Novel Visuo-Tactile Deep Learning Model for Robotic Arm Manipulation

Atakan Topcu
Dept. of Electrical & Electronics Eng.
Bilkent University

Ecem Şimşek
Dept. of Electrical & Electronics Eng.
Bilkent University

Abdallah Zaid Alkilani
Dept. of Electrical & Electronics Eng.
Bilkent University

***Abstract*—This study introduces a novel visuo-tactile, cross-modal, deep learning architecture for multi-modal representation learning that extends the visual deep learning models to incorporate both visual and tactile multi-modal sensory information. This architecture utilizes tactile feedback to generate latent representations that effectively highlight crucial task-related features within the visual domain. This study offers a simple yet effective framework for exploiting the synergy between vision and touch, thereby enhancing manipulation dexterity, resilience, and sample efficiency in various applications in the domain of robotics. This report covers the final state of the implementation and outline future work and improvements.**

***Index Terms*—Multi-modal Learning, Reinforcement Learning, Robotic Manipulation, Long Short-Term Memory, Attention Mechanism**

I. Introduction

Most research in the field of robotic manipulation focuses on visual data, especially in the case of deep reinforcement learning (RL) [1]. Nonetheless, the sense of touch, coupled with vision, plays an important role in robotic manipulation. However, the cross-modal representation of manipulation has not yet been thoroughly investigated [2], [3]. Though there are existing methods that utilize both modalities, their representations of the data can smooth over fine details and struggle with locality in images. These issues are particularly troublesome for manipulation, as the difference between in-contact and out-of-contact can be small in pixel space. Though some models offer performance enhancements, they usually require a large amount of data and/or long training time [3], [4]. In this work, we propose a novel cross-modal deep learning framework that is envisioned to be data efficient and faster to train, while also being able to extract fine details from a benchmark with visual occlusions and high levels of tactile perception uncertainty.

This work aims to extend the approach by Chen et al. [3], focusing on mainly reducing the data requirements imposed by the transformer. Hence, our motivation is to process the image data by using ConvNet architecture; such architectures have lower training times and higher efficiency in their usage of collected samples, especially in terms of in-domain learning [5]. It is especially important to be time and sample-efficient in robotic manipulation tasks, such as pushing and picking, since they are considered short-term benchmark tasks. Overall, our proposed improvements aim to decrease the training time and increase sample efficiency over the transformer-based architecture of Chen et al. [3]. As for the tactile data, we introduce a long short-term memory (LSTM) model for sequential processing of time-series data [6]. Furthermore, to further fuse the two modalities, we introduce a simple yet efficient attention mechanism [7]. The models are fused down the line using a multilayer perceptron (MLP) approach which is then passed onto our attention model. We seek to demonstrate how this approach cab achieve manipulation while integrating visual and physical information with more efficiency in terms of sample usage.

II. Related Work

In their architecture titled visuo-tactile transformers (VTT), as illustrated in Figure 1, Chen et al. produce compact latent representations by fusing visuo-tactile inputs with self and cross-modal attention mechanisms [3]. These novel additions to the already-existing vision transformer (ViT) [8] structure aim to further improve the multi-modal reasoning capability of VTT and increase success rates in four different robotic tasks.

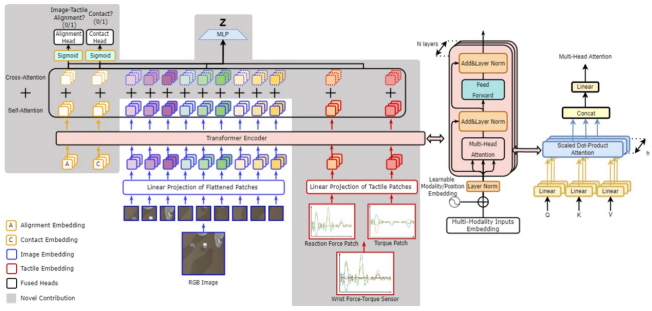

Fig. 1. State-of-the-art Visuo-Tactile deep learning architecture for robotic manipulation [3].

