Inductive Representation Learning on Large Graphs

William L. Hamilton*, Rex Ying*, Jure Leskovec
(with many thanks to Dan Jurafsky, Alex Ratner, and Bryan He)

STANFORD INFOLAB

Inductive Representation Learning on Large Graphs

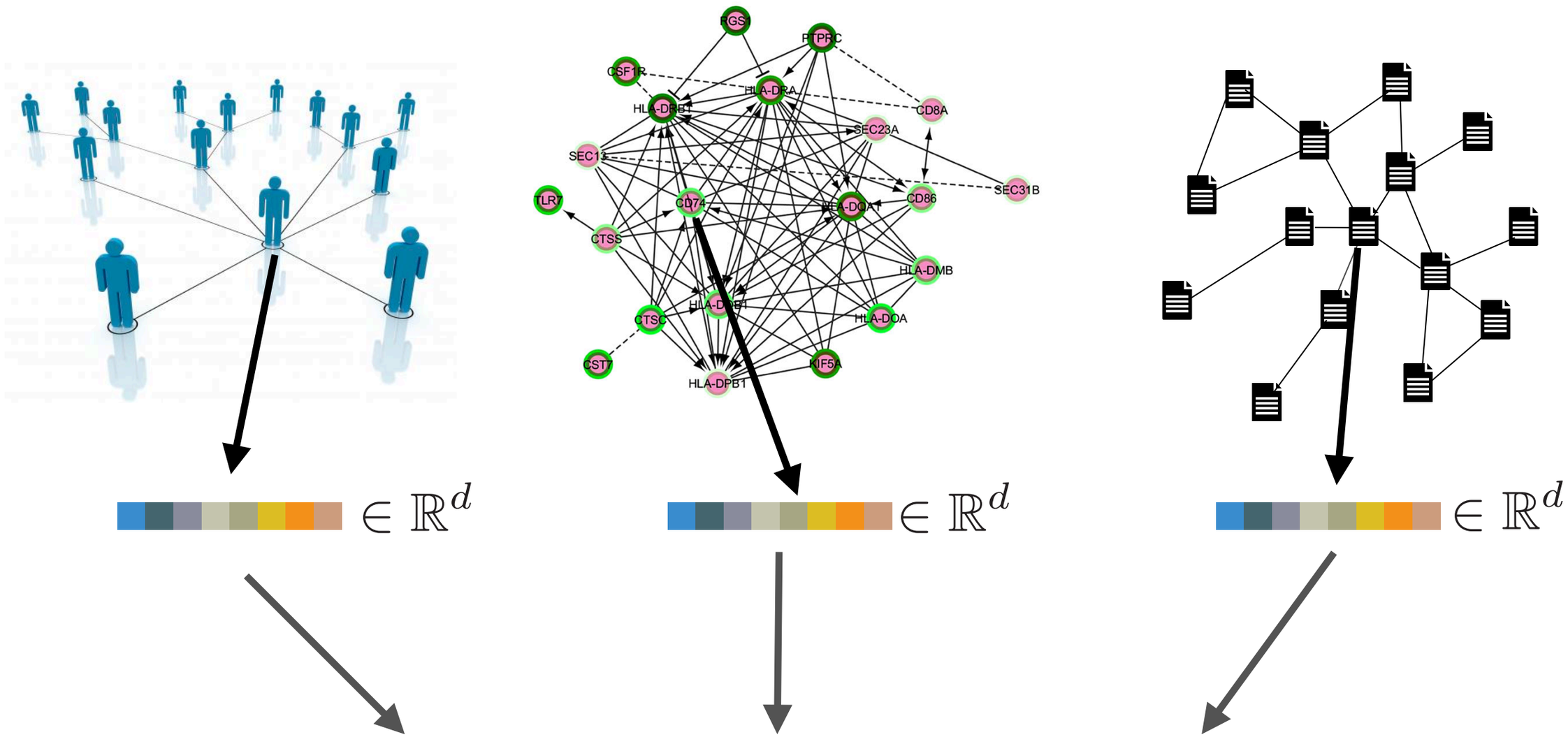
William L. Hamilton*, Rex Ying*, Jure Leskovec
(with many thanks to Dan Jurafsky, Alex Ratner, and Bryan He)

https://github.com/williamleif/GraphSAGE

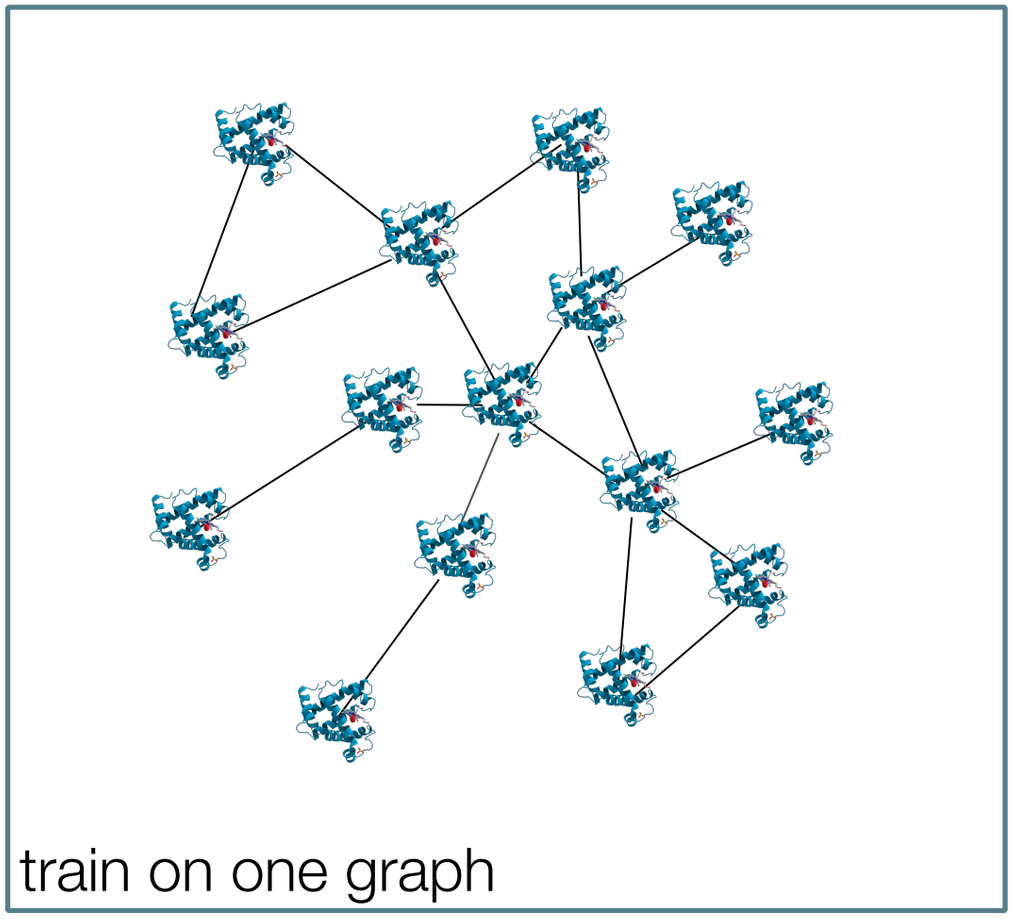
STANFORD INFOLAB

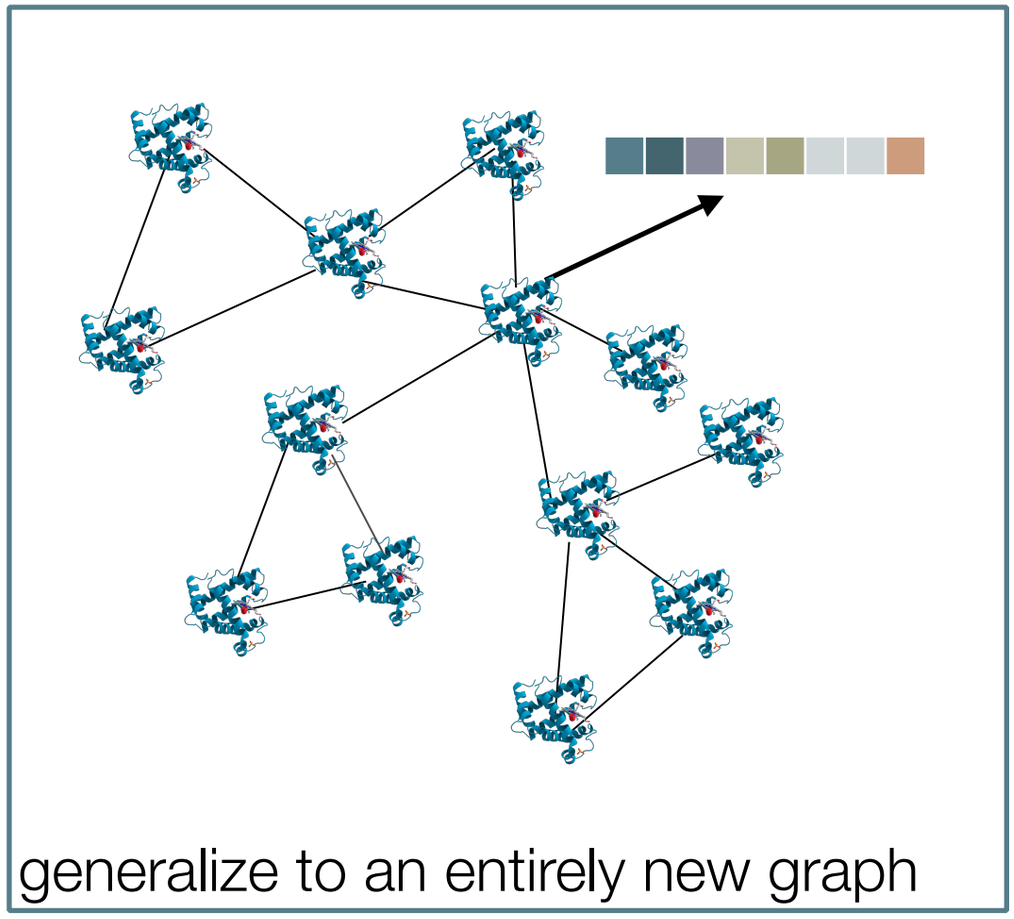
Stanford University S NLP Natural Language Processing

Representation learning on graphs



Embed nodes **inductively** (across graphs)

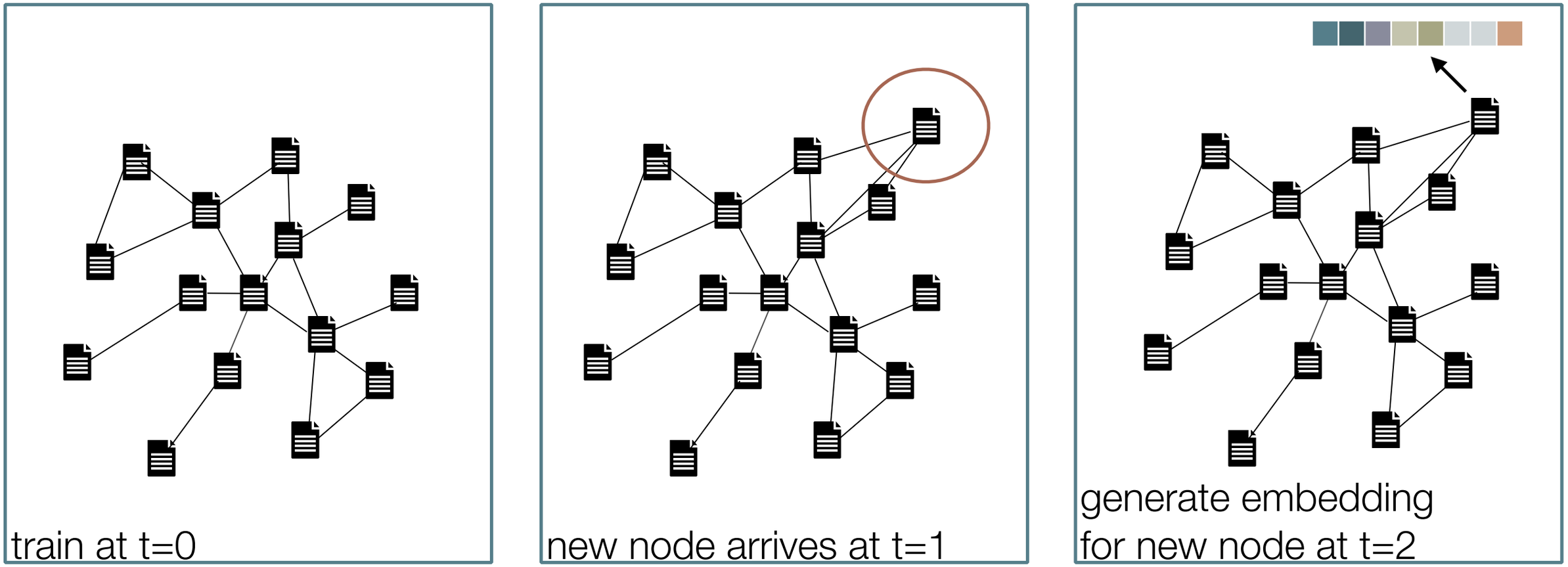




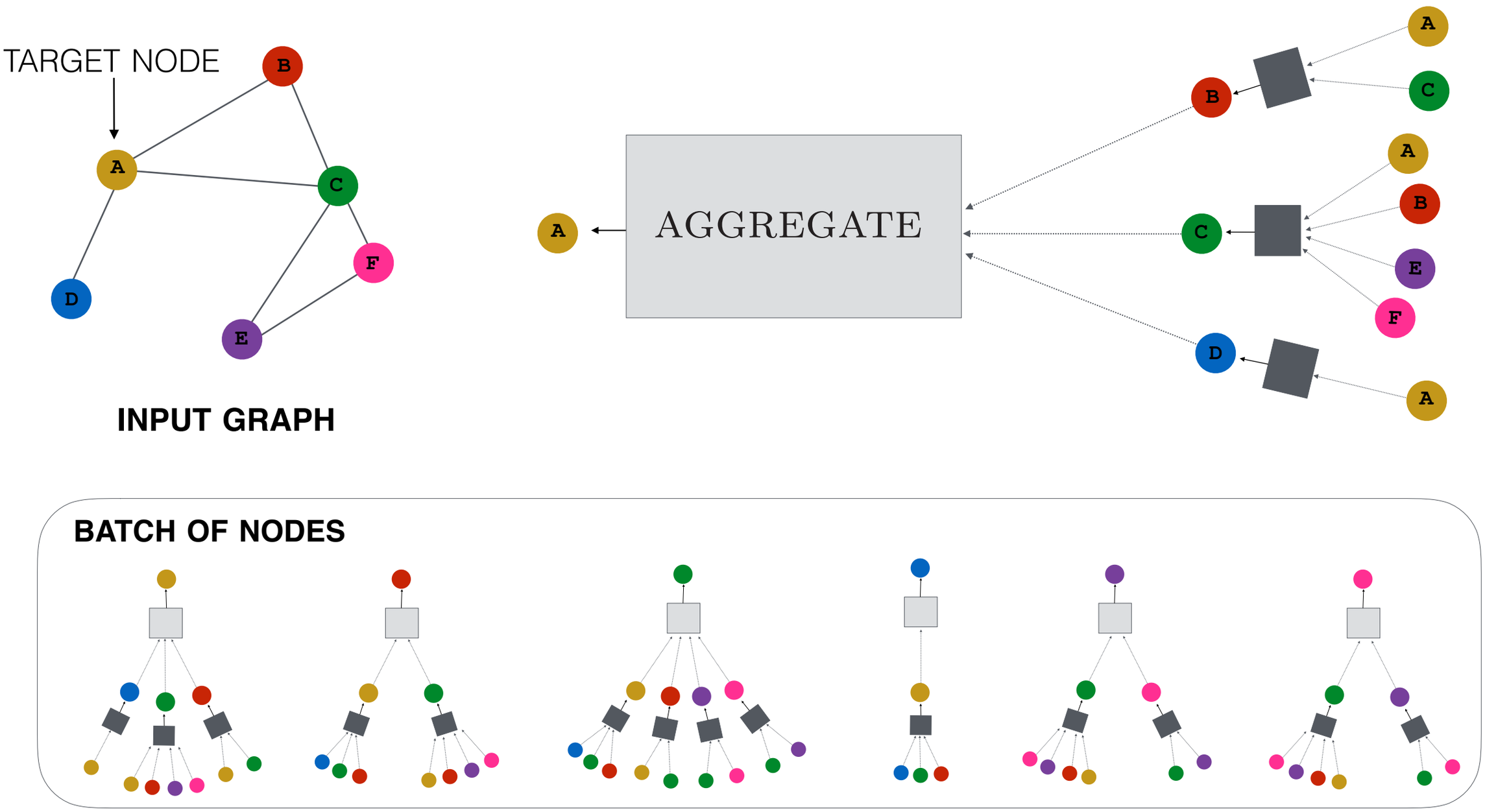
Inductive node embedding → generalize to entirely unseen graphs

e.g., train on protein interaction graph from model organism A and generate embeddings on newly collected data about organism B

