Object Detection via Haptic Sensory Information

Universität Hamburg

Seminar Neural Networks SoSe24

by

Nicolai Caspari, Louise Schop

Hamburg, 4th of July 2024

Abstract

This work explores the use of different neural network architectures to process haptic sensory information for object detection from a humanoid robot. Utilizing the child-sized humanoid robot NICO, equipped with high-resolution cameras, binaural microphones, and three-fingered articulated hands with haptic sensors, a novel dataset of 6240 haptic measurements from six differently colored cubes with varying hardness was generated. Inspired by previous work, we implemented and tested various neural network models, including a Simple Classifier, LSTM, GRU, MLP, and CNN, to classify the cubes. Despite extensive testing and hyperparameter tuning, the highest accuracy to predict the right cube achieved was 58% with the LSTM model, which was below the target accuracy of 60% set by the provided baseline machine learning classifiers. Our findings indicate that further model optimization, larger datasets, and potentially different neural network architectures are necessary to improve the performance. This paper underscores the importance of haptic sensory processing in advancing robotic capabilities for human-robot interaction and highlights areas for future exploration.

Contents

1 Introduction

The ability to detect objects using visual or haptic information and measurements is an essential skill for robotic hardware to interact effectively with humans and their environment. To implement these skills accurately and reproducibly, it is usually necessary to have datasets for training and adapting robots to specific requirements. One method to generate these datasets is by using robots specifically designed for the corresponding tasks. These robots can also be in a humanoid form like NICO (Neuro-Inspired COmpanion) [1]. The child-sized humanoid robot is designed as a platform for multimodal interaction [1]. NICO is equipped with high-resolution cameras for stereo vision and binaural microphones for accurate sound source localization, enabling advanced sensory processing [1]. Additionally, it features three-fingered articulated hands and haptic sensors, allowing it to perform delicate manipulations and gain tactile feedback from its environment [1].
NICO's ability to grasp and manipulate objects provides an excellent foundation for generating datasets with haptic measurements. Kerzel et al. [2] present the development of a neural model for haptic object classification through active exploration using the humanoid robot NICO. A large comprehensive haptic dataset with 83,200 measurements was created, and various neural models (MLP, CNN, LSTM) were developed to integrate and classify the haptic sensory data, achieving a classification accuracy of 66.6% [2].
Our work is inspired by the work of Kerzel et al. [2] and aims to develop a neural network based on a novel dataset generated by NICO [1], capable of classifying six differently colored cubes with varying degrees of hardness. The newly developed dataset with 6240 haptic measurements was provided by our advisor from the Knowledge Technology research group[1] at the Department of Informatics at the University of Hamburg, Germany. Our advisor also provided a basic script (Simple Classifier), implemented in Python 3 [3], for classifying the cubes, based on scikit-learn [4] and lazypredict[2] with a prediction accuracy of 60% with the `DecisionTreeClassifier`. Our main goal in this work is to implement and try out four neural network models (MLP, LSTM, GRU, CNN) that ideally exceed the accuracy of the given classifiers from lazypredict. These models could serve as a base for further research in the future, enhancing prediction accuracy even further.

[1]Knowledge Technology Research Group at the University of Hamburg, `https://www.inf.uni-hamburg.de/en/inst/ab/wtm`

