Computer Vision

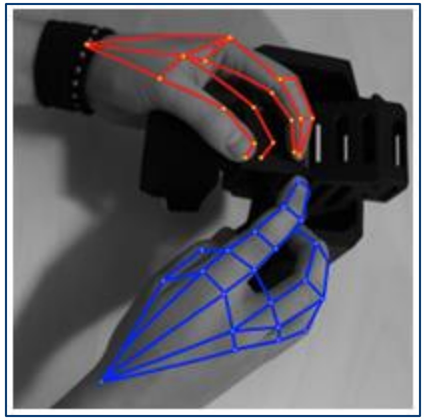
EqMotion Meets Assembly101

Matteo Onger - September 2025

Scope of the Project

- Explore the applicability of the ***EqMotion*** model in domains beyond those previously investigated.

- Goal: evaluate EqMotion for **predicting hand motions** in procedural tasks.
- Dataset: ***Assembly101***.
- Framework: integration into ***4D-Hands*** for **fair comparison** with other models (baseline, equivariant and GMN-based models) thanks to shared hyperparameters and same training/test set.

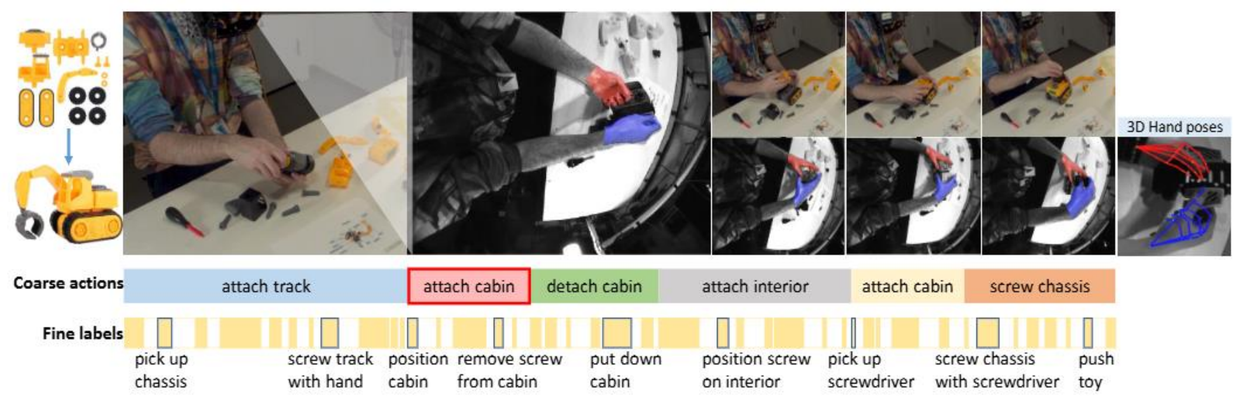


UNIVERSITÀ DEGLI STUDI DI MILANO

Assembly101

- The **Assembly101** dataset was used to evaluate/compare the EqMotion's performance.

- Main characteristics:
  - It consists of **362 unique sequences** of people dis/assembling 101 toy vehicles.
  - Recorded by 12 (8 static and 4 egocentric) **synchronized cameras**.
  - **No strict** ordered recipe/**script**.



An example of sequence. [Source]

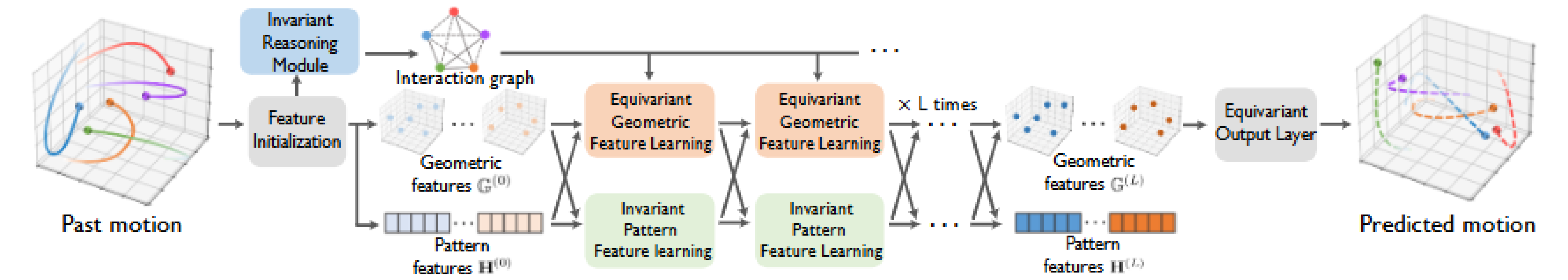
- Sequences are annotated with:
  - Fine-grained actions (ST/ET).
  - Coarse actions (ST/ET).
  - Correct, wrong or recovery action.
  - **42-node** representation of **3D hands positions** (21 per hand).

UNIVERSITÀ DEGLI STUDI DI MILANO

EqMotion

- The **EqMotion** model was specifically designed for **motion prediction tasks**.
- Characteristics:
  - Input: $\boldsymbol{X} \in \mathbb{R}^{N \times T_p \times d}$.
  - Output: $\boldsymbol{Y} \in \mathbb{R}^{N \times T_f \times d}$.
  - **5** main **components** (FI, IRM, EGFL, IPFL, EOL).
  - Preserve **equivariance** and **invariance**.
    - → Ensure consistent predictions.
    - → Lead to faster training.

