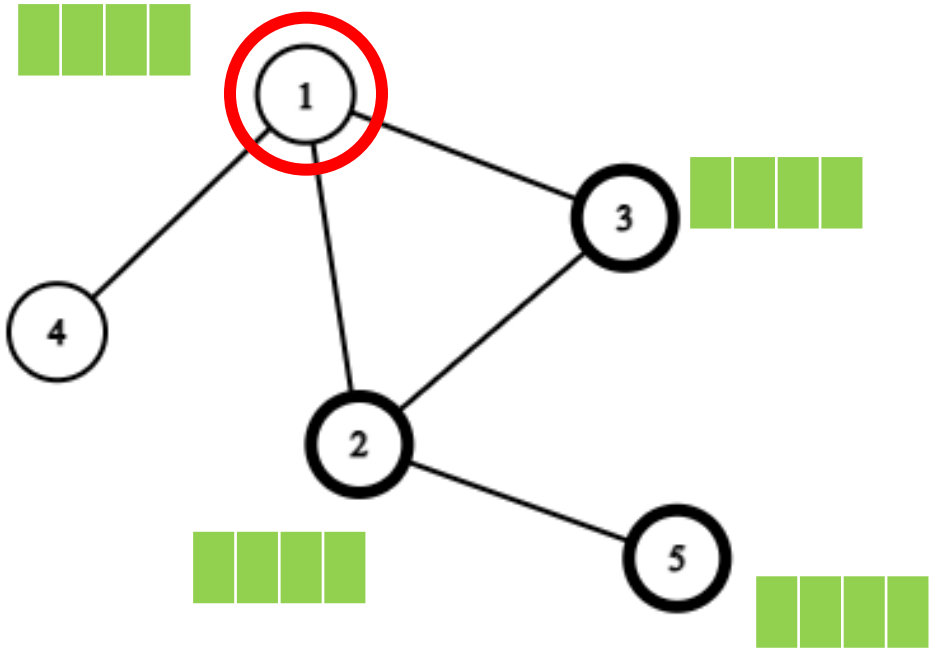


$k = 1, v = 2$

GraphSAGE

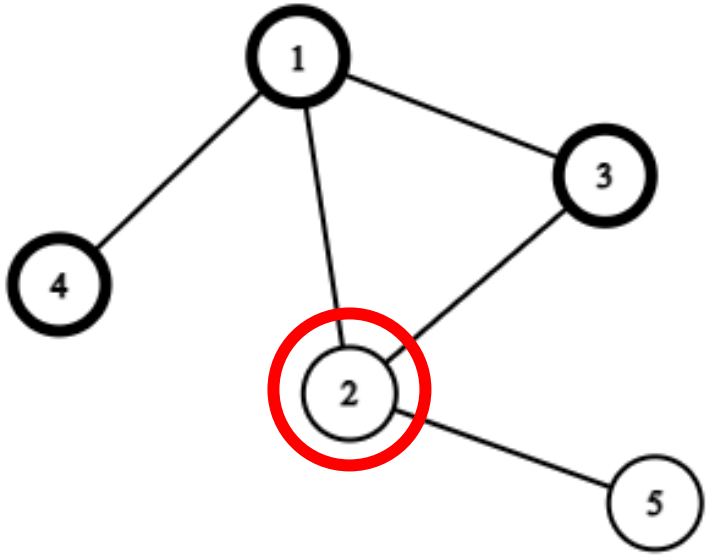
k = 1, v = 3
k = 1, v = 4
k = 1, v = 5

GraphSAGE



$k = 2, v = 1$

GraphSAGE



$k = 2, v = 1$

GraphSAGE

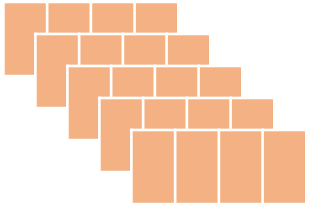
$k = 2, v = 3$
$k = 2, v = 4$
$k = 2, v = 5$

