JoTyMo: Joint type and movement detection from RGB images of before and after interacting with articulated objects

1st Ehsan Forootan
ECE department (University of Tehran)
Human and Robot Interaction Laboratory
Tehran, Iran
esa.forootan@ut.ac.ir

2nd Hamed Ghasemi
ECE department (University of Tehran)
Human and Robot Interaction Laboratory
Tehran, Iran
hamed.ghasemi@ut.ac.ir

3rd Mehdi Tale Masouleh
ECE department (University of Tehran)
Human and Robot Interaction Laboratory
Tehran, Iran
m.t.masouleh@ut.ac.ir

***Abstract*—We introduce JoTyMo, a system designed to classify and detect joint types and movements of articulated objects within the SAPIEN PartNet-Mobility Data set. Our approach utilizes a two-stage process, employing consecutive CNN encoders based on the VGG architecture to classify joints from pre and post-interaction images. Additionally, we utilize another CNN layer to regress both an effect point and movement vector, based on the joint type provided. Our extensive evaluation showcases impressive results, achieving 96% accuracy in joint classification and 94% accuracy in regression on the dataset. Furthermore, we provide a comprehensive comparison with existing methods, highlighting JoTyMo's superior performance and its ability to effectively generalize on simulated objects.**

I. INTRODUCTION

Grasping is one of the fundamental but most challenging skills of robots. Grasping is also important for enabling robotics arms to perform various tasks in various environments. One of the main challenges of robotic grasping is how to deal with objects that have restrictions in movement such as articulated objects. Using the convolutional neural networks(CNNs) in articulated object grasping has been a challenging problem due to the number of reasons such as estimating the position and orientation of the joint axis and the current rotation angle of the object. Moreover, various applications, such as robotic manipulation, object recognition, and human-computer interaction calls for processing more than one image per object.There are also works such as Multi-Camera Video Tracking and Recognition [13], that utilize CNN to track objects and classify them, promote the usage of CNN in classifying images. There are different approaches to tackle this problem, such as model-free method suggested in [14], where Deep Reinforcement Learning (DRL) and also the Deep Deterministic Policy Gradient (DDPG) network is used. Model-based methods, which use a model of the object and its joint parameters similar to [15], or learning-based methods, which use deep neural networks to learn the joint state from data have also been suggested. CNN learning based methods can be more flexible and generalizable, as they do not rely on a fixed or predefined model of the environment, but instead learn from data and experience. This can allow them to adapt to changing or uncertain situations, and to handle complex or nonlinear dynamics that might be difficult to model [4]. They can also leverage existing techniques and frameworks for deep learning, such as neural networks, convolutional layers, and back-propagation, which can enable fast and parallel learning on large datasets [5]. Subsequently, Methods such as DEEP-SEE [1] ,CNN-based Joint State Estimation [2] and Single-view 3D Openable Part Detection [3] have been suggested.

This paper presents a CNN-based method that can identify joint types and movements of articulated objects from RGB images. We use the PartNet-Mobility Dataset [6], which contains images of objects before and after an interaction or a view change, to train and test our method. We classify the objects into four joint types: P-joint, R-joint, Push-P joint, and Not articulated. Figure 1 shows some examples of these images. We also use another CNN structure, trained on a specific joint type, to estimate the movement vector and the joint effect point of the object after the classification. We compare our method with different classical machine learning algorithms and CNN architectures to evaluate its accuracy and design. We also review the related works before concluding this section.

A. Deep Convolutional Networks for Image Classification and Regression

The paper [11] introduces a convolutional neural network (CNN) architecture that consists of multiple layers of 3x3 convolution filters, followed by max-pooling and fully-connected layers [11]. The paper shows that increasing the depth of the network can improve its accuracy on large-scale image recognition tasks, such as the ImageNet challenge [11]. The paper [11] also demonstrates that the VGG networks can be used as feature extractors for other vision tasks, such as object detection and face recognition.

The VGG networks [11] are known for their deep architecture and have been widely used for various computer vision tasks, including image classification.These networks have a structure, consisting of multiple convolutional layers

followed by fully connected layers. The convolutional layers are responsible for learning hierarchical features from the input images, while the fully connected layers perform the final classification or regression. Overall, the VGG based design was chosen over other networks like Alexnet [12] because of its utilization of smaller kernel sizes. In addition, networks such as Inception-V3 [27] and various ResNets available in [26] and [25] where utilized for comparison in performance. As for in image regression, works such as [29] and [30] have been developed.