T_f = num. next positions predicted (1)
T_p = num. previous positions (90)
d = num. of dimension (3)
N = num. of nodes (42)



Schema of the architecture of the EqMotion model. [Source]

EqMotion: Two Key Properties

Equivariance

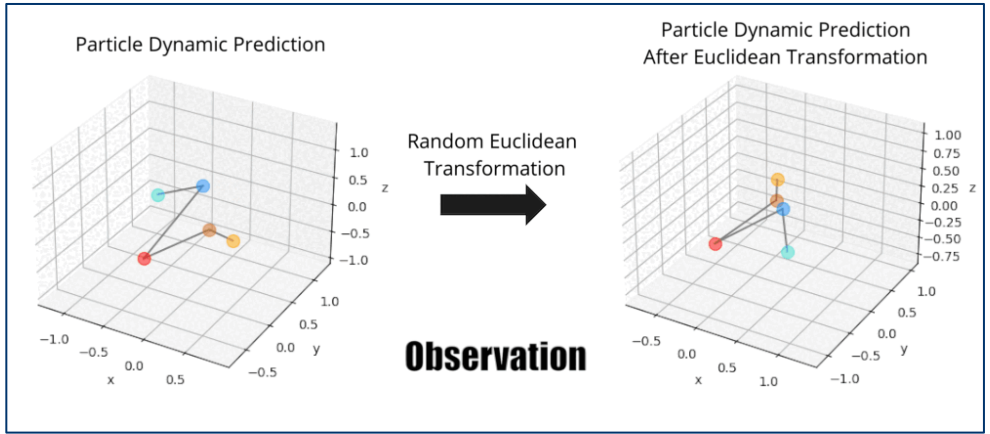
Let x denote an input and $f(\cdot)$ a given operation. The operation f is said to be **equivariant** under and Euclidian transformation if, for every transformation T, it holds that:

$$f\big(T(x)\big) = T(f(x)).$$

Invariance

Let x denote an input and $f(\cdot)$ a given operation. The operation f is said to be **invariant** under and Euclidian transformation if, for every transformation T, it holds that:

$$f(x) = f(T(x)).$$



Example of equivariant and invariant prediction. [Source]

UNIVERSITÀ DEGLI STUDI DI MILANO

EqMotion: Feature Initialization

- The **feature initialization** (FI) layer is designed to obtain the initial **geometric features** $\boldsymbol{G}^{(0)} \in \mathbb{R}^{N\times C\times d}$ and **pattern features** $H^{(0)} \in \mathbb{R}^{N\times D}$.

- For each node i, the initial features are computed according to the following formulas:

$$G_i^{(0)} = \phi_{init\_g}(X_i - \overline{\mathbf{X}}) + \overline{\mathbf{X}} = W_{init\_g} \cdot (X_i - \overline{\mathbf{X}}) + \overline{\mathbf{X}}$$

$$V_i = \Delta X_i \in \mathbb{R}^{T_p\times d}$$
$$p_i^t = \|V_i^t\|_2$$
$$\theta_i^t = \mathrm{angle}(V_i^t, V_i^{t-1})$$
$$h_i^{(0)} = \phi_{\mathrm{init\_h}}([p_i; \theta_i]) \in \mathbb{R}^D$$

C = latent space size of geometric features
D = latent space size of pattern features
ϕ = linear layers or MLPs
W = learnable weights
V = velocity

Implementation details

- In `EqMotion.forward(…)`.
- Main differences:
  - Velocity as input.
  - Concatenation after the embedding.

EqMotion: Invariant Reasoning Module

- The **invariant reasoning module** (IRM) is used to infer the **type of interaction** (if not known a priori) between each **pair of nodes** (complete graph).

- Each node pair (i, j) receives a soft assignment c_{ij} over K (hyperparameter) possible categories according to the following formula:

$$m'_{ij} = \phi_{rm}([h_i^{(0)}; h_j^{(0)}; ||G_i^{(0)} - G_j^{(0)}||_{2,col}])$$

$$p'_i = \sum_{j \in \mathcal{N}_i} m'_{ij}$$

$$h'_i = \phi_{rh}([p'_i; h_i^{(0)}])$$

$$c_{ij} = sm\left(\frac{\phi_{rc}([h'_i; h'_j; ||G_i^{(0)} - G_j^{(0)}||_{2,col}])}{\tau}\right) \in [0,1]^K$$

τ = temperature (1.)

sm = softmax function

Implementation details

- In `EqMotion.calc_category(…)`.
- Main differences:
  - Velocity embedding concatenated to $\boldsymbol{G}^{(0)}$.

UNIVERSITÀ DEGLI STUDI DI MILANO

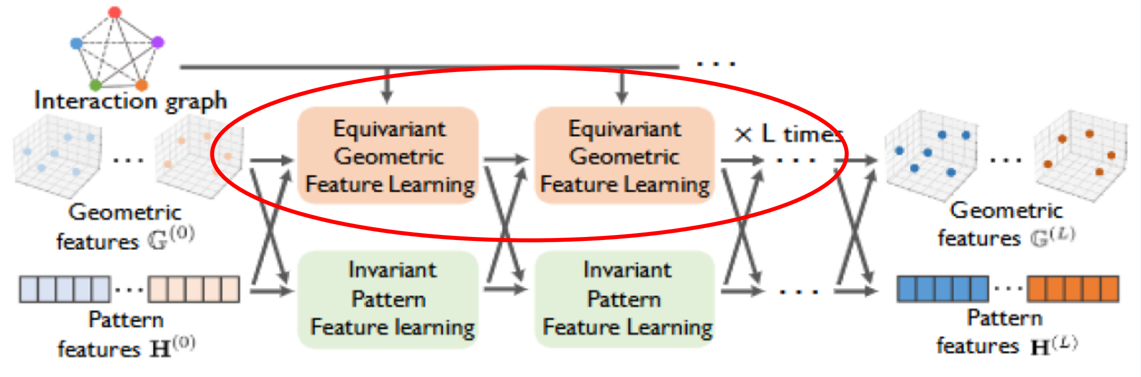
EqMotion: Equivariant Geometric Feature Learning (1)

- The L (hyperparameter) **equivariant geometric feature learning** (EGFL) layers update the geometric features.