[2]lazypredict library for Python 3, `https://pypi.org/project/lazypredict/`

2 Background and Related Work

To classify objects through haptic exploration, Kerzel et al. [5] advanced the state-of-the-art in robot haptic perception by employing a novel approach using an optical 3-axis force sensor mounted on a robot arm. They demonstrated that material classification can be achieved using a multi-channel neural network, primarily based on 1D-Convolutional Neural Networks (CNNs), which processes datasets containing information on vibrations and pressure gathered through the robot arm's lateral and pressure movements. They revealed their intention to enhance their method for more advanced haptic classification tasks, including the identification of uneven surfaces and 3D objects [5], a goal they achieved through the use of NICO (Neuro-Inspired COmpanion) [1].
NICO serves as a flexible and modular research platform equipped with rich sensory and motor capabilities, enabling the embodiment of complex human-like sensorimotor skills in real-world environments [1]. NICO can express and recognize emotions, classify objects haptically, and perform binaural sound localization, supporting a wide range of research scenarios [1]. Due to its adaptability, NICO was used by Kerzel et al. [2] to generate a new dataset of haptic measurements with a new robotic sensor hand. This hand is based on the child-sized, three-fingered Seed Robotics RH4D[3] hand, featuring an embedded OptoForce 3-Axis optical force sensor[4] in the thumb tip that measures forces in three dimensions with high precision [2]. This underactuated, tendon-driven hand securely grasps objects of varying sizes and records 52 measurements, including finger positions, motor currents, and tactile forces, ensuring robust and maintenance-free operation [2]. The paper by Kerzel et al. [2] presents a novel embodied neural model for haptic object classification through active haptic exploration using NICO. An extensive haptic dataset with 83,200 measurements from 16 different objects was created and made available to facilitate the integration and classification of haptic data by neural models. The best models, including MLP, CNN, and LSTM, achieved a classification accuracy of 66.6%, highlighting the networks' ability to integrate data across time domains and different sensory channels [2]. In their work, the Convolutional Network performed as the best model for predicting different objects right (0.666 ± 0.029). Recurrent Architecture (LSTM) performed with a lower accuracy rate at 0.598 ± 0.058, but still higher than the simpler Multi-Layer Perceptron Architecture (0.534 ± 0.021) [2]. This result is supported by Bai et al. [6], who indicated that convolutional architectures are "more effective across diverse sequence modeling tasks than recurrent architectures such as LSTMs". CNNs also show improvement in language modeling compared to established recurrent architectures [7]. Nevertheless, recurrent architectures "have been established as best practice when processing sequential data" [2, p. 3].

[3]For further information, visit `http://www.seedrobotics.com/rh4d-manipulator.html`
[4]From OnRobot as OMD-3 axis sensor `https://onrobot.com/products/omd-force-sensor/`

3 Methodology

The humanoid robot NICO, developed by the Knowledge Technology group in Hamburg [1], was used as the platform of choice for this experiment. Kerzel et al. [2] equipped NICO with a child-sized three-fingered seed robotics RH4D manipulator[5] with an OptoForce 3-Axis optical force sensor[6] [2] at the tip of the thumb. The tips of the index and ring fingers were extended with the same sensors, enabling all three fingers to measure haptic sensory data when squeezing an object.

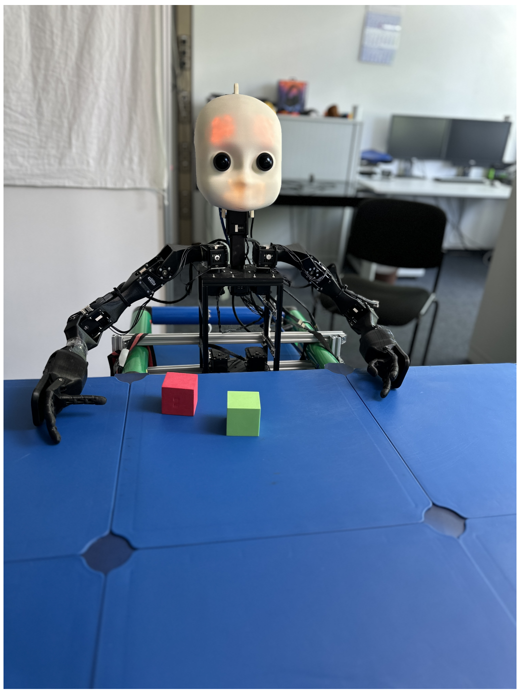

Figure 1: Example of NICO, the child-sized humanoid robot. The hands in this picture differ from the hands that were used for this experiment.

With this Setup, a dataset of 6240 haptic measurements based on 6 cubes with different haptic features in their softness was created. Each cube was grasped 20 times by the robotic hand of NICO and movement steps were measured at each grasp. During the grasp, 52 measurements were taken, recording the following: position of the fingers, current in the motors associated with the fingers, and tactile data of all sensors in the fingers.