B. CNN-based Joint State Estimation

The paper [2], provides a general overview on the image classification tasks using convolutional neural networks (CNNs), and introduces the differences and contributions of the deep learning models such as LeNet, AlexNet, Inception, VggNet and ResNet. This paper also explains how different units in these CNN models, such as pooling, activation, and dropout, function to support better results for these models. The paper aims to offer a guideline for deeply understanding the utility of CNN units.

Image Classification with Classic and Deep Learning Techniques [7] paper is a report that compares the performance of classic and deep learning techniques for image classification on three datasets: MNIST [8], CIFAR-10 [9], and Caltech-101 [10]. The report shows that CNNs outperform the classic methods such as k-NN, SVM, and Random Forests. Our bases for using CNNs is based on [2], while the inspiration for comparing classification and regression with classical methods is [7].

C. Using Multiple CNN Networks

The fundamental essence of the suggested technique in [1] relies on an object tracking method, which in turn relies on the utilization of two convolutional neural networks that have been trained offline [1]. The primary principle entails the alternation between tracking through the utilization of motion information and predicting the temporal location of the object based on visual similarity [1]. The paper [28] also utilizes a similar approach of using two layers of neural networks where the second one uses data from the first layer that has been applied to input images.

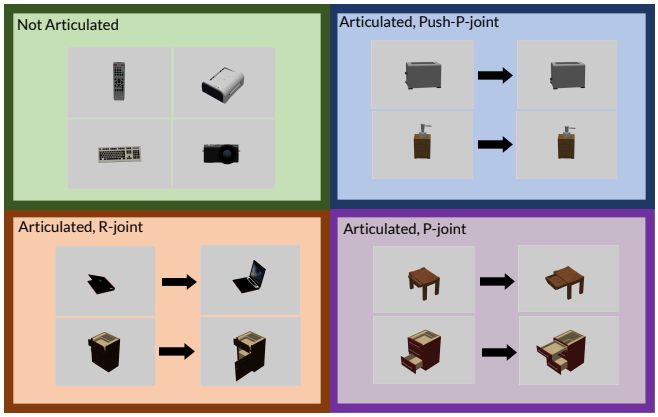

Fig. 1. This visual representation showcases various examples of distinct objects along with their potential interactions within the PartNet-Mobility Dataset [6].

D. Open-able Part Detection

Published in 2022, the paper [3] presents the OPDRCNN, a neural network architecture capable of detecting openable parts and predicting their motion parameters using a single RGB image. Additionally, two datasets of 3D objects with openable parts, namely OPDSynth and OPDReal, were created. OPDSynth consists of synthetic objects, while OPDReal comprises real objects reconstructed from RGB-D images. The paper demonstrates the superior performance of OPDRCNN over baselines and prior work on both datasets [3]. In addition, the paper [31] generalizes the openable part detection task to scenes with multiple objects, and creates a corresponding dataset based on real-world scenes. The paper also proposes a part-aware transformer architecture that outperforms prior work on this task.

Our work extends the scope of these papers by incorporating RGB images of objects after undergoing interaction. However, the collection of real-world data on openable objects is a labor-intensive process, necessitating the use of synthetic images for our research. Moreover, it is essential to acknowledge that our proposed architecture is more intricate than the VGG-based CNN, namely OPDRCNN. Conversely, while OPD solely relies on a single frame, our approach considers the interaction performed on the object.

II. Methodology

Our goal is to classify RGB images generated based on interaction in the simulation environment of [6]. Moreover, we aim to find where the effect point and movement vector of the mentioned interaction is, considering that joint type is known. We also assume that the joint interaction can be seen in the RGB image, if it exists.

First, we formally introduce the task at hand and the data set used. Then, we show the architecture used to achieve the task and explain about the building blocks inside it. A summary of the design has been presented in figure 2 .