- Each layer $l \in \{1, \dots, \mathrm{L}\}$:
  - Input: $\boldsymbol{G}^{(\mathrm{l}-1)}, \mathrm{H}^{(\mathrm{l}-1)}, \mathrm{c}$.
  - **3** main **operations** :
    1. Equivariant inner-agent attention;
    2. Equivariant inter-agent aggregation;
    3. Equivariant non-linear function.
  - Output: $\boldsymbol{G}^{(\mathrm{l})}$.

Implementation details

- In `FeatureLearning.forward(…)`.
- Main differences:
  - Between step (2) and (3), a new velocity embedding is added to $\boldsymbol{G}^{(\mathrm{l})}$.



UNIVERSITÀ DEGLI STUDI DI MILANO

EqMotion: Equivariant Geometric Feature Learning (2)

- **Equivariant inner-agent attention**: attention mechanism is applied to geometric features to capture and exploit **temporal dependencies**.

- So, the geometric features of each node i are first updated according to the following formula:

$$G_i^{(l)} \leftarrow \phi_{att}^{(l)}(h_i^{(l)}) \cdot (G_i^{(l)} - \overline{\mathbf{G}}^{(l)}) + \overline{\mathbf{G}}^{(l)}$$

Equivariant inner-agent attention → Equivariant inter-agent aggregation → Equivariant non-linear function

Equivariant Geometric Feature Learning Layer

Implementation details

- In `FeatureLearning.inner_agent_attention(…)`.

EqMotion: Equivariant Geometric Feature Learning (3)

- **Equivariant inter-agent aggregation**: aggregation is performed by exploiting the previously computed interaction categories to capture **spatial dependencies**.

- After the attention mechanism, the geometric features are updated as follows:

$$e_{ij}^{(l)} = \sum_{k=1}^{K} c_{ij,k} \cdot \phi_k^{(l)}([h_i^{(l)}; h_j^{(l)}; ||G_i^{(l)} - G_j^{(l)}||_{2,col}])$$

$$G_i^{(l)} \leftarrow G_i^{(l)} + \sum_{j \in \mathcal{N}_i} e_{ij}^{(l)} \cdot (G_i^{(l)} - G_j^{(l)})$$

Equivariant inner-agent attention → Equivariant inter-agent aggregation → Equivariant non-linear function

Equivariant Geometric Feature Learning Layer

Implementation details

- In `FeatureLearning.inter_agent_aggregation(…)`.

EqMotion: Equivariant Geometric Feature Learning (4)

- **Equivariant non-linear function**: apply non-linear operations to enhance neural network performance. The terms are adopted in analogy with multi-head attention.

- Finally, each c-th coordinate of geometric features of each node is transformed as follows:

$$Q_i^{(l)} = W_Q^{(l)} \cdot (G_i^{(l)} - \overline{\mathbf{G}}^{(l)})$$

$$K_i^{(l)} = W_K^{(l)} \cdot (G_i^{(l)} - \overline{\mathbf{G}}^{(l)})$$

$$g_{i,c}^{(l+1)} = \begin{cases} q_{i,c}^{(l)} + \overline{\mathbf{G}}^{(l)} & \text{if } \langle q_{i,c}^{(l)}, k_{i,c}^{(l)} \rangle \geq 0 \\ q_{i,c}^{(l)} - \langle q_{i,c}^{(l)}, \frac{k_{i,c}^{(l)}}{\|k_{i,c}^{(l)}\|_2} \rangle \cdot \frac{k_{i,c}^{(l)}}{\|k_{i,c}^{(l)}\|_2} + \overline{\mathbf{G}}^{(l)} & \text{otherwise} \end{cases}$$

Equivariant inner-agent attention → Equivariant inter-agent aggregation → Equivariant non-linear function

Equivariant Geometric Feature Learning Layer

Implementation details

- In `FeatureLearning.non_linear(…)`.

EqMotion: Invariant Pattern Feature Learning

- In parallel, L **invariant pattern feature learning** (IPFL) layers update the pattern features.

- Each layer $l \in \{1, \dots, \mathrm{L}\}$:
  - Input: $\boldsymbol{G}^{(\mathrm{l}-1)}, \mathrm{H}^{(\mathrm{l}-1)}$.
  - Pattern features updated according to the equation below.
  - Output: $\mathrm{H}^{(\mathrm{l})}$.

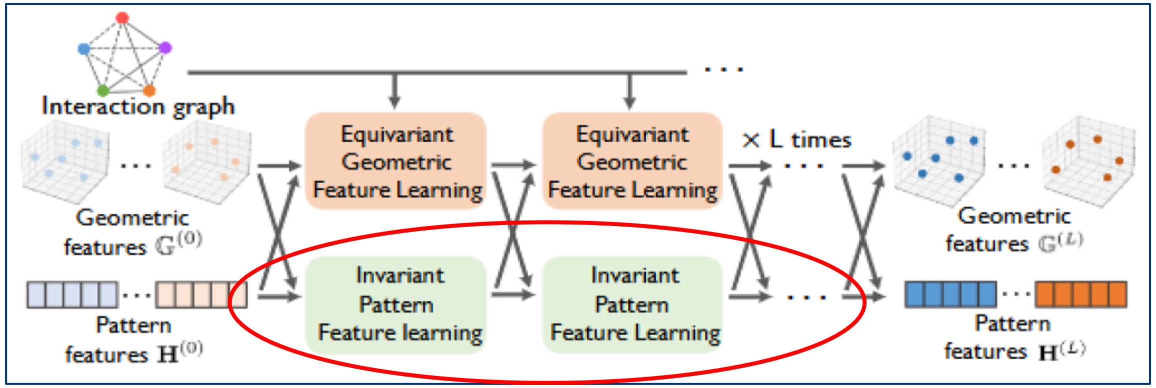
Implementation details

- In `FeatureLearning.edge_model(…)`, `FeatureLearning.node_model(…)` and `FeatureLearning.forward(…)`.

