Misan Journal of Engineering Sciences ISSN-E: 2957-4250
Vol. 3, No. 1, July 2024 ISSN: 2957-4242

Enhancing Robotic Grasping Performance through Data-Driven Analysis

Sarah Sabeeh

Computers Engineering Department, University of Basrah ,Basrah ,Iraq
Author E-mail: sarahsabeeh2020@gmail.com

(Received 24 April, Revised 30 June, Accepted 30 June)

Abstract: In automation, reliability in robotic grasping in dynamic environment is still a problem encountered. Further, there is the need to consider deep learning methods, as traditional approaches are not easily flexible in dealing with different objects and situations. In this work, we aim to analyze how well deep neural network models perform in predicting grasp strength based on data collected from the Smart Grasping Sandbox simulation trials. Hence, the proposed approach for analyzing the joint positions, velocities and efforts led to the design of a deep neural network for improving robot grasp performance. The question here could is if deep learning could help better predict grasp stability. To implement the method, we underwent rigorous data pre-processing such as outlier detection, and feature normalization and employed a structured neural network model for training. Our model got a training accuracy of nearly 99% and the test accuracy of nearly 96% demonstrating significant promise. These results were also better than CNN and LSTM where their accuracy rate was 94.12% and 91.81%, respectively. High deep learning performance was proven in robotic grasping, which can contribute to the creation of more flexible and sophisticated robotic platforms. This work also provides the foundation for subsequent research in robotics and automation, with emphasis on the role of data-driven methods in robotic grasping tasks.

Keywords: Deep Learning, Grasp Robustness, Neural Network Model, Predictive Modeling, Robot Grasping

1. Introduction

Among all the basic tasks of robotics and automation, robot grasping is one of the significant and vital tasks where robots are actively used in manufacturing, logistics, healthcare, and as home helpers [1]. Not only the sophistication noted in the interactions of the robots with the objects, but also accuracy and reliability, assessed in the working robots with objects, to enhance safety in various territories [2].

Robotic grasping, however, has emerged as one of the most important tasks in Robotics, the problem of achieving effective grasps remains a difficult one because of variation in the objects in terms of their shape, size, and response to environmental conditions in any complex real life environment [2].

DOI: https://doi.org/10.61263/mjes.v3i1.78

Notably, objects exist in numerous forms, including size, texture, and density which leads to inconsistencies in the effectiveness of the robotic grasp [3]. Further, physical attributes like lighting conditions at the scene, occlusion and clutter also pose another level of challenge to the grasping task, which makes it necessary for the robotic grasping systems to be learning and adaptive with the changing scene [4].

Recently, there occurs a breakthrough on the robot grasping research work caused by the development of deep learning methods. Deep learning models which are based on huge datasets and also sophisticated neural network architectures have demonstrated remarkable prowess in identifying the interdependence between sensory input and grasp results [5]. With the help of these models, it becomes possible to enhance grasp performance for different objects and environments, thus enhancing the development of robotic manipulators of higher efficiency [1].

Despite these advances, current approaches to robotic grasping can still only adapt somewhat to contingency and specific aspects of the objects [6]. It can be observed that conventional approaches such as analytic or geometric types have had a problem of not being easily extendable to new objects or new environment kinds because the features used and assumptions are hand crafted [7].

1.1 Objective and Contribution

The objective of this research is to determine the accuracy of deep neural networks in estimating grasp strength with data derived from simulation trials in the Smart Grasping Sandbox. The goal here is to contribute to the literature on robotic grasping through showing how deep learning can enhance its performance:

1. In this research, we provide a comprehensive study of robot grasping based on deep learning methods, with the help of a dataset obtained from simulation studies performed in the Smart Grasping Sandbox.
2. We show the benefit of using deep neural network models in the prediction of successful grasps, which can be used to enhance grasp ability in real life scenarios.
3. We discuss the conditions that affect grasp strength and give suggestions on how to modify the grasping techniques in a manner that enhances the robustness of robotic systems.

In the subsequent parts of this study, we will describe the data, outline prior research, preprocess the data, develop models, train/evaluate models, present the results, reflect on insights gained, and research we have planned for future.

2. Exploratory Data Analysis

Exploratory data analysis (EDA) is very important to understand the complexity and the characteristics of a dataset as a lever to design or improve models [8].