A. Data set

There are several data sets available that contain articulated objects. These data bases are presented in table I. In addition, figure 1 showcases some examples of objects that exist in the data set [6]. The importance of using different view angles and

TABLE I
DIFFERENT DATA SETS COMPARISON

Database name	Database features		
	Data type	*Number of Categories*	*Data Environment*
Shape2Motion [16]	Point Cloud	10	Real and Simulated
Shape Net Core [17]	3D CAD models	3135	Simulated
SAPIEN PartNet-Mobility [6]	**RGB**	**2037**	**Simulated**

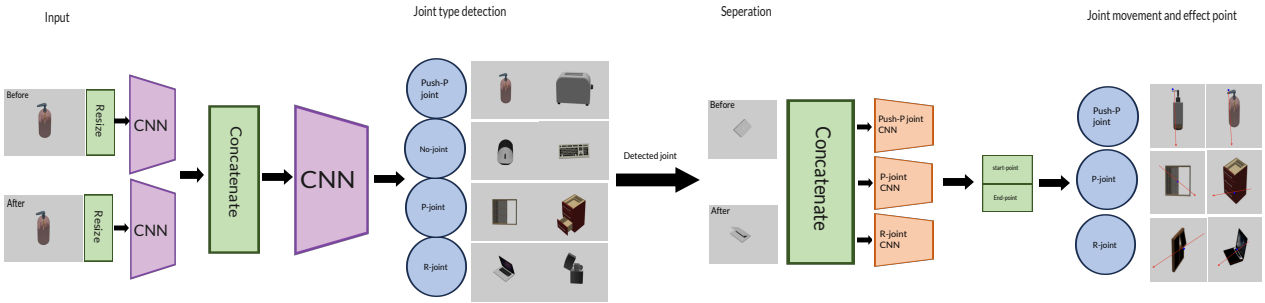

Fig. 2. Our method, JoTyMo, consists of two steps: classification and detection. We use a two-layer CNN to classify the joint types of the objects from RGB images before and after an interaction. Then, we use another CNN, trained on a specific joint type, to detect the movement vector and the effect point of the object. The inputs to the CNNs are the same images used for classification. If the object has no joints, the images are taken from different viewpoints.

significance of having Push-P joints means that using [6] is the only viable option as the two other do not include such a joint type, while the other two data bases provide either more groups of data [17] or a range of simulated and real world options [16].

Overall, a total of 200 images were captured from 4 different joint types across 40 different objects. Each image had dimensions of $224 \times 224 \times 3$ and was selected from a database either before or after an interaction.

To create a comprehensive dataset, a total of 600 training images and 120 validation images were generated from the initial 200 selected baseline images. Each data point in the dataset had dimensions of $2 \times 224 \times 224 \times 3$, including information from both before and after the same interaction.

To enhance the performance and convergence of the neural networks used, Keras augmentation techniques were applied. Specifically, horizontal and vertical flips were utilized as the chosen augmentations. These specific augmentations were selected to avoid any complications during the training and validation testing processes.

B. Problem Formulation and Proposed solution

Two RGB images, each with a size of $96 \times 96 \times 3$, are obtained from resized images before and after in [6]. These images are then fed into a CNN, as shown in Figure 1. The CNN consists of several layers, as depicted in Figure 2. The output shape of each smaller CNN is $21 \times 21 \times 96$. These two outputs are concatenated to form a $21 \times 21 \times 192$ input for the classification CNN. The output of the CNN is $1 \times 1 \times 64$, which is then passed through two Dense layers in Keras [18], followed by a Softmax layer. Categorical Cross Entropy is used as the loss function for the Keras back-propagation algorithm. The probability of being each groups is then outputted. In essence, the objective of the first half of the architecture is to learn a transformation function denoted by the equation:

$$T : \mathbb{R}^{2\times224\times224\times3} \rightarrow ß \tag{1}$$

where ß is defined in equation (2). Additionally, the largest value among the four elements in each output vector indicates the most probable group that the input belongs to.

The set ß can be represented as:

$$\left\{ \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \mid a, b, c, d \in [0.0, 1.0] \subseteq \mathbb{R} \& 0 \leq a + b + c + d \leq 1 \right\} \tag{2}$$

A similar transform is used for finding the joint movement vector and effect point, based on the fact that joint type is known. What it means is that each joint type has it's own dedicated trained Network. The only difference here is that instead of having two paths, we only will be using one path for the regression because the input both images are $224 \times 224 \times 3$ and no resize is used. The output of the CNN is $3 \times 3 \times 32$. After the CNN, two layers of Keras Dense is used to transform the flattened input into 4 regression points. The loss function used is referred to as "Huber Loss." It was chosen because it provides a smoother learning curve compared to other loss functions like "Mean Square Error Loss (MSE Loss)." However, for evaluation purposes, MSE Loss is used because it allows for better interpretations.

