Exercise 3

Timeline

- 2 weeks for each exercise + Office hours (OH) for questions in between

Holidays and New Year

Deadline always 23:59 CET on due date

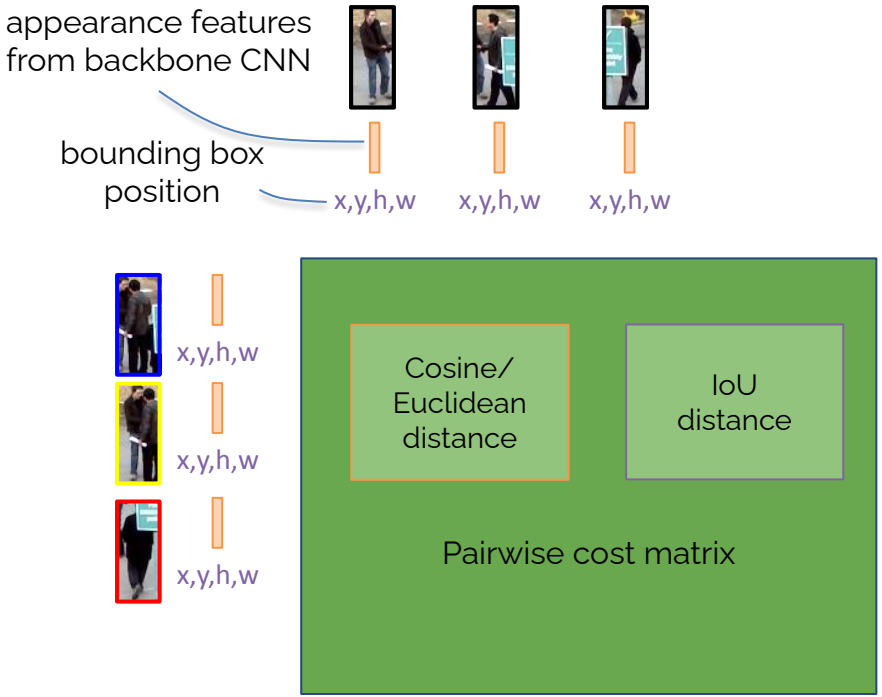
Exercise 3

- Improving the ReID-based tracker from the previous exercise
- Implementing a GNN-based tracker from scratch

Exercise 3

- **Improving the ReID-based tracker from the previous exercise**
- Implementing a GNN-based tracker from scratch

Recap: Distance Matrix



Recap: Distance Matrix

bounding box position

x,y,h,w x,y,h,w x,y,h,w

x,y,h,w

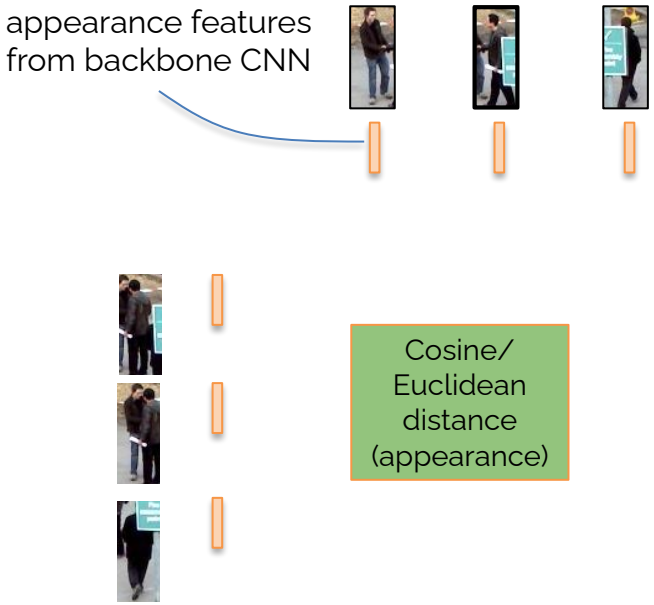
x,y,h,w

x,y,h,w

IoU distance

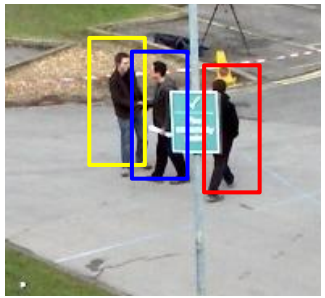
- advanced motion models not covered in this exercise