$$m_{ij}^{(l)} = \phi_m^{(l)}([h_i^{(l)}; h_j^{(l)}; ||G_i^{(l)} - G_j^{(l)}||_{2,col}])$$

$$p_i^{(l)} = \sum_{j \in \mathcal{N}_i} m_{ij}^{(l)}$$

$$h_i^{(l+1)} = \phi_h([p_i^{(l)}; h_i^{(l)}])$$

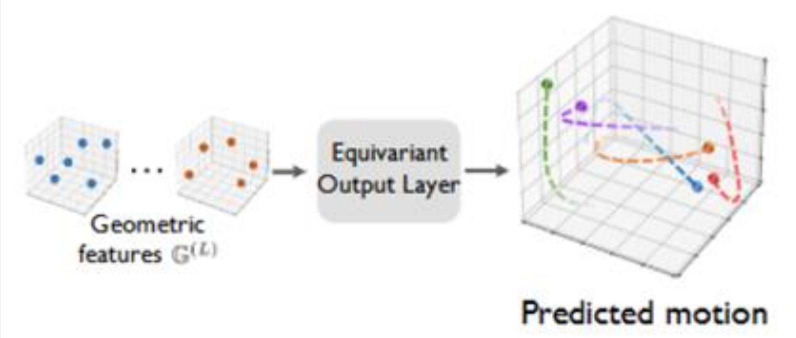


UNIVERSITÀ DEGLI STUDI DI MILANO

EqMotion: Equivariant Output Layer

- After the L feature learning layers, the final geometric features $\boldsymbol{G}^{(\mathrm{L})}$ and pattern features $\mathrm{H}^{(\mathrm{L})}$ are obtained.
- So, for each node i, the final prediction is computed by the **equivariant output layer** (EOL) as follows:

$$\widehat{Y}_i = W_{out} \cdot (G_i^{(L)} - \overline{\mathbf{G}}^{(L)}) + \overline{\mathbf{G}}^{(L)}$$



Implementation details

- In `EqMotion.forward(…)`.

EqMotion: Optional Operations

- A series of **optional operations** can be performed during the computation:
  - **Discrete Cosine Transform**: the data in input can be converted into the frequency domain. The inverse DCT is then applied to the final prediction to recover the motion back in the original domain.
  - **Agent token**: pattern features can be enriched with a learnable token specific to each agent, which is directly optimized during the training.
  - **Residual connections**: they can be enabled within the feature learning layers allowing the model to stabilize training and improve the flow of information across layers.

```python
class EqMotion(nn.Module):
    def forward(self, h: torch.Tensor, x: torch.Tensor, vel: torch.Tensor, edge_attr: torch.Tensor|None = None) ->
        # Sec. B.1: Optional Operation - DCT transform
        if self.apply_dct:
            # Normalize `x` (keep dims for broadcasting)
            x_mean = torch.mean(x, dim=(1,2), keepdim=True)  # -> (B, 1, 1, C)
            x = x - x_mean

            # Discrete cosine transform
            dct_m, _ = self.get_dct_matrix(self.in_channel, x)
            dct_m = dct_m[None, None, :, :].repeat(batch_size, num_agents, 1, 1)  # -> (B, N, T_in, T_in)

            # Inverse discrete cosine transform
            _, idct_m = self.get_dct_matrix(self.out_channel, x)
            idct_m = idct_m[None, None, :, :].repeat(batch_size, num_agents, 1, 1)  # -> (B, N, T_out, T_out)

            # Eq. x_i <- W_dct * (x_i - x_mean)
            x = torch.matmul(dct_m, x)      # -> (B, N, T_in, C)
            vel = torch.matmul(dct_m, vel)  # -> (B, N, T_in, C)
```

```python
class FeatureLearning(nn.Module):
    def node_model(self, h: torch.Tensor, edge_feat: torch.Tensor) -> torch.Tensor:
        return out + h if self.residual else out
```

```python
class EqMotion(nn.Module):
    def forward(self, h: torch.Tensor, x: torch.Tensor, vel: torch.Tensor, edge_attr: torch.Tensor|None = None) ->
        # Optional Operation: Agent Token
        if self.add_agent_token:
            h = self.agent_mlp(
                torch.cat(
                    [h, self.agent_token.repeat(batch_size, 1, 1)],
                    dim = -1
                )
            ) #  (B, N, 2*H_nf) -> (B, N, H_nf)
```

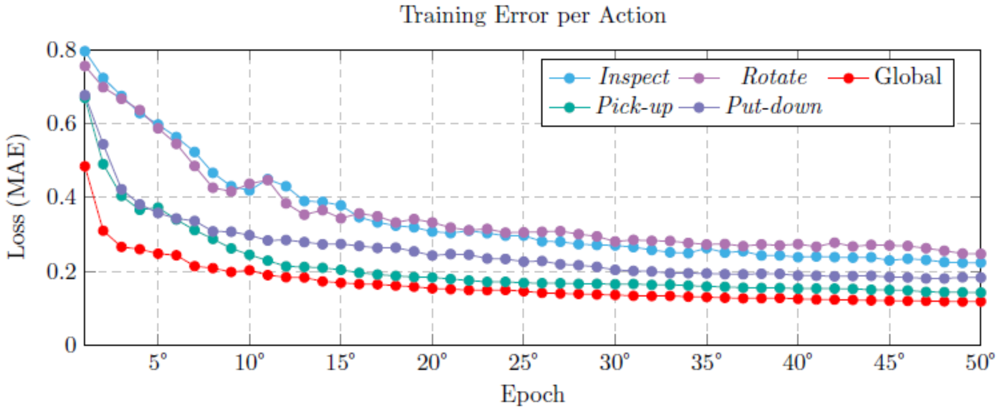
Code snippet showing the implementation.