[5]For further information, visit `http://www.seedrobotics.com/rh4d-manipulator.html`

[6]From OnRobot as OMD-3 axis sensor `https://onrobot.com/products/omd-force-sensor/`

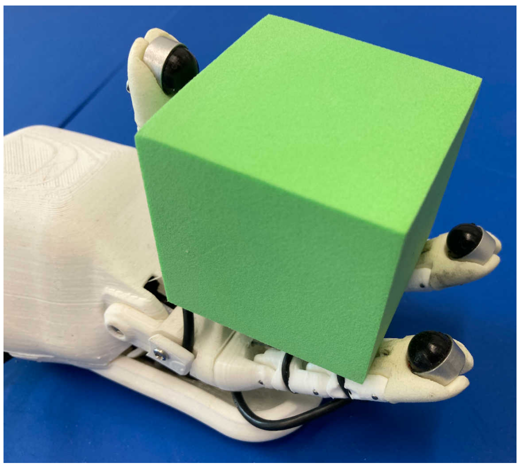

Figure 2: The robotic hand of the humanoid robot NICO holding the green cube, one of the 6 test cubes with different softness. The hand consists of three fingers (index-, ring finger and thumb) which each holding a OptoForce 3-Axis optical force sensor on the fingertip.

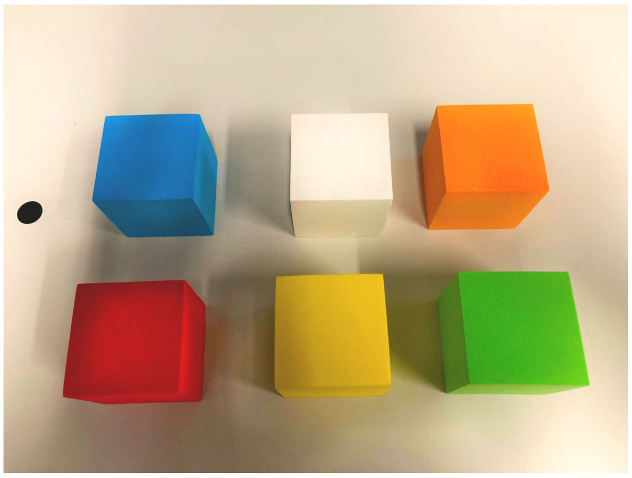

Figure 3: Six colored cubes with different kind of softness were used as test objects to create a dataset with the robotic hand of NICO. These were used to create 6240 haptic measurements, which built the dataset for the training and testing of our neural network models.

With the resulting data, a series of neural network classifiers were trained to evaluate the recorded dataset and to provide a baseline for further research. The calculated accuracy of correct object prediction reached 60% with the `DecisionTreeClassifier` and `NuSVC` from lazypredict, all other classifiers yielded below. Our aim was to develop algorithms using different neural network techniques to experiment on different models and ideally improve that accuracy in classifying the different cubes.
To archieve this, we created four different neural network models (LSTM, MLP,

GRU and CNN) based on a given `SimpleClassifier` model that we received from our advisor. The following hardware and software were used:

- **Hardware:**
  - **Processor:** Intel Core i7 2,6 GHz 6-Cores
  - **RAM:** 16 GB DDR4 2667 MHz
  - **GPU:** Intel UHD 630
- **Software:**
  - **Operating System:** macOS Sonoma 14.1.1
  - **Development Environments:** Spyder 5.5.1 (IDE)[7], TensorBoard [8]
  - **Programming Language:** Python 3.11.5 [3]
  - **Libraries:** NumPy [9], pandas ([10], [11]), PyTorch [12], TorchMetrics [13], SQLite3 [14], PyTorch Lightning [15], TensorFlow [8], Grid Search [16]

To display our results and interpret learning curves, we used Tensorflow's visual extension TensorBoard [8].
The Hyperparametertuning was mainly done manually, but for comparison and experimenting we also used Grid Search [16] to automate the search for potentially matching parameters. Genreal parameters that were tested are (not every parameter was used for every neural network architecture):