Recap: Distance Matrix



- features from trained backbone ResNet34

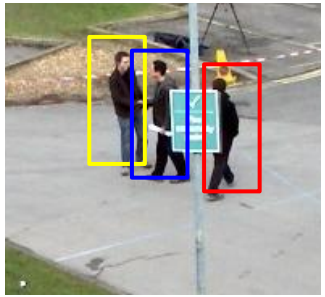
Recap: Hungarian Tracker



Tracks in past frame

Current Frame detections

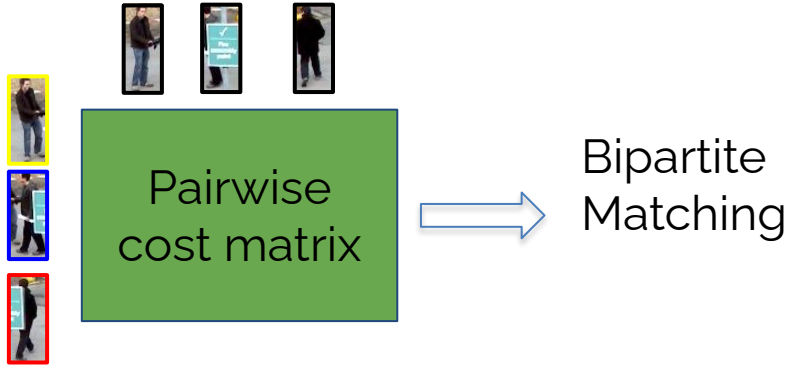
Recap: Hungarian Tracker

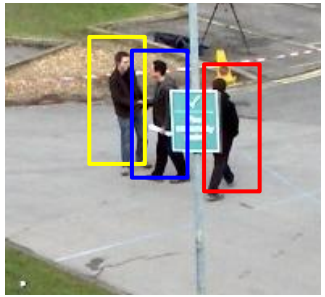


Tracks in past frame

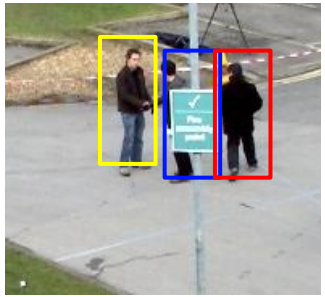
Current Frame detections

Recap: Hungarian Tracker



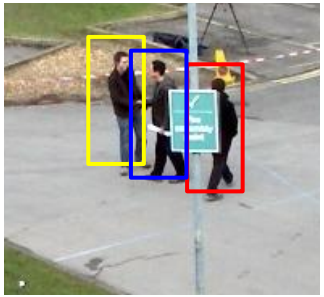


Tracks in past frame



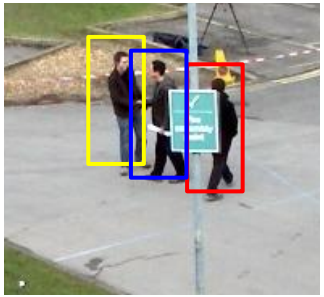
Tracks Frame detections

Recap: Missing Detections



Tracks in past frame

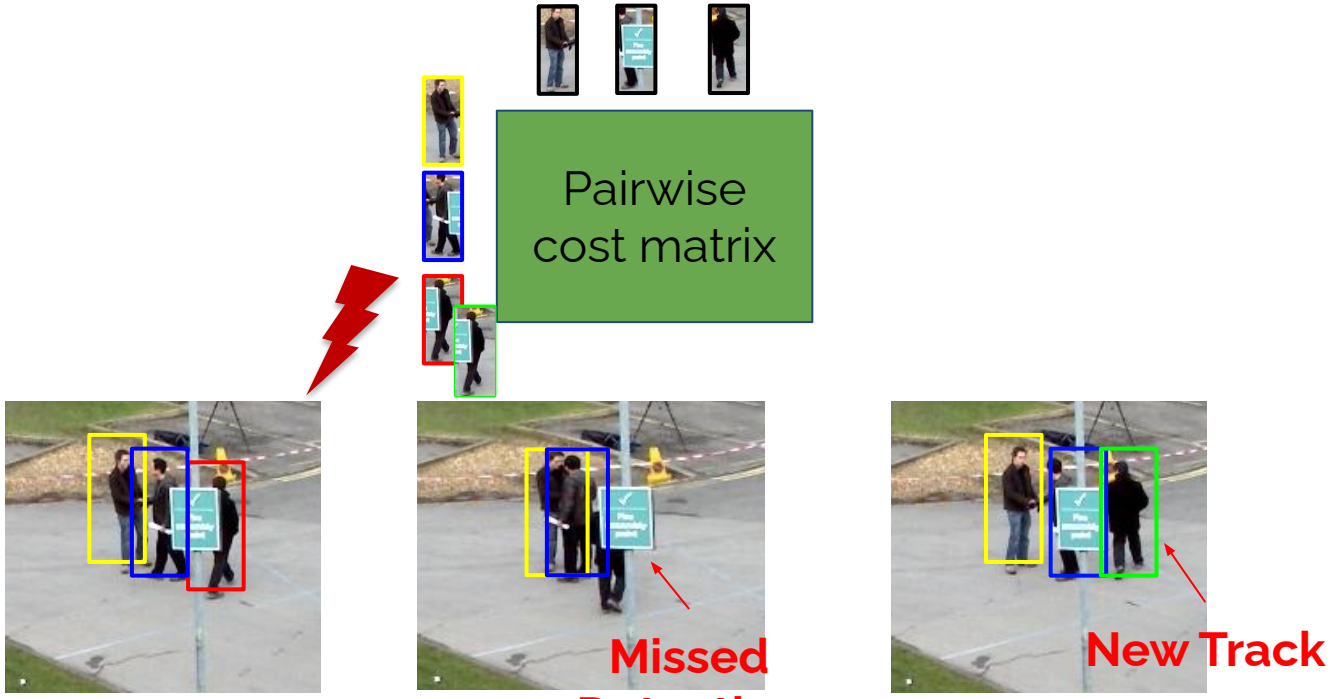
Recap: Missing Detections



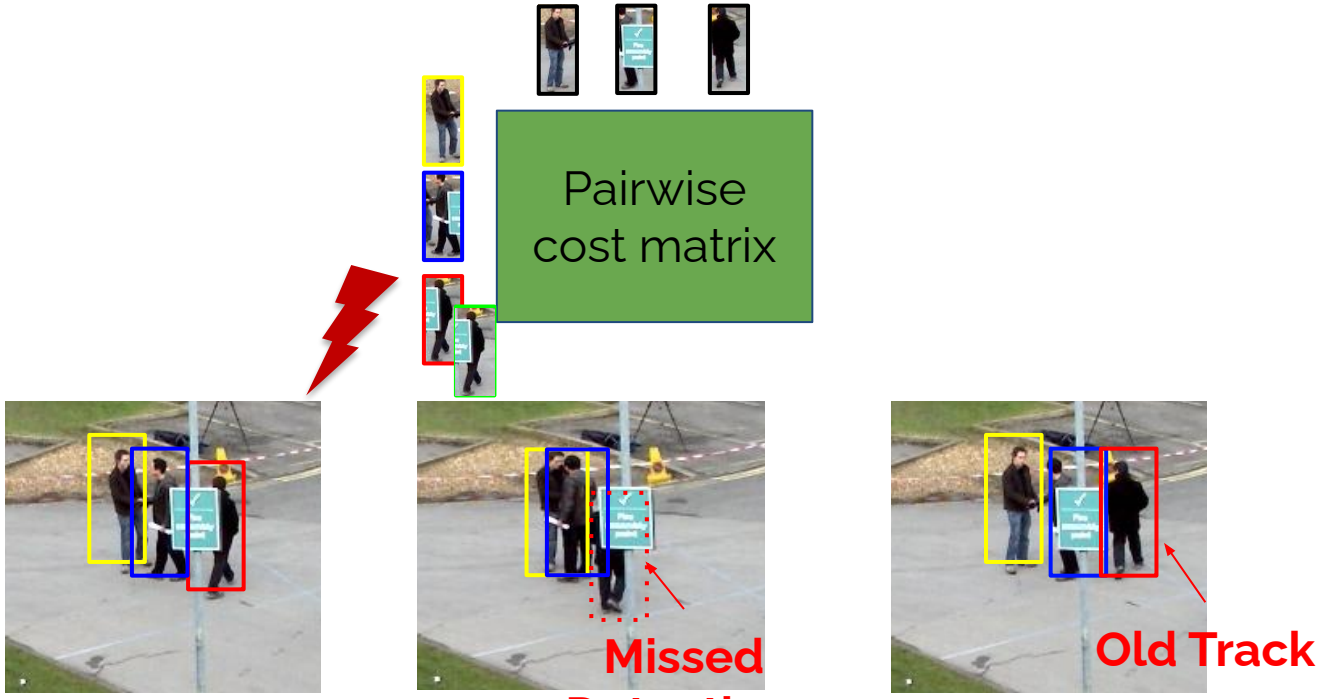
Tracks in past frame

Missed Detection

Recap: Missing Detections



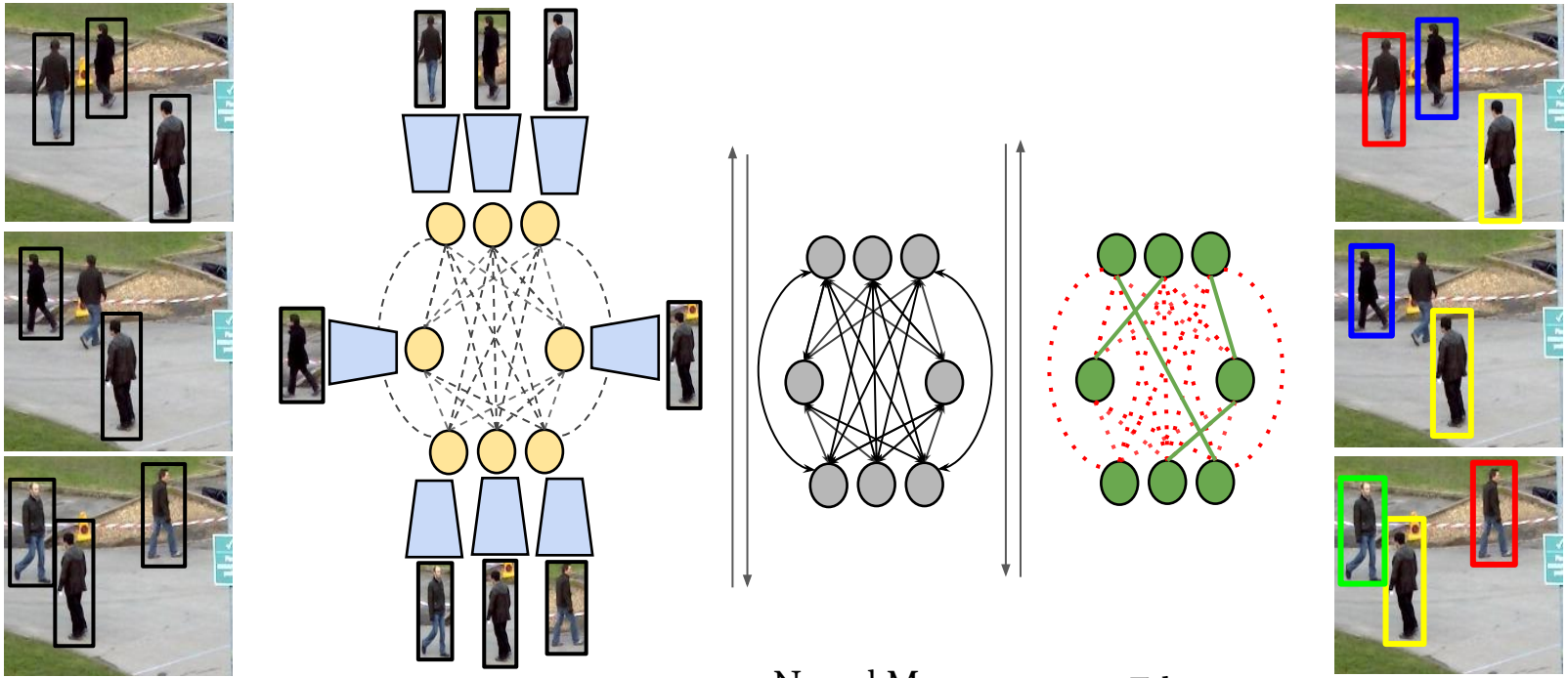
What we want …


Pairwise
cost matrix
Tracks in past
frame
Missed
Detection,
Keep Track
Old Track

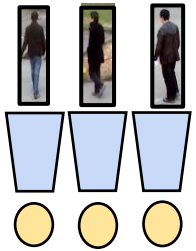
Exercise 3

- Improving the ReID-based tracker from the previous exercise
- **Implementing a GNN-based tracker from scratch**

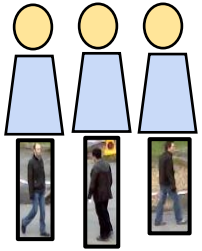
Learning a GNN based Assignment



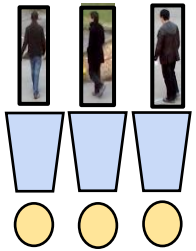
GNN: Initialization



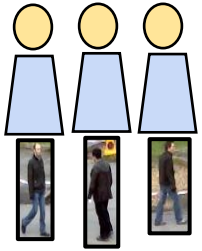
Pairwise cost matrix



GNN: Initialization



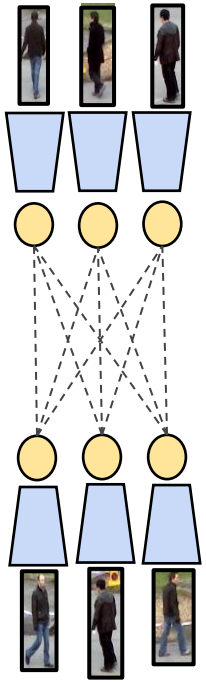
