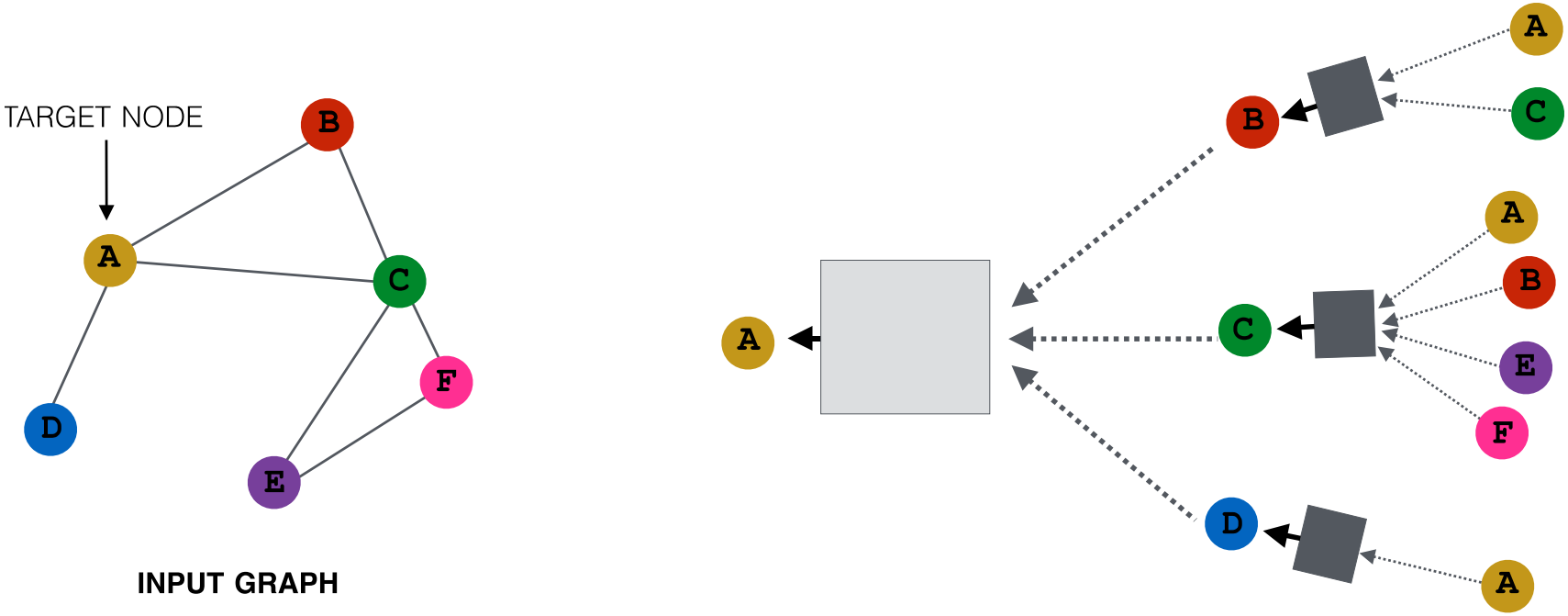
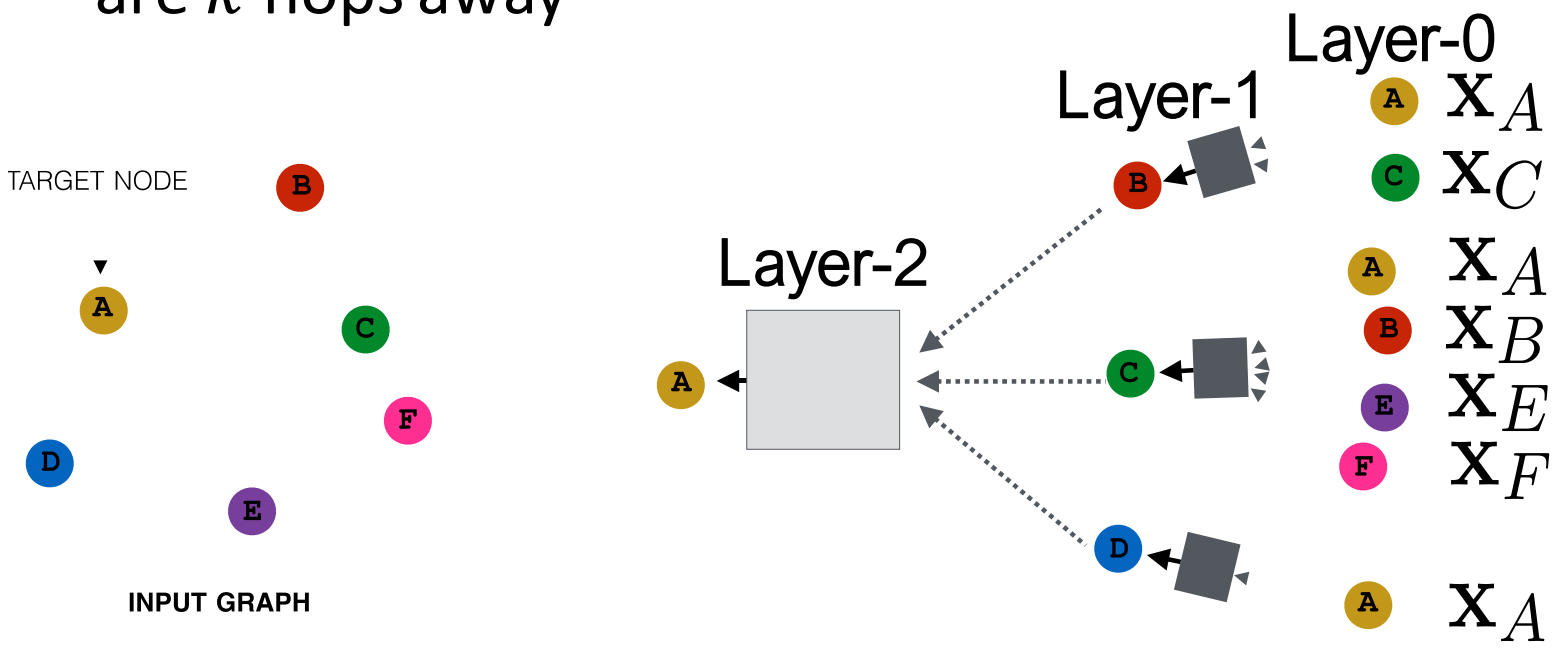
Idea: Aggregate Neighbors

