dortmund university

Stanford University

Auto-Scaling Graph Neural Networks

Matthias Fey

matthias.fey@tu-dortmund.de

https://rusty1s.github.io

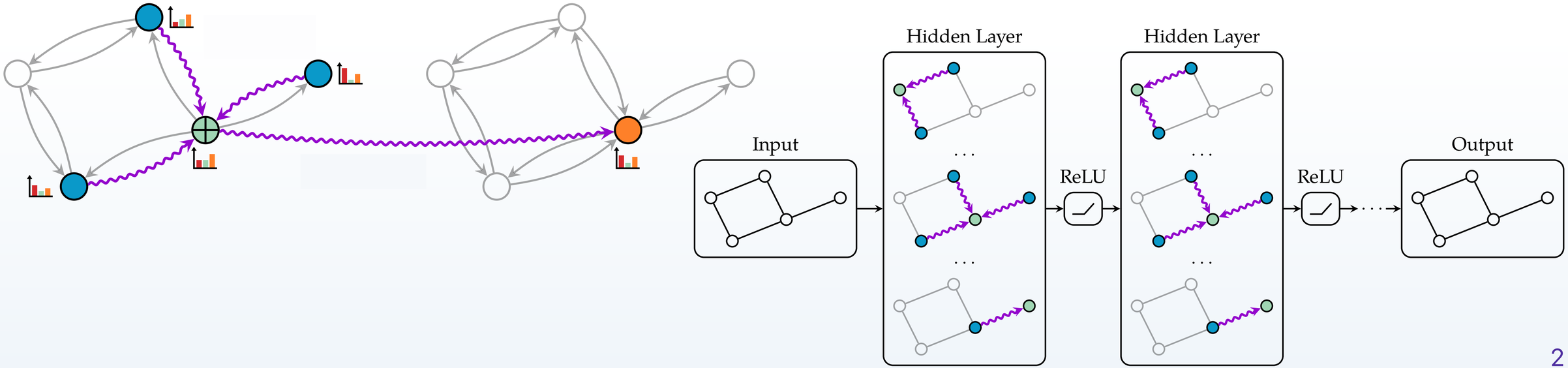
/rusty1s

Graph Neural Networks

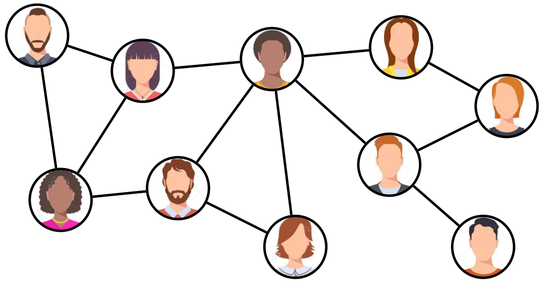
Graph Neural Networks (GNNs) generalize **Convolutional Neural Networks** to arbitrary structured data by following a differentiable **message passing scheme**:

$$\mathbf{h}_v^{(\ell+1)} = f_\theta^{(\ell+1)}\left(\mathbf{h}_v^{(\ell)}, \left\{\!\!\left\{\mathbf{h}_w^{(\ell)} : w \in \mathcal{N}(v)\right\}\!\!\right\}\right)$$

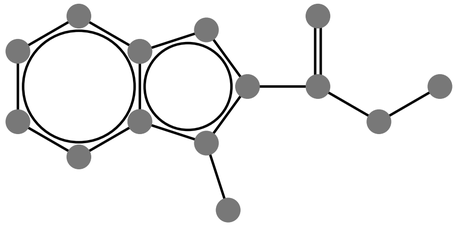
where $\mathbf{h}_v^{(\ell)}$ denotes a feature vector for every node $v \in \mathcal{V}$.