Pairwise cost matrix



Feature Extraction

- Node Embeddings (each of size node_dim)
  - Appearance
- Edge Embeddings (each of size edge_dim)
  - Distance of Appearance
  - Distance of Motion Feature

GNN: Initialization



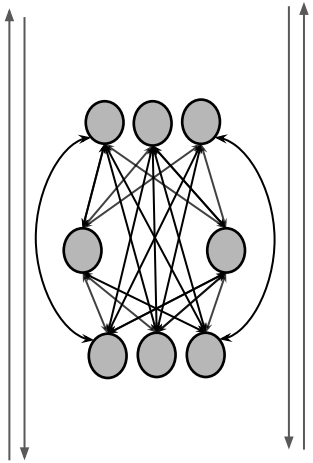
Feature Extraction

```
# Get initial edge embeddings
edge_feats_app = cosine_distance(track_app, current_app)
edge_feats_motion = self.compute_motion_edge_feats(track_coords, current_coords, track_t, curr_t)
edge_feats = torch.cat((edge_feats_motion, edge_feats_app.unsqueeze(-1)), dim=-1)
edge_embeds = self.edge_in_mlp(edge_feats)
initial_edge_embeds = edge_embeds.clone()

# Get initial node embeddings, reduce dimensionality from 512 to node_dim
node_embeds_track = F.relu(self.cnn_linear(track_app))
node_embeds_curr = F.relu(self.cnn_linear(current_app))
```

GNN Update

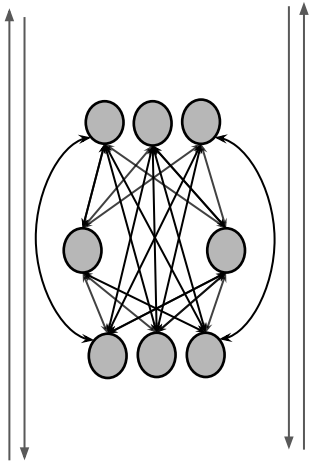
How is the MPN Update different from a Convolutional Layer?



Neural Message Passing

GNN Update

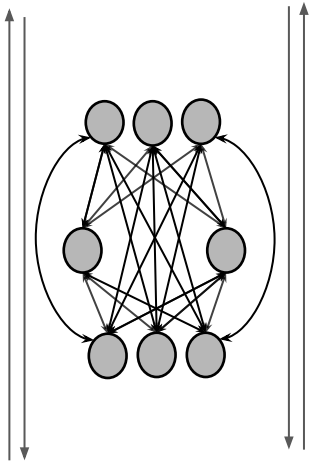
How many frames can we look backwards and include in the decision?



Neural Message Passing

GNN Update

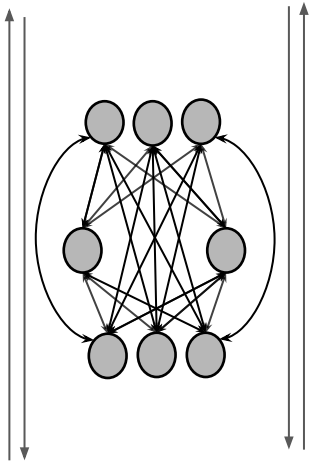
What properties do the update function have?



Neural Message Passing

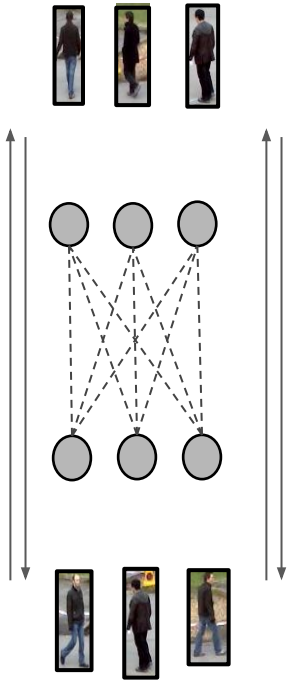
GNN Update

Can we process an arbitrary number of detections?