- **Key idea:** Generate node embeddings based on **local network neighborhoods**



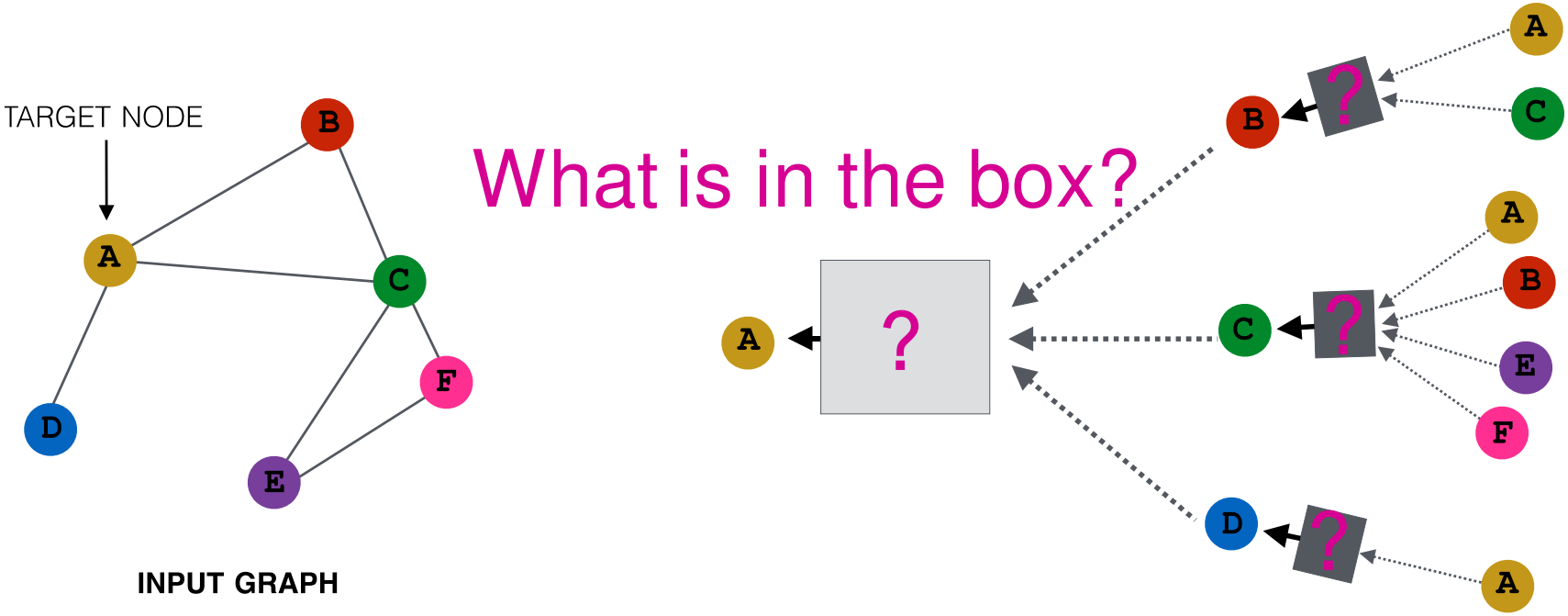
Deep Model: Many Layers

- Model can be of arbitrary depth:
  - Nodes have embeddings at each layer
  - Layer-0 embedding of node v is its input feature, x_v
  - Layer-k embedding gets information from nodes that are k hops away