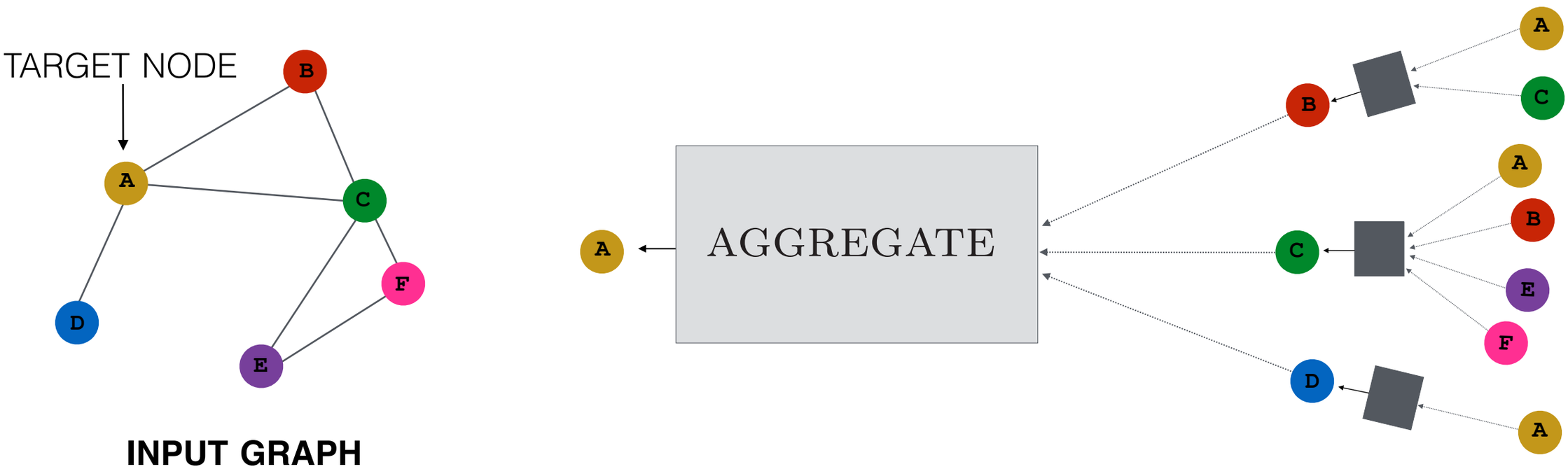
Embed nodes **inductively** (in evolving graphs)



GraphSAGE: The basic idea



GraphSAGE: Neighborhood aggregation



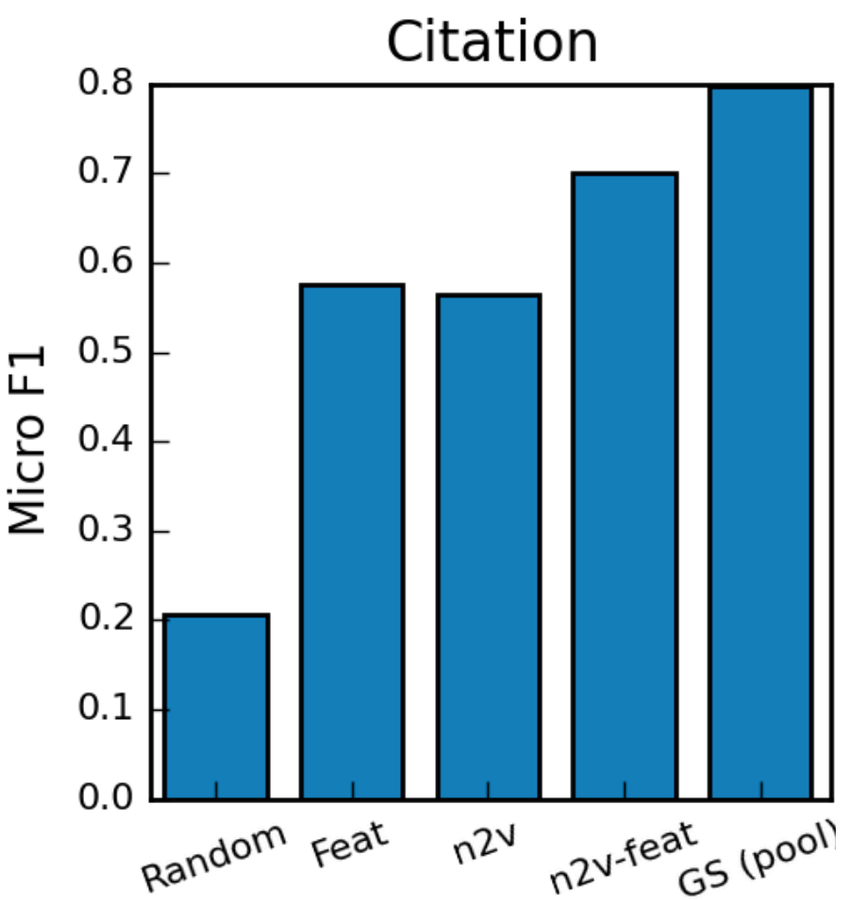
Aggregator options:

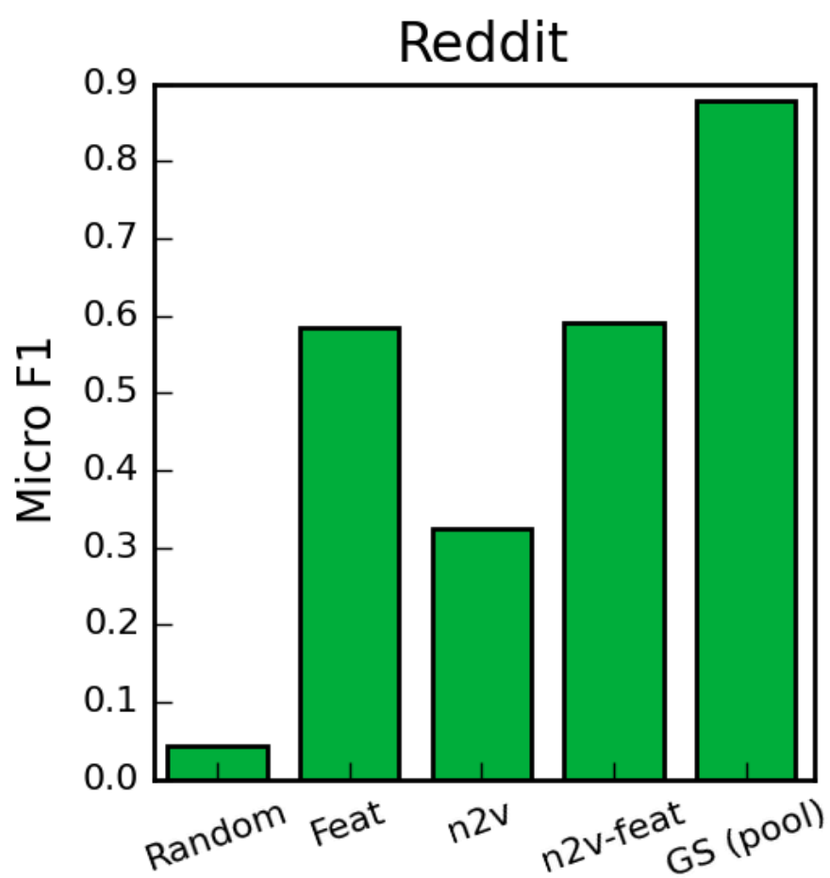
mean: simple, element-wise mean

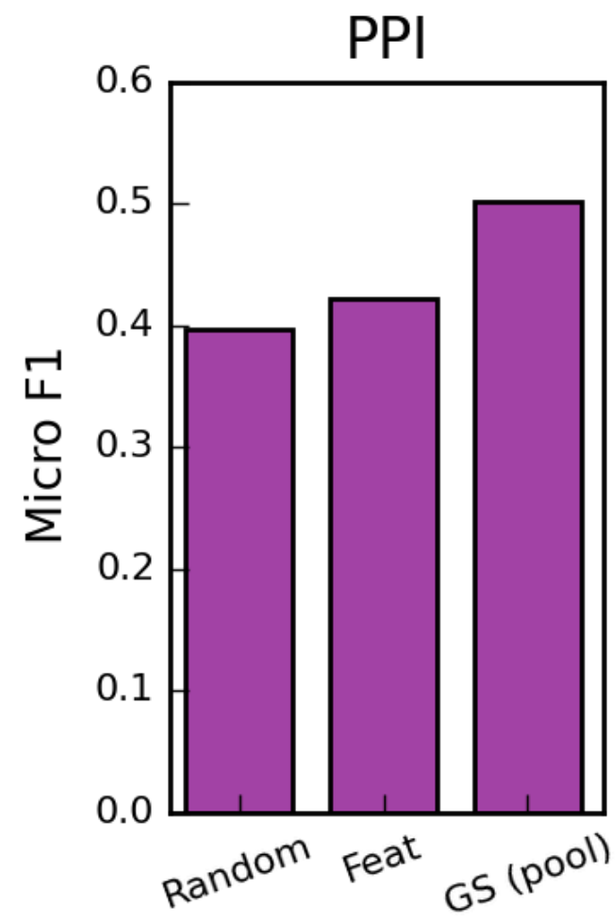
LSTM: apply LSTM to random permutation of the points

max-pooling: $\max(\{\sigma(\mathbf{W}\mathbf{h}_u + \mathbf{b}), \forall u \in \mathcal{N}(v)\})$

Performance: Improvement over baselines

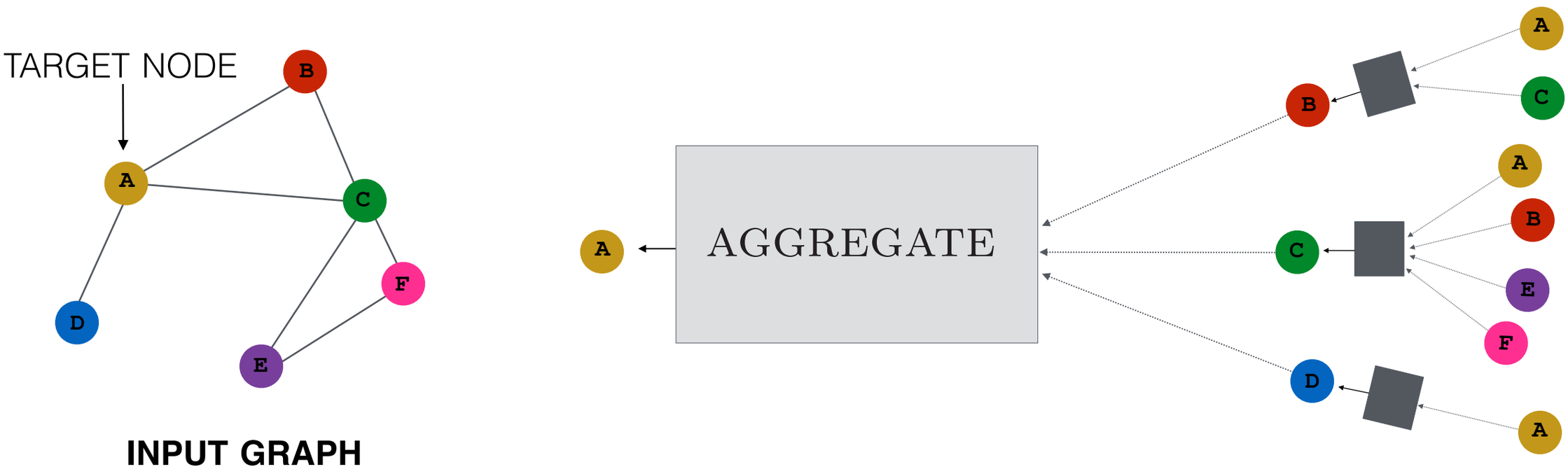






~40% average improvement over raw features.

GraphSAGE: Neighborhood aggregation



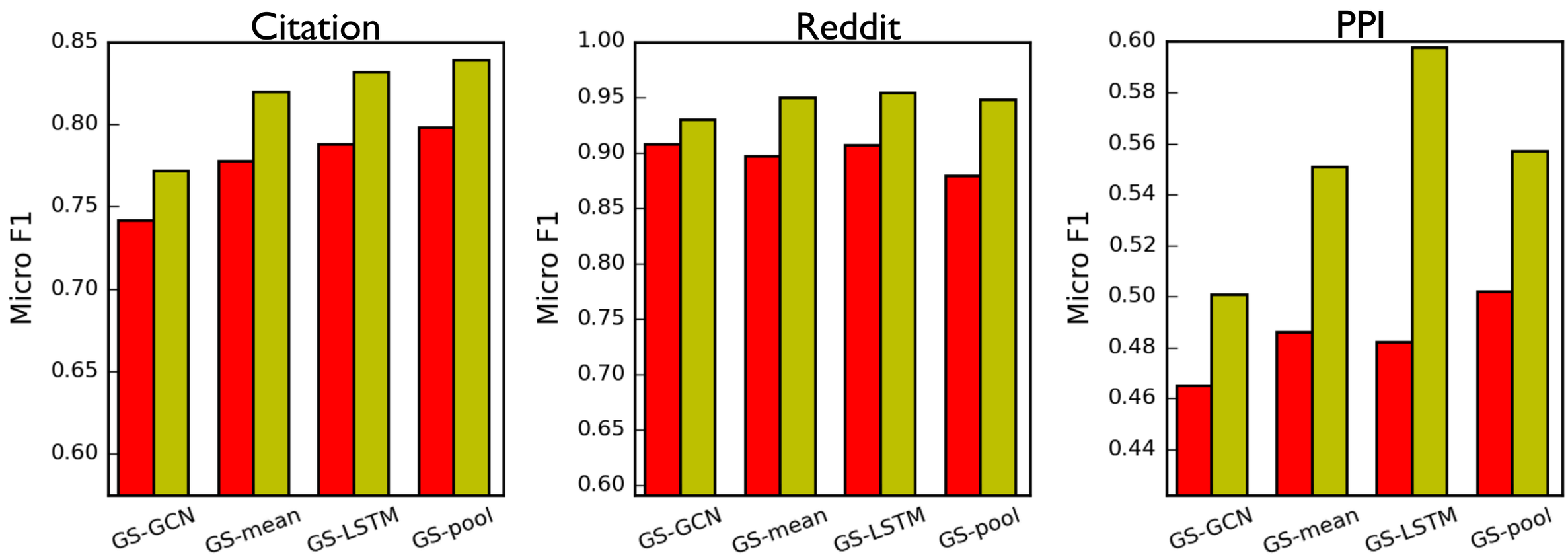
Aggregator options:

mean: simple, element-wise mean

LSTM: apply LSTM to random permutation of the points

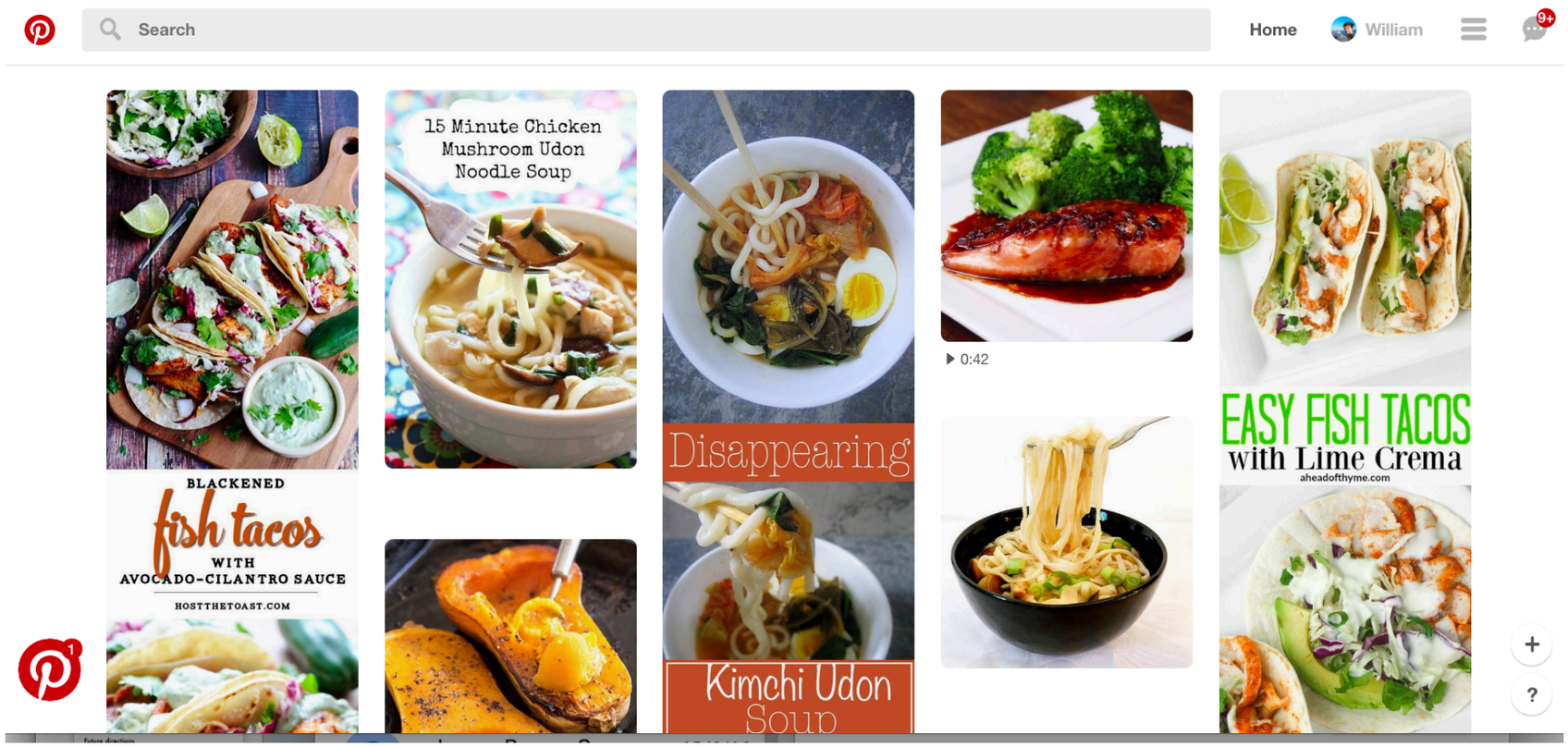
max-pooling: $\max(\{\sigma(\mathbf{W}\mathbf{h}_u + \mathbf{b}), \forall u \in \mathcal{N}(v)\})$

Performance: Aggregator comparison



On average, the trainable LSTM/pool aggregators give a gain of ~6.5% compared to the GCN-based aggregator.