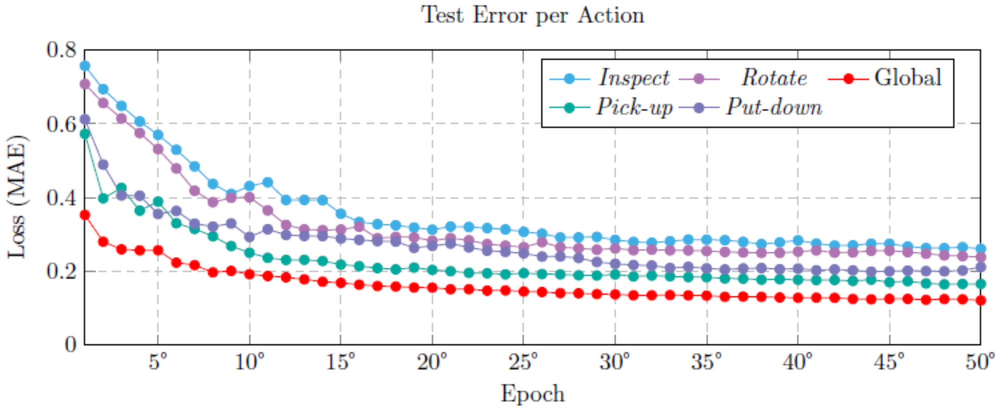
Experimental Results: Action Prediction

- The first experiment investigates how EqMotion's performance varies when **training is restricted to** predicting hand movements during the execution of **specific actions**.
- All hyperparameters are set to the default values:
  - K = 4;
  - L = 2;
  - C = 32.
  - D = 32.

Training & Test Error per Action

Epoch	*Inspect* $\mathcal{L}_{\text{train}}$	*Inspect* $\mathcal{L}_{\text{test}}$	*Rotate* $\mathcal{L}_{\text{train}}$	*Rotate* $\mathcal{L}_{\text{test}}$	Global $\mathcal{L}_{\text{train}}$	Global $\mathcal{L}_{\text{test}}$	*Pick-up* $\mathcal{L}_{\text{train}}$	*Pick-up* $\mathcal{L}_{\text{test}}$	*Put-down* $\mathcal{L}_{\text{train}}$	*Put-down* $\mathcal{L}_{\text{test}}$
1°	0.80	0.76	0.76	0.71	0.48	0.35	0.67	0.57	0.68	0.61
5°	0.60	0.57	0.59	0.53	0.25	0.26	0.37	0.39	0.36	0.36
10°	0.42	0.43	0.44	0.49	0.20	0.19	0.24	0.25	0.30	0.29
25°	0.30	0.31	0.31	0.27	0.15	0.15	0.17	0.20	0.23	0.25
35°	0.25	0.28	0.27	0.25	0.13	0.13	0.16	0.18	0.19	0.21
50°	0.22	0.26	0.25	0.24	0.12	0.12	0.14	0.17	0.18	0.21
Size	163		148		2216		514		631	



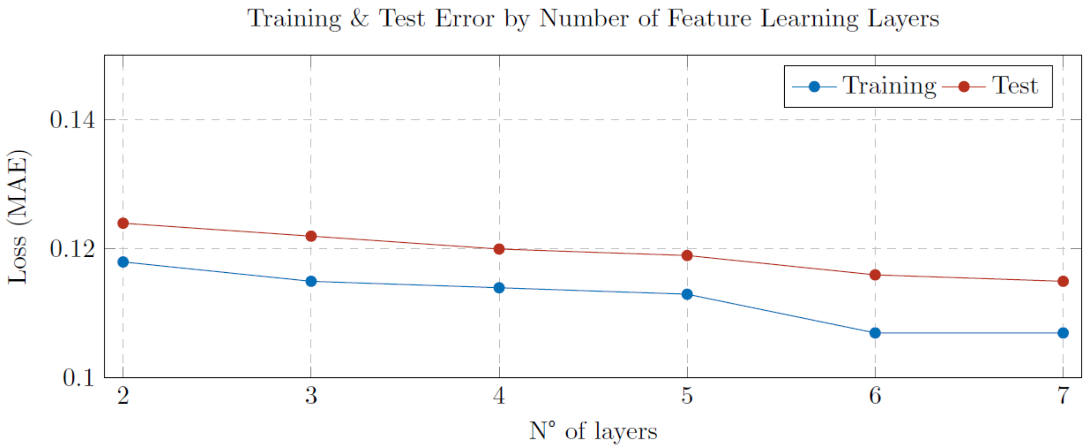


UNIVERSITÀ DEGLI STUDI DI MILANO

Experimental Results: Impact of Hyperparameters (1)

- The first hyperparameter analysed is the **number of feature learning layers** L.
  It directly affects the number of trainable parameters, and so the level of complexity it can handle.
- A training set comprising all actions was used and all the other hyperparameters were fixed at their default values.

Training & Test Error by Number of Feature Learning Layers			
N° of Layers	N° of parameters	$\mathcal{L}_{\text{train}}$	$\mathcal{L}_{\text{test}}$
2	69,188	0.118	0.124
3	91,044	0.115	0.122
4	112,900	0.114	0.120
5	134,756	0.113	0.119
6	156,612	0.107	0.116
7	178,468	0.107	0.115