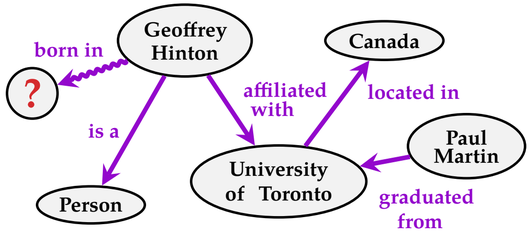
Graph Neural Networks: Applications



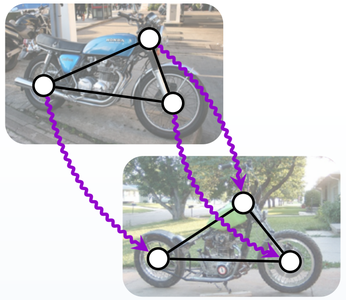
(a) Recommendation



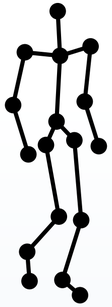
(b) Drug Discovery



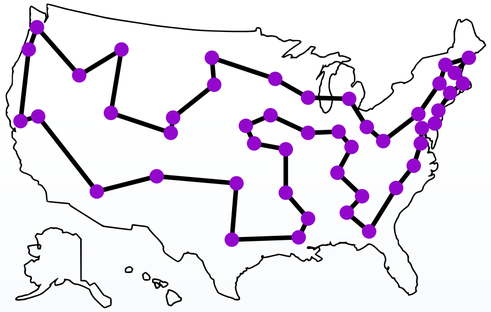
(c) Reasoning in KGs



(d) Graph Matching



(e) Scene Understanding

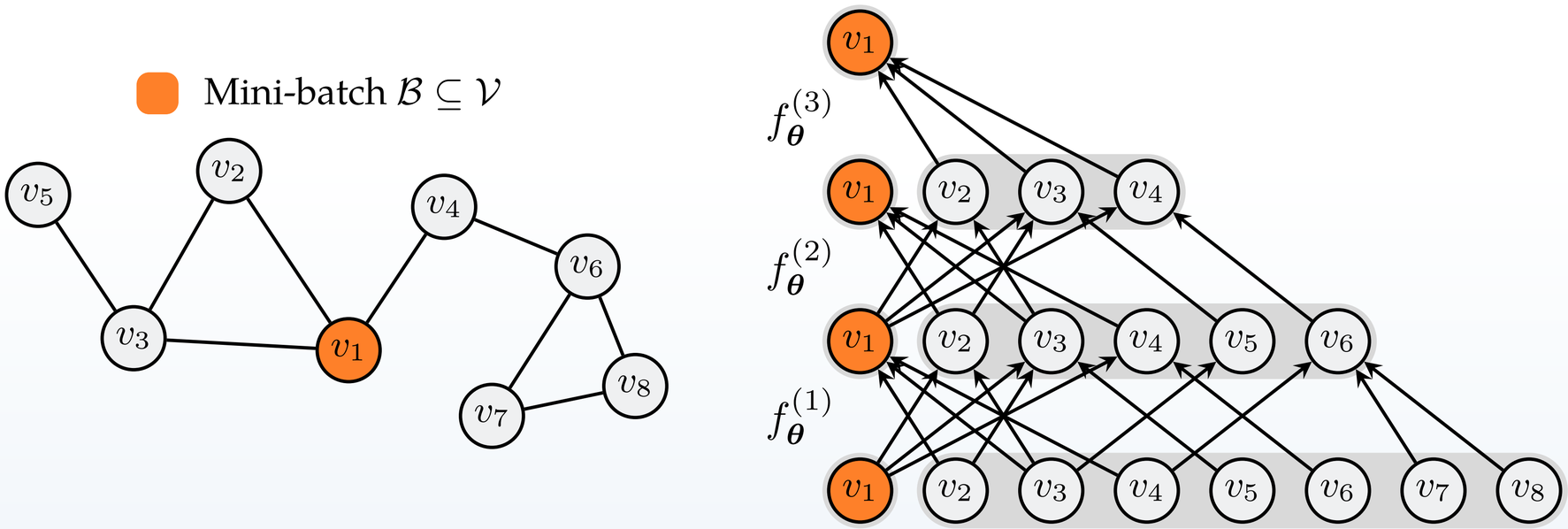


(f) Combinatorial Optimization

Scalable Graph Neural Networks

Scalability techniques are **indispensable** for applying GNNs to large graphs

Mini-batch processed GNNs are subject to the **neighbor explosion problem:**
exponentially increasing dependency of nodes over layers