Performance: Scalable to massive graphs

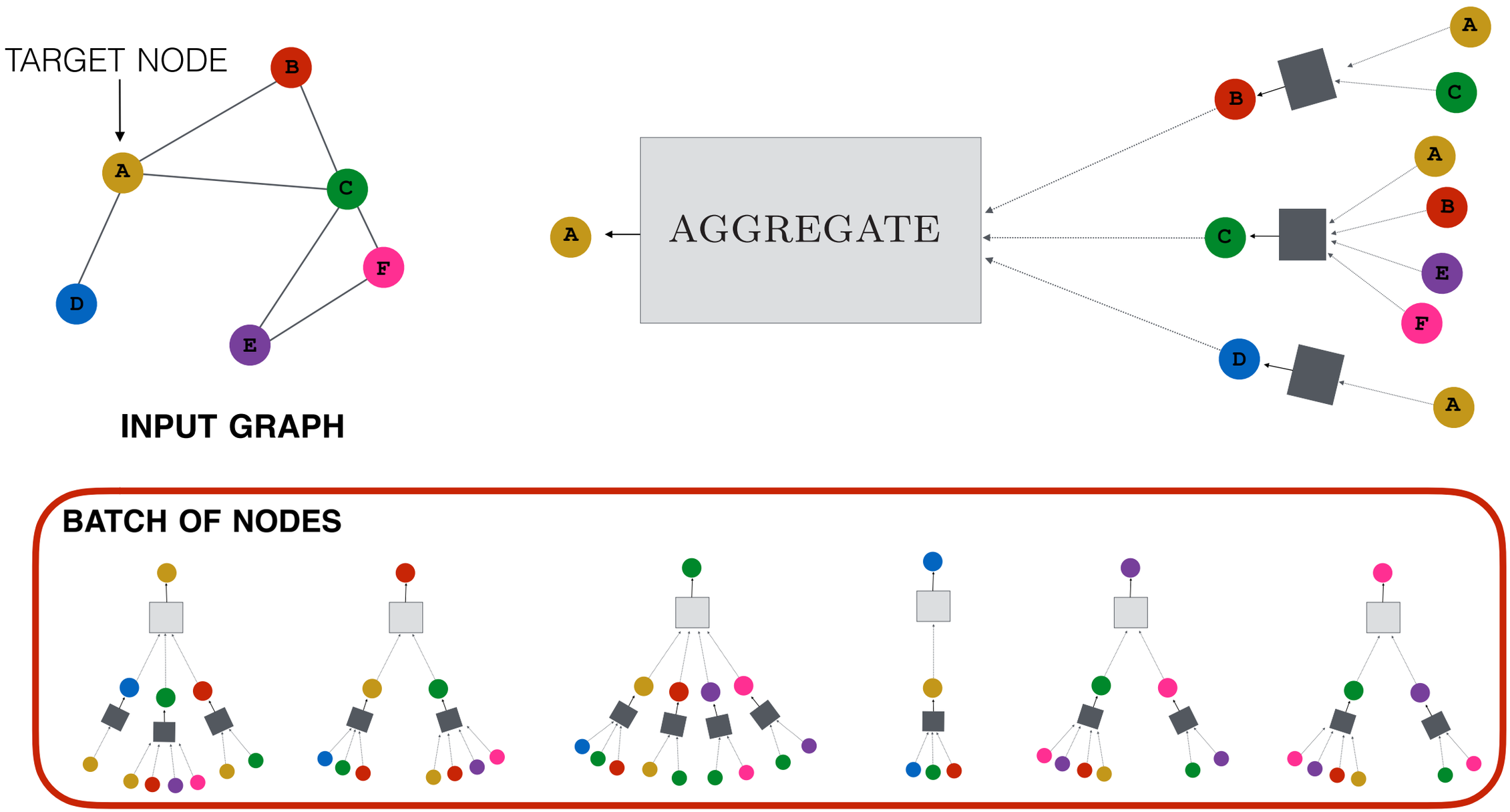


Pinterest

Task: Retrieve content that users are likely to “pin” (prior to ranking). **1+ billion possible candidates**.

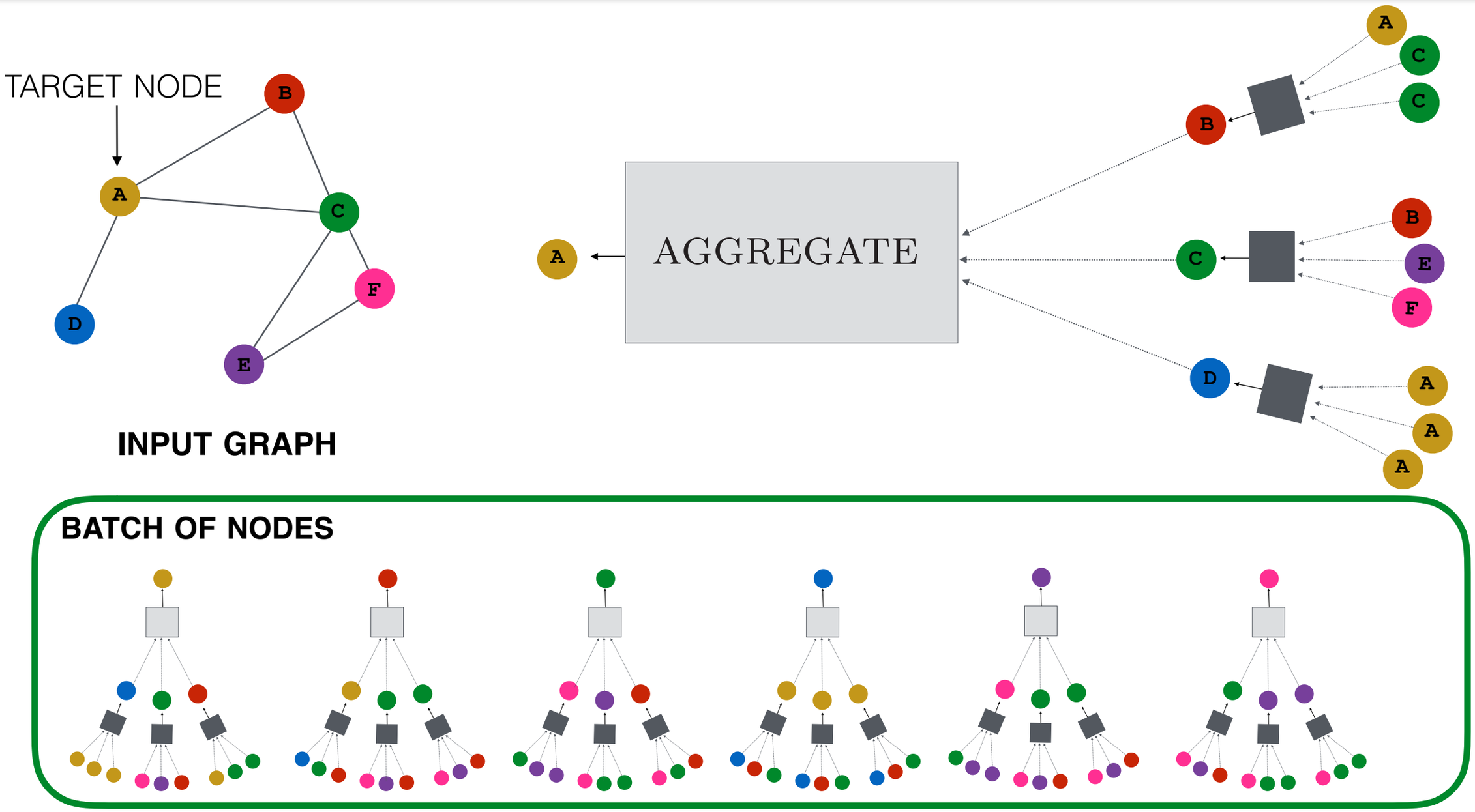
Results: GraphSAGE gives 350% improvement over PageRank-like baseline (36% vs. 11% prob@k).

Implementation challenge: Dynamic batch sizes



Every node has unique compute graph. Can't batch on GPU!

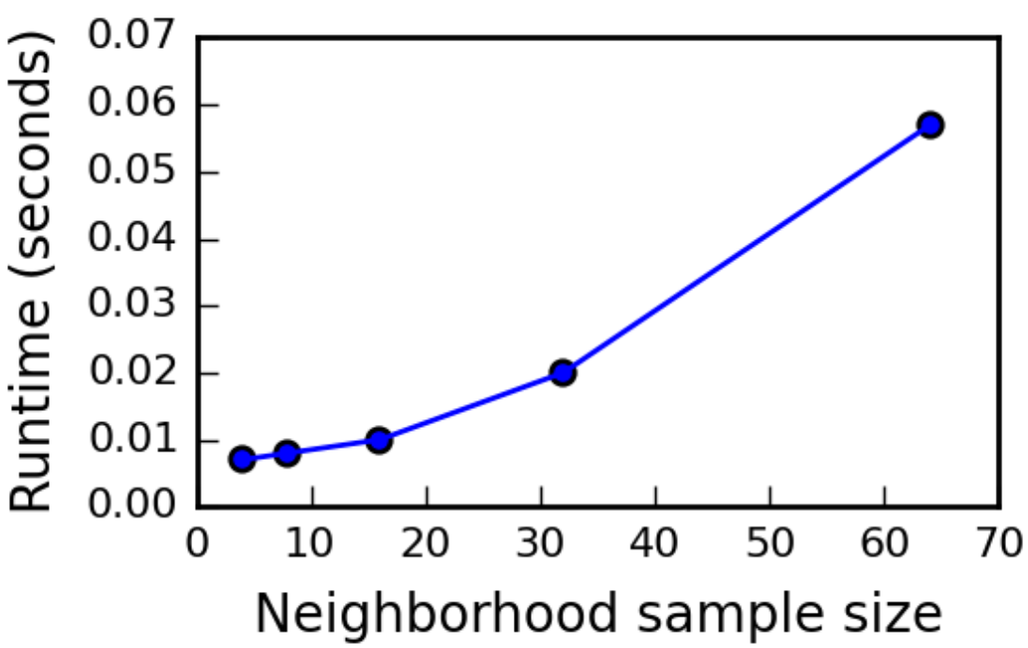
Solution: Sample fixed-sized neighborhoods

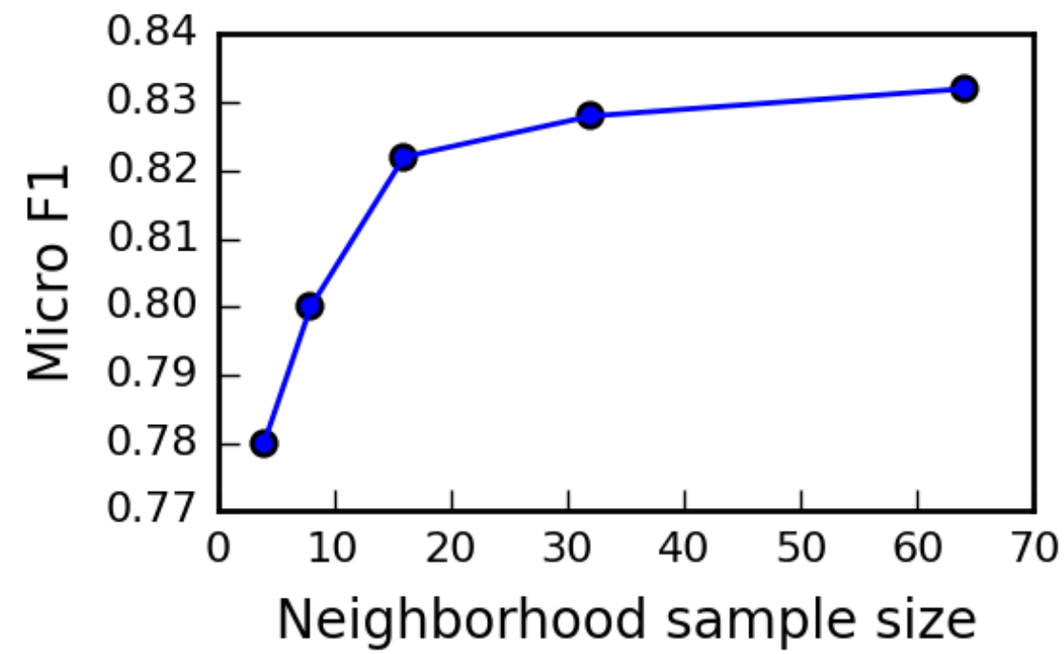


Compute graphs have same structure = efficient GPU batching

Performance: Impact of sampling

Diminishing returns for sampling larger neighborhoods





(Results on citation data using GraphSAGE-mean)

Implementation sketch: Overview

Algorithm 2: GraphSAGE minibatch forward propagation algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$;
input features $\{\mathbf{x}_v, \forall v \in \mathcal{B}\}$;
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 sampling functions, $\mathcal{N}_k : v \rightarrow 2^{\mathcal{V}}, \forall k \in \{1, ..., K\}$

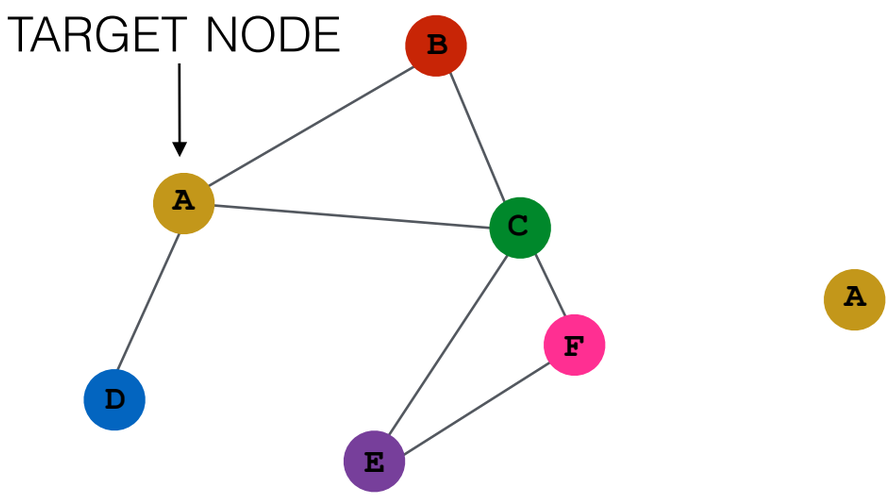
Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{B}$

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

???