Neural Message Passing

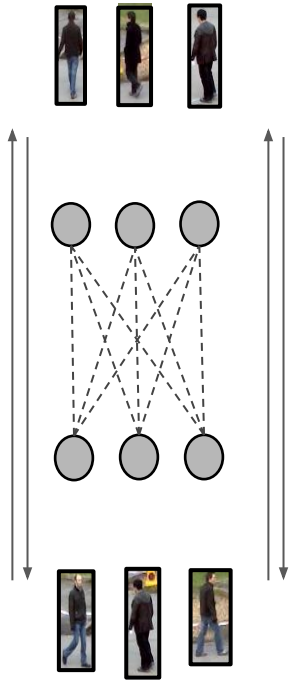
GNN: Update



Neural Message Passing

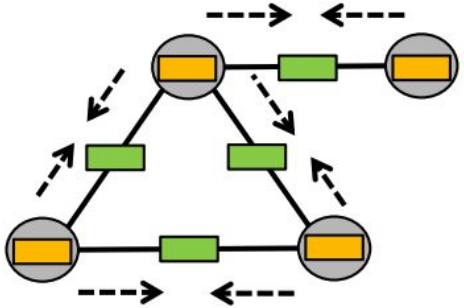
```
classified_edges = []
for _ in range(self.num_steps):
    # Concat current edge embeds with initial edge embeds, increasing the feature dimension
    edge_embeds = torch.cat((edge_embeds, initial_edge_embeds),dim=-1)
    # Override edge_embeds, node_embeds
    edge_embeds, node_embeds_track, node_embeds_curr = self.graph_net(
        edge_embeds=edge_embeds,
        nodes_a_embeds=node_embeds_track,
        nodes_b_embeds=node_embeds_curr
    )
    # Run the classifier on edge embeddings
    classified_edges.append(self.classifier(edge_embeds))
classified_edges = torch.stack(classified_edges).squeeze(-1)
similarity = torch.sigmoid(classified_edges)
return similarity
```

GNN: Update

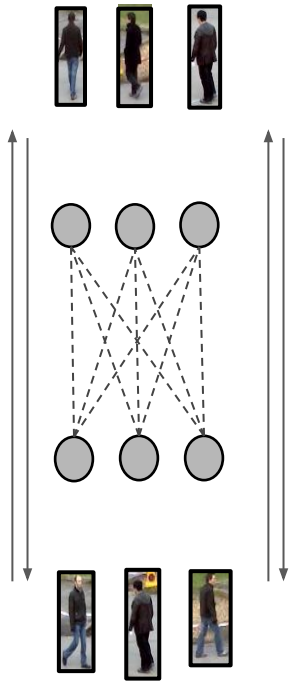


Neural Message Passing

Edge Update

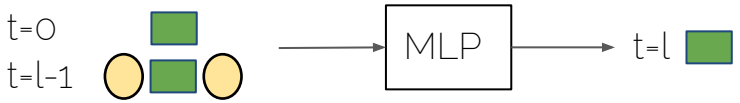


GNN: Update

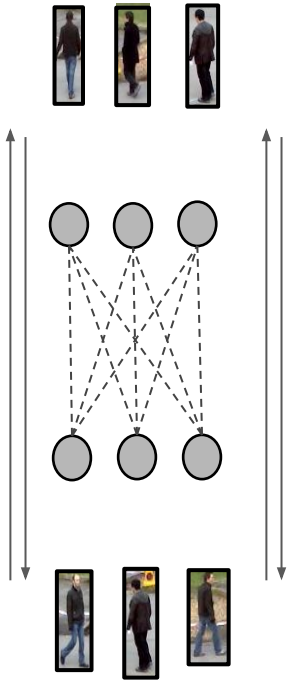


Neural Message Passing

Edge Update

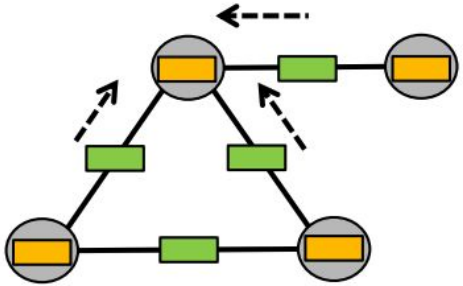


GNN: Update

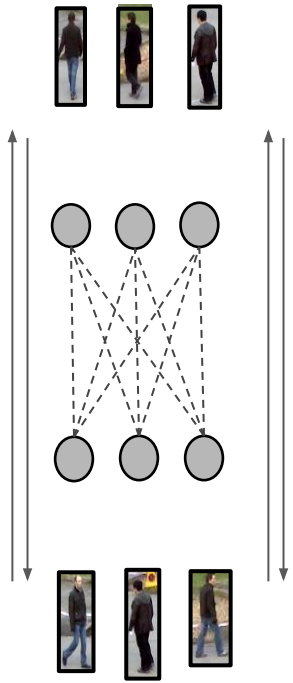


Neural Message Passing

Node Update

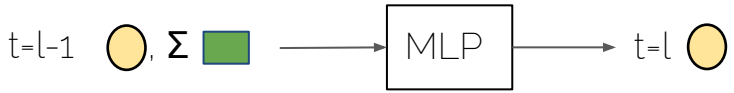


GNN: Update

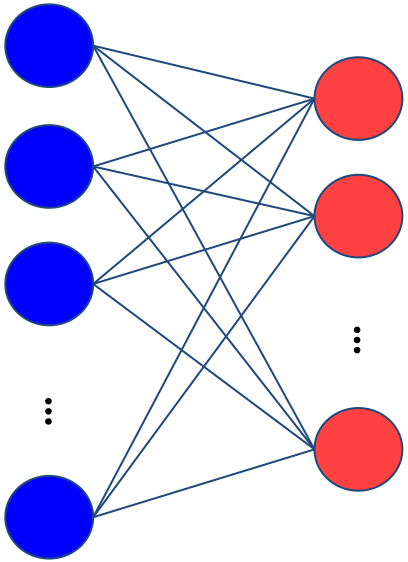


Neural Message Passing

Node Update



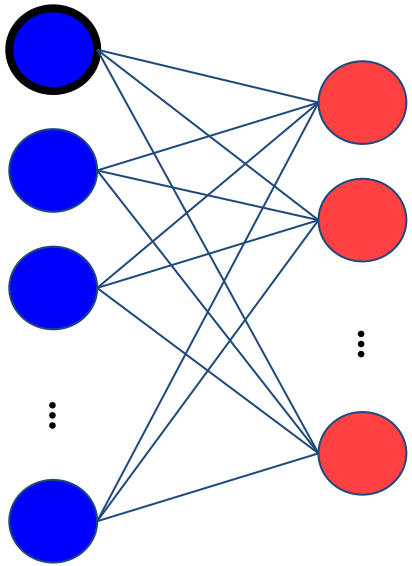
Dimensions of Feature Embeddings



A
(e.g. past tracks)

B
(e.g. detections)

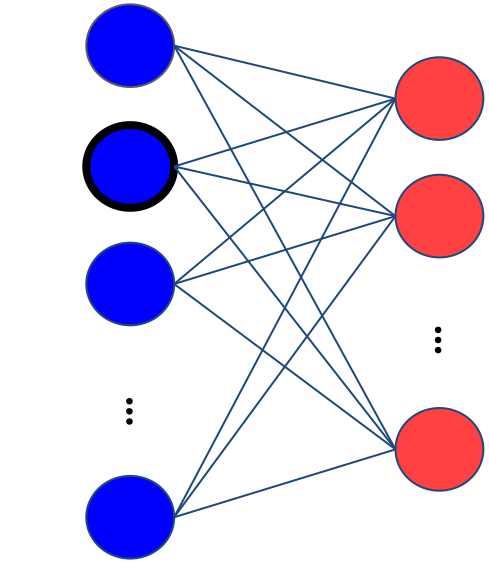
Dimensions of Feature Embeddings



A
(e.g. past tracks)