2.1 Dataset Description

All of the data for this research was obtained through experiments with robotic grasping that were conducted in the controlled environment of the Smart Grasping Sandbox simulation environment. This information is crucial for assessing the reliability of various grip types and comparing the effectiveness of a deep learning model for the tasks related to robotic manipulation. Stored as a flat table format and often saved in CSV format, the dataset can be readily downloaded for analysis with basic data manipulation tools as well as standard

libraries. The rows in the dataset represent a particular experiment while the columns present certain measurements conducted during the grasping maneuver. Some of the feature variables of the data set include:

- **Joint positions, velocities, and efforts**: These features offer an opportunity to receive the detailed data on the kinematics and dynamics of the robotic hand. Joint positions depict the configuration of the robotic hand in space, velocities address the rate of change in the joints, and efforts represent forces or torques applied upon the joints. These attributes are tasks related to grasp strength since they define the control and stability of the grasp.
- **Grasp robustness**: This is an indication of the security of the grasp and the stability, which is determined by the reliability of the distance between the objects and the palm in the course of the shake. This is a vital measurement of how much grasp efficiency is maintained during transient conditions, which is a real-world setting in most uses of a robot.
- **Experiment and measurement identifiers**: These identifiers assist in specifically identifying certain experiments or measurements that are incorporated within the dataset in order to evaluate variations and trends among multiple trials.

In the process of data collection, performing of simulated grasping experiments took place at the Smart Grasping Sandbox. In each trial, the robotic hand was employed to grasp a ball object and then to perform shaking movements, in order to replicate realistic tasks. During these experiments various measurements were recorded such a joint parameters and grasp robustness so that expectations to grasp dynamics and performance factors may be evaluated.

2.2 Dataset Information

The given dataset has 992,641 records for 30 columns where each of them is a number for experiments like experiment number, robustness, and various features having prefixes like H1_F1J2_pos. Most importantly, the analysis of the dataset reveals that there are no missing values signifying that the data is reliable for further analysis. An interesting finding is the fact that 28 from 30 columns have a floating-point data type, meaning that the data are mostly numeric. One column is of int64 and it can handle discrete numerical data while the other handles categorical data that is string data in this case. The memory usage for the said dataset is approximately 227. 2 MB, which is a clear indication of the large amount of memory resources required for its execution.

2.3 Data Splitting

In order to split the dataset as training and testing sets required for training and testing the model the data was split as follows. Particularly, the training set was made from 80 % of the dataset, whereas the test set comprise of the remaining 20%. This partition allows the model to be trained well, and at the same time, there is an independent dataset that would test its accuracy, as well as determining how well the model could generalize.

2.4 Statistical Summary

Among all the types of summaries, the most significant one calculates the dataset's qualities and shows essential information about them. The average robustness is approximately 77.22, with a standard deviation of about 53.07, indicating the range in grasp robustness across different tests. To get a better understanding as to

how the H1_F1J2_pos and H1_F1J2_vel factors deviate from the mean during grasp actions, it is possible to consider average values and standard deviations. Most specifically, the identification of outliers, here represented by very big and very small values, highlights the importance of advanced analysis techniques in order to ensure the reliability and precision of future modeling and inferential operations.

2.5 Distribution of Numerical Variables

Histograms and box plots are very useful ways of graphically displaying a numeric variable to help understand how the data is distributed and the degree of variability. The above graphical systems are useful in identifying trends, anomalies and any suspicious observation in a set of data. Histograms enable one to understand how the data values are distributed and thus the trend and the tendency of values.

In Fig. (1) below, we can observe the distribution of the nominal variables in the given dataset. Regarding the histogram resulting from the code, the X-axis is labeled according to the range of values or bin number for each numerical variable in the dataset, and for the Y-axis there are units of the numerical variables, which are meters for length, seconds for time and so on. The Y-axis represents the frequency or count of the data points which has been grouped under a particular bin where the values are expressed in terms of dimensionless counts. The histograms provide dioptric analysis of the distribution of each of the numerical variables and inform on the spread and the center of gravity of the data.