TensorFlow

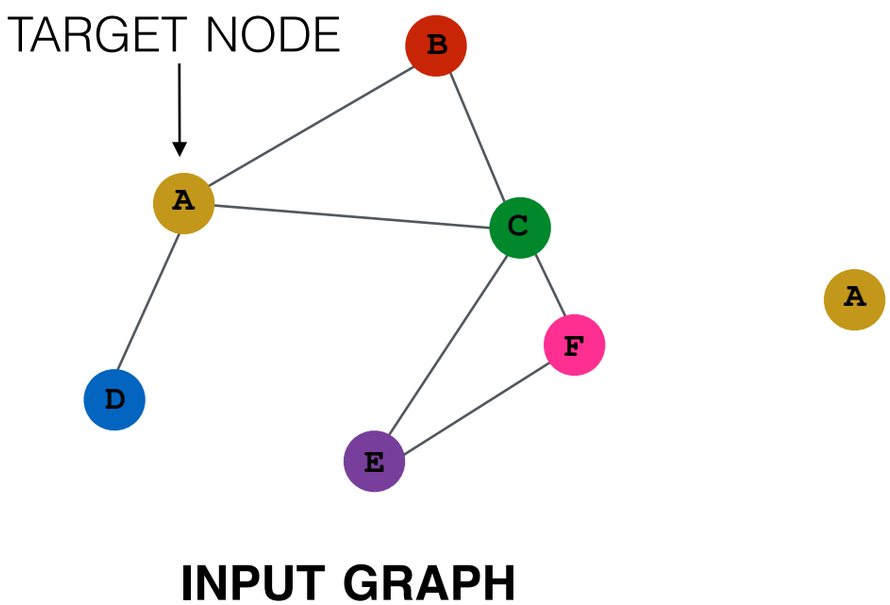
Implementation sketch: Overview



INPUT GRAPH

[redacted]

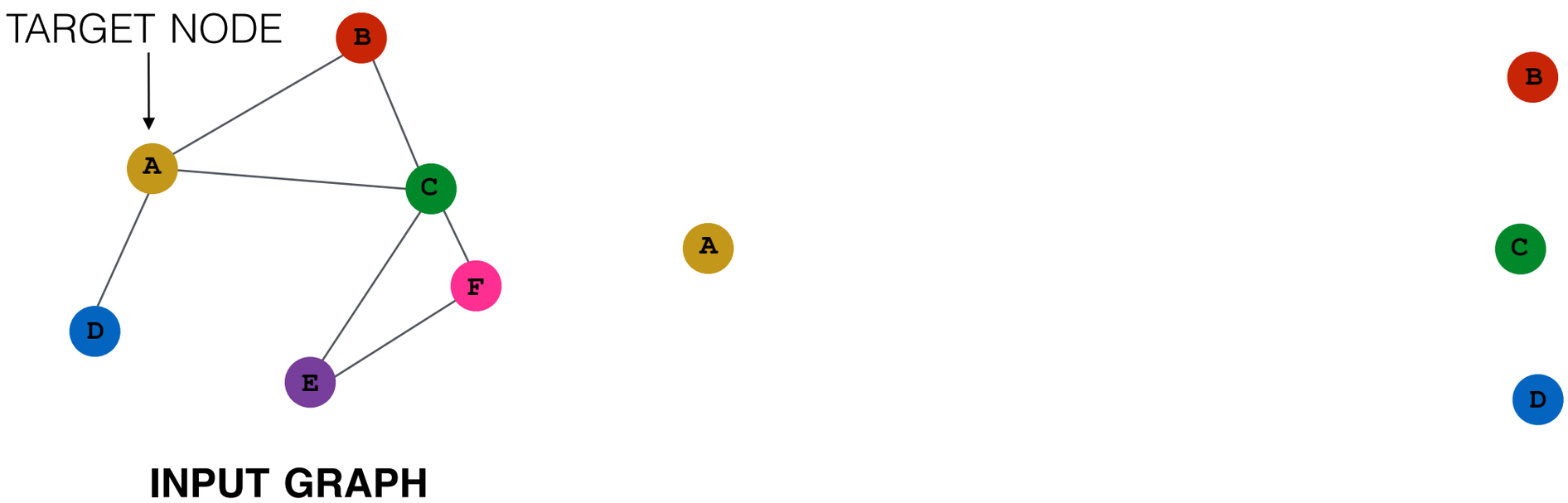
Implementation sketch: Overview




```
samples, support_sizes = sample(input_nodes, …)
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254

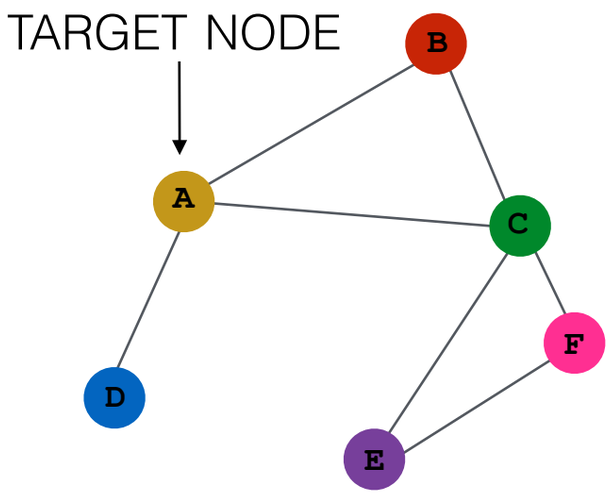
Implementation sketch: Overview




```
samples, support_sizes = sample(input_nodes, …)
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254

Implementation sketch: Overview

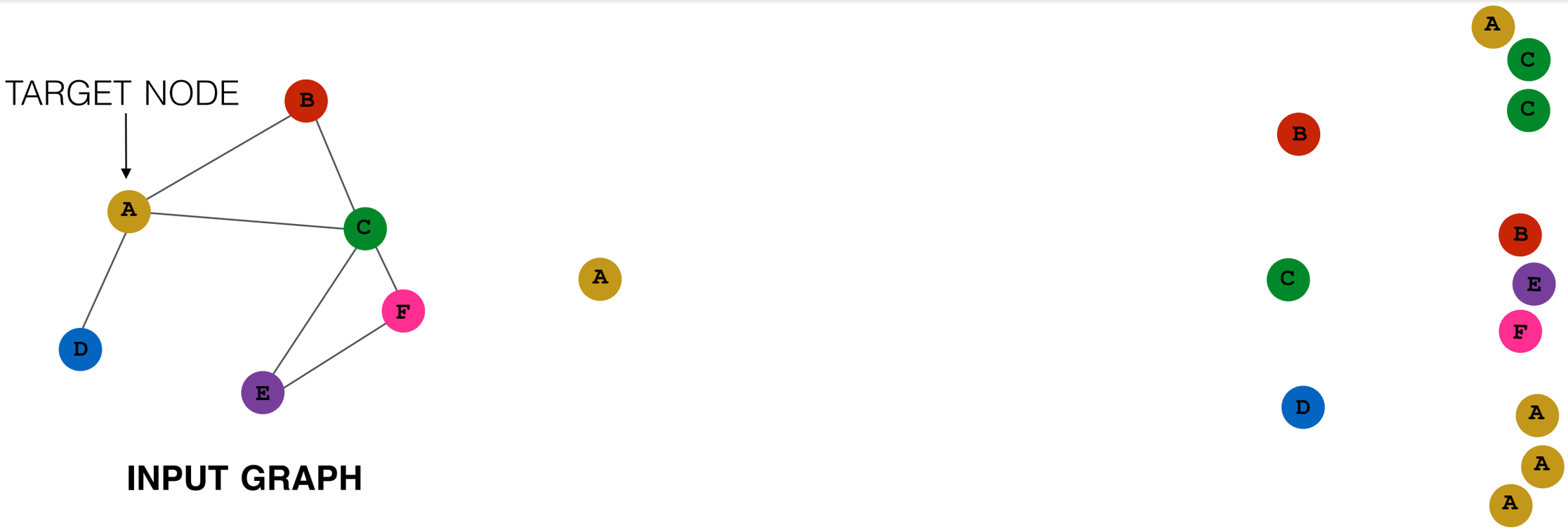


INPUT GRAPH


```
samples, support_sizes = sample(input_nodes, …)
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254

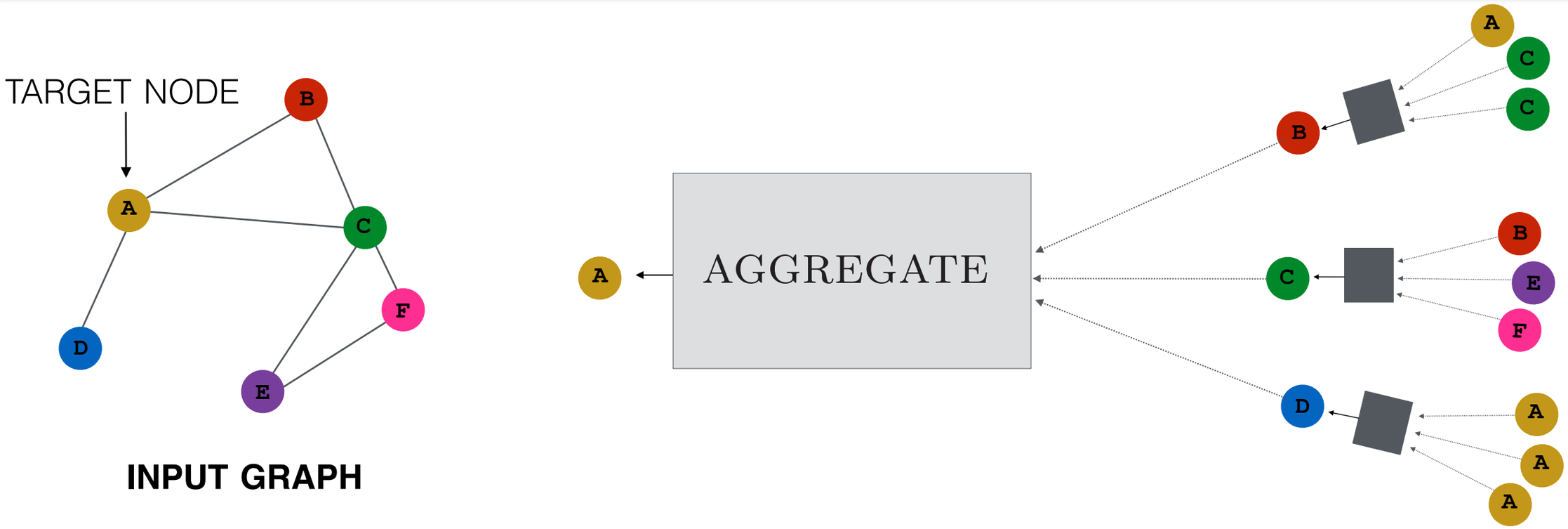
Implementation sketch: Overview




```
samples, support_sizes = sample(input_nodes, …)
```
https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254

```
embeddings = aggregate(samples, …)
```
https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278

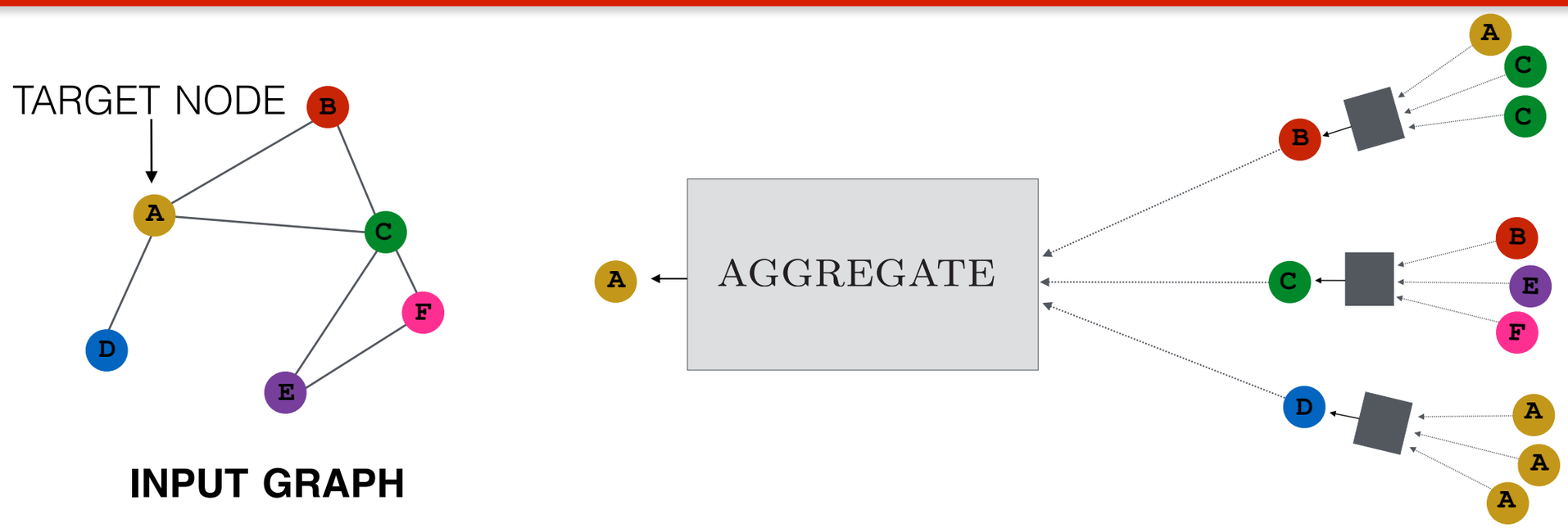
Implementation sketch: Overview




```
samples, support_sizes = sample(input_nodes, …)
```
https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254

```
embeddings = aggregate(samples, …)
```
https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278

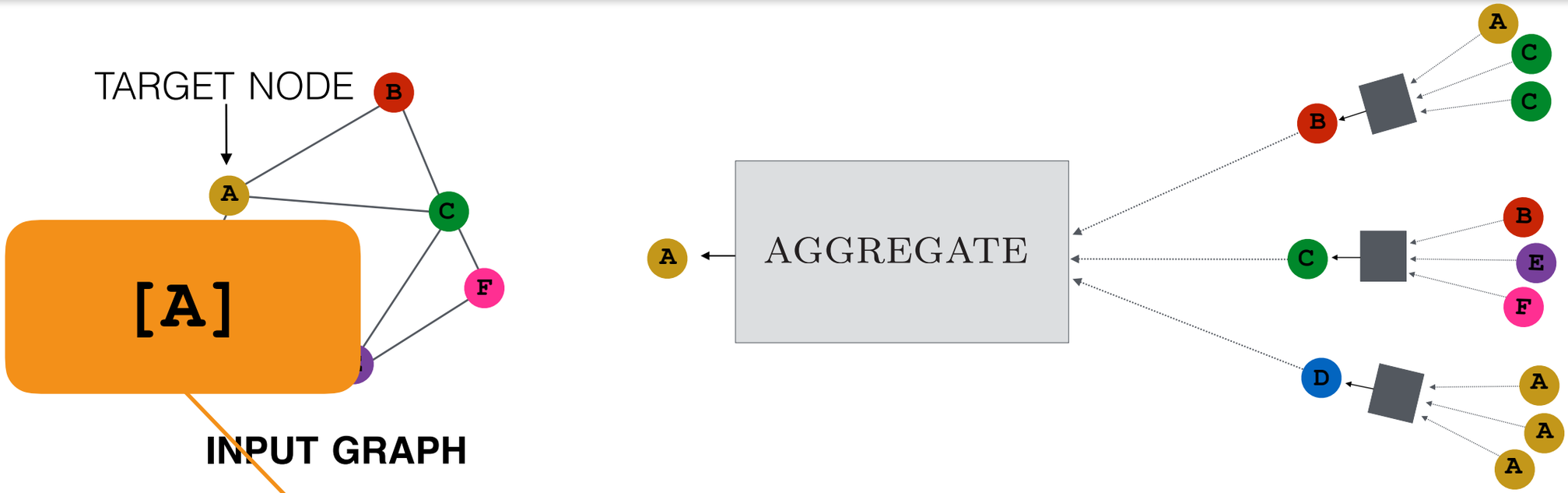
Implementation sketch: Sample stage




```
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1) # shuffle columns
        samples = neighs[:, :num_samples].flatten()
   return samples, support_sizes
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254

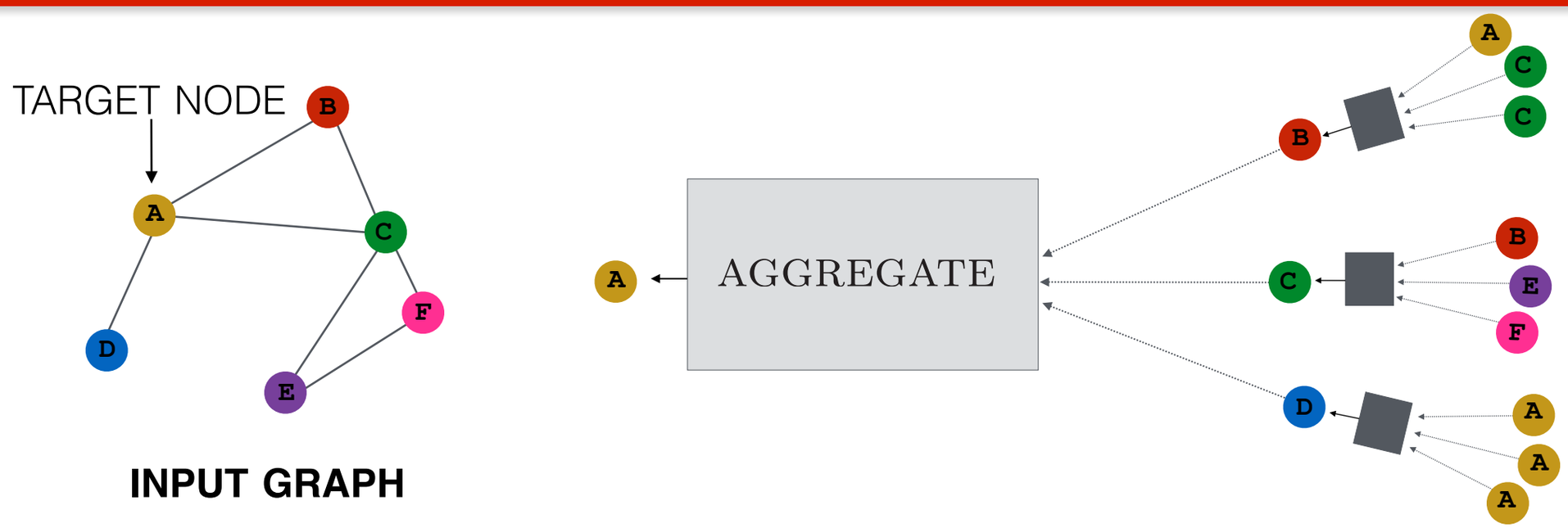
Implementation sketch: Sample stage




```
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1) # shuffle columns
        samples = neighs[:, :num_samples].flatten()
   return samples, support_sizes
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254

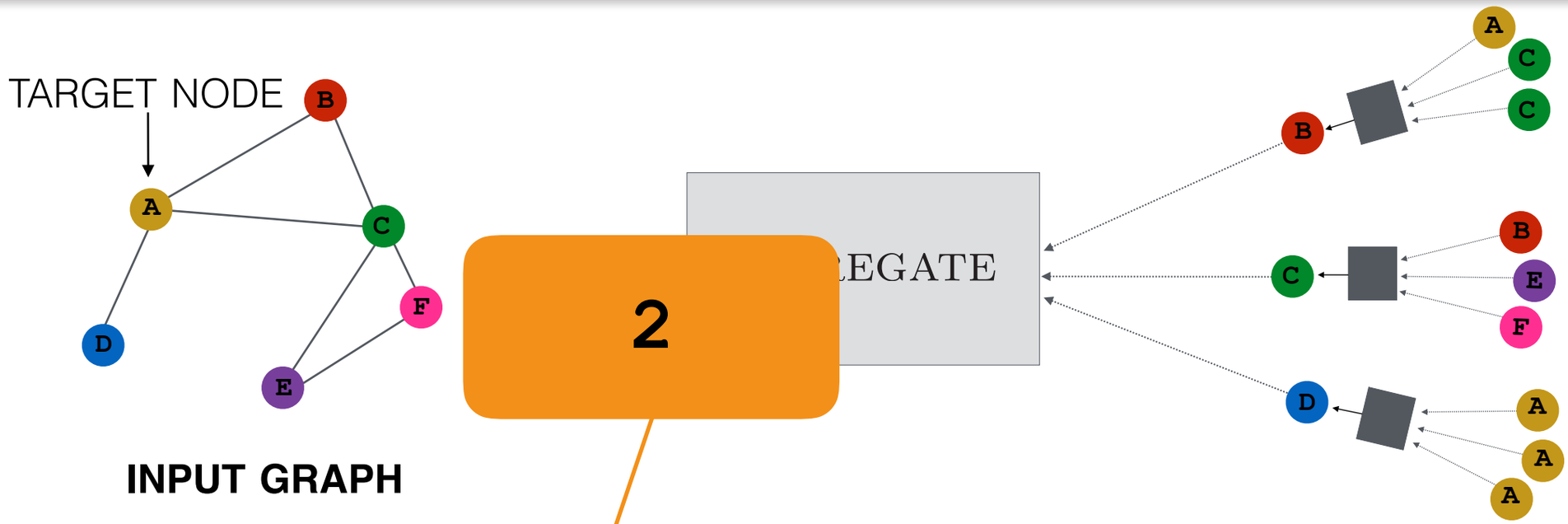
Implementation sketch: Sample stage