In the case of regression, (3) describes the output space. Each vector member represents an axis value related to pixel points. Specifically, each vector in a $224 \times 224 \times 3$ image represents two points: the first two points indicate the starting point, while the second point represents the resulting position after the interaction, giving the movement vector.

The output space can be defined as:

$$\left\{ \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \mid a, b, c, d \in [0.0, 224.0] \subseteq \mathbb{R} \right\} \tag{3}$$

In(3), each coordinate in the output space represents a pixel point within the image, with values ranging from 0.0 to 224.0.

When it comes to training, it is worth noting the effectiveness of using mini-batches in conjunction with the Adam

optimization technique [19]. This combination has gained significant popularity within the deep learning community, especially among those utilizing Keras for network implementation. By implementing mini-batches and leveraging the power of Adam optimization, the training process becomes more efficient and allows for better convergence of the neural network model.

III. Results and Discussion

In this section, we showcase the remarkable results achieved by our system, JoTyMo, in detecting the joint types and movements of articulated objects. We conduct a comprehensive comparison between JoTyMo, classical methods, and deep learning approaches to highlight the superior performance and architectural significance of our system. Furthermore, we assess JoTyMo's generalization ability by evaluating its performance on augmented and previously unseen inputs. Finally, we delve into a detailed examination of the limitations and challenges encountered by our system.

A. Training Curves

Figure 3 shows the training and testing losses during the learning process for both the classification and regression.

In Figure 5, it can be observed that the losses of both the validation and training data decreased significantly and eventually reached a plateau across all sub-figures. Additionally, it is worth noting that the curves representing the validation and training data displayed a smooth progression. This can be attributed to the utilization of L1 and L2 regularizations in the initial and final layers. By incorporating this regularization technique, the learning process was expedited, resulting in a more pronounced reduction in the loss functions.

B. Performance against Classical methodology

In this analysis, two comparisons are to be made: the effectiveness of the classification and regression tasks. Table II presents the accuracy and other metrics concerning the classifier, while Table IV showcases the performance of the regression task through two different metrics: the mean absolute error percentile (**MAEP**) and the mean squared error percentage (**MSEP**).These metrics provides a quantitative measure to assess the accuracy of predictions. By calculating the MAEP and MSEP, evaluation of the performance of JoTyMo can be done.

The results in Table II demonstrate the superior capability of our architecture to discern joint types by considering the observed interaction, rather than simply selecting features from the input. Moreover, the regression results highlight the CNN's ability to accurately estimate the movement vector, both in terms of magnitude and direction. This is evidenced by the higher MAEP and smaller angle difference, as indicated by the higher MSEP. Additionally, as shown in Table III, the CNN surpasses most classical methods in performance for each joint type. Although Random Forest Regression also yields acceptable results in certain cases, it is worth noting that MLP achieves notable accuracy. However, MLP is more complex than Random Forest and less precise compared to the CNN or Random Forest approaches.

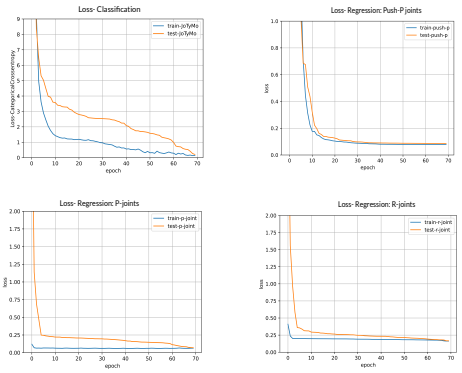

Fig. 3. The top left curve represents the loss function employed for joint detection, whereas the top right graph demonstrates the loss function specifically designed for the detection of Push-P joint movement vectors. Furthermore, the two bottom-line graphs depict the training curves for the loss function utilized in detecting P-joint and R-joint respectively.

C. Comparison of classifier CNN to alternative deep learning methods

To examine the impact of architecture on our design's performance, we assessed various pre-trained networks by employing transfer learning [22]. Specifically, we provided these networks with concatenated images representing the state before and after the interaction, serving as a single input. To ensure fairness in our comparison, we trained these networks for the same duration as our proposed CNN, using identical loss functions and optimization algorithms. This approach minimizes the influence of hyperparameters on the outcomes. The results, displayed in Table IV, showcase the accuracy of both training and testing data, thereby demonstrating the models' capacity to learn and generalize.