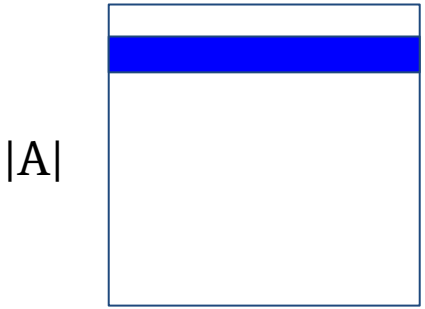
B
(e.g. detections)

Dimensions of Feature Embeddings

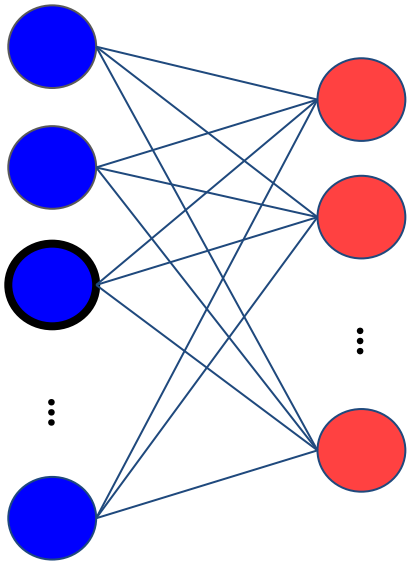


A
(e.g. past tracks)

B
(e.g. detections)

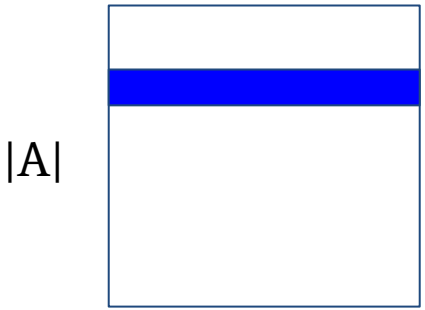


Dimensions of Feature Embeddings

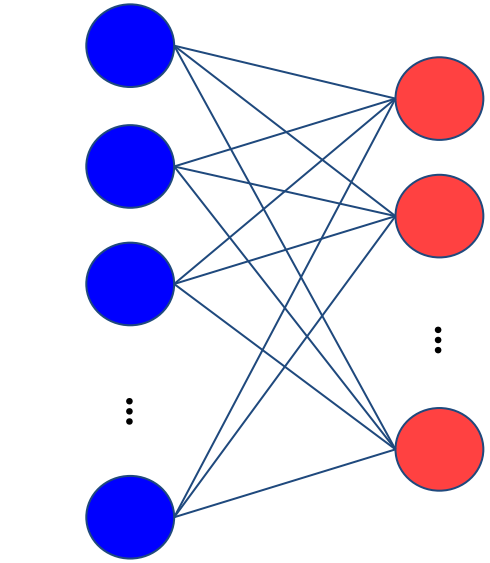


A
(e.g. past tracks)

B
(e.g. detections)

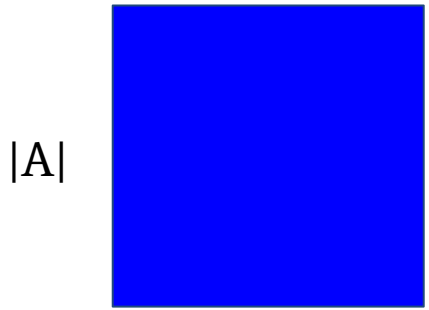


Dimensions of Feature Embeddings

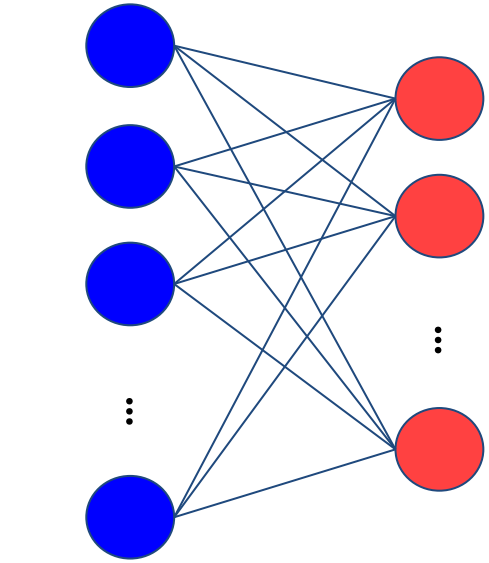


A
(e.g. past tracks)

B
(e.g. detections)

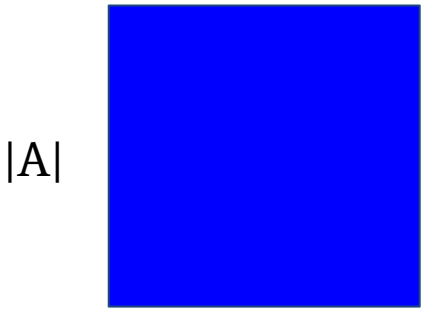


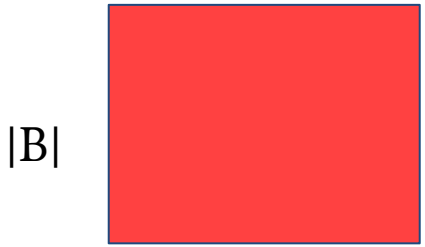
Dimensions of Feature Embeddings



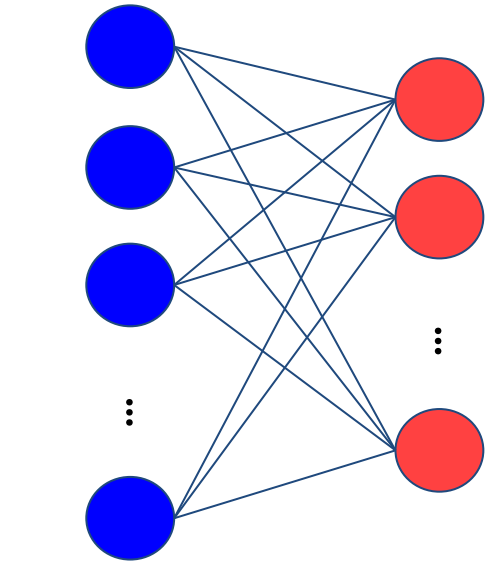
A
(e.g. past tracks)

B
(e.g. detections)