```
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1) # shuffle columns
        samples = neighs[:, :num_samples].flatten()
   return samples, support_sizes
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254

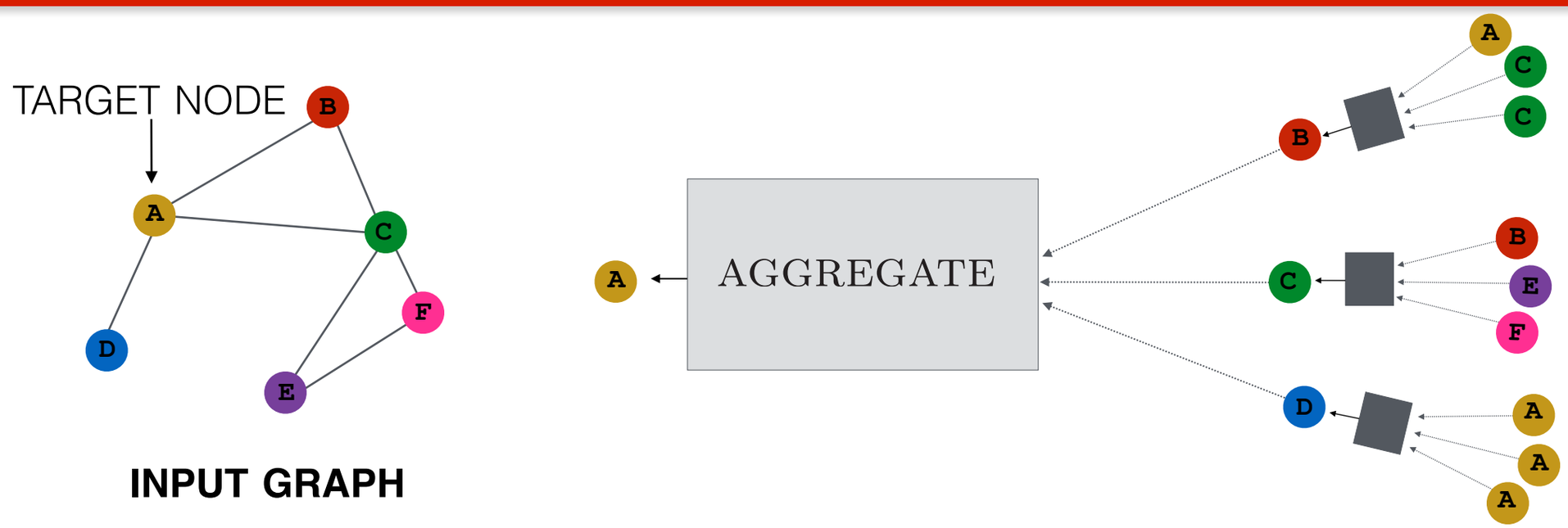
Implementation sketch: Sample stage




```
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1) # shuffle columns
        samples = neighs[:, :num_samples].flatten()
   return samples, support_sizes
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254

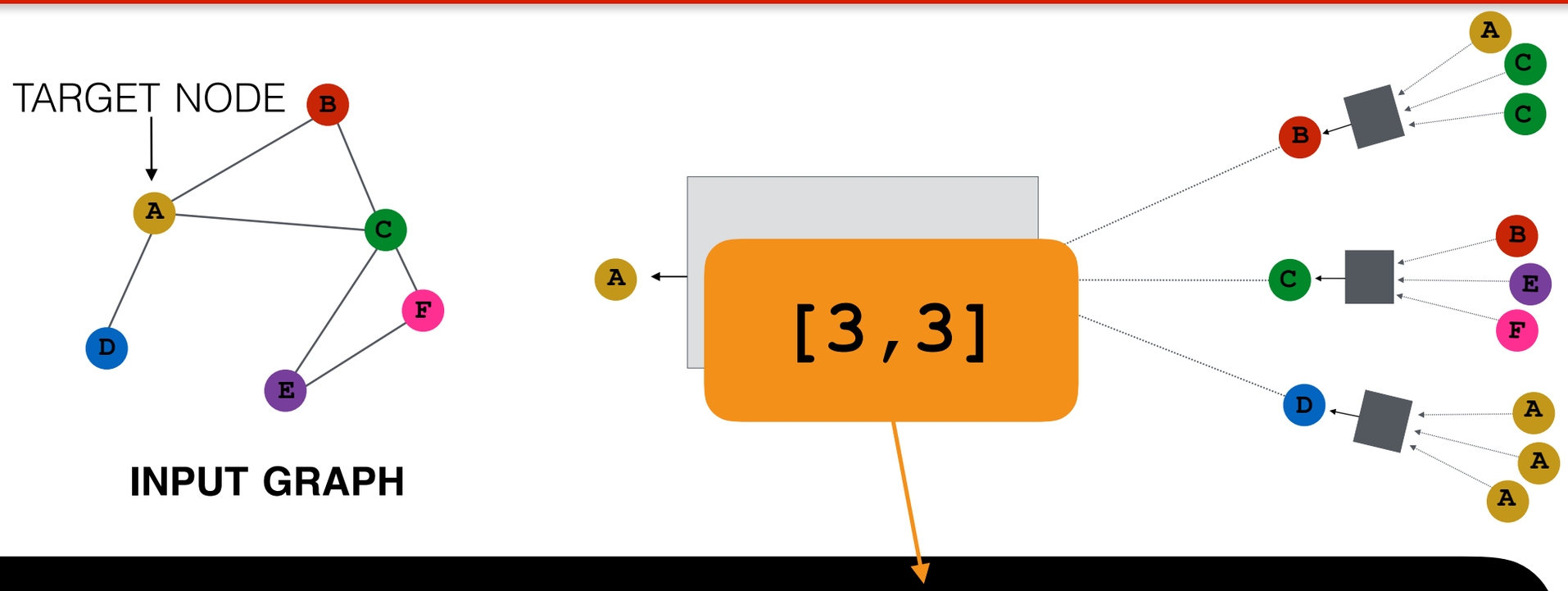
Implementation sketch: Sample stage




```
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1) # shuffle columns
        samples = neighs[:, :num_samples].flatten()
   return samples, support_sizes
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254

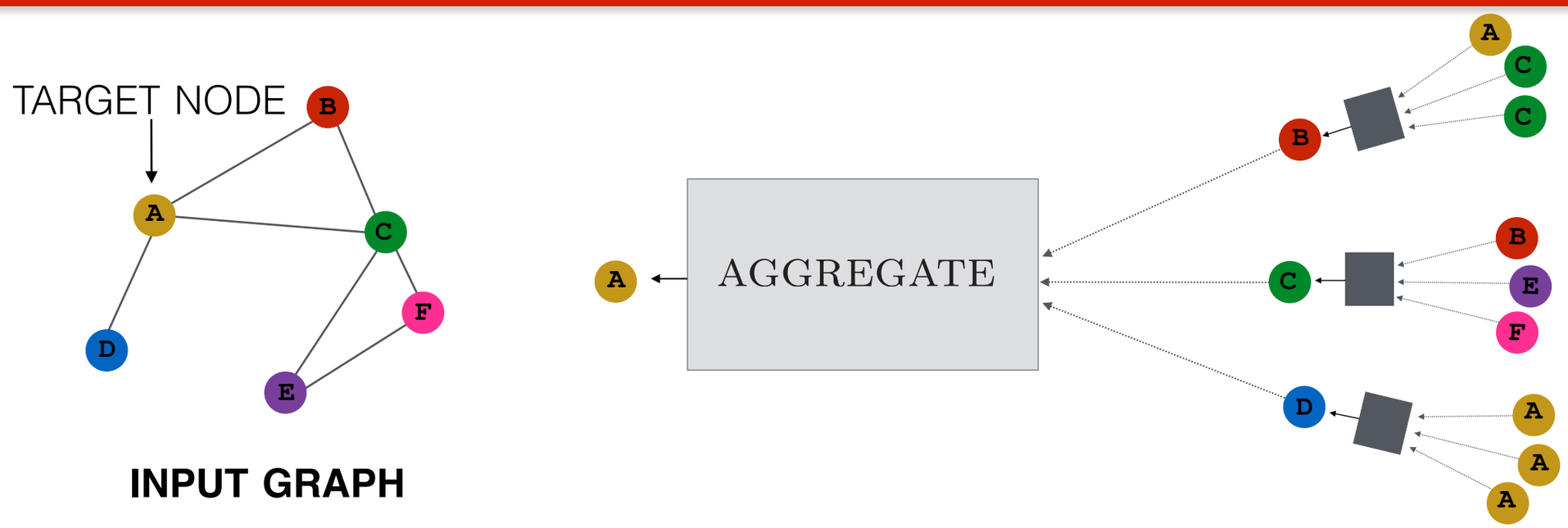
Implementation sketch: Sample stage




```
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1) # shuffle columns
        samples = neighs[:, :num_samples].flatten()
   return samples, support_sizes
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254

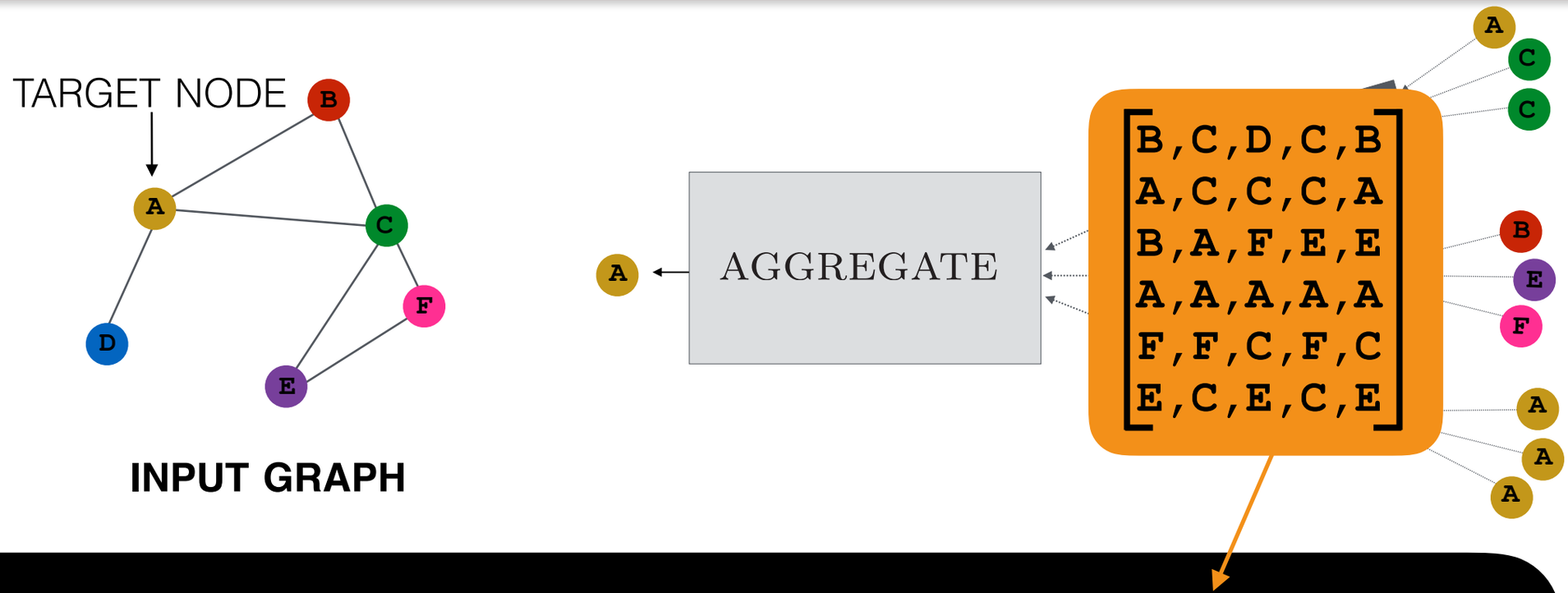
Implementation sketch: Sample stage




```
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1) # shuffle columns
        samples = neighs[:, :num_samples].flatten()
   return samples, support_sizes
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254

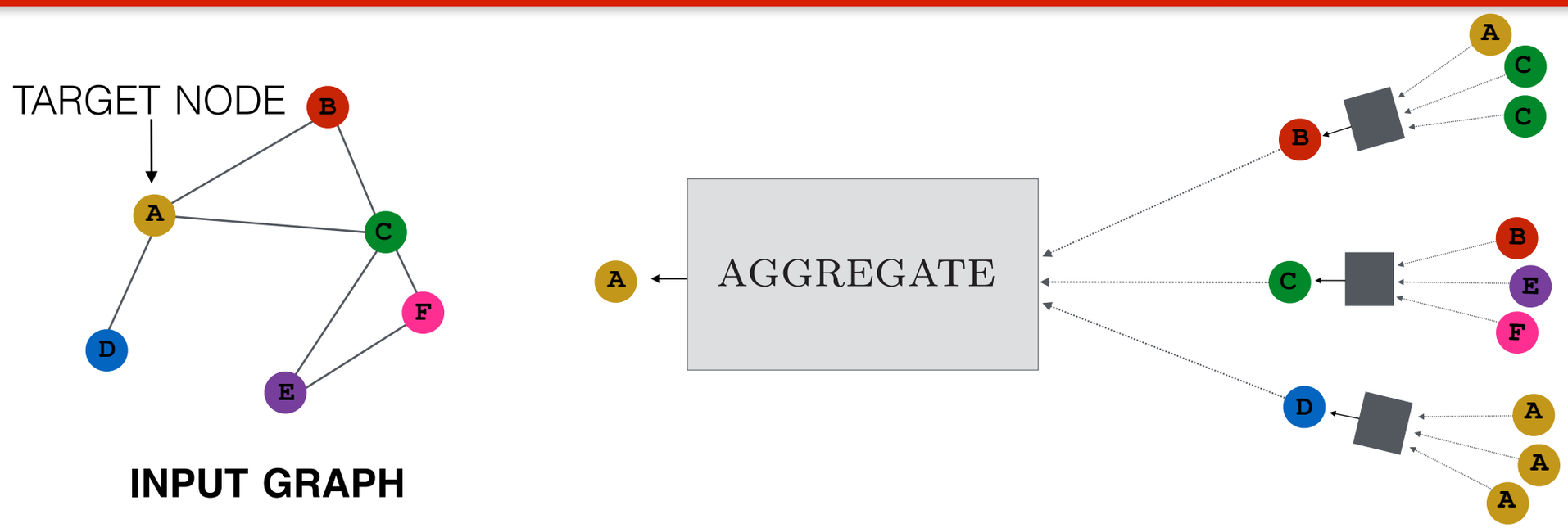
Implementation sketch: Sample stage




```
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1) # shuffle columns
        samples = neighs[:, :num_samples].flatten()
   return samples, support_sizes
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254

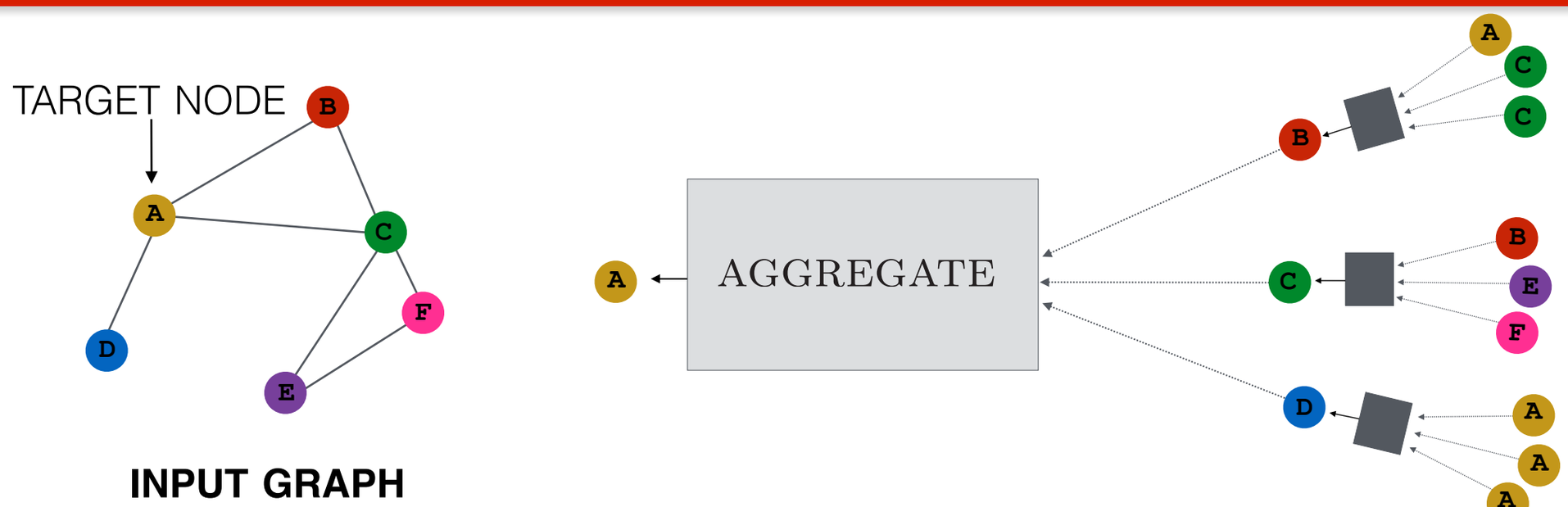
Implementation sketch: Sample stage