As Table III indicates, our network distinguishes different categories of joints. This pattern is not observed in other networks. ResNet and Inception exhibit overfitting, while VGG-based design shows underfitting. Despite having fewer parameters than all transfer learned networks, our classifier achieves an optimal result due to its structure.

TABLE II
Comparing Classification methods on the same Training data[a]

Method	Classification Metrics[b]		
	Precision-Avg%	*f1-Score-Avg%*	*Recall-Avg%*
LDA(LSQR) [20]	87	86	86
SVM(Poly) [21]	92	92	92
MLP(dept=6) [23]	84	82	83
Random Forests(dept=3) [23]	82	81	82
Convolutional Neural Network	**96**	**95**	**95**

[a] Done on Validation data, as all models are fitted on the training perfectly.

[b] The bigger the percentage, the better.

TABLE III
COMPARING REGRESSION METHODS ON THE SAME TRAINING DATA[a]

Methods\Metrics	Joint Type							
	P-joint		*Push-P-joint*		*R-joint*		*Total*	
	MAEP%	*MSEP%*	*MAEP%*	*MSEP%*	*MAEP%*	*MSEP%*	*MAEP%*	*MSEP%*
SVM(poly) Regression [21]	69	76	85	87	79	81	77.66	81.33
MLP Regression [23]	73	80	71	74	68	71	70.66	75.00
Random Forest Regression [23]	95	96	90	91	85	87	90.00	91.33
Voting Regression (linear + gradient boosting) [24]	73	81	85	87	58	62	72.00	76.67
Convolutional Neural Network	**92**	**95**	**91**	**91**	**99**	**96**	**94.00**	**94**

[a] Done on Validation data, as all models are fitted on the training perfectly.

D. Accuracy on augmented images

We assessed the performance of JoTyMo on unseen data through the pre-training of the CNN using augmented images that varied in parameters such as intensity and scale. Subsequently, we tested the accuracy of our design on the validation data described in II-A, and the results are provided in Table V. Additionally, Figure 4 showcases some examples of the augmented images and the respective detected joint types.

Observing Table V, it is evident that, due to the system's reliance on interaction detection, complete separation between objects without joint and articulated objects has been achieved. Furthermore, R-joints exhibit distinct movement patterns compared to P-joints and Push-P joints, allowing for a more accurate classification. On the other hand, due to the similarity in movement between P-joints and Push-P joints, their categorization is not as precise as that of the former two.

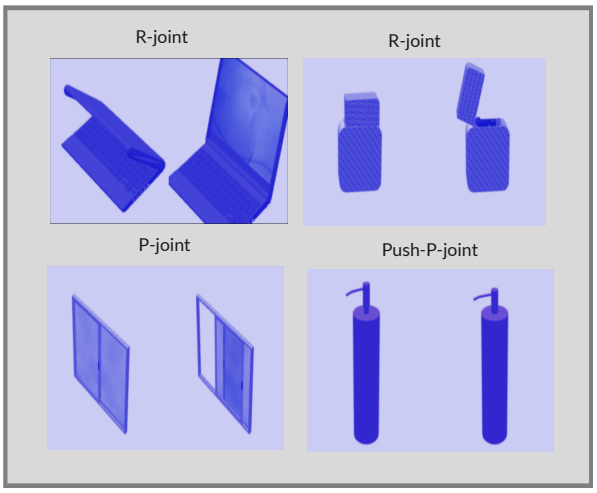

Fig. 4. This figure illustrates how the original images are transformed by applying different types of augmentations, such as changing the color intensity and scaling the size.

E. Limitations

We trained JoTyMo on a simulated environment, and we expect that it will require some fine-tuning and increasing the number of filters in each layer of our design to produce reliable results in reality. Moreover, the interaction with the object should be visible in the RGB image. Otherwise, the system will classify it as an object without joints. We also encountered some challenges during the testing phase, such as push-p joints that were opened being detected as p-joints, and p-joints that were closing being occasionally predicted as push-p joints. Figure 5 illustrates some of these examples.