Furthermore, Chen et al. compare VTT to concatenation (Concat) and product of experts (PoE) baselines. In Concat, visual data is passed onto a convolutional encoder while the tactile data is passed to a MLP. Then, the output of these two branches is concatenated to output the latent representation as well as the embeddings of image-tactile alignment and contact. Utilization of these embeddings assists in constraining the model in the case of occlusion or noise in the data and in obtaining a more stable representation. PoE is just an extension of Concat where the product-of-experts algorithm is employed to merge the two branches instead of directly concatenating.

VTT, on the contrary, does not have two separate branches for each modality. Hence, the VTT structure also acts as a fusion module in combining the tactile data, such as reaction force patch and torque patch, with image data. Additional details of VTT, particularly its RL formulation/integration, are provided below in subsection III-A.

III. Theoretical Background and Methodology

We conduct simulated experiments of two manipulation tasks in PyBullet. There are four tasks: Pushing, Door-Open, Picking, and Peg-Insertion (in this report, we focus on Door-Open and Peg-Insertion due to time/hardware constraints). These tasks are illustrated in Figure 2.

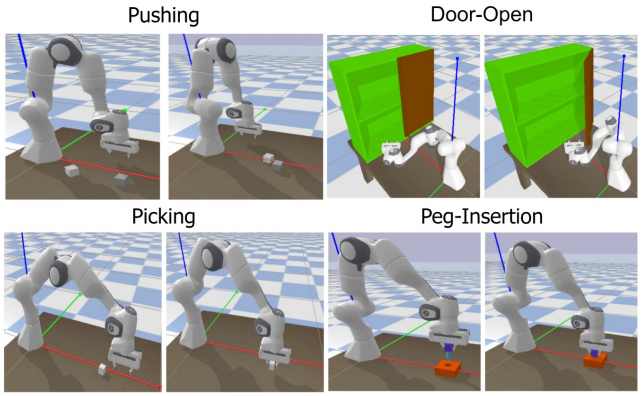

Fig. 2. Sample short-term benchmarks available to our simulator, Pybullet.

The initial simulation benchmark was developed by ElementAI [9]. The benchmark was further modified by a team at the University of Michigan, Ann Arbor [3]; we also performed additional necessary fixes to get the simulator working for our case. The agent receives an $84 \times 84 \times 3$ RGB image and a 1×6 wrist reaction wrench. This agent, specifically a Panda robotic arm, is equipped with various sensors that enhance its manipulation capabilities, such as force and torque sensors. These sensors, located at the wrist, end effector, and both fingers, allow the robot to gauge interaction forces during tasks. Additionally, the robot arm design includes sensors for monitoring the positions of its joints and fingers. This sensor setup is crucial for precise movement control and enables the robot to perform complex manipulation tasks by providing detailed feedback. This feedback is not only limited to its own state; the agent also receives feedback from its interaction with the environment. This sensor setup therefore makes the robot arm highly effective for tasks that require delicate handling and dexterity, such as the manipulation tasks we consider in this work.

Because our manipulation benchmarks involve frequent visual obstructions and a significant degree of uncertainty in tactile perception, our manipulation tasks are formulated as partially observable Markov decision processes (POMDP) [10]. While there have been notable advancements in model-based RL for POMDP, and notwithstanding the abundance of choices [11], the stochastic latent actor-critic (SLAC) algorithm stands out due to its demonstrated stability and robustness [12]. These specific attributes make SLAC well-suited for accommodating high levels of randomness and serve as an effective baseline for evaluating visuo-tactile fusion methods.