```
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1) # shuffle columns
        samples = neighs[:, :num_samples].flatten()
   return samples, support_sizes
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254

Implementation sketch: Sample stage

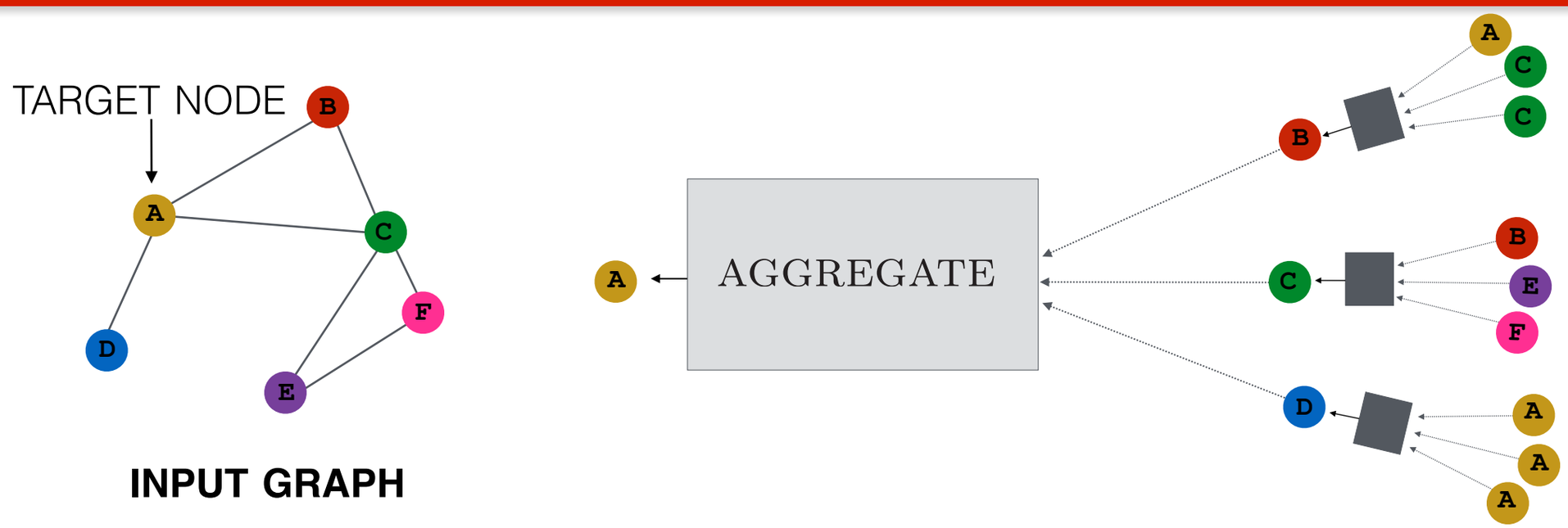



```
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1) # shuffle columns
        samples = neighs[:, :num_samples].flatten()
   return samples, support_sizes
```

[B C D C D]

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254

Implementation sketch: Sample stage

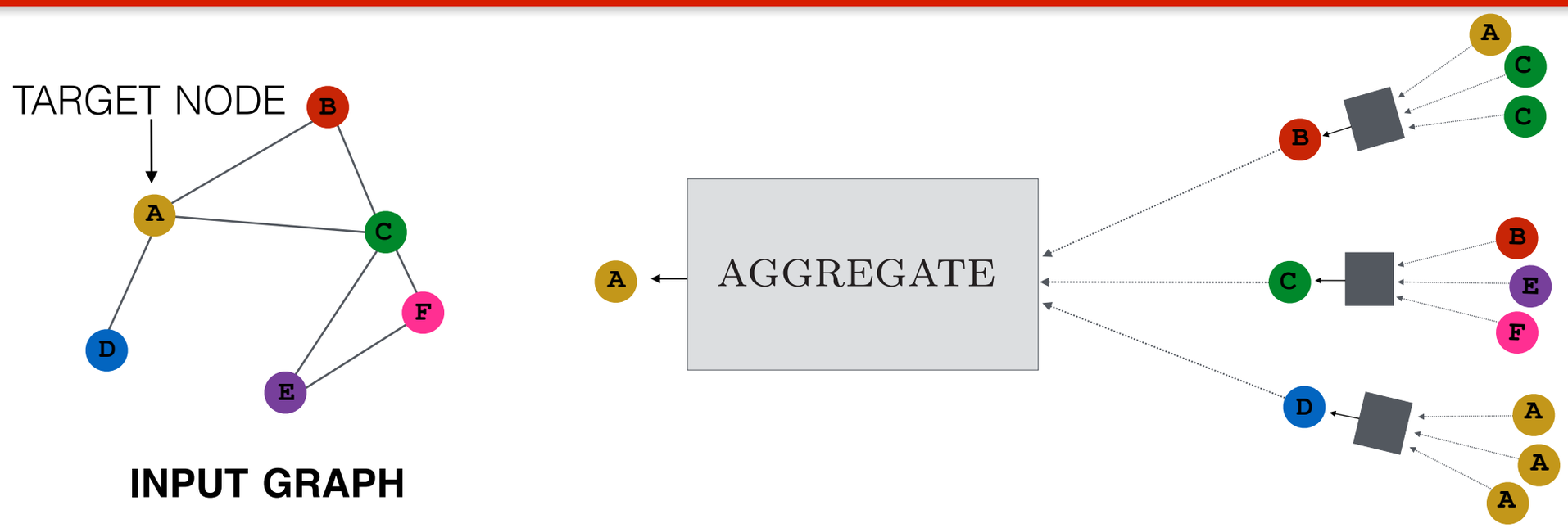



```
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1) # shuffle columns
        samples = neighs[:, :num_samples].flatten()
   return samples, support_sizes
```

[D C B D C]

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254

Implementation sketch: Sample stage

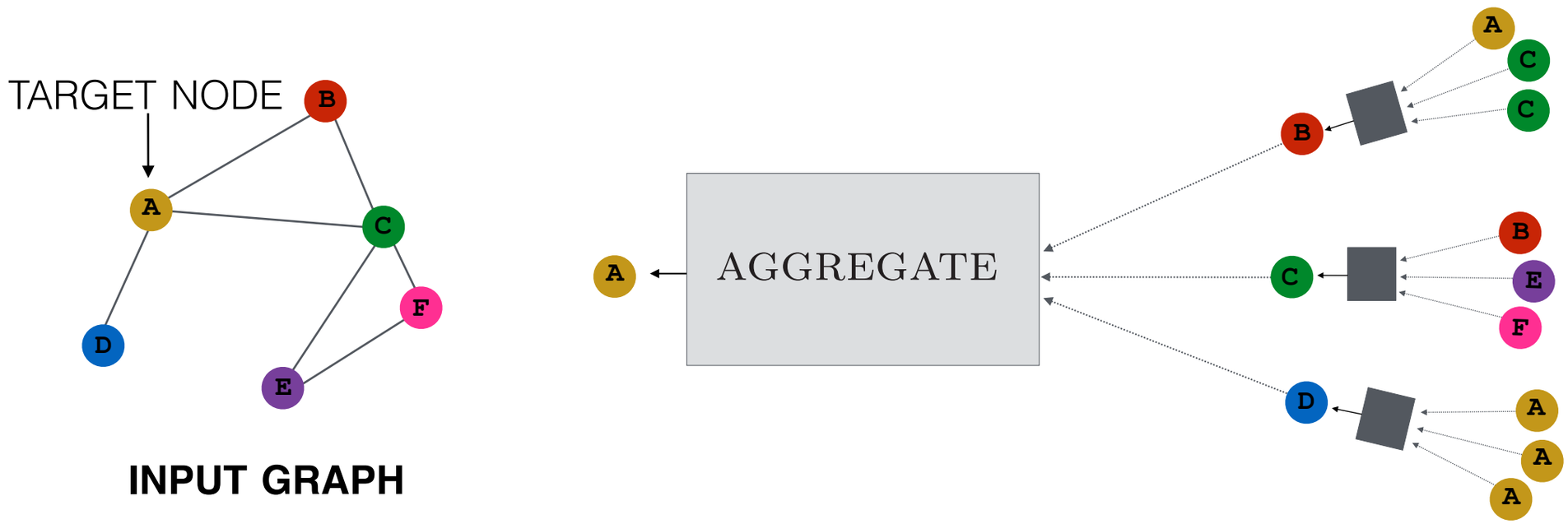



```
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1) # shuffle columns
        samples = neighs[:, :num_samples].flatten()
   return samples, support_sizes
```

[D C B]

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254

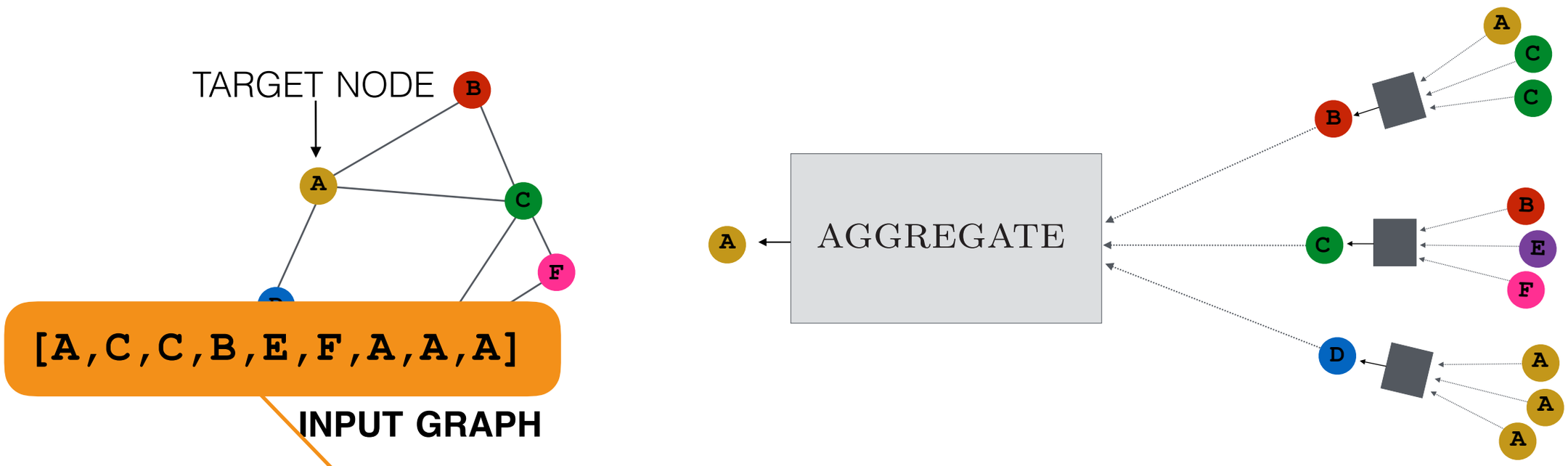
Implementation sketch: Aggregation stage




```
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)

    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
        messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)

    return messages
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278

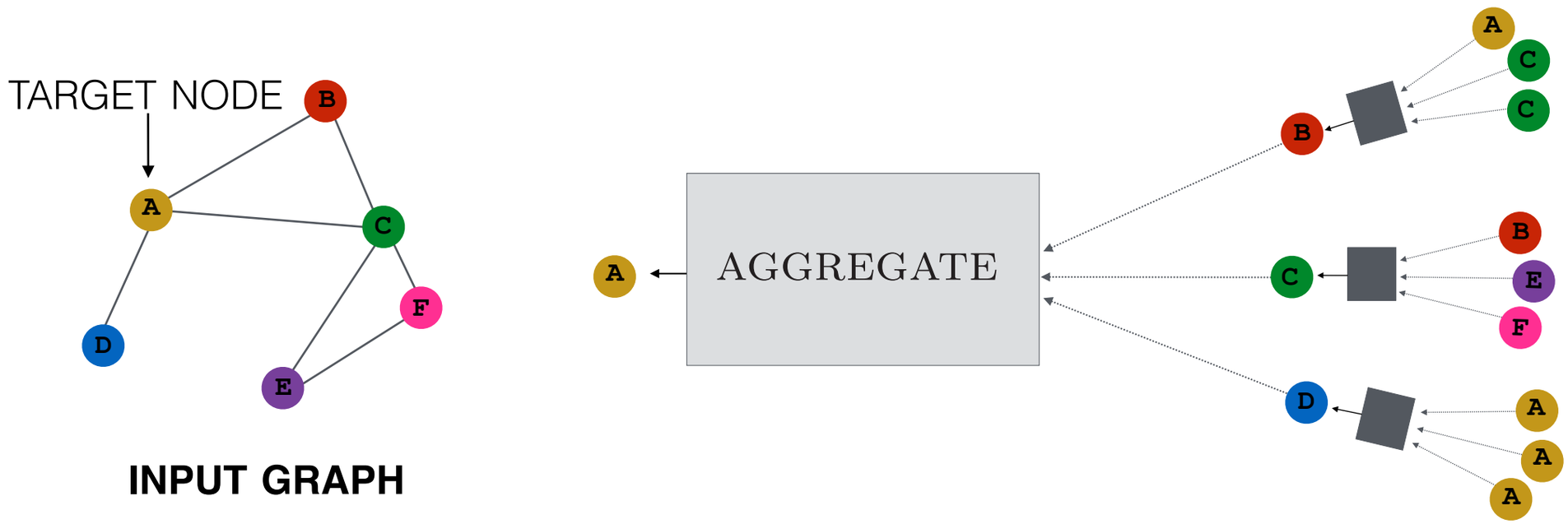
Implementation sketch: Aggregation stage