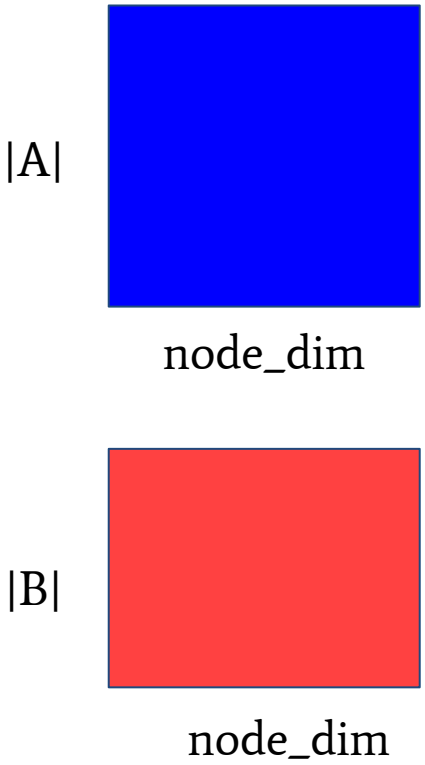


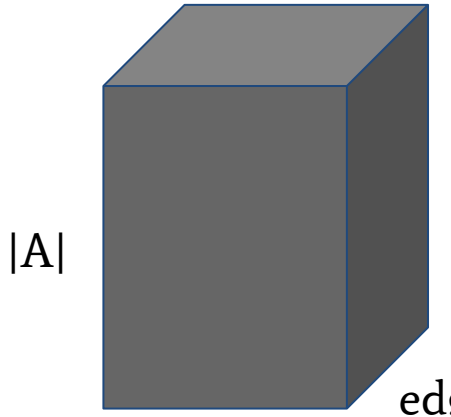
Dimensions of Feature Embeddings



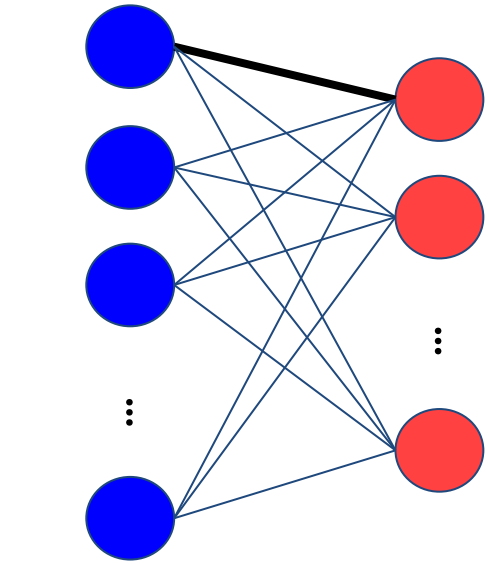
A
(e.g. past tracks)

B
(e.g. detections)



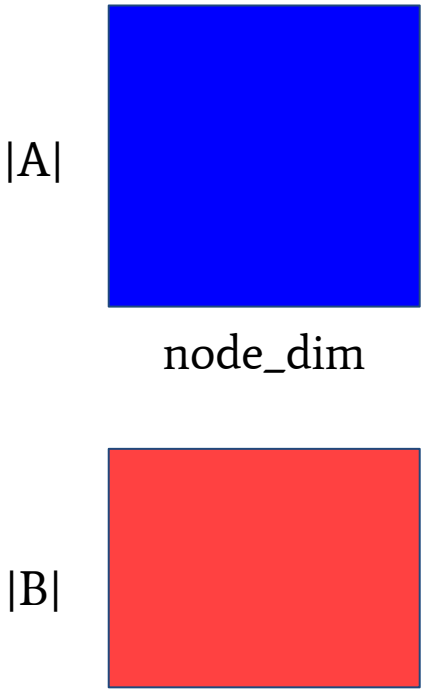


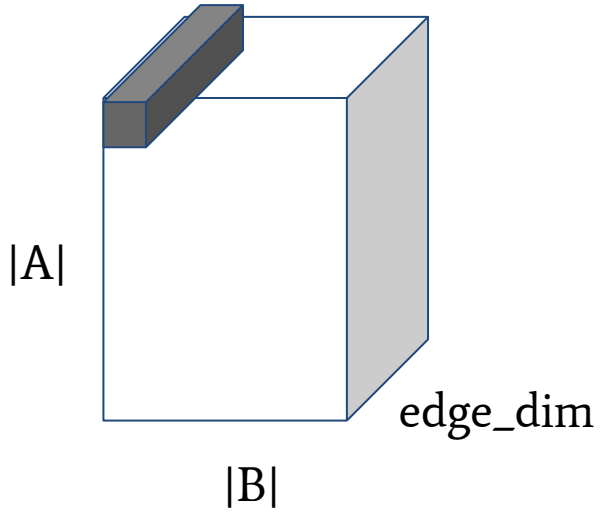
Dimensions of Feature Embeddings



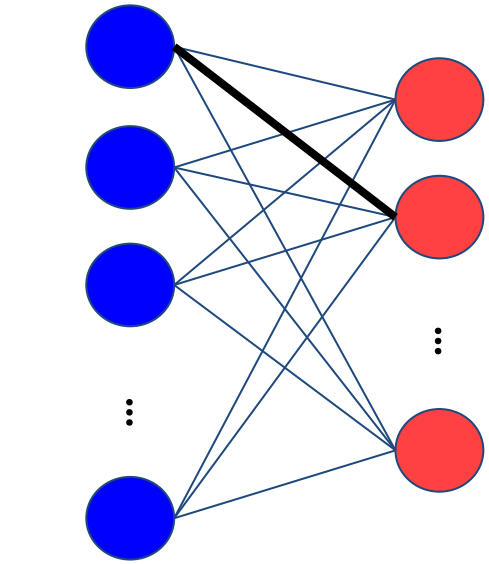
A
(e.g. past tracks)

B
(e.g. detections)



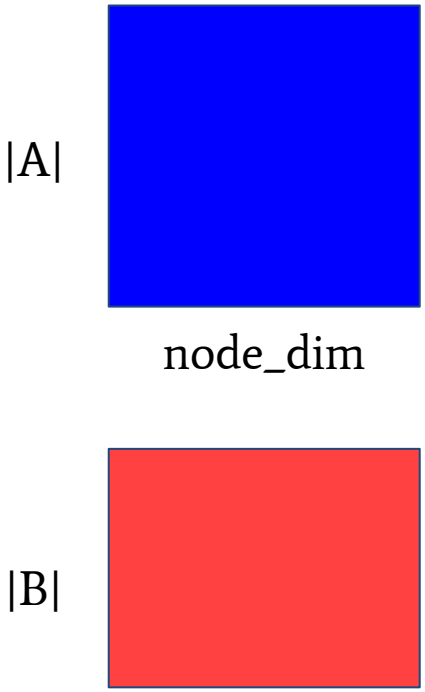


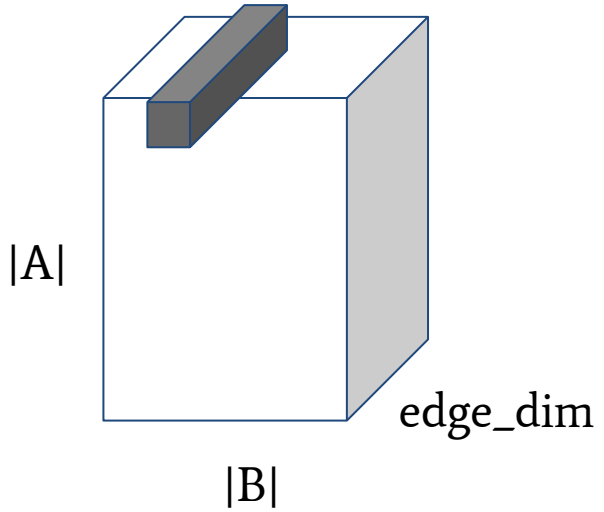
Dimensions of Feature Embeddings



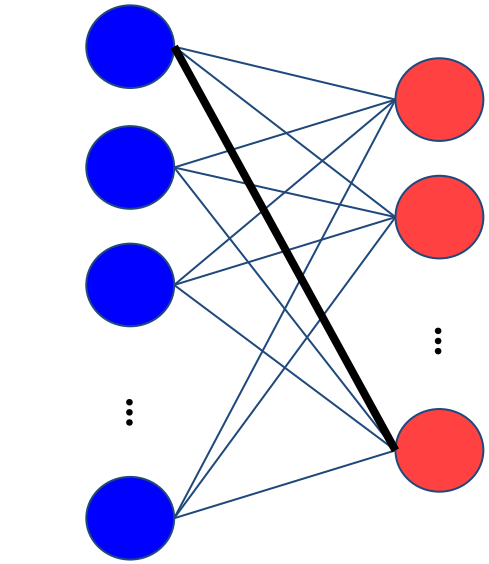
A
(e.g. past tracks)