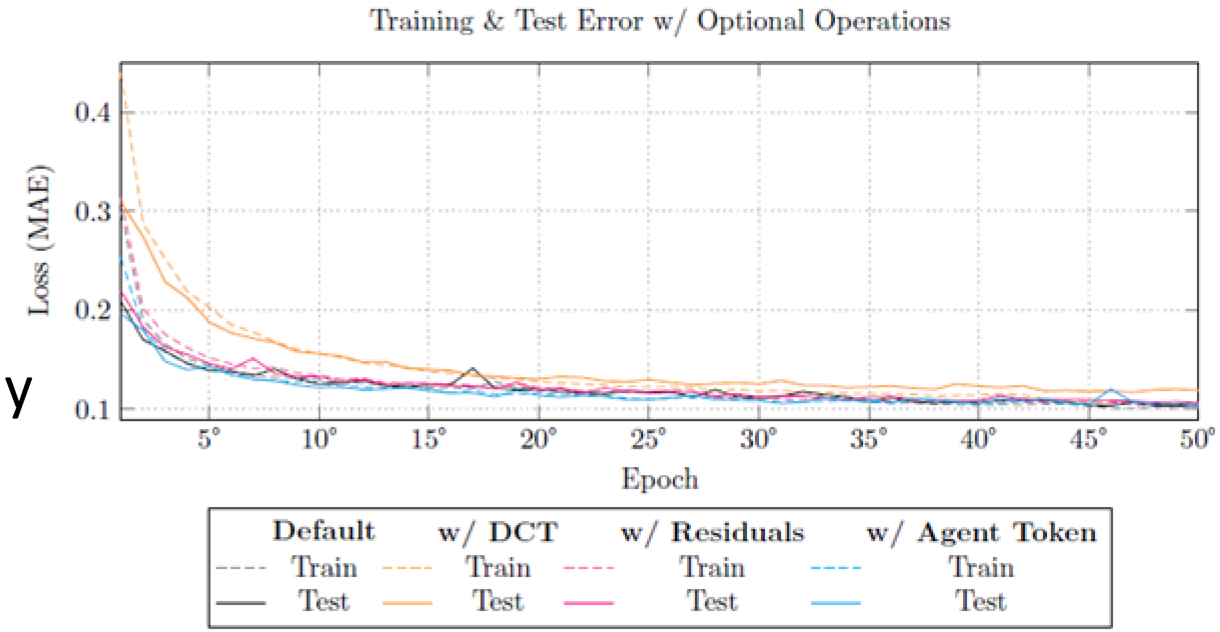


Experimental Results: Impact of Hyperparameters (2)

- Subsequently, **the impact of the optional operations** on the model's performance was examined.
- Even when these operations are combined, the loss does not fall below 0.10 in either the training or testing phases.
- Number of parameters remains essentially unchanged.

Training & Test Error w/ Optional Operations

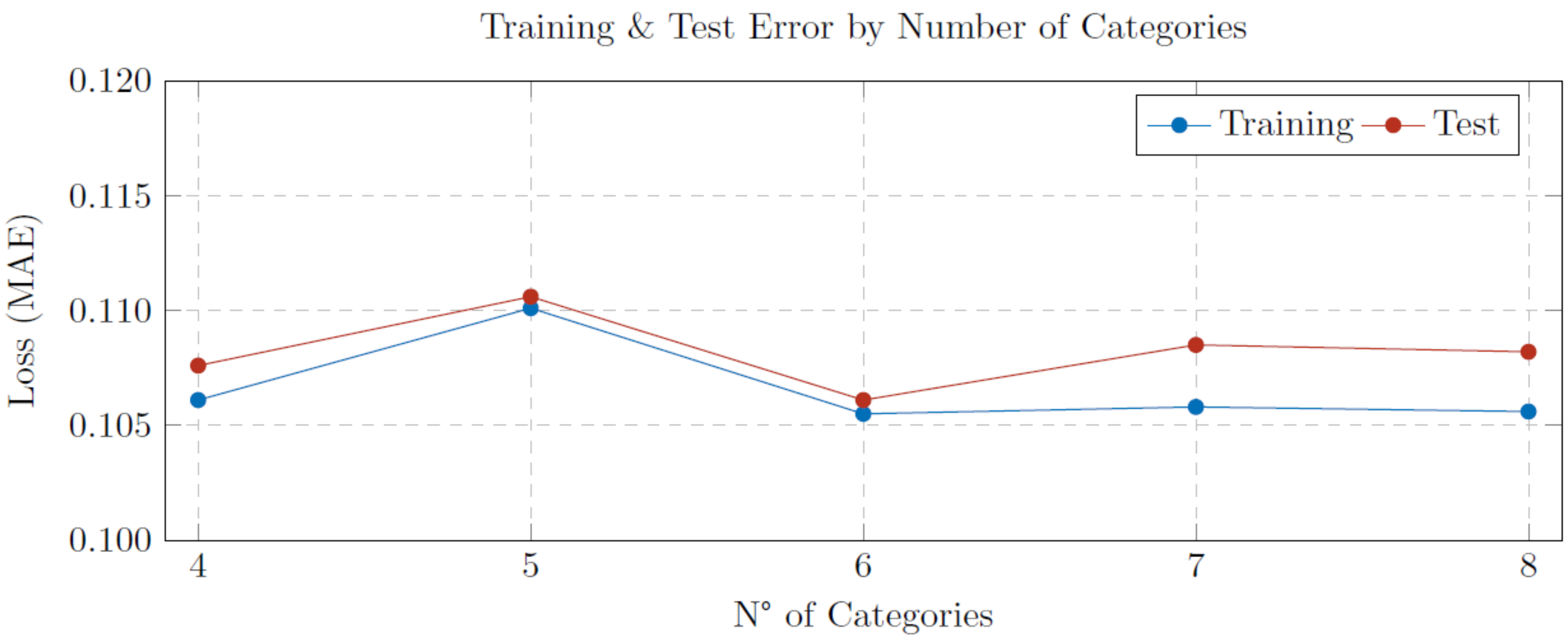
Epoch	Default		w/ DCT		w/ Residuals		w/ Agent Token	
	$\mathcal{L}_{train}$	$\mathcal{L}_{test}$	$\mathcal{L}_{train}$	$\mathcal{L}_{test}$	$\mathcal{L}_{train}$	$\mathcal{L}_{test}$	$\mathcal{L}_{train}$	$\mathcal{L}_{test}$
1°	0.21	0.21	0.44	0.31	0.22	0.22	0.20	0.20
5°	0.14	0.14	0.20	0.19	0.15	0.15	0.14	0.14
10°	0.13	0.13	0.16	0.16	0.13	0.13	0.12	0.12
35°	0.11	0.11	0.12	0.12	0.11	0.11	0.11	0.11
50°	0.10	0.10	0.11	0.12	0.11	0.11	0.10	0.10



UNIVERSITÀ DEGLI STUDI DI MILANO

Experimental Results: Impact of Hyperparameters (3)

- By contrast, increasing the **number of categories** K has a stronger influence on the model's parameter count; however, similar to the optional operations, it does **not significantly affect the final loss**.