- **LSTM-layers:** 1, 2, 3
- **Hidden size** (number of neurons in LSTM and GRU layers)**:** 50, 75, 85, 100, 150, 200
- **Batchsize (All models):** 16, 18, 20, 32
- **Activation Function (All models):** Linear, ReLU, GELU
- **Dropout rates (LSTM, GRU):** 0.1, 0.2, 0.3, 0.4, 0.5, 0.7
- **Fully connected layers (All models except CNN):** 1, 2, 3, 4
- **Neurons in FC layers (All models exept CNN):** 100, 128, 150, 200, 300, 500, 256, 512, 1024
- **Learning rate (All models):** $10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$
- **Patience** (in learning rate scheduler only LSTM)**:** 2, 3, 4, 4.5, 5, 6

[7]Spyder IDE Homepage, `https://www.spyder-ide.org/`

4 Implementation

Here we are going to describe our implementation. For better maintenance, we modularized our script into separate files for the main script, header, and different neural network models:

- main.py
- header_Classifiers.py
- LSTM_Classifier.py
- Simple_Classifier.py
- GRU_Classifier.py
- MLP_Classifier.py
- CNN_Classifier.py

`main.py` serves as the entry point for our program, which creates and manages four different types of neural networks and also handles data preparation. We implemented a configuration block as global variables for easy access, allowing changes like the seed value (`SEED`) for reproducibility or maximum number of epochs per run (`MAX_EPOCHS`). A dictionary is created containing all five different classifier models (”`simple`”, ”`lstm`”, ”`gru`”, ”`mlp`”, ”`cnn`”). New models/classifiers can easily be added to this dictionary for further research. The script then trains each classifier using PyTorch Lightning's Trainer, which manages the training process over the specified number of epochs. After training, the script proceeds to test each model and logs the results, which can be viewed using TensorBoard. This structure allows easy addition of new classifiers and ensures consistent handling of data preparation, training, and testing across different models.

`header_Classifiers` loads the dataset of the haptic sensory experiment (`"experiment.db"`), handles the seed configuration of the script for reproducibility and the dataloaders for the models, to split the dataset into training, validation, and testing sets.

`LSTM_Classifier.py`, a model based on Long Short-Term Memory (LSTM) networks was implemented first. It initializes the LSTM network architecture with one LSTM layer containing 75 hidden neurons, two Fully Connected Layers with 128 connections and finally no dropout rate set. The Activation Function in the end was set to ReLU.

`GRU_Classifier.py` depicts a model based on Gated Recurrent Unit (GRU) networks. The GRU network is initialized with parameters comparable to the LSTM, since they are similar architectures. We implemented two GRU layers with 75 hidden neurons and two additional Fully Connected Layers with 128 connections. There was also no dropout rate in the final model. The Activation Function in the end was also set to ReLU.

`MLP_Classifier.py` defines the Multi-Layer Perceptron (MLP) network architecture. The MLP architecture consists of four fully connected layers with ReLU Activation Functions and descending numbers of connections throughout the model and a final softmax layer for classification.

`Simple_Classifier.py` is the script we received for reference and start. It defines a simple classifier using a straightforward neural network architecture. The model architecture consists of two fully connected layers with ReLU Activation Functions and a softmax output layer.

`CNN_Classifier.py` is the neural network architecture that performed the best in the given paper[2]. Here it defines an architecture consisting of 7 layers: two 1D Convolutional Layers, two Max-Pooling Layers, one Flatten Layer and two Linear Layers. The model uses ReLU Activation Functions and a softmax output layer.

During each training and validation step, our script calculated the loss and accuracy, logging these metrics for TensorBoard analysis. After training, the model was evaluated on a test dataset to determine its performance. Confusion Matrices were built to receive an overview showing the cubes that were recognized correctly and incorrectly (see Results and Discusion section below).

5 Results and Discussion

We evaluated five different neural network architectures (Simple Classifier, LSTM, GRU, MLP, CNN) regarding their prediction accuracy using the dataset of 6240 haptic measurements. The aim was to find at least one model that is able to predict the right cube with better performance than the lazypredict classifiers, meaning a minimum of 60% accuracy. Through a lot of testing, our highest accuracy yielded 58% using the LSTM model. Automated Hyperparameter Tuning with Grid Search and Random Search for the LSTM suggested best possible parameter combinations of: learning rate = 0.001, dropout rate = 0.3 or 0.7, hidden size = 100 and number of layers = 2. We tested the suggested parameters but only found accuracy levels below 40%, so we discarded the approach and continued manually. All models performed best with a learning rate of 0.001 and 120 epochs. The following results illustrate the outcome of our manual tuning process (Table 1).