```
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)

    for i in range(len(samples)-1, 1, -1):
         messages = aggregate_func(messages, embed_dim)
         messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)

    return messages
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278

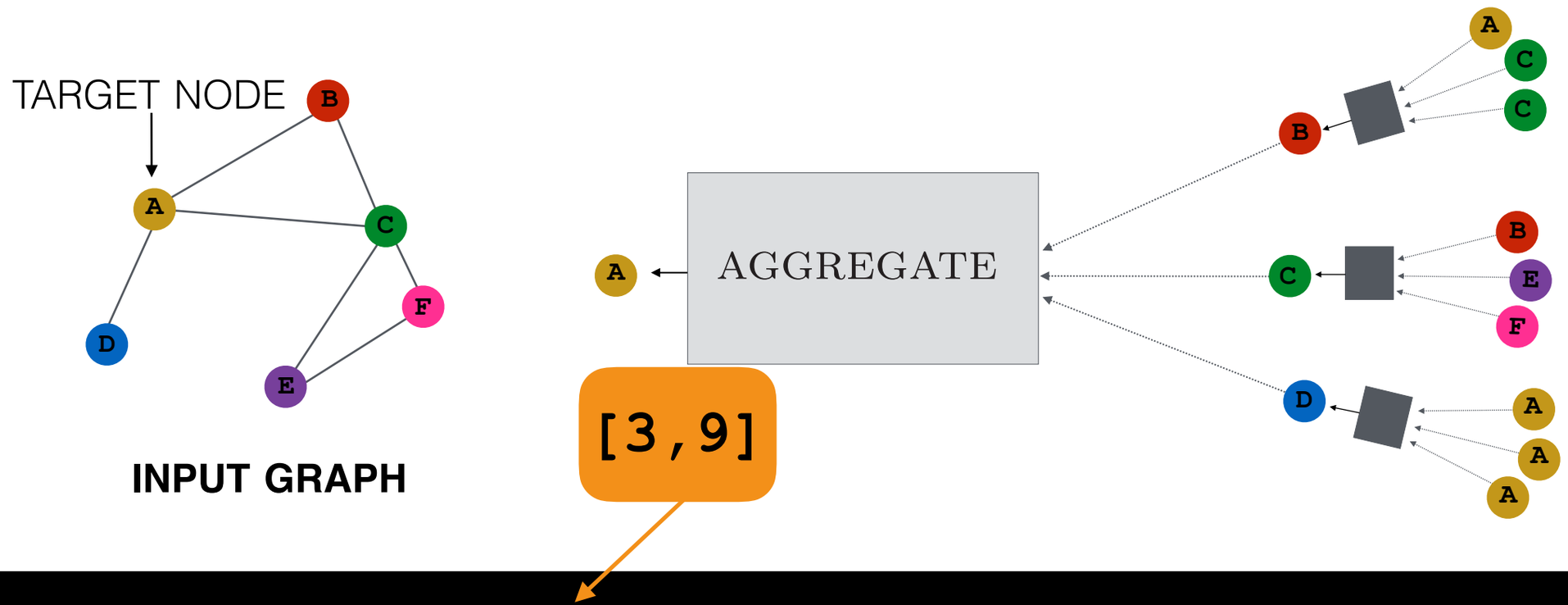
Implementation sketch: Aggregation stage




```
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)

    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
        messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)

    return messages
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278

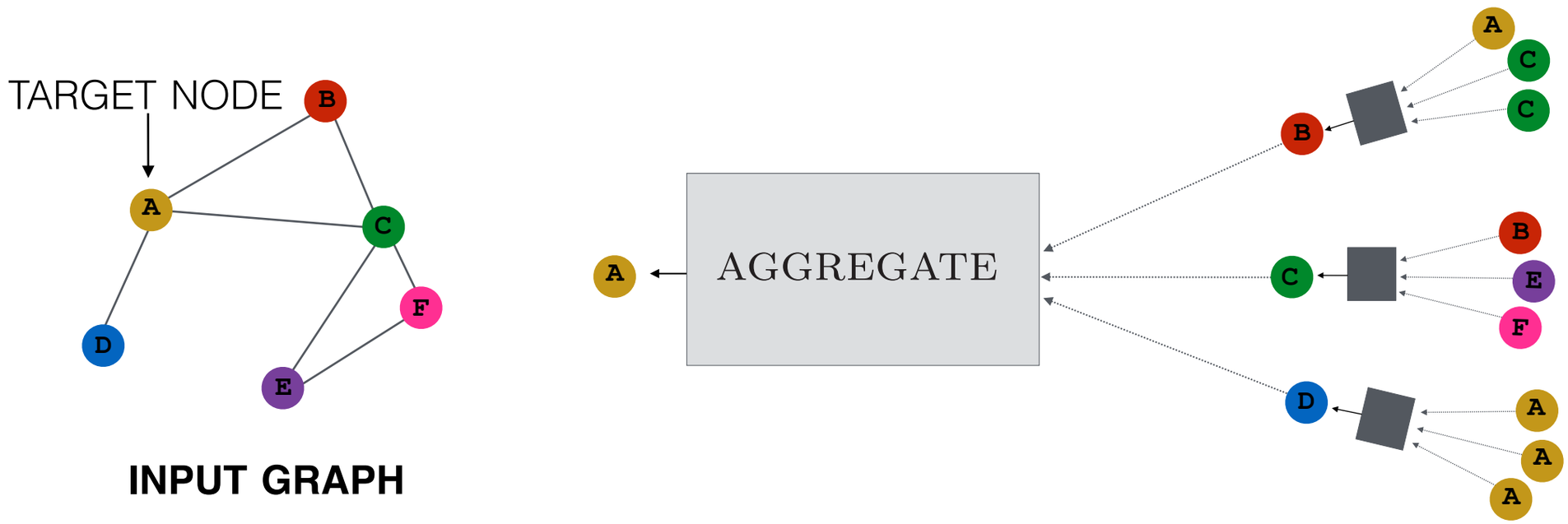
Implementation sketch: Aggregation stage




```
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)

    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
        messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)

    return messages
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278

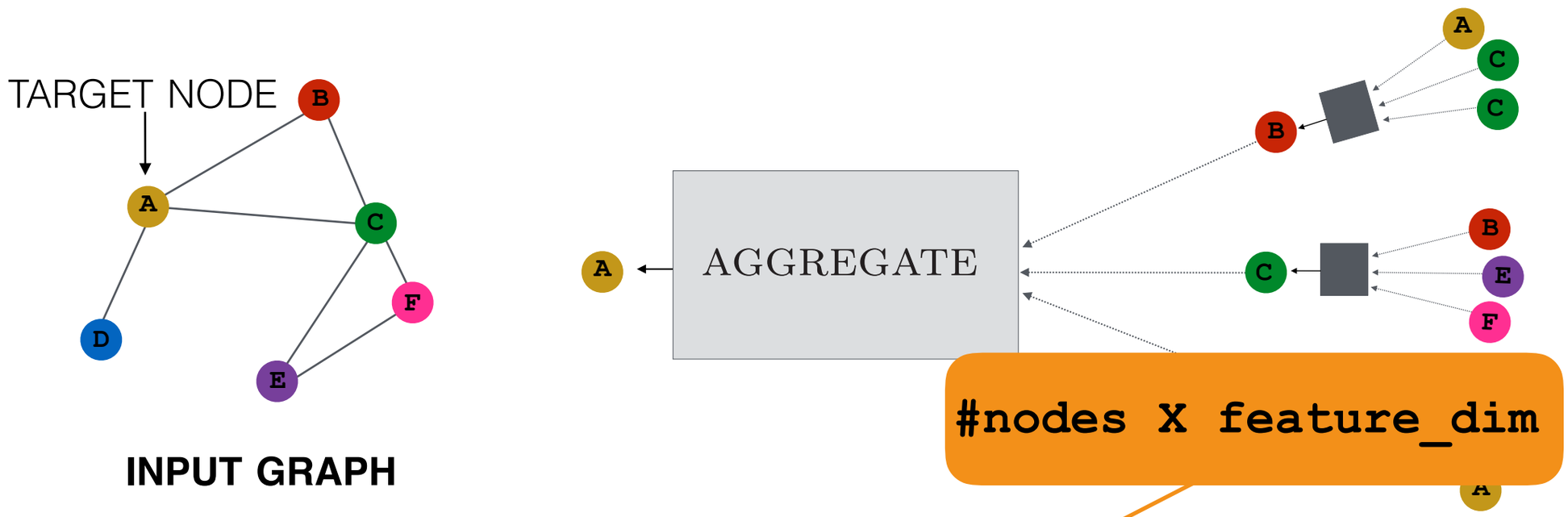
Implementation sketch: Aggregation stage




```
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)

    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
        messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)

    return messages
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278

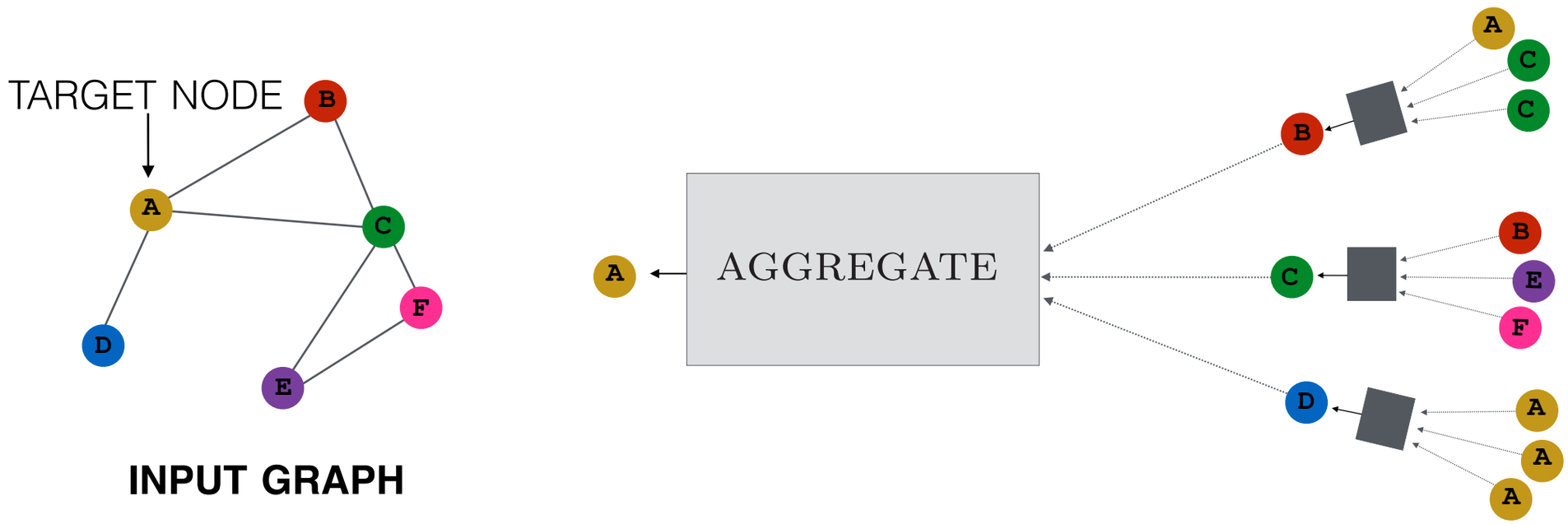
Implementation sketch: Aggregation stage




```
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)

    for i in range(len(samples)-1, 1, -1):
         messages = aggregate_func(messages, embed_dim)
         messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)

    return messages
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278

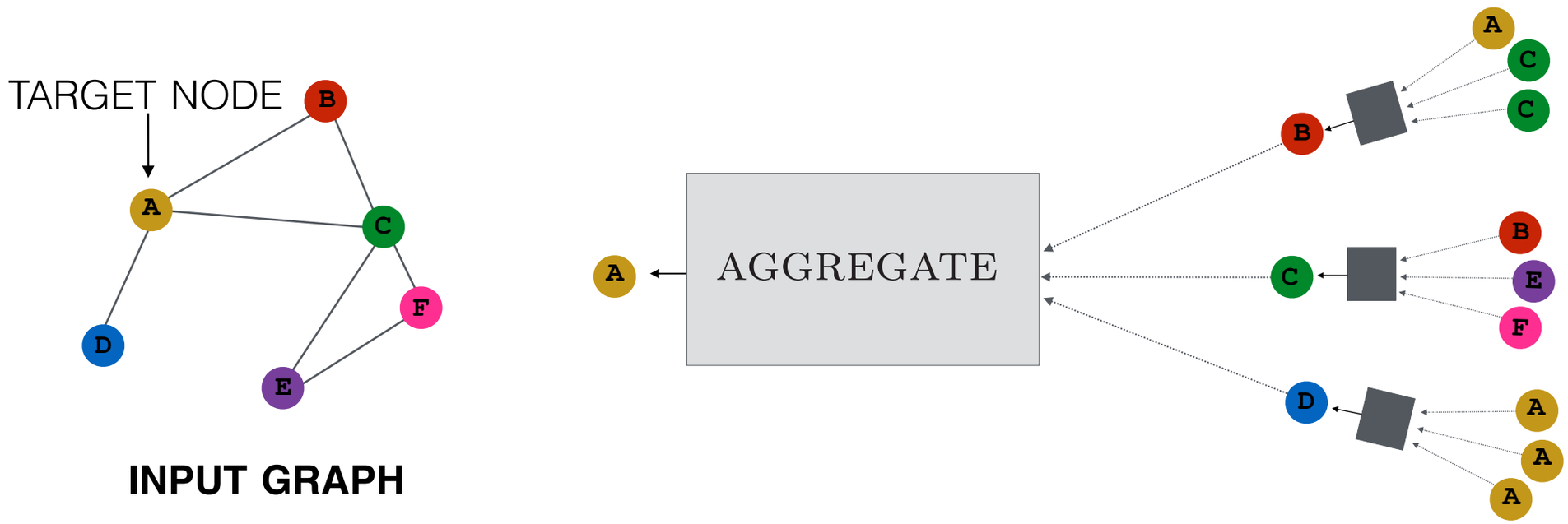
Implementation sketch: Aggregation stage




```
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)

    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
        messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)

    return messages
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278

Implementation sketch: Aggregation stage

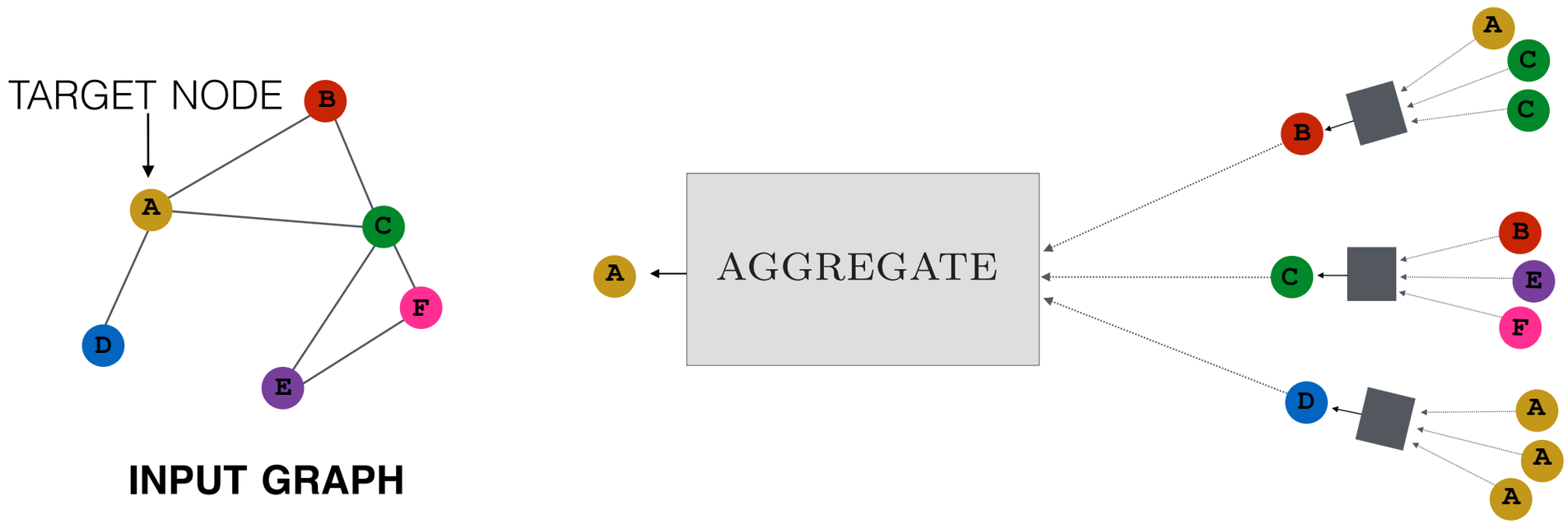


9 X feature_dim

```
def aggregate(samples, su                    matrix, embed_dim):
    messages = feature_mat                   (samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)