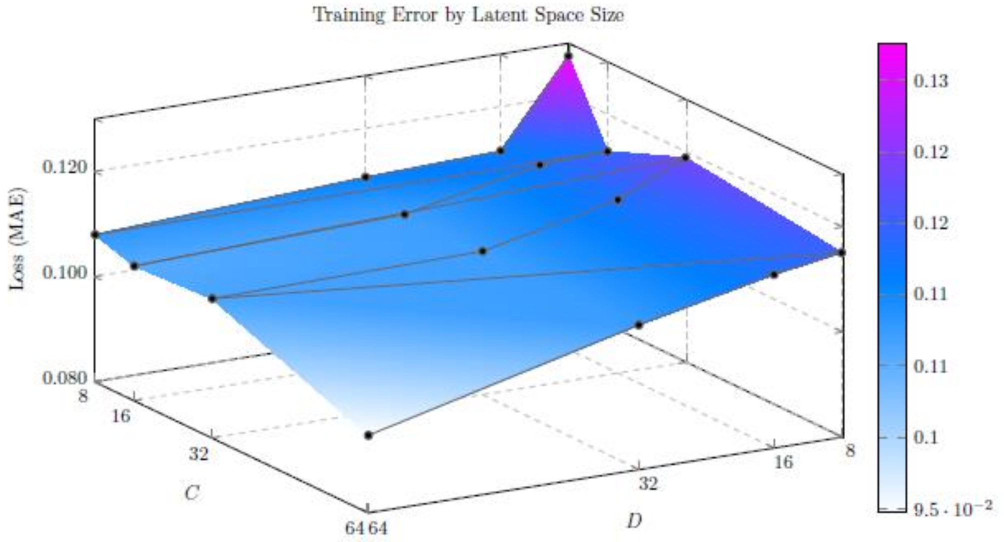
Experimental Results: Impact of Hyperparameters (4)

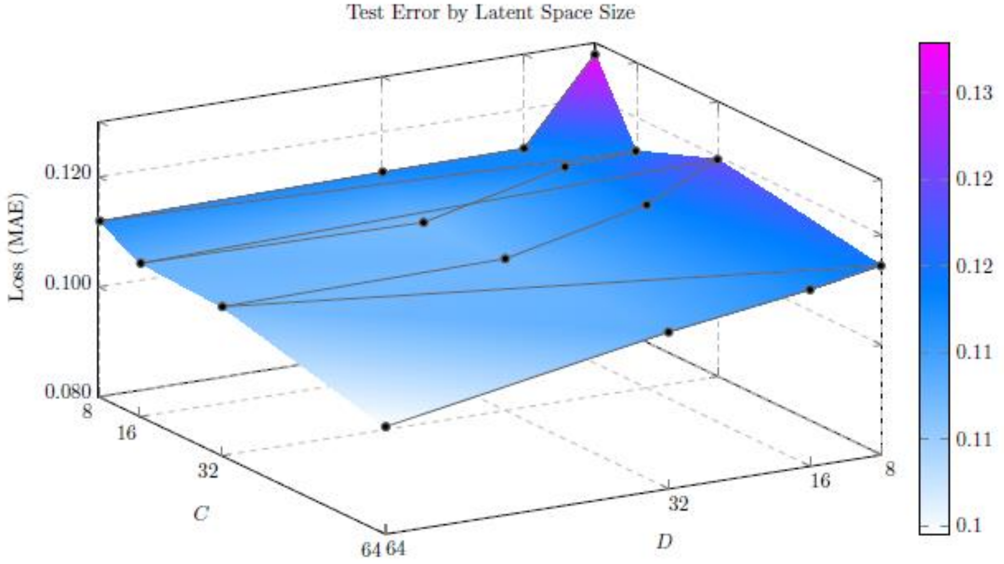
- Last hyperparameters tested are **latent dimensions** of geometric (C) and pattern (D) features.

- Enlarging latent spaces *(C, D)* **reduces final loss**, but requires a **trade-off**:
  - **Slow decrease in final loss**.
  - **Rapid increase in trainable parameters** → higher computational demands.

Training & Test Error by Latent Space Size

		D: 8	D: 16	D: 32	D: 64
C	8	0.128/0.128 (6K)	0.112/0.113 (11K)	0.111/0.112 (24K)	0.108/0.112 (66K)
	16	0.113/0.114 (14K)	0.112/0.113 (20K)	0.107/0.107 (36K)	0.106/0.108 (83K)
	32	0.119/0.119 (40k)	0.113/0.113 (66K)	0.107/0.108 (69k)	0.106/0.107 (127K)
	64	0.115/0.114 (134K)	0.113/0.112 (148K)	0.107/0.108(178K)	0.095/0.098 (257K)