IV. CONCLUSION

In conclusion, we developed JoTyMo, a system that can achieve higher accuracy in detecting joint types and movements of articulated objects than existing methods. Our model also performs well on unseen images, demonstrating its generalization ability. JoTyMo can serve as an effective and fast

TABLE IV
COMPARING NETWORKS

Network Name	Number of parameters	Accuracy Metrics	
		Accuracy-test%	*Accuracy-train%*
VGG-16 [11]	50.3M	25.73	19
VGG-19 [11]	55.7M	25.87	19
Resnet50 [25]	23.6M	28	93.6
Resnet50-V2 [26]	23.5M	41	98.62
Inception-V3 [27]	21.8M	6	98.84
JoTyMo	**1.2M**	**94**	**99.85**

TABLE V
RESULTS ON ALL UNSEEN CATEGORIES

Joint Type\ Metric	*Precision on unseen data%*	f1-score on unseen data%
No-joint	100	22
P-joint	50	67
Push-P-joint	100	77
R-joint	86	92
Maco-Avg	**84**	**65**

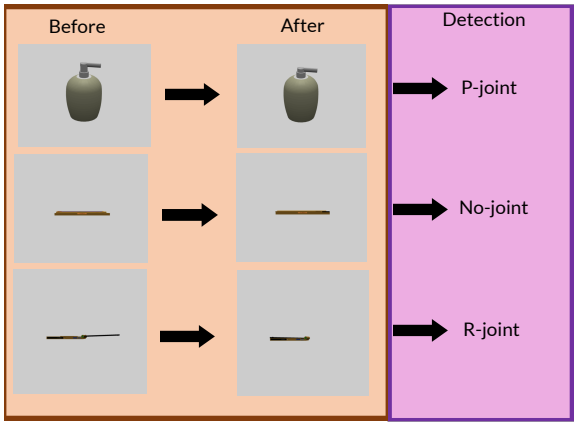

Fig. 5. The figure demonstrates how the algorithm detects different types of joints in the images. In row 1, the algorithm misclassified a push-p joint as a p-joint, because it did not capture the opening between the two objects. In rows 2 and 3, the algorithm correctly identified a R-joint, but failed to recognize the interaction between the objects in row 2, and labeled it as a single object without a joint.

preprocessor for a larger system that aims to learn how to grasp articulated objects through interactive learning. Moreover, JoTyMo can offer a novel approach to classify and detect objects that have visible moving parts based on their motion rather than their type. However, JoTyMo is not a complete solution and there are many ways to enhance and modify it. For instance, we can design a multimodal JoTyMo that can use text or audio inputs from humans to guide the detection process. Another possibility is to use visual transformers and large language models instead of CNNs to improve generalization and create a system that can handle the grasping task end-to-end rather than as a preprocessing step.

REFERENCES

[1] R. Tapu, B. Mocanu, and T. Zaharia, "DEEP-SEE: Joint Object Detection, Tracking and Recognition with Application to Visually Impaired Navigational Assistance," Sensors, vol. 17, no. 11, p. 2473, Oct. 2017, doi: 10.3390/s17112473.

[2] K. Młodzikowski and D. Belter, "CNN-based Joint State Estimation During Robotic Interaction with Articulated Objects," 2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore, Singapore, 2022, pp. 78-83, doi: 10.1109/ICARCV57592.2022.10004277.

[3] Jiang, Hanxiao, Yongsen Mao, Manolis Savva and Angel X. Chang. "OPD: Single-view 3D Openable Part Detection." European Conference on Computer Vision (2022), in press.

[4] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53 (2021). https://doi.org/10.1186/s40537-021-00444-8.

[5] Aman Shrivastav, "Different types of CNN models" OpenGenus. https://iq.opengenus.org/different-types-of-cnn-models(Auguest 21th, 2023).

[6] Xiang, Fanbo, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, et al. 2020. "SAPIEN: A SimulAted Part-Based Interactive ENvironment." ArXiv.org. March 18, 2020. https://doi.org/10.48550/arXiv.2003.08515.

[7] Òscar Lorente, "Image Classification with Classic and Deep Learning Techniques," arXiv.org, May 11, 2021, https://arxiv.org/abs/2105.04895.

[8] Y. LeCun, "MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges," Lecun.com, 2009. http://yann.lecun.com/exdb/mnist/.

[9] A. Krizhevsky, "CIFAR-10 and CIFAR-100 datasets," Toronto.edu, 2009. https://www.cs.toronto.edu/ kriz/cifar.html

[10] F.-F. Li, M. Andreeto, M. Ranzatoand P. Perona, "Caltech 101". CaltechDATA, Apr. 06, 2022. doi: 10.22002/D1.20086.

[11] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv.org, Apr. 10, 2015. https://arxiv.org/abs/1409.1556.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, pp. 84–90, May 2012, doi: https://doi.org/10.1145/3065386.