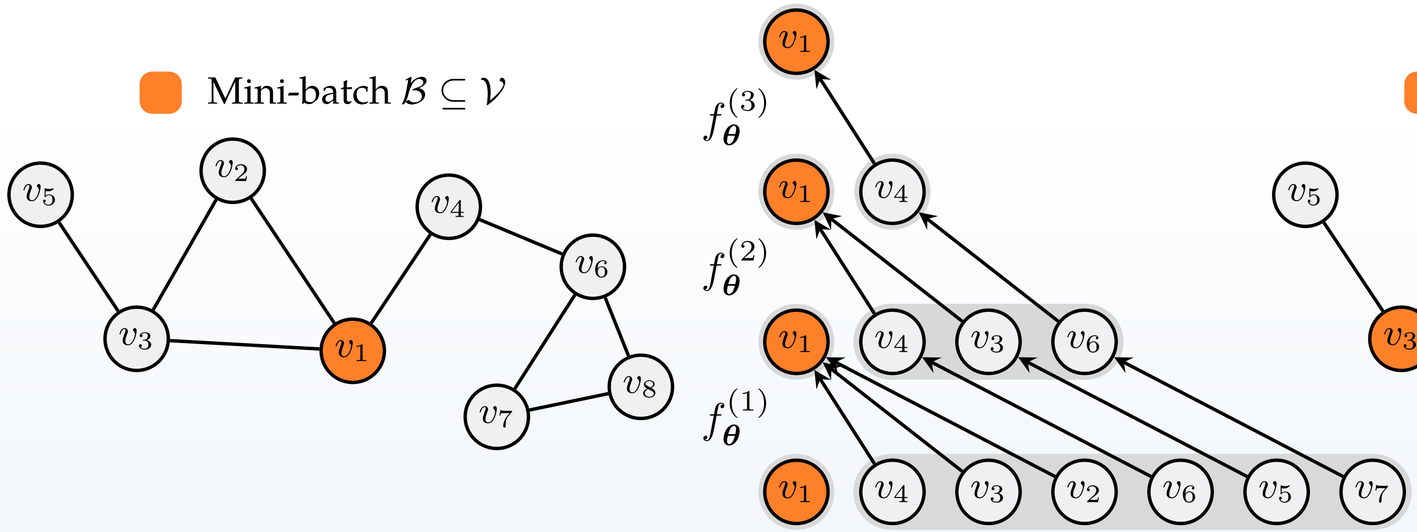
Scalable Graph Neural Networks

The most common approaches for scaling GNNs work by **sampling edges**

Hamilton et al., 2017; Chen et al., 2018; Zou et al., 2019; Huang et al., 2018; Chiang et al., 2019; Zeng et al. 2019;...

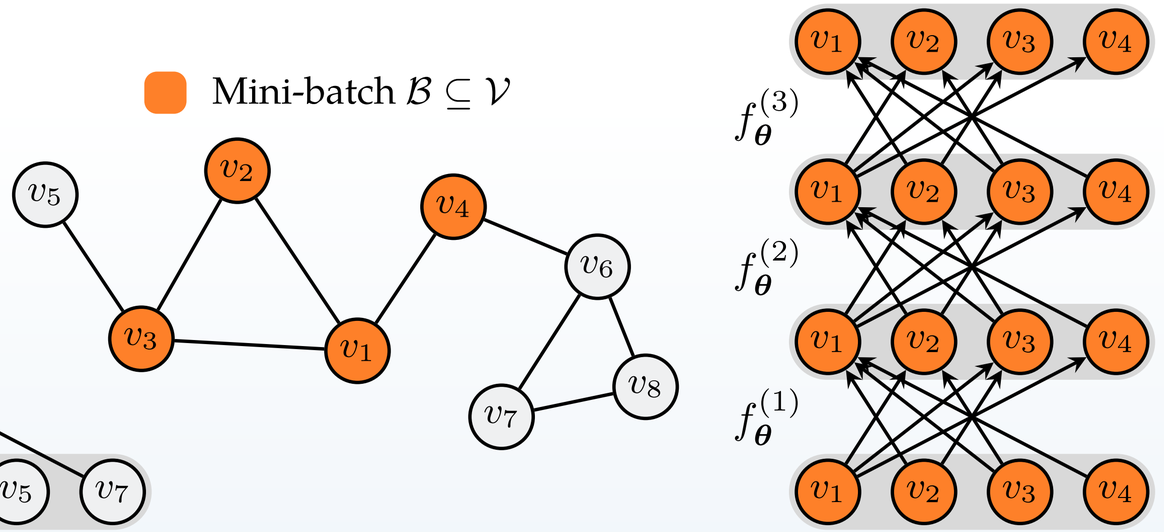
Node-wise Sampling

Samples a fixed amount of neighbors per node/layer



Subgraph-wise Sampling

Samples a subgraph to learn embeddings for all sampled nodes



Scalable Graph Neural Networks

The most common approaches for scaling GNNs work by **sampling edges**

Hamilton et al., 2017; Chen et al., 2018; Zou et al., 2019; Huang et al., 2018; Chiang et al., 2019; Zeng et al. 2019;...