B
(e.g. detections)



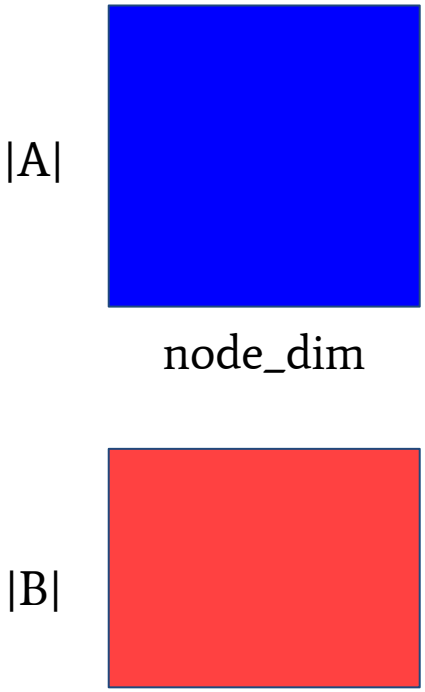


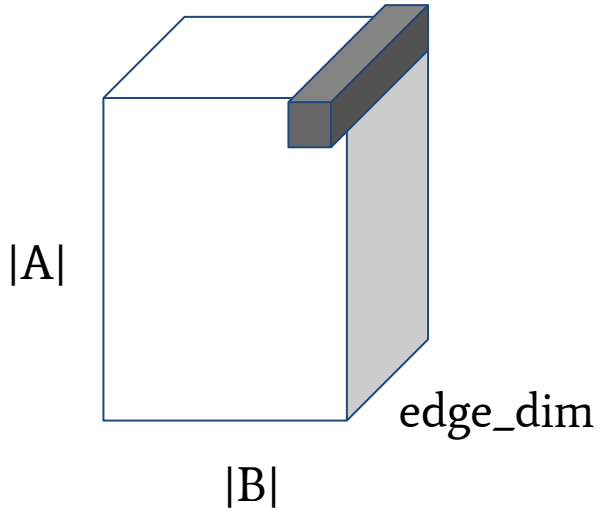
Dimensions of Feature Embeddings



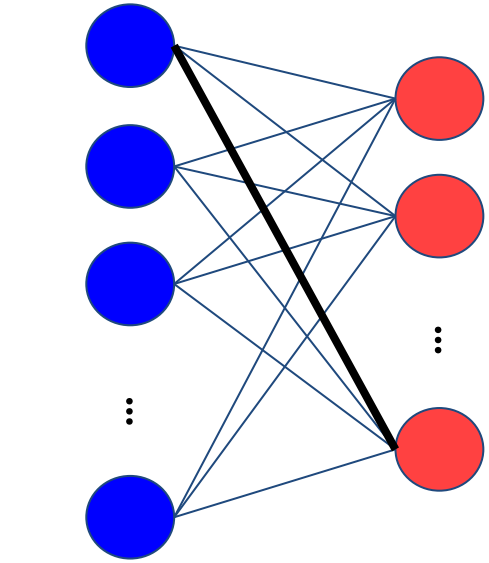
A
(e.g. past tracks)

B
(e.g. detections)



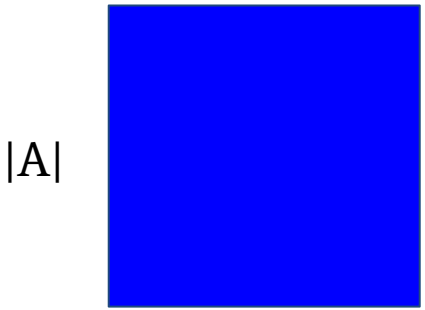


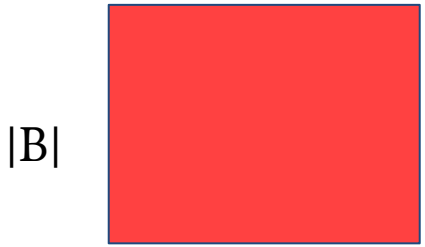
Dimensions of Feature Embeddings

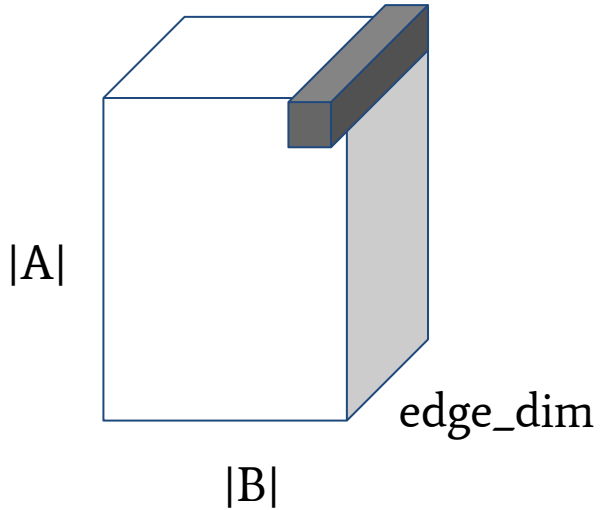


A
(e.g. past tracks)

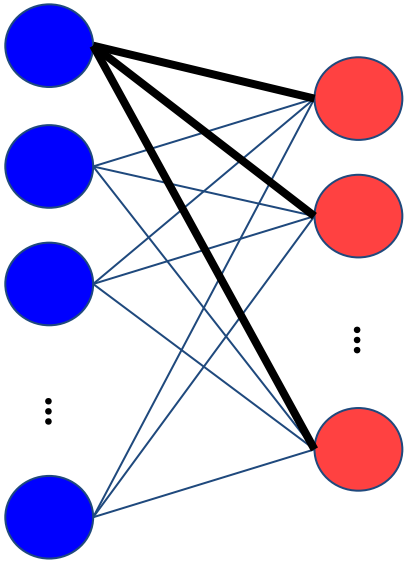
B
(e.g. detections)