Table 1: The test accuracy and loss of each neural network model was individually calculated with a seed value of 42 for reproducibility.

Classifier	**Accuracy**	**Loss**
Simple	0.417	1.68
LSTM	0.583	1.75
GRU	0.5	1.63
MLP	0.417	1.61
CNN	0.417	1.67

5.1 Simple Classifier

The simple model we received was meant to serve as a starting point and reference. The all over accuracy did not exceed 41,7% and thus was lower than the top classifiers from the lazypredict method.

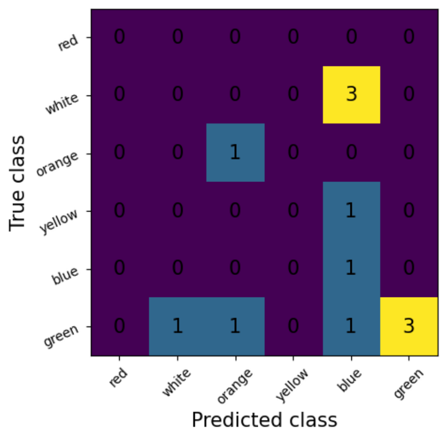

Figure 4: Confusion Matrix Output for the Simple Classifier.

5.2 Long-Short-Term-Memory

The LSTM model was built by us and was expected to perform better than the Simple Classifier. The best accuracy was found with a batch size of 16, hidden layer size of 75, two Fully Connected Layers with 200 connections and number of LSTM layers set to two. The highest performance regarding the prediction of the right cube was found to be at 58.3%. Thus, it was slightly lower than the two best classifiers from the lazypredict method and higher than the Simple Classifier. We did change the hyperparameters systematically but it still remained a trial end error concept.

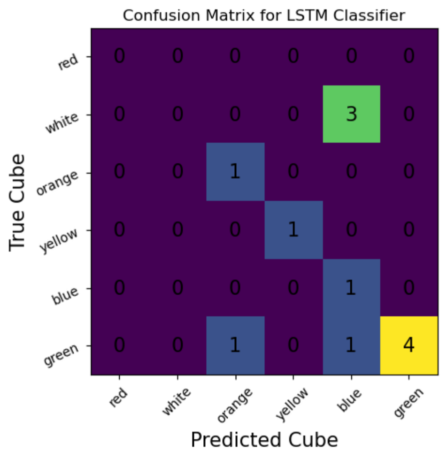

Figure 5: Confusion Matrix Output for the LSTM Classifier.

5.3 Gated Recurrent Unit Classifier

We set up a GRU model to compare an easier and maybe faster approach to the LSTM with a gating mechanism. The expectation was to find similar results to the LSTM. The all over accuracy with the seed set to 42, batch size of 18, no dropout, two Fully Connected Layers with 128 connections, hidden layer size of 75 and number of GRU layers set to two yielded slightly below the LSTM. The highest performance regarding the prediction of the right cube was found to be at 50% accuracy. Here we also changed the hyperparameters systematically, and it remained a trial end error concept as well.

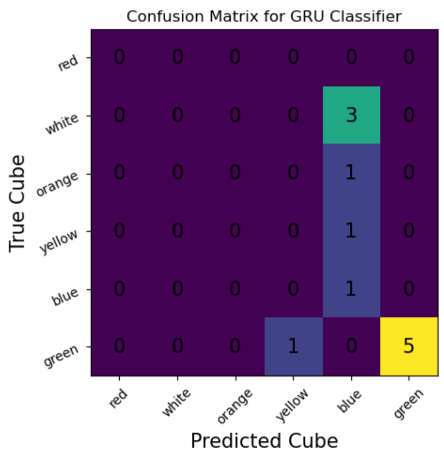

Figure 6: Confusion Matrix Output for the GRU Classifier.