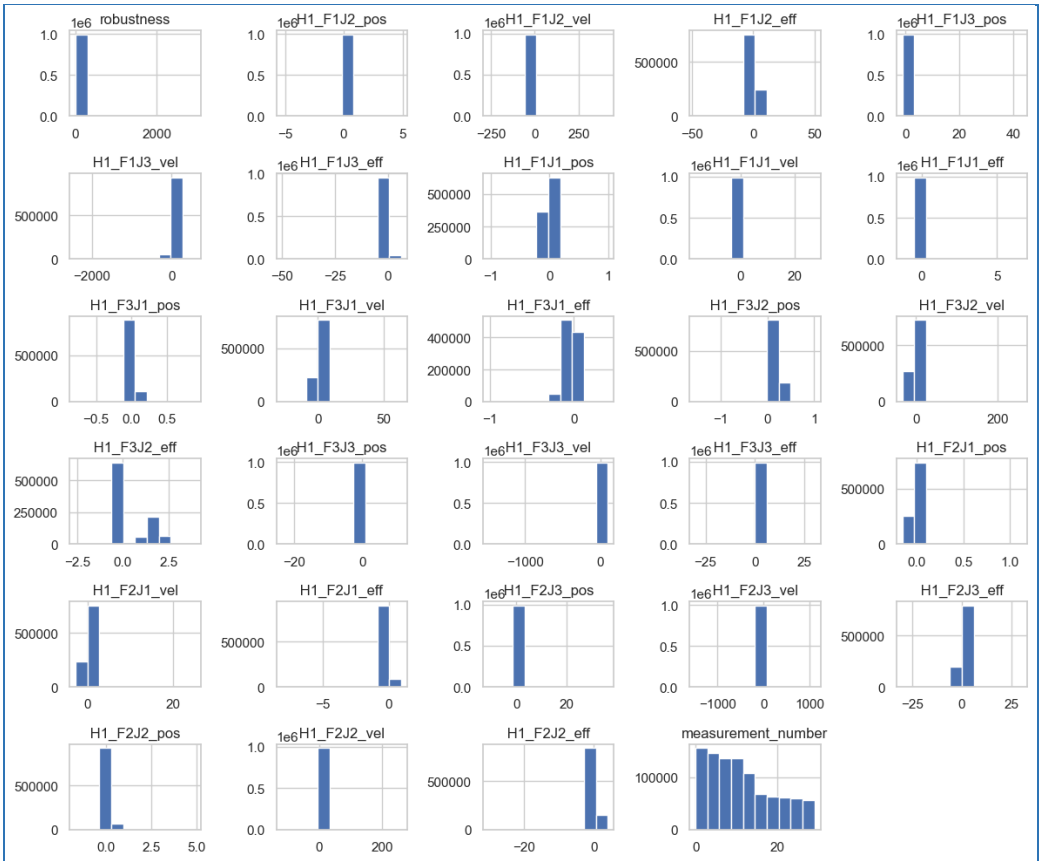

Fig. 1: Distribution of Numerical Variables of the Dataset.

2.6 Outlier Detection

Outlier identification and management is crucial from the standpoints of data quality, measurement accuracy, and analytic and modeling methodology. We used valid techniques in handling outliers especially box plots to help us identify any anomalous sets. Therefore, through preventing and eliminating outliers, we minimize the probability of influencing the results or forecasting of the models, thus enhancing the reliability of our results. Consequently, this measured approach of detecting outliers ensures our analyses are based on sound data and encompassing, thus paving the way for informative results and accurate models.

The Box Plot method, on the other hand, uses the interquartile range (IQR) to identify outliers. More specifically, any value that is outside of the range of plus or minus 1.5 times the Interquartile Range above the third quartile or below the first quartile is considered as an outlier. This method was chosen because it offers ease of use and provides a good way of sorting out outliers that may affect the modeling process. You can refer to Fig. (2) to get the graphical representation of the numeric variable in the form of box plots.