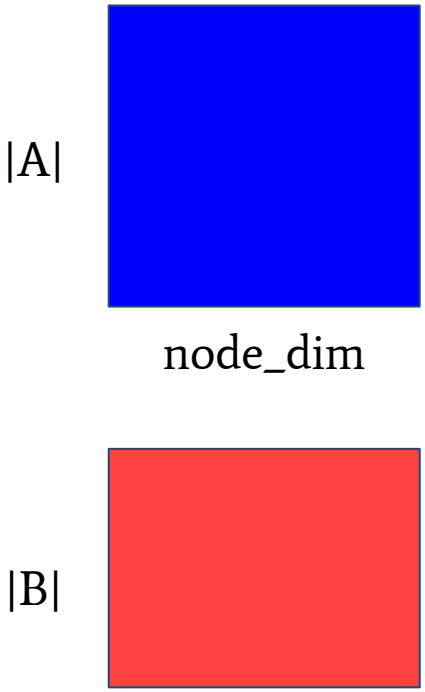


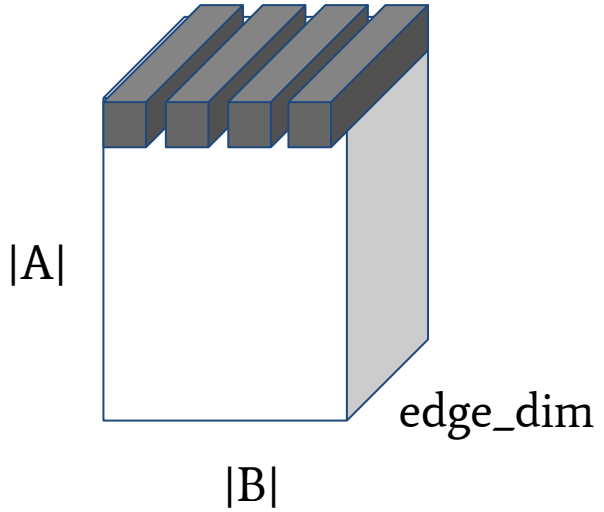
Efficient Node/Edge Update



A
(e.g. past tracks)

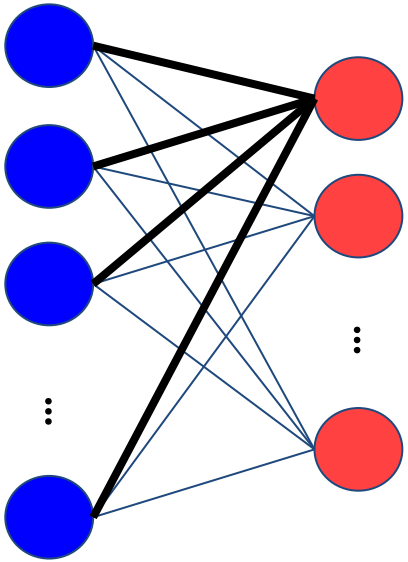
B
(e.g. detections)





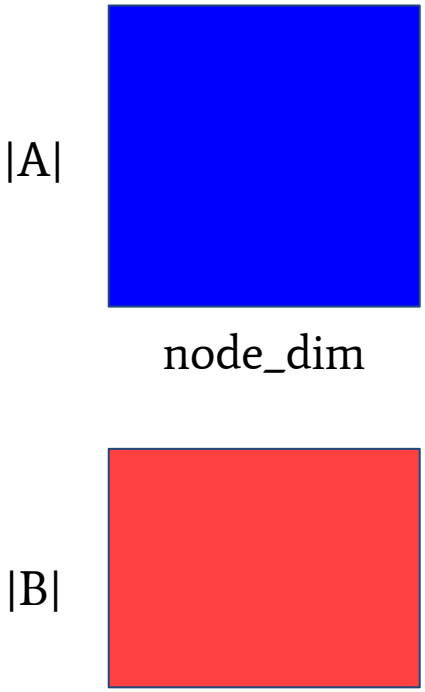
- Node Update: Aggregating over neighbors can be done by summing over rows/ columns
- Edge Update: Expanding the Node Embeddings to concatenate with Edge Embeddings

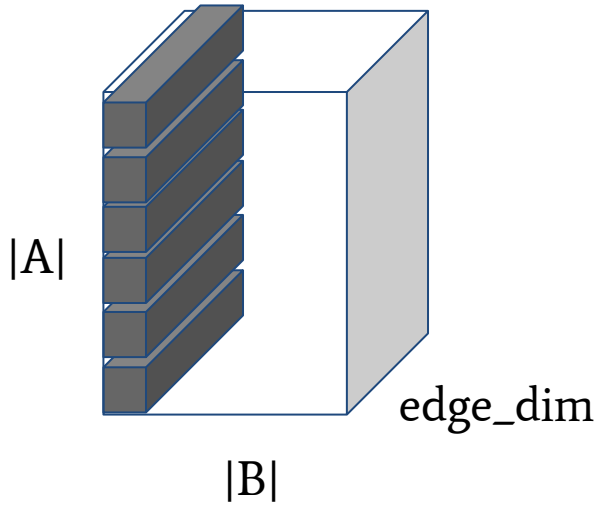
Efficient Node/Edge Update



A
(e.g. past tracks)

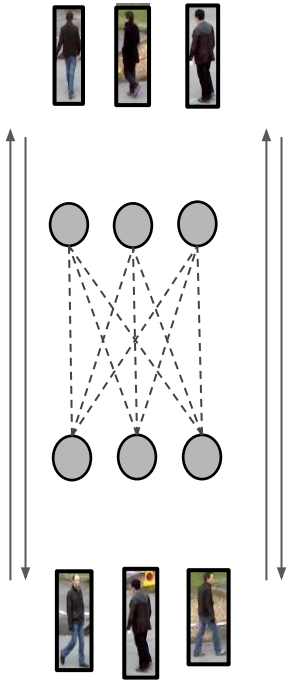
B
(e.g. detections)





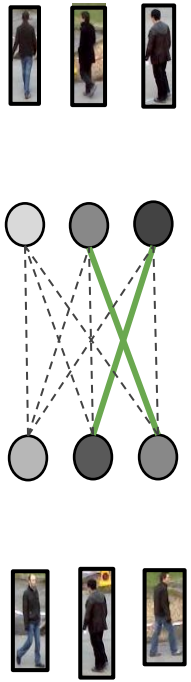
- Node Update: Aggregating over neighbors can be done by summing over rows/ columns
- Edge Update: Expanding the Node Embeddings to concatenate with Edge Embeddings

Applying GNN based Assignment



Neural Message Passing

Applying GNN based Assignment



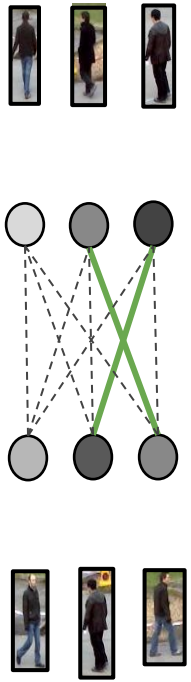
Edge Classification

```
pred_sim = pred_sim[-1]  # Use predictions at last message passing step
distance = 1 - pred_sim

# Do not allow matches when sim < 0.5, to avoid low-confident associations
distance = np.where(pred_sim < 0.5, _UNMATCHED_COST, distance)

# Perform Hungarian matching.
row_idx, col_idx = linear_assignment(distance)
```

Training GNN based Assignment



Edge Classification

Binary Cross Entropy Loss

$$\sum_{i,j} w \cdot y_{i,j} \log\left(\hat{y_{i,j}}\right) + \left(1 - y_{i,j}\right) \log\left(1 - \hat{y_{i,j}}\right)$$

Links

- Test server: https://cv3dst.cvai.cit.tum.de/login
- If you have trouble registering https://forms.gle/yZkZiDiyHxWuNqQG7
- Data for Exercise 03: https://vision.in.tum.de/webshare/g/cv3dst/exercise_03.zip

Links for the individual datasets

- MOT https://vision.in.tum.de/webshare/g/cv3dst/datasets/MOT16.zip
- market https://vision.in.tum.de/webshare/g/cv3dst/datasets/market.zip
- obj_seg https://vision.in.tum.de/webshare/g/cv3dst/datasets/obj_seg.zip
- reid_gnn https://vision.in.tum.de/webshare/g/cv3dst/datasets/reid_gnn.zip