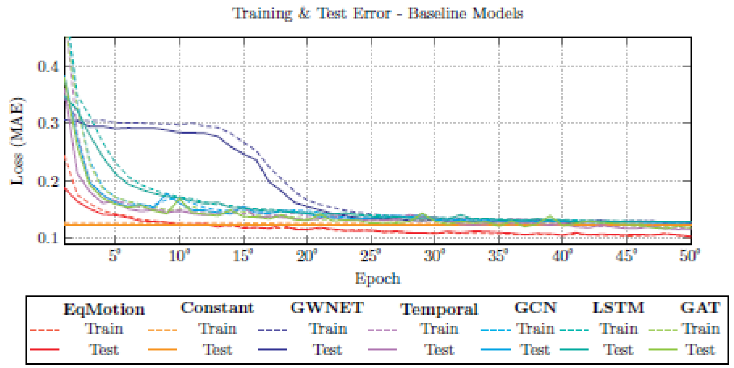


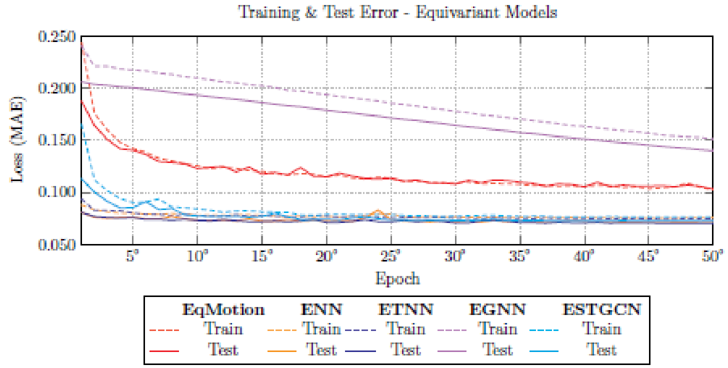


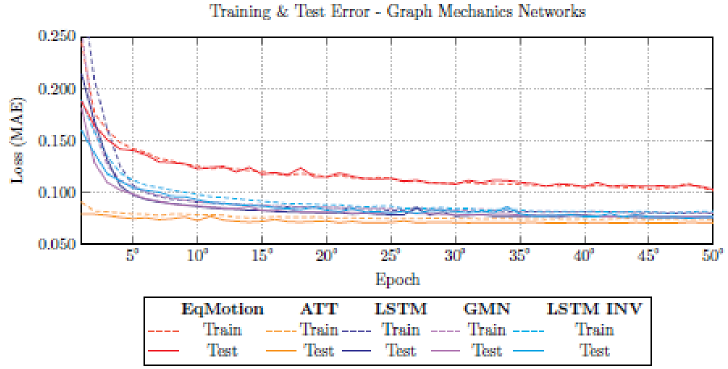
Experimental Results: Benchmarking

- Lastly, evaluate **EqMotion against commonly adopted reference models**: (i) baseline models, (ii) equivariant models and (iii) GMN-based models.

Training & Test Error												
		Epoch										
	Model	1°		5°		10°		25°		50°		Size
		$\mathcal{L}_{\text{train}}$	$\mathcal{L}_{\text{test}}$	$\mathcal{L}_{\text{train}}$	$\mathcal{L}_{\text{test}}$	$\mathcal{L}_{\text{train}}$	$\mathcal{L}_{\text{test}}$	$\mathcal{L}_{\text{train}}$	$\mathcal{L}_{\text{test}}$	$\mathcal{L}_{\text{train}}$	$\mathcal{L}_{\text{test}}$	
	EqMotion	0.244	0.188	0.142	0.140	0.126	0.123	0.113	0.114	0.102	0.103	74K
Baseline	Constant	0.127	0.124	0.126	0.124	0.127	0.124	0.127	0.124	0.127	0.124	—
Baseline	GWNET	0.345	0.306	0.301	0.291	0.297	0.284	0.142	0.135	0.127	0.126	334k
Baseline	Temporal	0.502	0.361	0.166	0.165	0.146	0.148	0.133	0.131	0.116	0.116	15K
Baseline	GCN	0.525	0.382	0.171	0.162	0.170	0.161	0.136	0.136	0.128	0.128	17K
Baseline	LSTM	0.525	0.348	0.231	0.212	0.172	0.170	0.140	0.139	0.129	0.129	558k
Baseline	GAT	0.484	0.381	0.171	0.159	0.150	0.169	0.130	0.126	0.122	0.118	17K
Equivariant	ENN	0.088	0.081	0.080	0.076	0.078	0.073	0.079	0.074	0.075	0.072	3K
Equivariant	ETNN	0.094	0.081	0.080	0.076	0.078	0.073	0.075	0.072	0.074	0.070	17K
Equivariant	EGNN	0.241	0.206	0.217	0.201	0.210	0.193	0.186	0.171	0.151	0.140	17K
Equivariant	ESTGCN	0.166	0.113	0.090	0.085	0.084	0.077	0.078	0.074	0.076	0.072	44K
GMN	ATT	0.091	0.079	0.079	0.075	0.078	0.073	0.075	0.071	0.074	0.071	42K
GMN	LSTM	0.300	0.214	0.107	0.098	0.091	0.086	0.083	0.078	0.080	0.076	33K
GMN	GMN	0.256	0.181	0.106	0.098	0.091	0.087	0.084	0.081	0.080	0.077	27K
GMN	LSTM INV	0.189	0.160	0.112	0.104	0.098	0.093	0.087	0.008	0.082	0.077	33K



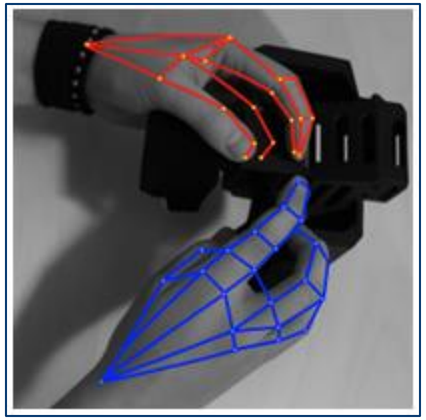




UNIVERSITÀ DEGLI STUDI DI MILANO

Conclusion Remarks

- In conclusion:
  - Model performance:
    - Effective on full dataset.
    - Performance influenced by data quantity and action complexity → more complex actions → higher errors.
  - Hyperparameter insights:
    - More feature layers & larger latent spaces → better accuracy.
    - Optional operations & number of categories → minor effect on final loss.
  - Comparison with other models:
    - Outperforms baseline models thanks to equivariance.
    - Underperform (slightly) other equivariant or GMN-based models.



- Future work:
  - Further theoretical & experimental analyses:
    - To better understand strengths and limitations.
    - To improve efficiency and robustness.

Grazie