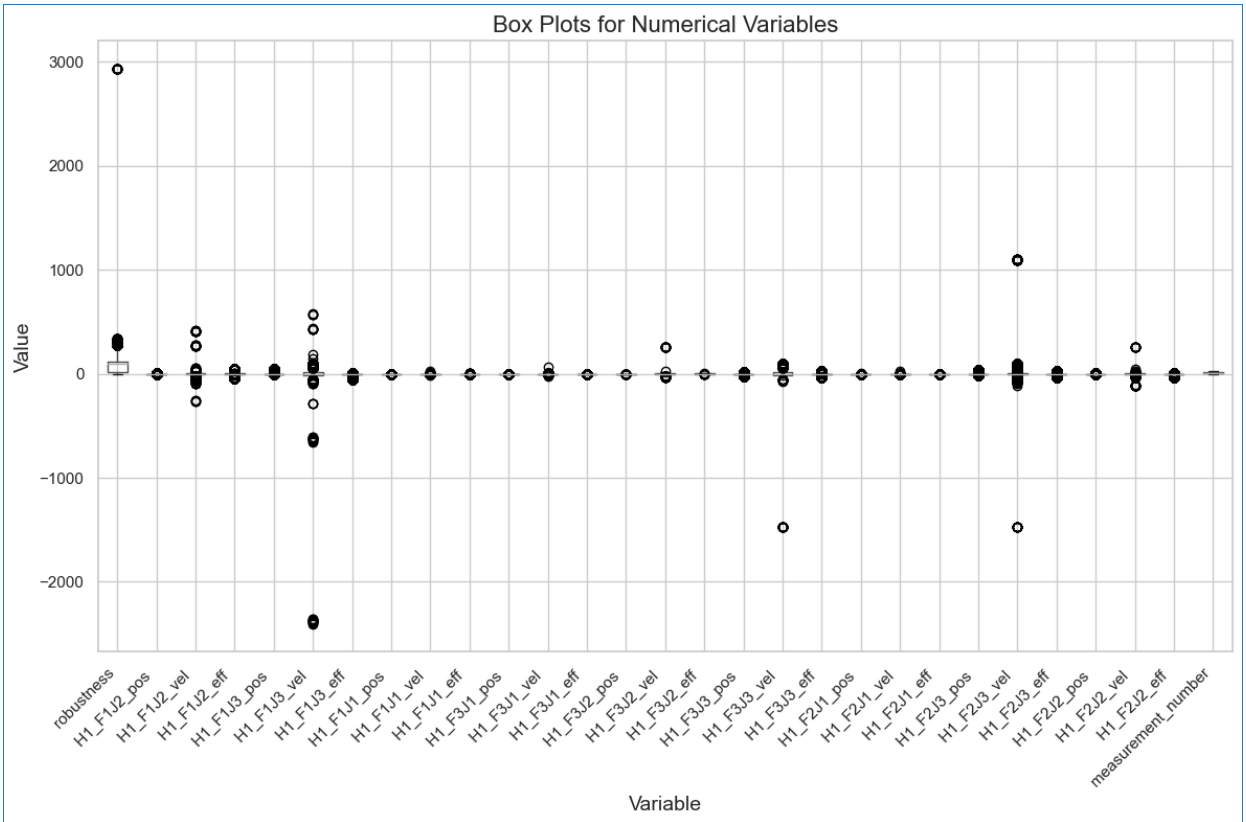

Fig. 2: Box Plot Representation for the Numerical Variables.

2.7 Correlation Matrix

Correlation matrix heatmap allows us to see the relation between the variables at a glance. It assists in selection of features to use, model building as well as exploring data. This way, we can see dependencies and relations in the provided dataset, since we are searching for patterns of correlation.

Fig. (3) illustrates the relationships between different features in the form of heatmap. As we can see, this information can be useful for making better decisions and gaining new knowledge. This heatmap allows to see the positive and negative correlations between any two variable in a data set and the strength of these correlations. This colormap highlights positive correlation values in red color and negative correlation values in blue color and no correlation in white color.