5.4 Multi Layer Perceptron

For a basic approach, we decided to also try out a Multi-Layer Perceptron (MLP) model for comparison. The expectation was to find moderate results compared to the other models. The all over accuracy with four fully connected layers (1042, 512, 256 connections), and six output neurons correlating with the 6 different cubes achieved a performance regarding the prediction of the right cube of 41.7%. Compared to the LSTM and GRU, performance of the MLP model so far was lower and comparable to that of the Simple Classifier. Since there are not many parameters compared to LSTM and GRU, the Hyperparameter Tuning took less time.

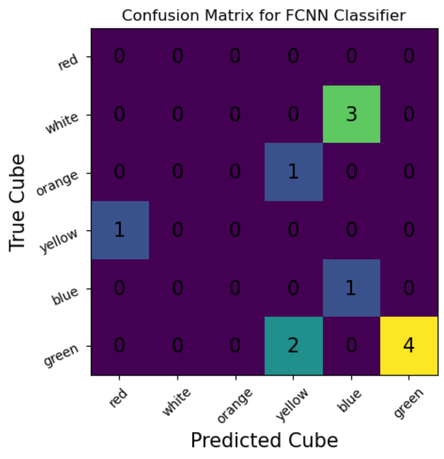

Figure 7: Confusion Matrix Output for the MLP Classifier.

5.5 Convolutional Neural Network

Since the results in the paper of Kerzel et al.[2] yielded best using a CNN, we also tried that model for comparison. The expectation was to find better results compared to the other models. The all over accuracy with seven layers in total, ReLU and Linear Activation Functions and six output neurons achieved a performance of 41.7%. Thus, it resulted in the same accuracy as the Simple and MLP Classifiers and did not outperform the LSTM as expected, like shown in the work of Kerzel et al. [2].

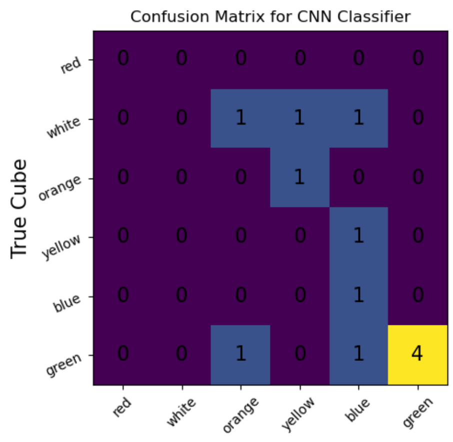

Figure 8: Confusion Matrix Output for the CNN Classifier.

The LSTM and GRU models took the longest time to process but were still compilable in under two minutes. What stands out is that all models are performing good when it comes to recognizing the green cube (seen in the Confusion Matrices). Regarding the red, white, orange and yellow cubes, all the models performed badly. The blue cube is recognized once in each model, but often (mainly three times) confused with the white cube.

6 Conclusion and Outlook

Our work attempted to find a neural network model that could classify the type of cube with the highest possible accuracy. So far, we haven't managed to achieve our goal. Especially, the Convolutional Network performed worse than expected, compared to [2]. The results are still far from being perfectly reliable. To optimize the outcome, other neural network architectures should be tested to find the optimal method for this kind of data, and intensified and extended Hyperparameter Tuning could be applied to the existing models. Additionally, it should be mentioned that the dataset used doesn't contain too much data. Therefore, a larger dataset might positively influence the outcome and performance of our models or any enhanced models implemented in the future.
The use cases for a humanoid robot like NICO and potential descendants are numerous. Haptic sensory data processing plays a huge role in developing machines for all kinds of purposes. An example is the approach to realize teleoperation, where research tries to combine haptic sensory data with visual data to get higher performance of the robots [17]. This field of research is crucial to achieve further goals in building machines that can support humanity in various areas. Exploring data augmentation techniques and integrating advanced neural network architectures such as Transformers [18] and Graph Neural Networks [19] could be different approaches. Furthermore, interdisciplinary collaboration is essential to push the boundaries of what is achievable with haptic sensory data processing. While it is still a long way to go, neural network models will contribute an important share to the big picture. It's also important to consider the ethical implications of deploying such advanced robotic systems to ensure they augment human capabilities effectively and ethically.