GraphSAGE

Feature vectors for node 1 - 5

Edge List
[1, 1, 1, 2, 3, 2, 4, 5]
[3, 2, 4, 5, 1, 1, 1, 1]

GraphSAGE

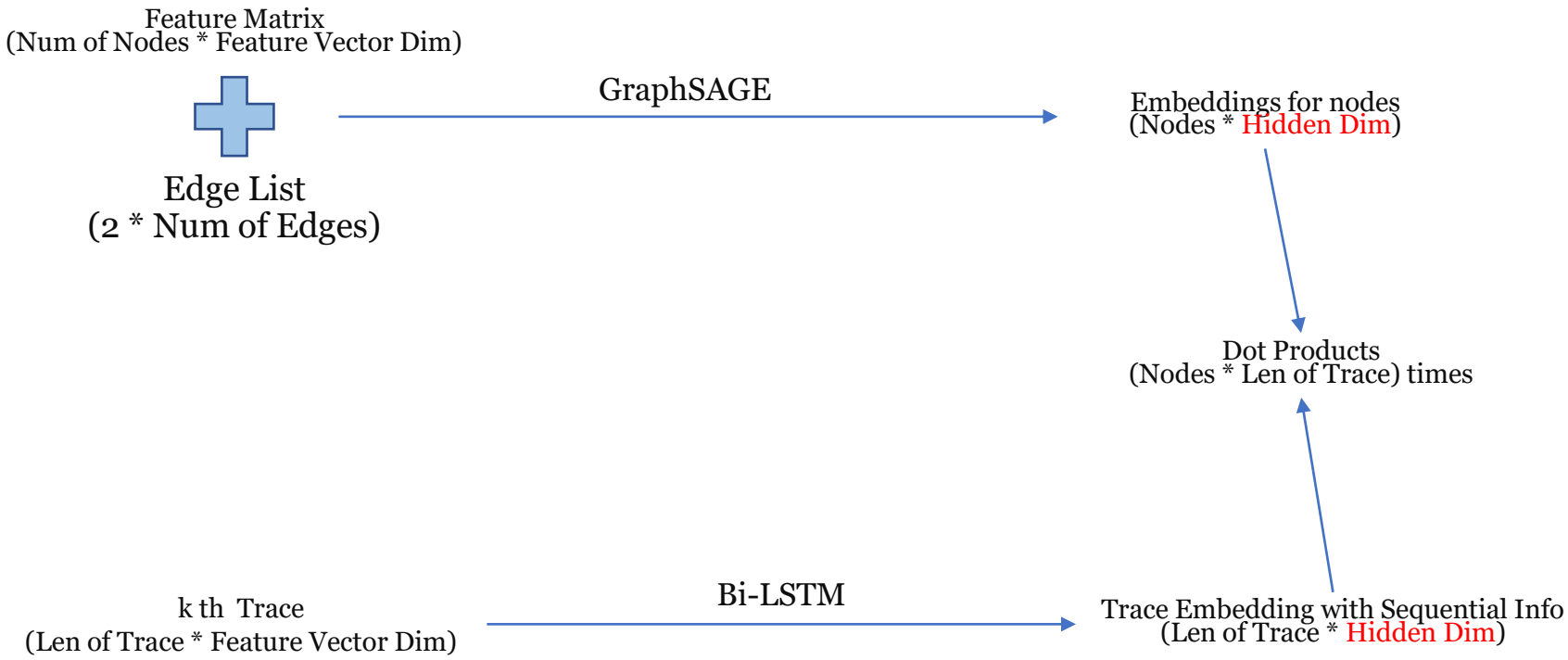


Embeddings for node 1 – 5

GraphSAGE

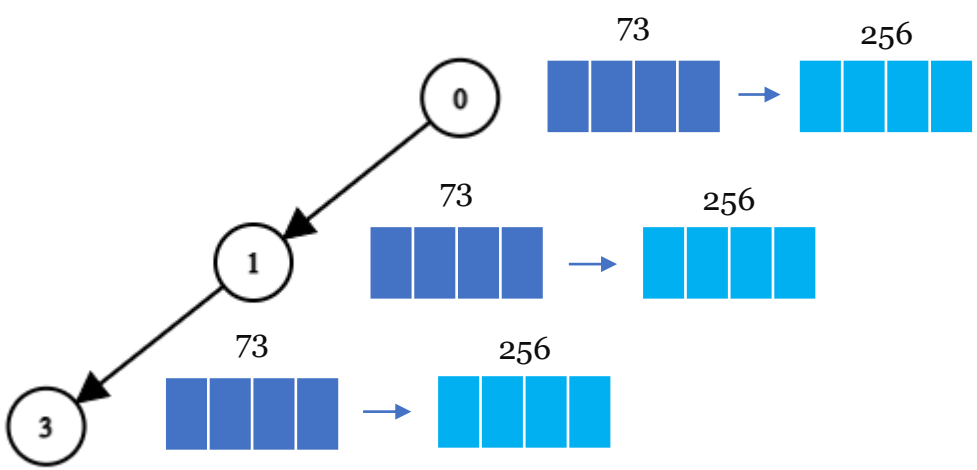
```python
class GraphSAGE(torch.nn.Module):
    def __init__(self, in_channels, hidden_channels, out_channels):
        super(GraphSAGE, self).__init__()
        self.conv1 = SAGEConv(in_channels, hidden_channels)
        self.conv2 = SAGEConv(hidden_channels, out_channels)