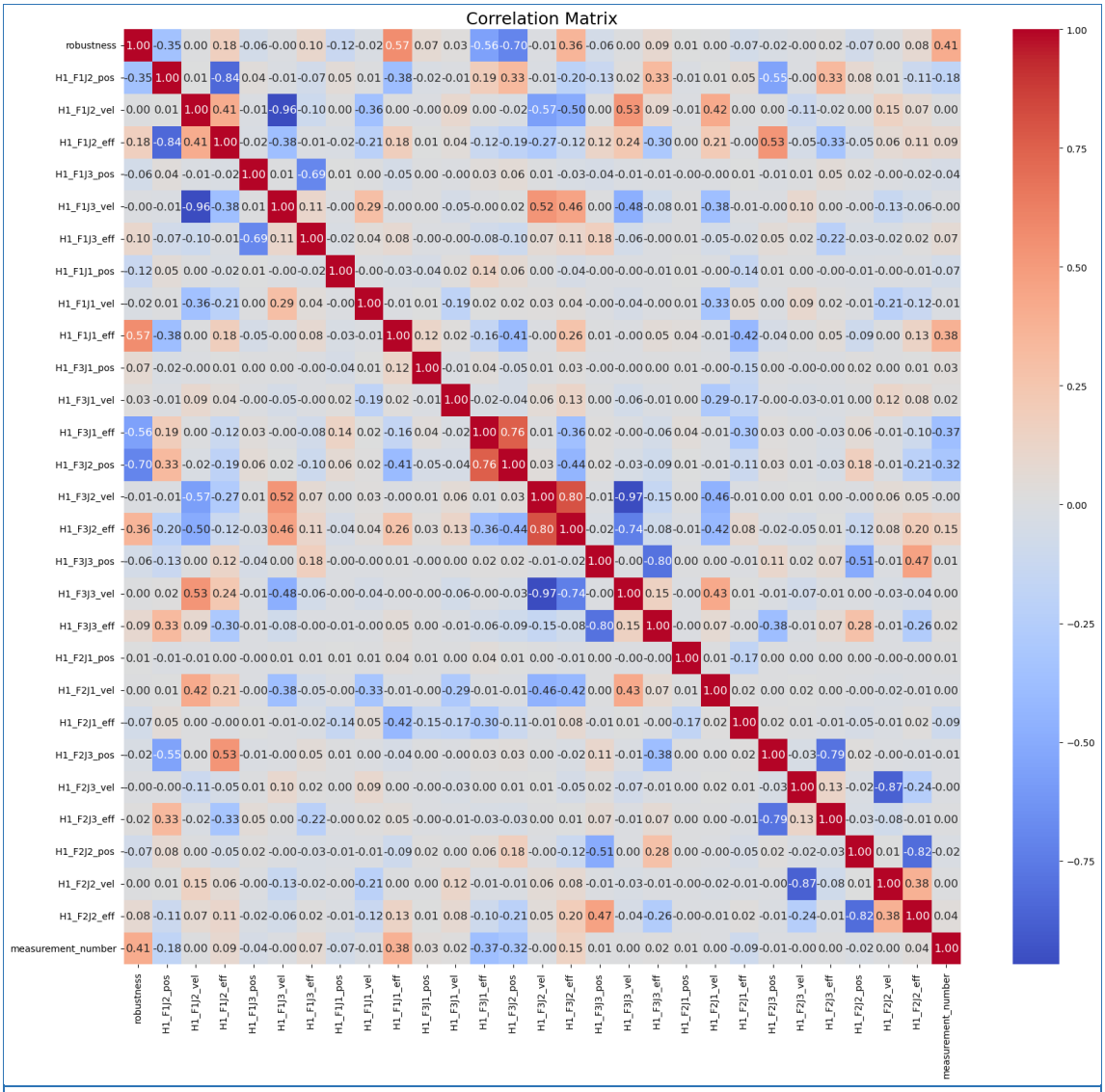

Fig. 3: Heatmap Representing the Relationships between Features.

2.8 Relationship between Experiments and Measurements

Knowing how experiments and measurements are related can give us clues about how data is distributed and varies from experiment to experiment. Understanding maximum as well as the mean and the standard deviation of measurements of each experiment will give an insight into the general structures and distribution of data sets. Table (1) illustrates the connection between experiments and measurements.

Table 1: Relationship between Experiments and Measurements

Feature	Measurement
Number of experiments	53,937
Max measurements	30
Approx. Mean measurements	17.4037
Approx. Std. measurements	9.6501

2.9 Distribution of Grasp Robustness

Looking into the values of grasping robustness in discrete categories makes it easier to determine the distribution pattern. Thus, by dividing these values into such categories, the use of graphs will be much easier, giving more opportunities for future analysis and decision-making. This type of categorization enables us to explore more into grasp quality at multiple levels of stability so we can identify possible secularities that would impact model training and assessment.

Fig. (4) highlights the density function of the column labeled “Robustness. “ This plot shows four histograms of robustness values grouped into four quantiles (q0, q1, q2, and q3). Each of the subplots represents one of the groups marked as ‘0’, ‘1’, ‘2’, and ‘3’, which presents the distribution of the robustness values within the groups. These histograms show the distribution of robustness values to highlight the variation and relative concentration of the quantile-based categorizations over the total dataset.