- Neighbor explosion **still takes place** in node-wise sampling for deeper GNNs 😭
- Subgraph-wise sampling restricts learning to **shallow subgraphs** 😭

Just by the act of sampling edges,
a GNN will **fail to learn** about **structural graph properties**

No longer access to the full neighborhood information (reduced expressivity)

How can we learn **structural graph properties** while still **being scalable?** 🤔

GNNAutoScale: Historical Embeddings

We can utilize **historical embeddings** to **approximate** the **missing out-of-mini-batch information** for each layer

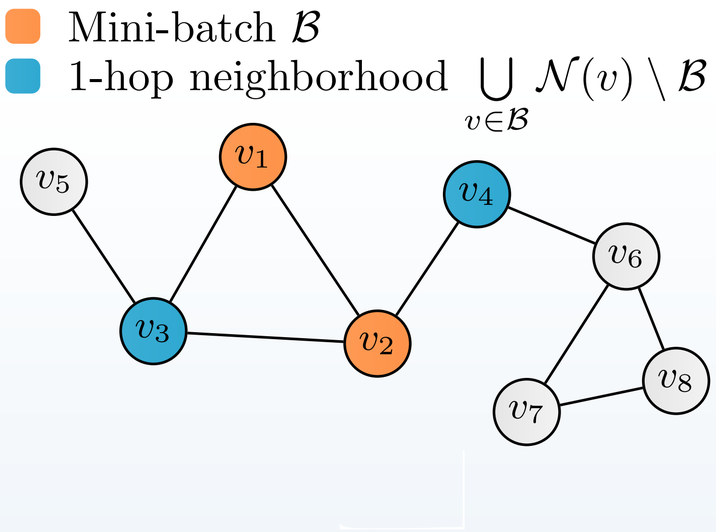
$$
\begin{aligned}
\mathbf{h}_v^{(\ell+1)} &= f_\theta^{(\ell+1)}\left(\mathbf{h}_v^{(\ell)}, \left\{\!\!\left\{ \mathbf{h}_w^{(\ell)} : w \in \mathcal{N}(v) \right\}\!\!\right\}\right) \\
&= f_\theta^{(\ell+1)}\left(\mathbf{h}_v^{(\ell)}, \left\{\!\!\left\{ \mathbf{h}_w^{(\ell)} : w \in \mathcal{N}(v) \cap \mathcal{B} \right\}\!\!\right\} \cup \left\{\!\!\left\{ \mathbf{h}_w^{(\ell)} : w \in \mathcal{N}(v) \setminus \mathcal{B} \right\}\!\!\right\}\right) \\
&\approx f_\theta^{(\ell+1)}\left(\mathbf{h}_v^{(\ell)}, \left\{\!\!\left\{ \mathbf{h}_w^{(\ell)} : w \in \mathcal{N}(v) \cap \mathcal{B} \right\}\!\!\right\} \cup \underbrace{\left\{\!\!\left\{ \bar{\mathbf{h}}_w^{(\ell)} : w \in \mathcal{N}(v) \setminus \mathcal{B} \right\}\!\!\right\}}_{\text{Historical Embeddings}}\right)
\end{aligned}
$$

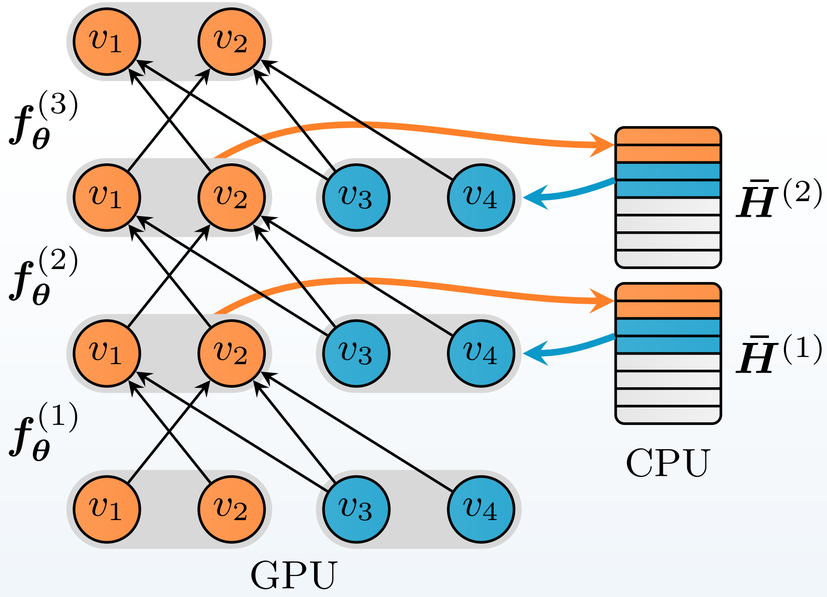
*Historical embeddings represent node embeddings **acquired in previous training iterations***