    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        hidden_embeddings = x.detach()  # Store the hidden node embeddings
        x = F.dropout(x, p=0.5, training=self.training)
        x = self.conv2(x, edge_index)
        return x, hidden_embeddings
```

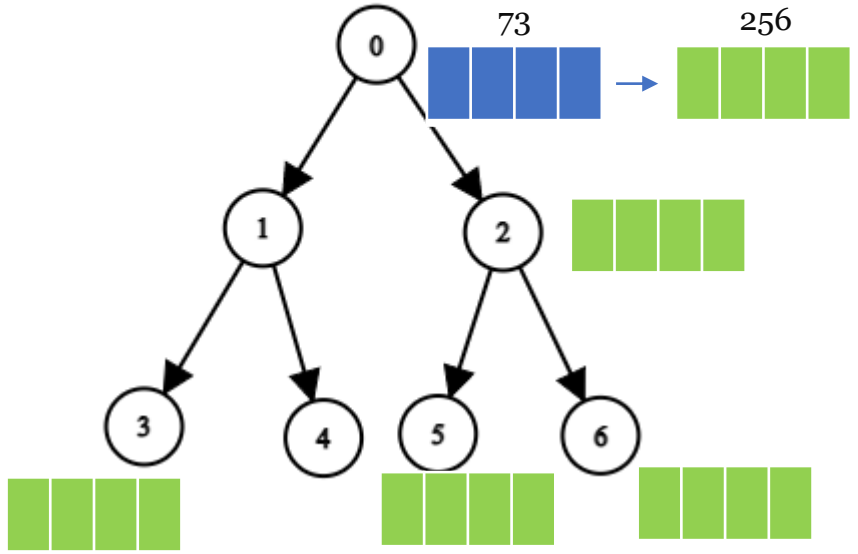
Progress: Model Specification



For k th Training Loop

Progress: Unique Labels





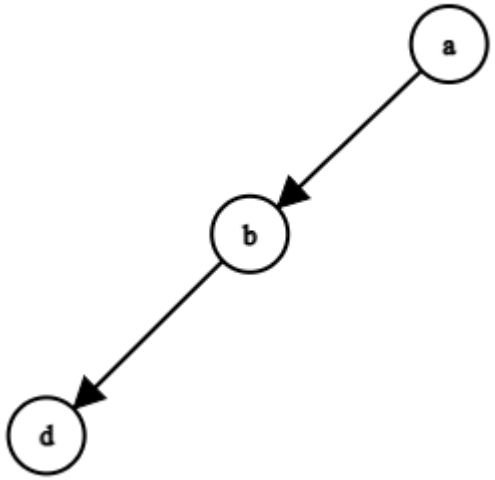
100 Accuracy Limit: ~ 64 traces, ~ 73 nodes,
fanout: 8
(~infinite traces, ~infinite nodes)