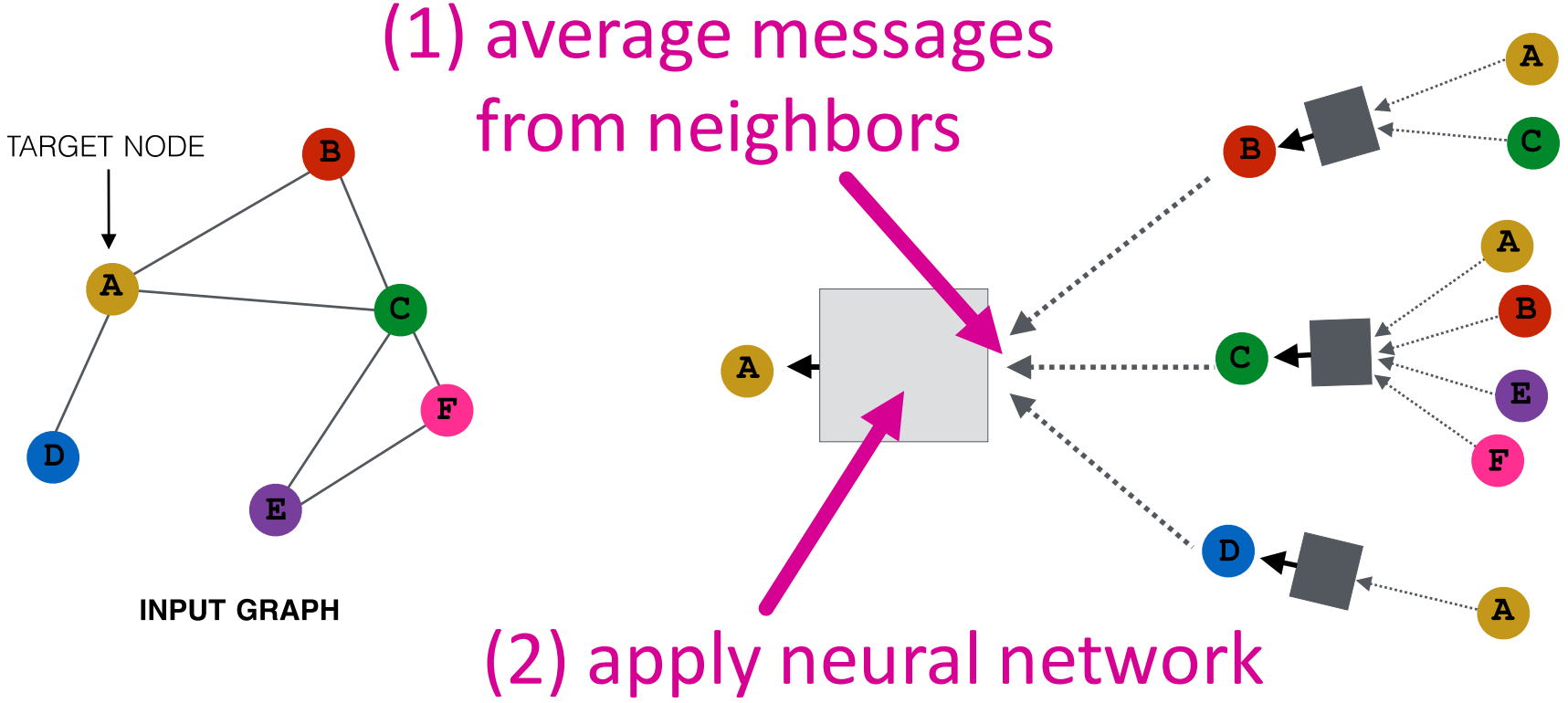
Neighborhood Aggregation

- **Neighborhood aggregation:** Key distinctions are in how different approaches aggregate information across the layers



