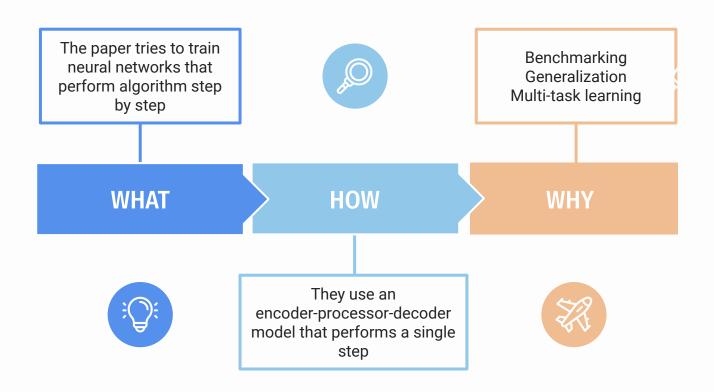
# **Neural execution** of graph algorithms

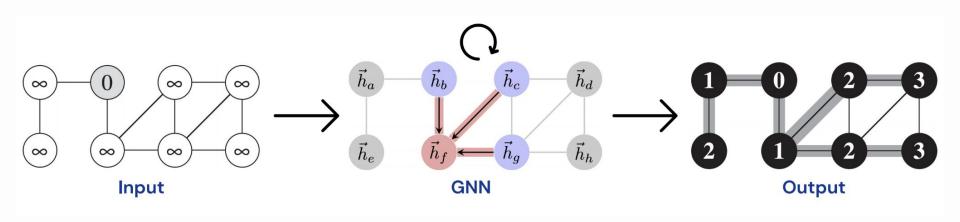
Implementation by Gbetondji Dovonon

# **Creative Process Infographics**



### What do we learn to execute

- The model is performed on the input
- It is called on its own output until terminated



# How do we learn an algorithm

Input Graph with features [n\_nodes, 1] **Encoder** Linear transformation to [n\_nodes, hidden dimension] **Processor** Performs the operations. Most of the modeling is here. Outputs to [n\_nodes, hidden dimension] **Termination** Decoder Linear transformation to [n\_nodes, 1] Int that predicts if the model is done

# Why do we learn algorithms

#### Better benchmarking

Current benchmarks are not reliable. Simplified GCNs achieve strong baselines. Algorithms can allow us to generate infinite controlled data with high complexity

#### Generalization

Learning general steps instead of an input/output mapping Algorithms generalize trivially

#### Multi task learning

Possibility to learn multiple algorithms

## **Breadth First Search**

All models are trained on graphs with 20 nodes

#### Mean step by step accuracy / last step accuracy

Model	20 nodes	50 nodes	100 nodes	1000 nodes
GAT	100% / 100%	99.47% / 100 %	99.47% / 99.8%	91% / 99%
MPNN-sum	95% / 100%	96% / 100%	93.64% / 99.96%	76.47% / 86.77%
MPNN-mean	100% / 100%	74% / 100%	71.99% / 99.84%	57.96% / 100%
MPNN-max	100% / 100%	100% / 100%	100% / 100%	100% / 100%

# THANK YOU

Questions?