GNNAutoScale: Historical Embeddings

We can utilize **historical embeddings** to **approximate** the **missing out-of-mini-batch information** for each layer

- *We **pull** the most recent histories from **out-of-mini-batch nodes***
- *We **push** newly estimated embeddings of **in-mini-batch nodes** to histories*





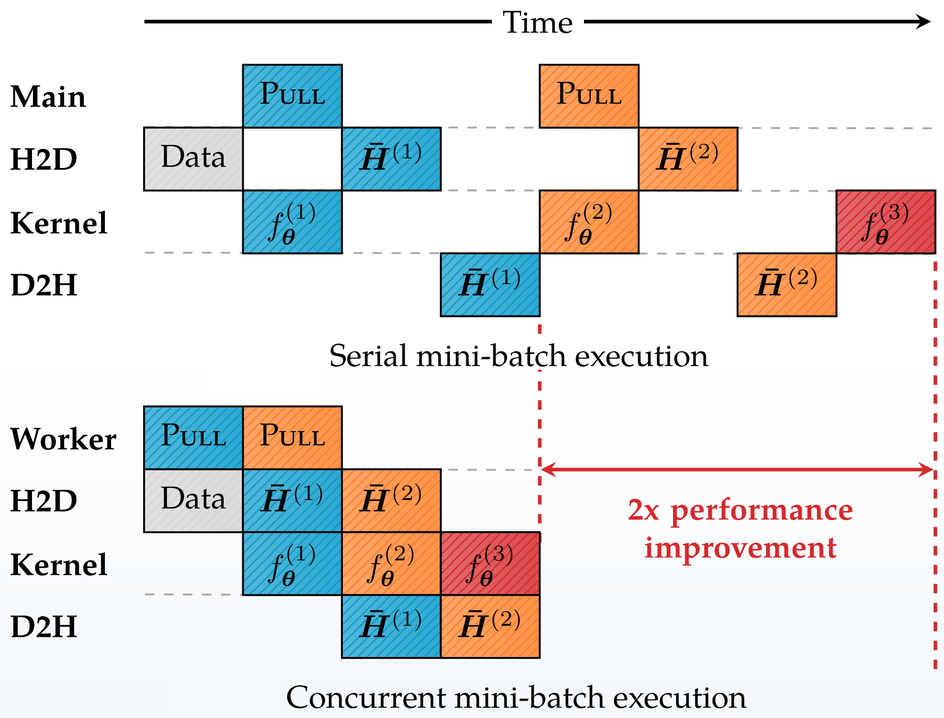
- *local information* 🤗
- *trains over all data* 🤗
- *maintains properties* 🤗
- *constant inference time* 🤗

GNNAutoScale: Fast Historical Embeddings

Frequent data transfers to and from the GPU can cause **major I/O bottlenecks** 😭

- We use **non-blocking device transfers** to counteract 🤗

  1. **Asynchronously copy** histories from out-of-mini-batch nodes to **pinned memory**
  2. ***Asynchronously transfer*** *pinned memory buffers to* ***GPU***
  3. ***Synchronize*** *individual CUDA stream before GPU access*



GNNAutoScale

- ✓ provably maintains the properties and power of the original GNN
- ✓ is fast and memory-efficient despite making use of all available neighborhood information
- ✓ resembles full-batch performance
- ✓ auto-scales any GNN backbone model to large-scale graphs
- ✓ fully open-sourced at `/rusty1s/pyg_autoscale`
  on top of PyTorch Geometric: `/pyg-team/pytorch_geometric`

Fey et al.: *GNNAutoScale: Scalable and Expressive Graph Neural Networks via Historical Embeddings* (ICML 2021)