A. Description of the SLAC Algorithm

The SLAC algorithm integrates a latent variable model into the actor-critic framework for reinforcement learning with high-dimensional observations [12]. The algorithm's strength lies in its ability to handle high stochasticity, making it effective for complex tasks like visuo-tactile manipulation in robotics [10]. A notable and advanced RL framework, SLAC is particularly suited for tasks modeled as POMDPs. POMDPs are an extension of Markov Decision Processes that handle scenarios where the agent does not have complete visibility of the environment's state. In a POMDP, the decision-making process is influenced by uncertainty about the current state due to partial observability. A POMDP is defined by the tuple (S, A, T, R, Ω, O), where:

- S is the set of states.
- A is the set of actions.
- $T : S \times A \times S \to [0, 1]$ is the transition probability function, with $T(s, a, s')$ representing the probability of transitioning from state s to state s' after taking action a.
- $R : S \times A \to \mathbb{R}$ is the reward function.
- Ω is the set of observations.
- $O : S \times A \times \Omega \to [0, 1]$ is the observation probability function, where $O(s', a, o)$ gives the probability of observing o after taking action a and ending up in state s'.

The main challenge in POMDPs is to make decisions without direct knowledge of the state. The agent maintains a belief state b, a probability distribution over all possible states, updated based on the observations and actions taken. The belief update rule, a fundamental aspect in POMDPs, is often represented as:

$$b'(s') = \frac{O(s', a, o) \sum_{s \in S} T(s, a, s') b(s)}{P(o|b, a)}, \tag{1}$$

where $b'(s')$ is the updated belief about being in state s', $b(s)$ is the previous belief about being in state s, and $P(o|b, a)$ is the probability of observing o given the belief b and action a. This rule captures the Bayesian update of the belief state, considering the dynamics of the environment and the latest observation.

SLAC consists of two interconnected components: a variational autoencoder (VAE) for learning latent dynamics and an actor-critic model for policy learning. The outline of the SLAC algorithm is provided in Algorithm 1.

The VAE in SLAC learns a compact representation of the state space. It involves an encoder q_ϕ that infers the latent state z_t from the current observation O_t, previous action a_{t-1}, and previous latent state z_{t-1}; and a decoder p_θ that reconstructs

Algorithm 1 Stochastic Latent Actor-Critic (SLAC) [12]

Require: Environment E and initial parameters ψ, ϕ, θ_1, θ_2 for the model, actor, and critics.
  $x_1 \sim E_{\text{reset}}()$
  $D \leftarrow (x_1)$
  for each iteration **do**
    for each environment step **do**
      $a_t \sim \pi_\phi(a_t|x_{1:t}, a_{1:t-1})$
      $r_t, x_{t+1} \sim E_{\text{step}}(a_t)$
      $D \leftarrow D \cup (a_t, r_t, x_{t+1})$
    end for
    for each gradient step **do**
      $x_{1:\tau+1}, a_{1:\tau}, r_\tau \sim D$
      $z_{1:\tau+1} \sim q_\psi(z_{1:\tau+1}|x_{1:\tau+1}, a_{1:\tau})$
      $\psi \leftarrow \psi - \lambda_M \nabla_\psi J_M(\psi)$
      for $i \in \{1, 2\}$ **do**
        $\theta_i \leftarrow \theta_i - \lambda_Q \nabla_{\theta_i} J_Q(\theta_i)$
      end for
      $\phi \leftarrow \phi - \lambda_\pi \nabla_\phi J_\pi(\phi)$
      for $i \in \{1, 2\}$ **do**
        $\bar{\theta}_i \leftarrow \nu\theta_i + (1-\nu)\bar{\theta}_i$
      end for
    end for
  end for

the observation from the latent state. The encoder is modeled as a Gaussian distribution:

$$q_\phi(z_t|z_{t-1}, a_{t-1}, O_t) = \mathcal{N}(z_t; \mu_\phi(z_{t-1}, a_{t-1}, O_t), \sigma_\phi(z_{t-1}, a_{t-1}, O_t)), \tag{2}$$

where μ_ϕ and σ_ϕ are neural networks parameterized by ϕ. The VAE loss function combines the reconstruction loss and the Kullback-Leibler (KL) divergence:

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q_\phi}[-\log p_\theta(O_t|z_t)] + \beta \cdot \text{KL}(q_\phi(z_t|z_{t-1}, a_{t-1}, O_t)||p(z_t|z_{t-1}, a_{t-1})), \tag{3}$$

where $p_\theta(O_t|z_t)$ is the likelihood of the observation given the latent state and β is a hyperparameter that balances the reconstruction loss and the KL term.

In the actor-critic framework, the policy (actor) $\pi_\psi(a_t|z_t)$, parameterized by ψ, maps latent states to a distribution over actions. The value function (critic) $Q_\theta(z_t, a_t)$, parameterized by θ, estimates the expected return of taking action a_t in latent state z_t. The actor-critic loss function is composed of two parts: the policy loss and the critic loss. The policy loss aims to maximize the expected return and the entropy of the policy for exploration:

$$\mathcal{L}_{\text{policy}}(\psi) = -\mathbb{E}_{\pi_\psi}[Q_\theta(z_t, a_t) - \alpha \log \pi_\psi(a_t|z_t)], \tag{4}$$

where α is the temperature parameter controlling the importance of the entropy term. The reparametrization trick is used in practice, where a deterministic transformation f_ψ from the state to the action is employed to obtain the desired latent variable values and thereby enabling gradient-based training through backpropagation. The critic loss is the mean squared Bellman error:

$$\mathcal{L}_{\text{critic}}(\theta) = \mathbb{E}_{\pi_\psi}\left[\left(Q_\theta(z_t, a_t) - \hat{Q}(z_t, a_t)\right)^2\right]; \quad \hat{Q}(z_t, a_t) = r_t + \gamma \mathbb{E}_{z_{t+1}}[V_\theta(z_{t+1})], \tag{5}$$

where $V_\theta(z_{t+1})$ is the state value function, and γ is the discount factor.

The overall loss function for SLAC is the sum of the VAE loss and the actor-critic losses:

$$\mathcal{L}_{\text{SLAC}} = \mathcal{L}_{\text{VAE}} + \mathcal{L}_{\text{policy}} + \mathcal{L}_{\text{critic}}. \tag{6}$$

This combined loss function enables the simultaneous learning of latent dynamics and policy optimization, resulting in efficient learning from high-dimensional sensory inputs.

In VTT, the latent representation z from the transformer itself is utilized in both model-learning and policy-learning components. Additionally, the critic's value function loss is backpropagated through VTT to the attention mechanism. The total model-learning losses are a combination of the VTT losses and SLAC losses, represented as:

$$\mathcal{L}_{\text{model}} = \mathcal{L}(O_t|z_t, a_{t-1}) + \mathcal{L}(r_t|z_t, a_{t-1}) + \mathcal{L}_{\text{KL}}(q\|p) + \mathcal{L}_{\text{VTT}}, \tag{7}$$

where $\mathcal{L}(O_t|z_t, a_{t-1})$ represents the loss for reconstructing the observation O_t based on the latent state z_t and the previous action a_{t-1}), $\mathcal{L}(r_t|z_t, a_{t-1})$ denotes the loss associated with inferring the reward r_t given the latent state and the previous action, $\mathcal{L}_{\text{KL}}(q\|p)$ is the KL divergence between the prior and posterior models in the SLAC framework, and $\mathcal{L}_{\text{VTT}}$ represents the specific loss components related to the VTT model. $\mathcal{L}_{\text{VTT}}$ incorporates alignment loss and contact loss by exploiting binary cross entropy (BCE) between logits, predicted via a MLP, and the ground truth labels:

$$\mathcal{L}_{\text{VTT}} = BCE_{logits}(MLP(Al_{head}, Al_{gt}) + BCE_{logits}(MLP(C_{head}, C_{gt}), \tag{8}$$

where Al_{head} and C_{head} indicate the alignment and contact heads, respectively, and Al_{gt} and C_{gt} their corresponding ground truth labels. Refer to Figure 1 for illustration.