[13] F. Siahkali, S. A. Alavi and M. T. Masouleh, "SIVD: Dataset of Iranian Vehicles for Real-Time Multi-Camera Video Tracking and Recognition," 2022 8th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS), Behshahr, Iran, Islamic Republic of, 2022, pp. 1-7, doi: 10.1109/ICSPIS56952.2022.10043932.

[14] H. Jalali, S. Samadi, A. Kalhor and M. T. Masouleh, "Model-Free Dynamic Control of a 3-DoF Delta Parallel Robot for Pick-and-Place Application based on Deep Reinforcement Learning," 2022 10th RSI International Conference on Robotics and Mechatronics (ICRoM), Tehran, Iran, Islamic Republic of, 2022, pp. 48-54, doi: 10.1109/ICRoM57054.2022.10025269.

[15] K. Morabia, J. Arora, and T. Vijaykumar, "Attention-based Joint Detection of Object and Semantic Part," arXiv.org, Jul. 05, 2020. https://arxiv.org/abs/2007.02419 (accessed Aug. 26, 2023).

[16] X. Wang, B. Zhou, Y. Shi, X. Chen, Q. Zhao, and K. Xu, "Shape2Motion: Joint Analysis of Motion Parts and Attributes from 3D Shapes," arXiv.org, Mar. 12, 2019. https://arxiv.org/abs/1903.03911 (accessed Aug. 26, 2023).

[17] A. X. Chang et al., "ShapeNet: An Information-Rich 3D Model Repository," arXiv:1512.03012 [cs], Dec. 2015, Available: https://arxiv.org/abs/1512.03012.

[18] Ketkar, N. (2017). Introduction to Keras. In: Deep Learning with Python. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-2766-4-7.

[19] Kingma, Diederik P, and Jimmy Ba. "Adam: A Method for Stochastic Optimization." ArXiv.org, 22 Dec. 2014, arxiv.org/abs/1412.6980.

[20] Ye, Jieping. (2007). Least squares linear discriminant analysis. Proceedings of the 24th international conference on Machine learning. 227. 1087-1093. 10.1145/1273496.1273633.

[21] Vinge, Rikard & Mckelvey, Tomas. (2019). Understanding Support Vector Machines with Polynomial Kernels. 1-5. 10.23919/EUSIPCO.2019.8903042.

[22] Hosna, Asmaul, et al. "Transfer Learning: A Friendly Introduction." Journal of Big Data, vol. 9, no. 1, 22 Oct. 2022, https://doi.org/10.1186/s40537-022-00652-w.

[23] Trappenberg, Thomas. (2019). Machine learning with sklearn. 10.1093/oso/9780198828044.003.0003.

[24] Erdebilli, Babek & Devrim-İçtenbaş, Burcu. (2022). Ensemble Voting Regression Based on Machine Learning for Predicting Medical Waste: A Case from Turkey. Mathematics. 10. 2466. 10.3390/math10142466.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," arXiv.org, Dec. 10, 2015. https://arxiv.org/abs/1512.03385.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Identity Mappings in Deep Residual Networks," arXiv:1603.05027 [cs], Jul. 2016, Available: https://arxiv.org/abs/1603.05027.

[27] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," arXiv.org, 2015. https://arxiv.org/abs/1512.00567.

[28] M. Seeland and P. Mäder, "Multi-view classification with convolutional neural networks," PLOS ONE, vol. 16, no. 1, p. e0245230, Jan. 2021, doi: https://doi.org/10.1371/journal.pone.0245230.

[29] V. Santhanam, V. I. Morariu, and L. S. Davis, "Generalized Deep Image to Image Regression," arXiv.org, Dec. 10, 2016. https://arxiv.org/abs/1612.03268 (accessed Sep. 02, 2023).

[30] Ranganathan, H., Venkateswara, H., Chakraborty, S., Panchanathan, S. (2020). Deep Active Learning for Image Regression. In: Wani, M., Kantardzic, M., Sayed-Mouchaweh, M. (eds) Deep Learning Applications. Advances in Intelligent Systems and Computing, vol 1098. Springer, Singapore. https://doi.org/10.1007/978-981-15-1816-4_7.

[31] [9]X. Sun, H. Jiang, M. Savva, and A. X. Chang, "OPDMulti: Openable Part Detection for Multiple Objects," arXiv.org, Mar. 24, 2023. https://arxiv.org/abs/2303.14087 (accessed Sep. 03, 2023).