References

[1] M. Kerzel, E. Strahl, S. Magg, N. Navarro-Guerrero, S. Heinrich, and S. Wermter, "Nico – neuro-inspired companion: A developmental humanoid robot platform for multimodal interaction," Aug. 2017. DOI: `10.1109/ROMAN.2017.8172289`.

[2] M. Kerzel, E. Strahl, C. Gaede, E. Gasanov, and S. Wermter, "Neuro-robotic haptic object classification by active exploration on a novel dataset," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2019.

[3] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual.* Scotts Valley, CA: CreateSpace, 2009, ISBN: 1441412697.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[5] M. Kerzel, M. Ali, H. G. Ng, and S. Wermter, "Haptic material classification with a multi-channel neural network," in *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2017, pp. 439–446.

[6] S. Bai, J. Z. Kolter, and V. Koltun, *An empirical evaluation of generic convolutional and recurrent networks for sequence modeling*, 2018. arXiv: `1803.01271 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/1803.01271`.

[7] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, Jun. 2017, pp. 933–941. [Online]. Available: `https://proceedings.mlr.press/v70/dauphin17a.html`.

[8] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: `https://www.tensorflow.org/`.

[9] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: `10.1038/s41586-020-2649-2`. [Online]. Available: `https://doi.org/10.1038/s41586-020-2649-2`.

[10] W. McKinney, “Data Structures for Statistical Computing in Python,” in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 56–61. DOI: `10.25080/Majora-92bf1922-00a`.

[11] T. pandas development team, *Pandas-dev/pandas: Pandas*, version 2.1.1, Feb. 2020. DOI: `10.5281/zenodo.3509134`. [Online]. Available: `https://doi.org/10.5281/zenodo.3509134`.

[12] A. Paszke, S. Gross, F. Massa, *et al.*, *Pytorch: An imperative style, high-performance deep learning library*, 2019. arXiv: `1912.01703 [cs.LG]`.

[13] N. S. Detlefsen, J. Borovec, J. Schock, A. H. Jha, T. Koker, L. D. Liello, D. Stancl, C. Quan, M. Grechkin, and W. Falcon, “Torchmetrics - measuring reproducibility in pytorch,” *Journal of Open Source Software*, vol. 7, no. 70, p. 4101, 2022. DOI: `10.21105/joss.04101`. [Online]. Available: `https://doi.org/10.21105/joss.04101`.

[14] R. D. Hipp, *SQLite*, version 3.41.2, 2020. [Online]. Available: `https://www.sqlite.org/index.html`.

[15] W. Falcon and The PyTorch Lightning team, *PyTorch Lightning*, version 2.2.2, Mar. 2019. DOI: `10.5281/zenodo.3828935`. [Online]. Available: `https://github.com/Lightning-AI/lightning`.

[16] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, “On the relationship between classical grid search and probabilistic roadmaps,” *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 673–692, 2004.

[17] Z. Chua and A. M. Okamura, “Characterization of real-time haptic feedback from multimodal neural network-based force estimates during teleoperation,” *CoRR*, vol. abs/2109.11488, 2021. arXiv: `2109.11488`. [Online]. Available: `https://arxiv.org/abs/2109.11488`.

[18] A. Zadeh, C. Mao, K. Shi, Y. Zhang, P. P. Liang, S. Poria, and L. Morency, “Factorized multimodal transformer for multimodal sequential learning,” *CoRR*, vol. abs/1911.09826, 2019. arXiv: `1911.09826`. [Online]. Available: `http://arxiv.org/abs/1911.09826`.

[19] J. Chang, C. Gao, Y. Zheng, Y. Hui, Y. Niu, Y. Song, D. Jin, and Y. Li, “Sequential recommendation with graph neural networks,” in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR ’21, Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 378–387, ISBN: 9781450380379. DOI: `10.1145/3404835.3462968`. [Online]. Available: `https://doi.org/10.1145/3404835.3462968`.