B. Description of Our Model

The architecture of our novel model is provided in Figure 3. The base of our architecture builds on and modifies the architecture provided by Chen et al. [3]. We have separate streams for visual and tactile data.

For the visual data, we pass the $84 \times 84 \times 3$ RGB image patches to a convolutional encoder. This encoder is designed to process image patches through multiple layers, progressively reducing their resolution while increasing the depth of feature representations. Starting with an input RGB image, the encoder uses a series of convolutional layers with 6×6 filter size and a stride of 2, together with appropriate padding and each followed by LeakyReLU activation for stability, described as $\text{LeakyReLU}(x) = \max(0, x) + 0.2 \cdot \min(0, x)$. These layers

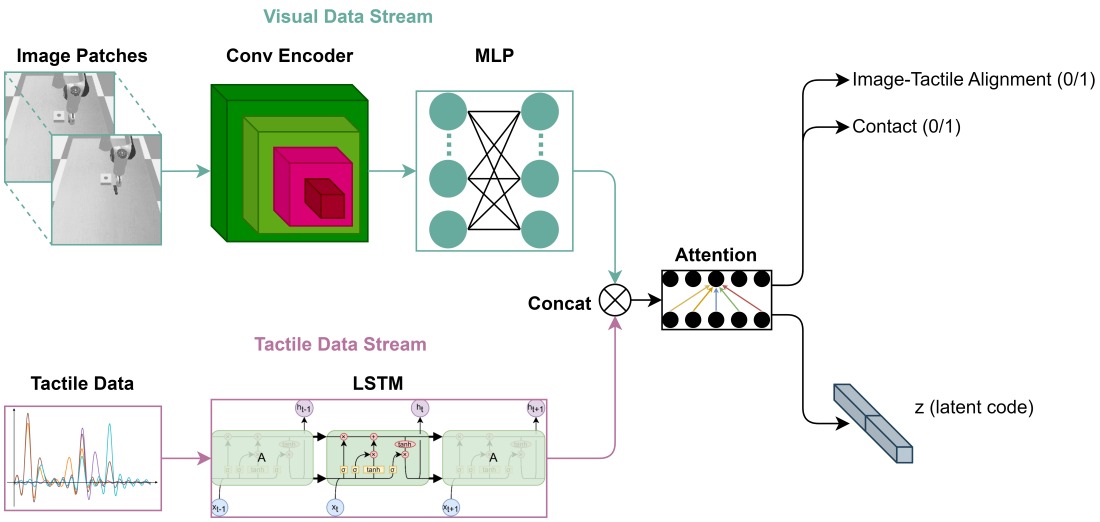

Fig. 3. The architecture of our proposed novel model. Visual and tactile data are processed in streams that are later concatenated and fused via an attention mechanism, from which binary alignment and contact information is predicted, as well as the latent code z for SLAC.

gradually transform the image from its original size to a more compact feature map, suitable for feature extraction. A MLP transforms the representation to a hidden dimension of 256 and singular dimension of 6, after which layer normalization is applied [13].

As for the tactile data, a LSTM handles the sequential 1×6 wrist reaction wrench tactile inputs, capturing temporal patterns and dependencies, and then compresses these learned features into a more compact latent representation with hidden dimension of 32 and singular dimension of 6, after which layer normalization is applied. This LSTM architecture is particularly suitable for understanding and analyzing time-series tactile data.