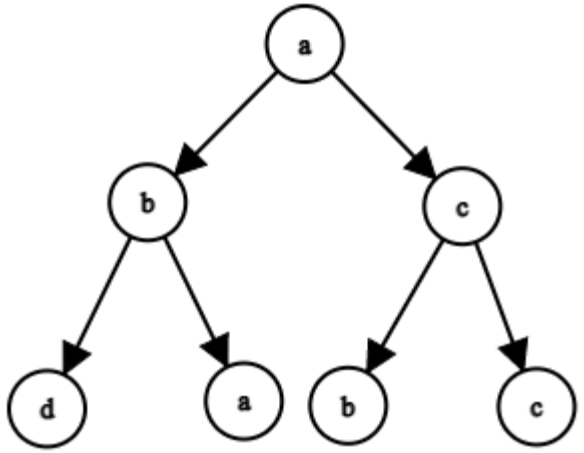
Progress: Unique Labels

Hidden Dim	Traces	Nodes	Fanout
256	**64**	**73**	**8**
100	**49**	**57**	**7**
64	**36**	**43**	**6**
48	**25**	**31**	**5**

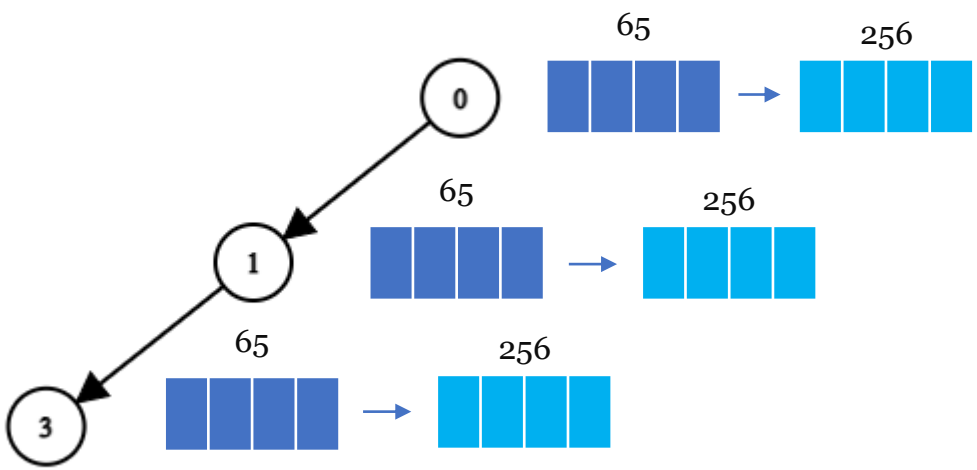
Hidden Dim Req. for 100% Acc

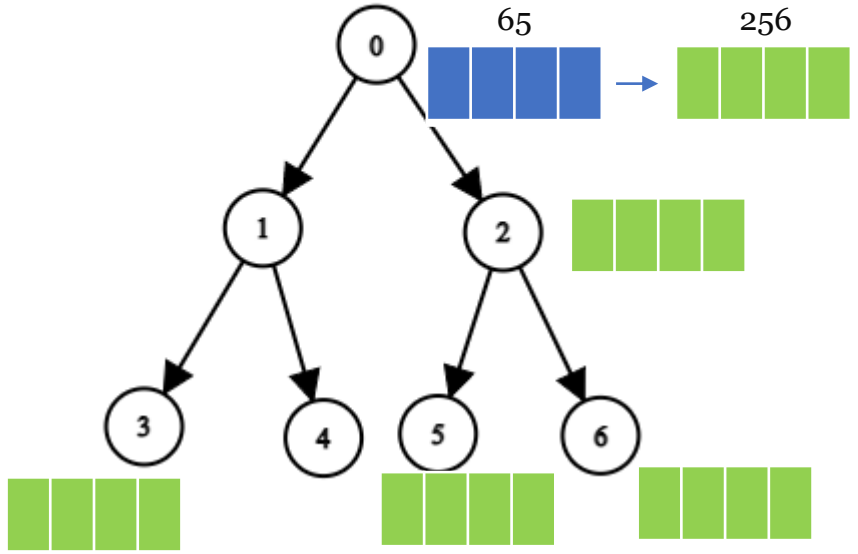
Progress: Non-Unique