    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
        messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)

    return messages
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278

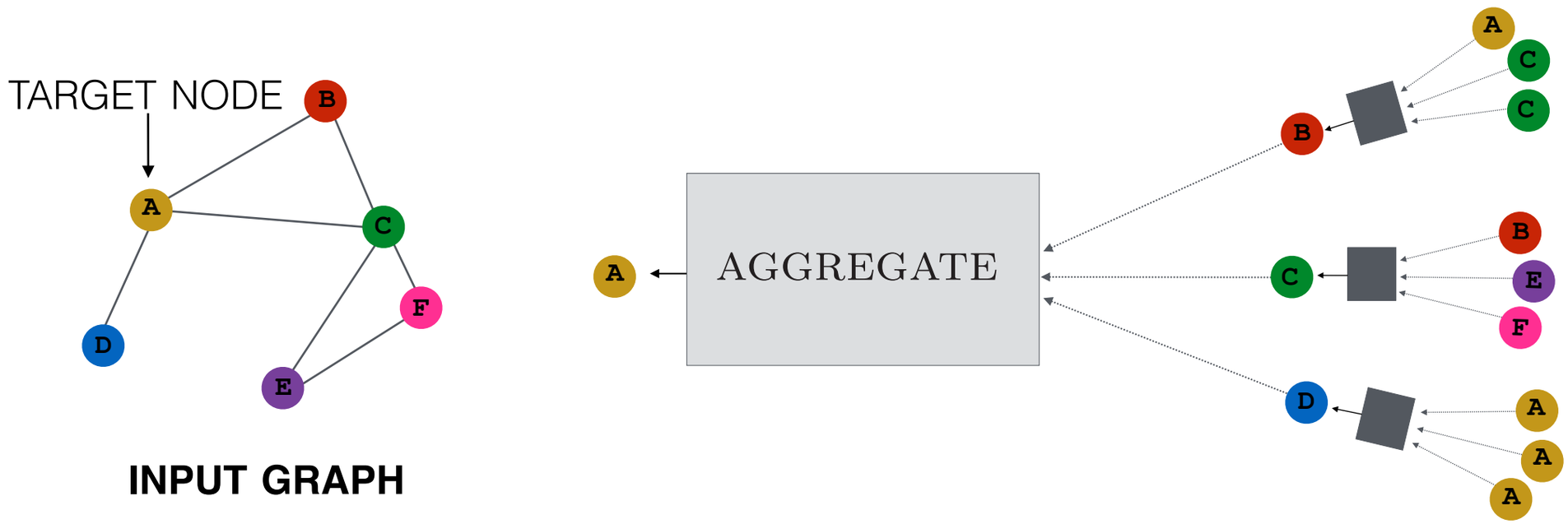
Implementation sketch: Aggregation stage




```
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)

    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
        messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)

    return messages
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278

Implementation sketch: Aggregation stage

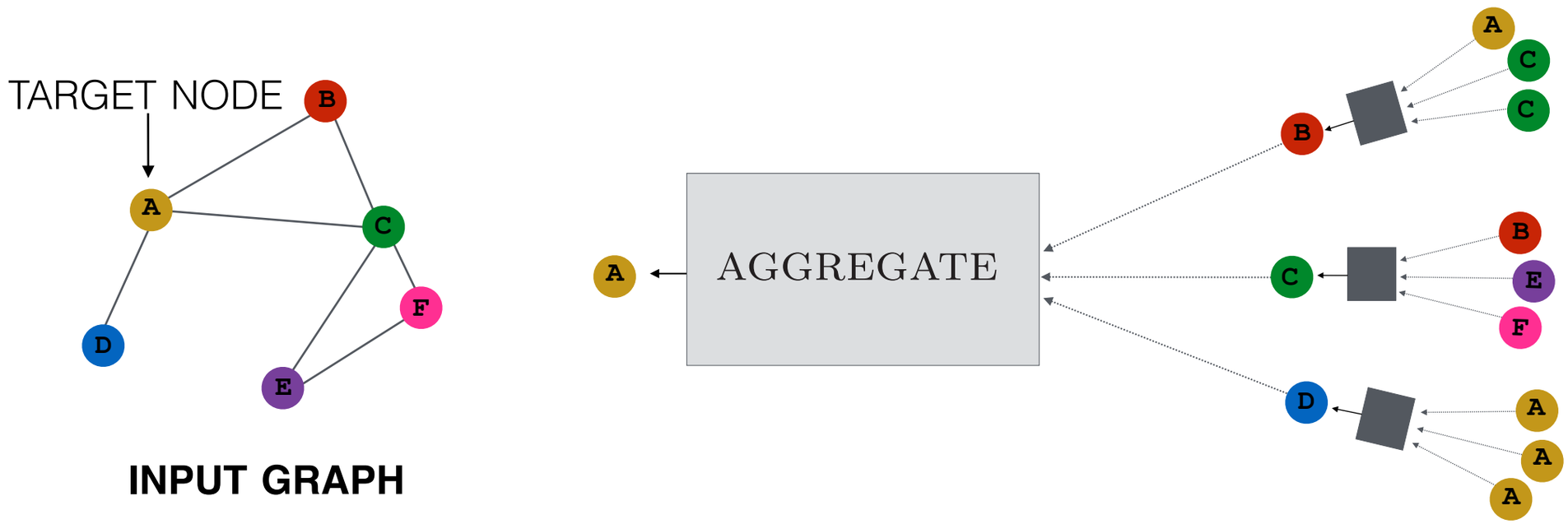



```
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.res                    _sizes[-1], feature_dim)

    for i in range(len(samp
        messages = aggregate_func(messages, embed_dim)
        messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)

    return messages
```

3 X 3 X feature_dim

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278

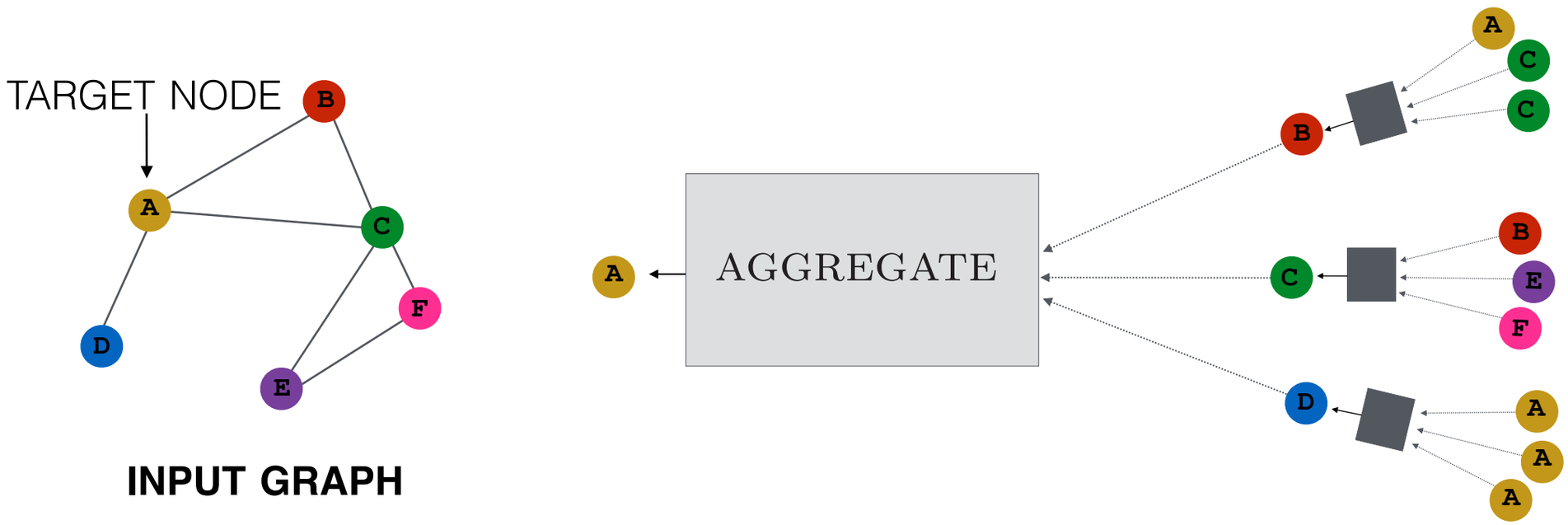
Implementation sketch: Aggregation stage




```
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)

    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
        messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)

    return messages
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278

Implementation sketch: Aggregation stage

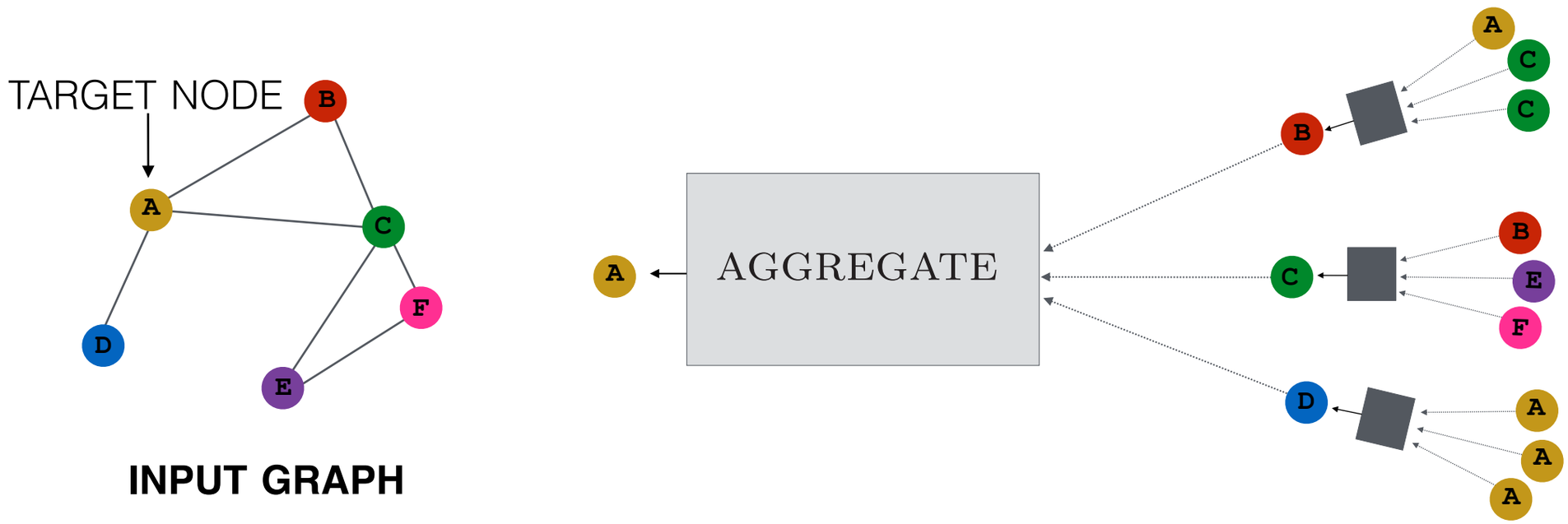



```
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)

    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_fu                    d_dim)
        messages = messages.res                    support_sizes[-2], embed_dim)

    return messages
```

3 X embed_dim

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278

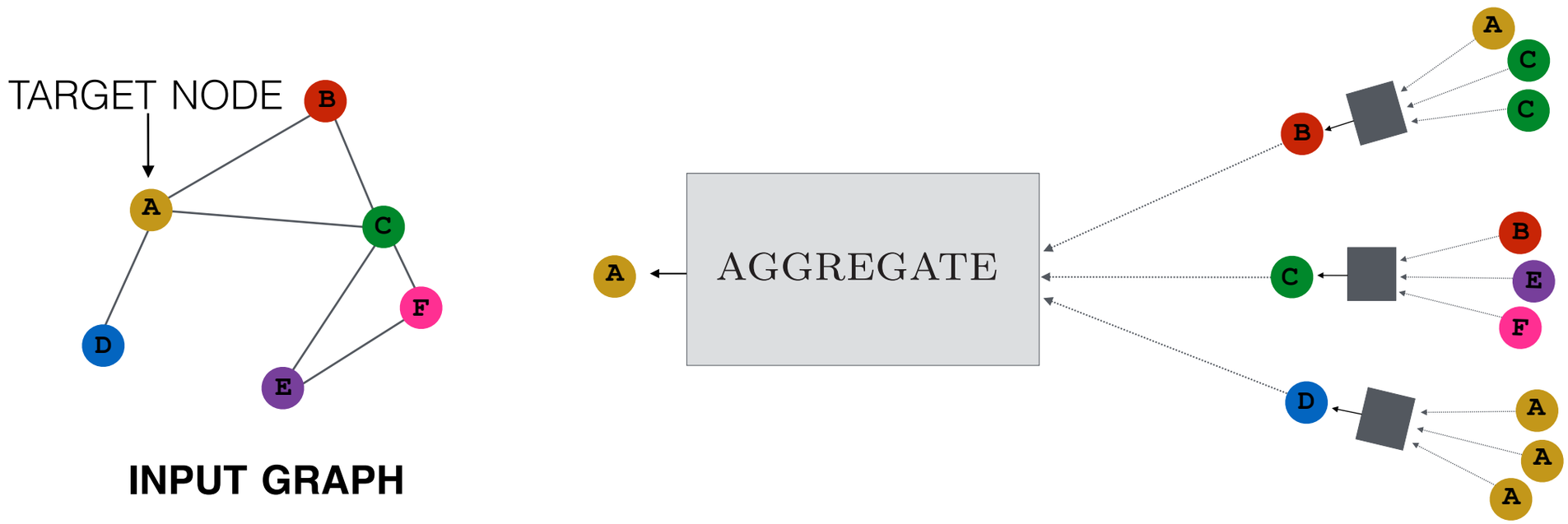
Implementation sketch: Aggregation stage




```
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)

    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
        messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)

    return messages
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278

Implementation sketch: Aggregation stage

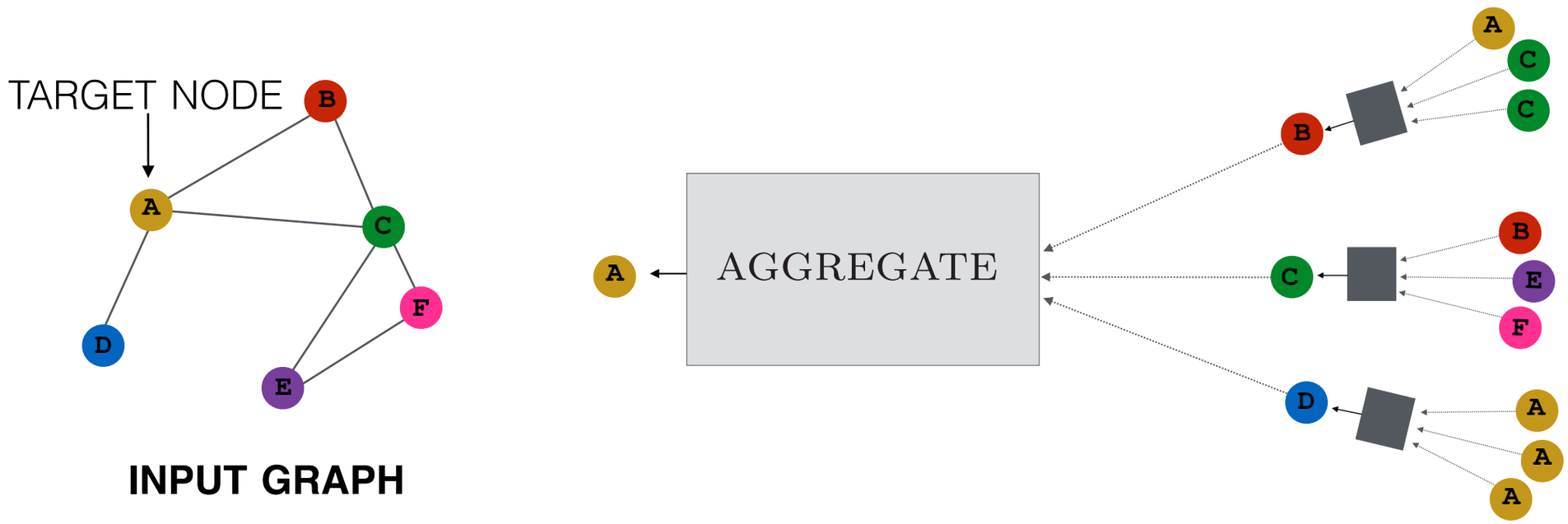



```
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)

    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
        messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)

    return messages
```

1 X 3 X embed_dim

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278

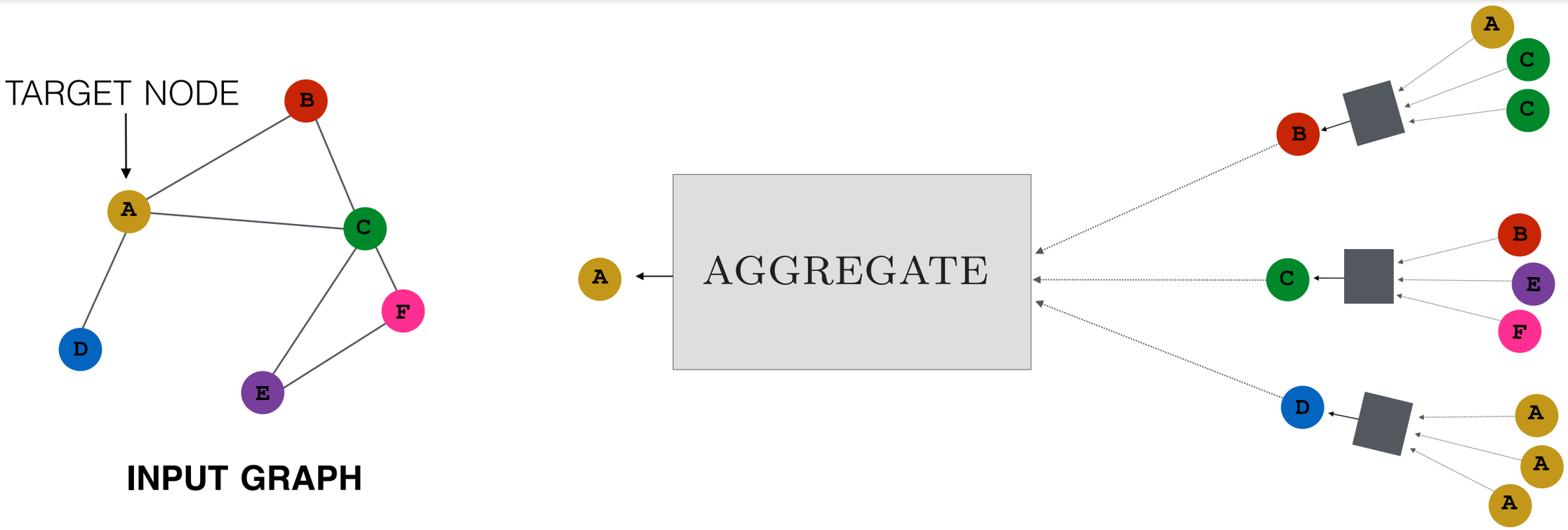
Implementation sketch: Aggregation stage




```
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)

    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
        messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)

    return messages
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278

Additional complexities.....



Various unsupervised vs. supervised loss functions

“Skip-connections”

Efficient data I/O.

https://github.com/williamleif/GraphSAGE