As for concatenation and fusion, an attention mechanism [7] is designed to fuse visual and tactile data. This attention mechanism processes the image and tactile features through a series of layers to compute attention weights. These weights are used to emphasize the most relevant combined features, enabling the model to focus on the most informative aspects of the multi-modal input. The output is also layer normalized. This attention-based approach efficiently integrates visual and tactile information, which is crucial for decision-making. To produce the prediction for the latent code z, a 1-layer MLP (i.e., a linear transformation layer) is employed (see subsection III-A for additional details on SLAC; our model surrogates VTT and fills in for the $\mathcal{L}_{\text{VTT}}$ term in Equation 7 as well as Equation 8). Likewise, for the image-tactile alignment and the contact binary variables, 1-layer MLPs with sigmoid output activation were employed to achieve those binary classification outputs.

IV. Results

Due to time and hardware constraints, we have only looked at two of the benchmark tasks: door opening and peg insertion. We have trained three different baselines (VTT, POE, and Concat) and our model, and report the evaluation rewards plotted versus the evaluation steps. The results are provided in Figure 4 and Figure 5 for the door opening and peg insertion task, respectively. Please refer to Appendix B for additional GIFs of the results.

It is clearly evident that our model outperforms VTT as well as the baselines. In Figure 4, our model climbs up to high rewards much faster and is able to sustain them much sooner and in a more stable manner. Likewise, in Figure 5, our model still maintains its head start performance and converges in the smoothest and quickest way amongst the compared methods.

Overall, the results in in Figure 4 and Figure 5 demonstrate that our method holds premise for achieving visuo-tactile multi-modal fusion with in a more efficient manner than VTT, while also outperforming VTT and other baselines methods in the process.

V. Discussion & Future Directions

Current results show promise as we successfully demonstrate that, with a simple attention-based fusion method, we can outperform transformer-based methods with better data efficiency. This opens the way for the implementation of complex, cross-model modalities in real-life scenarios with lower cost and ease of implementation. With upcoming usage of visuo-tactile data in robotics (see new Optimus by Tesla [14]), we have demonstrated a simple yet efficient way to utilize such multi-modal methodologies.

In future work, we plan to utilize more seeds and more tasks to compare the methods. For further research, we can focus on the graph neural network (GNN)-based neural network architecture and the effect of different graph convolution operators. Using different numbers of layers in these GNN-based neural networks can be explored for a trade-off between efficiency and representation capability. Graph-based models can be used instead of our current attention mechanism, while

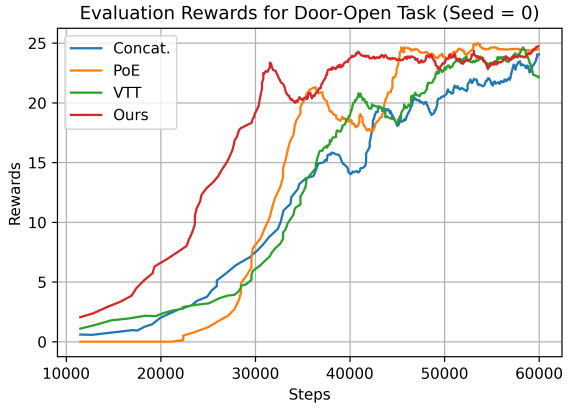

Fig. 4. Evaluation rewards per step for the various methods on the task of Door opening. Random seed was set to 0 while training to produce these results.

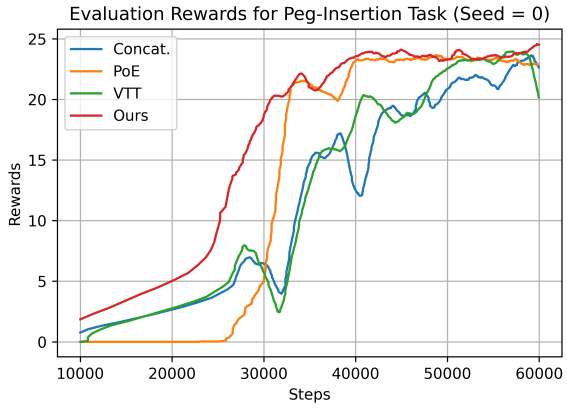

Fig. 5. Evaluation rewards per step for the various methods on the task of Peg Insertion. Random seed was set to 0 while training to produce these results.

still providing attention at a lower cost. As such, we can provide a more sophisticated attention mechanism without compromising on data efficiency.