Progress: Non-Unique



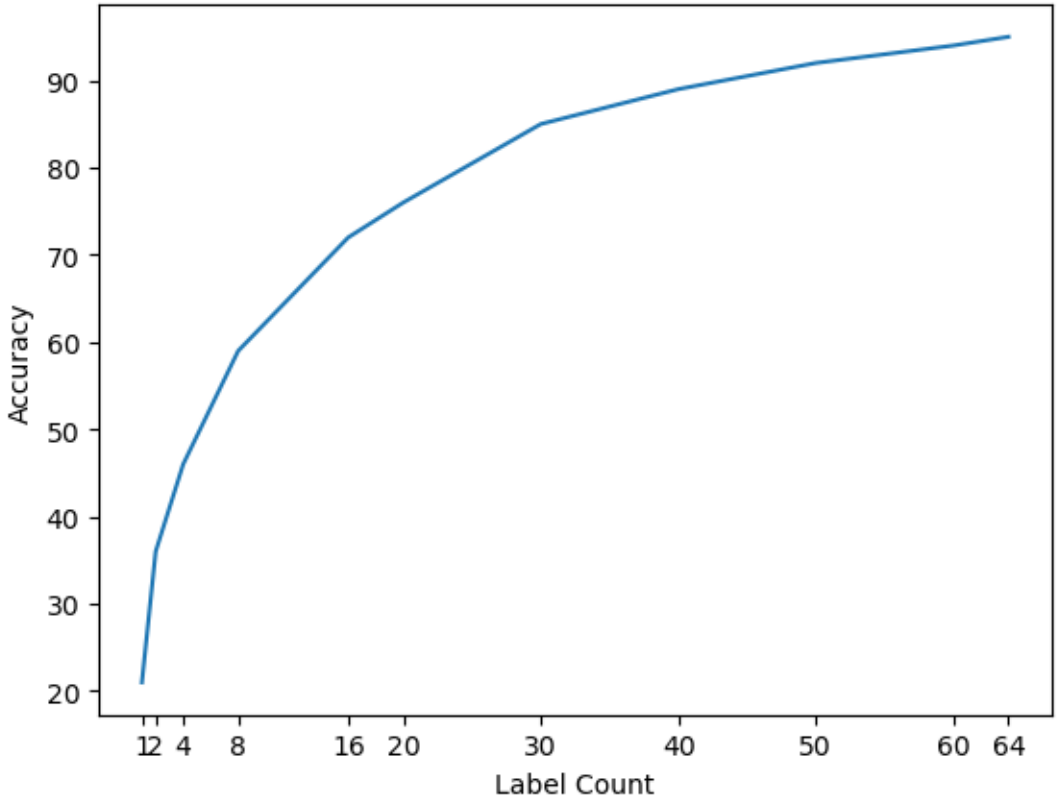


100 Accuracy Limit: ~121 traces, ~ 133 nodes,
fanout: 11, labels = nodes / 2
(~infinite traces, ~infinite nodes)

Progress: Non-Unique

Label Count	Traces	Nodes	Fanout	Accuracy
11	100	111	10	100
10	100	111	10	88
9	100	111	10	80

Progress: Non-Unique



Traces: 64, Nodes 127, Fanout 2