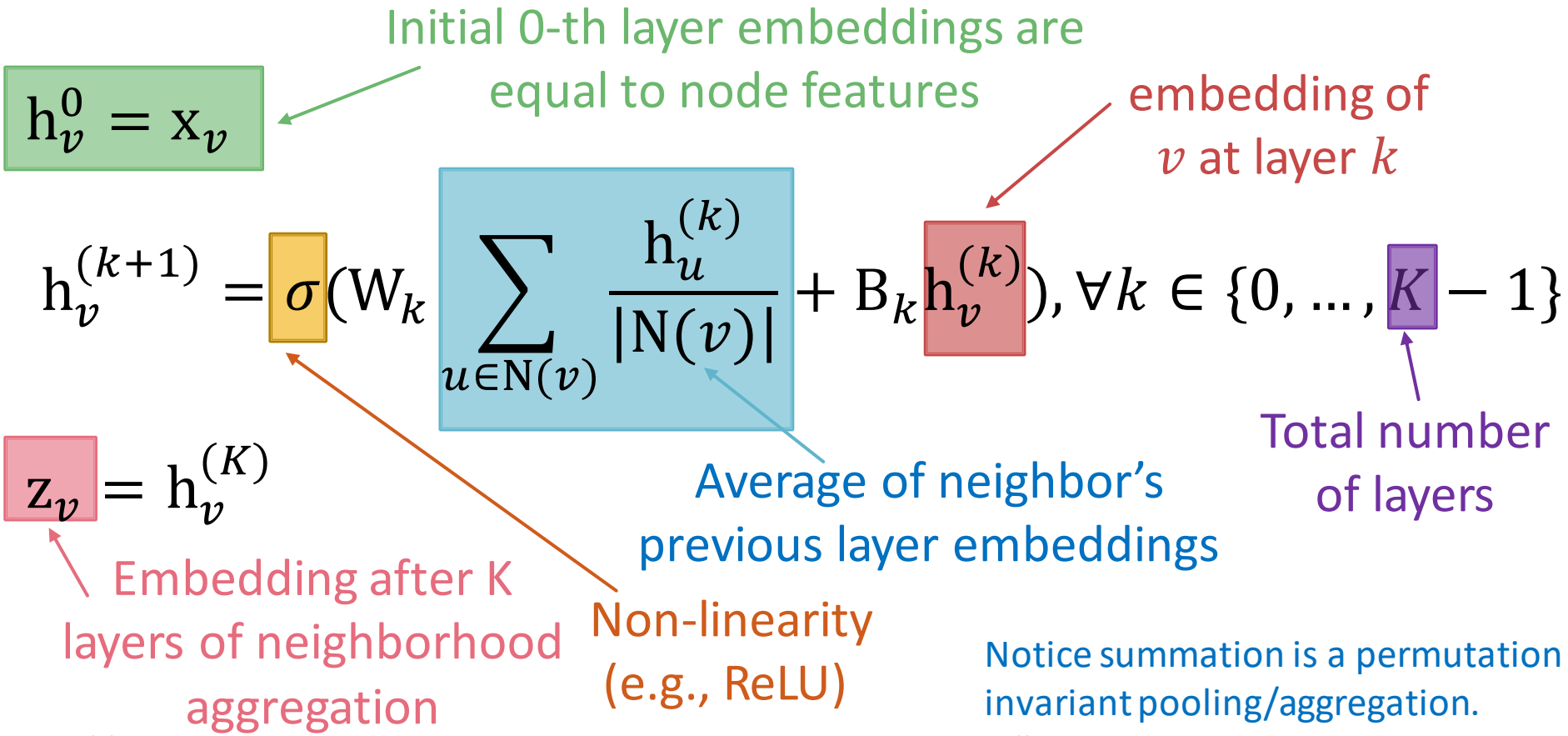
Neighborhood Aggregation

- **Basic approach:** Average information from neighbors and apply a neural network



The Math: Deep Encoder

- **Basic approach:** Average neighbor messages and apply a neural network



Model Parameters

Trainable weight matrices
(i.e., what we learn)

$$\mathrm{h}_v^{(0)} = \mathrm{x}_v$$

$$\mathrm{h}_v^{(k+1)} = \sigma\left(\mathrm{W}_k \sum_{u \in \mathrm{N}(v)} \frac{\mathrm{h}_u^{(k)}}{|\mathrm{N}(v)|} + \mathrm{B}_k \mathrm{h}_v^{(k)}\right), \forall k \in \{0..K-1\}$$

$$\mathrm{z}_v = \mathrm{h}_v^{(K)}$$

Final node embedding

We can feed these **embeddings into any loss function** and run SGD to **train the weight parameters**

h_v^k: the hidden representation of node v at layer k

- W_k: weight matrix for neighborhood aggregation
- B_k: weight matrix for transforming hidden vector of self