Another direction we can take is converting tactile data to two-dimensional image patches via recurrence plots [15]. This way, we might not have to utilize two different branches and instead directly merge both modalities from the very beginning.

In summary, our study shows that simple attention-based methods can outperform complex transformers in robotics, using visuo-tactile data more efficiently. Future work directions include using graph neural networks and the possibility of merging modalities from the onset, further improving our method's efficiency and applicability in real-world scenarios.

References

[1] L. Cong, H. Liang, P. Ruppel, Y. Shi, M. Görner, N. Hendrich, and J. Zhang, "Reinforcement learning with vision-proprioception model for robot planar pushing," *Frontiers in Neurorobotics*, vol. 16, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2022.829437

[2] W. Zheng, H. Liu, and F. Sun, "Lifelong visual-tactile cross-modal learning for robotic material perception," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 3, pp. 1192–1203, 2021.

[3] Y. Chen, M. V. der Merwe, A. Sipos, and N. Fazeli, "Visuo-tactile transformers for manipulation," in *CoRL 2022, Auckland, New Zealand*, ser. Proceedings of Machine Learning Research, K. Liu, D. Kulic, and J. Ichnowski, Eds., vol. 205. PMLR, 2022, pp. 2026–2040. [Online]. Available: https://proceedings.mlr.press/v205/chen23d.html

[4] D. Driess, Z. Huang, Y. Li, R. Tedrake, and M. Toussaint, "Learning multi-object dynamics with compositional neural radiance fields," 2022.

[5] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," 2022.

[6] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[9] S. Rajeswar, C. Ibrahim, N. Surya, F. Golemo, D. Vazquez, A. Courville, and P. O. Pinheiro, "Touch-based curiosity for sparse-reward tasks," 2021.

[10] H. Kurniawati, "Partially observable markov decision processes (pomdps) and robotics," 2021.

[11] M. Egorov, "Deep reinforcement learning with POMDPs," https://cs229.stanford.edu/proj2015/363_report.pdf, accessed: 2023-10-22.

[12] A. X. Lee, A. Nagabandi, P. Abbeel, and S. Levine, "Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 741–752. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/08058bf500242562c0d031ff830ad094-Paper.pdf

[13] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016.

[14] V. Savov, "Tesla shows off optimus gen 2 robot with improved hands, slimmer figure," https://www.bloomberg.com/news/articles/2023-12-13/tesla-optimus-gen-2-new-video-shows-dancing-robot-better-movement-updates, Dec. 2023, accessed: 2023-12-29.

[15] N. Hatami, Y. Gavet, and J. Debayle, "Classification of time-series images using deep convolutional neural networks," 2017.

Appendix A
Contribution

This project is not a subsequent project of any team member and the proposed methods have not been implemented hitherto.

Atakan Topcu is responsible for debugging the simulator. He found the simulator to malfunction or give faulty data and fixed it. He created the GIF function for getting the results. He also found and implemented the attention module. He ran several models for comparison for the opening task.

Abdallah Zaid Alkilani is responsible for baseline results. He successfully debugged/re-implemented each baseline for the peg insertion task. He implemented the LSTM algorithm in Python and did the hyperparameter tuning. Furthermore, he is responsible for analysis and conducting experiments for validation and quantitative comparison.

Ecem Şimşek oversaw the general framework and helped with the implementation of the codes. Additionally, she ran several model variations for comparison, as well as extensive formulation for establishing a theoretical framework.

Appendix B
Demo

Examples of the simulation environment and GIFs demonstrating each method are provided in the demo slides.