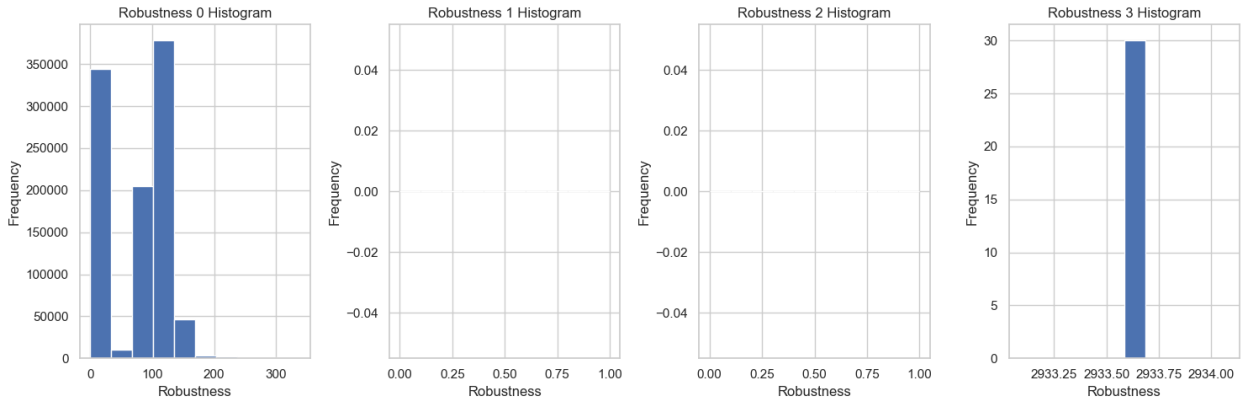

Fig. 4: Distribution Patterns of Grasp Robustness.

2.10 Distribution of Grasp Quality

When preparing the dataset for a binary classification task, a specific threshold is used to filter the data and apply binary labels accordingly. This process is significant in transforming the continuous grasp robustness values into categories that help in classification tasks. Graphing the distributions of these categories gives a perspective of the balance of the classification problem, which is useful for further modeling and analysis.

The distribution of grasping quality is presented in the Fig. (5). The plot shows a histogram that gives the distribution of the categories depending on the 'robustness' of the data points in the context of the dataset. The 'robustness' values have been transformed into binary categories: A score of 0 represents 'Bad' where robustness value is below 100 and 1 represents 'Good' where robustness value is 100 or above. Here, X-axis represents 'Bad' and 'Good' as two variables, and Y-axis represents the number of data points for each variable. The extent to which the dataset is split between these two types of robustness is shown in this visualization.

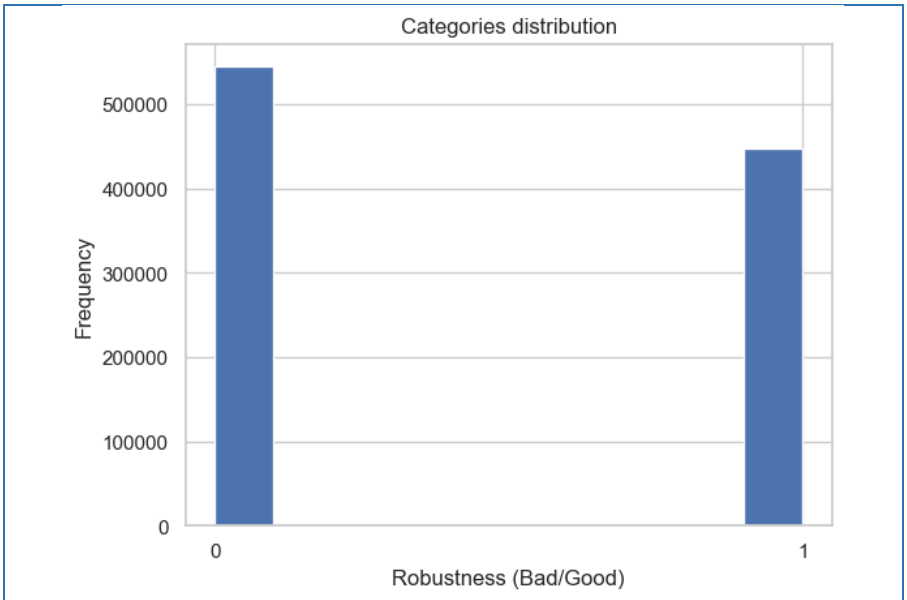

Fig. 5: Distribution of Grasp Quality.

3. Related Work

Robotic grasping is one of the most researched technologies over the years within the domain of robotics. Many techniques and strategies have been developed to address the challenges related to possessing powerful and effective grips.

Traditional approaches to robotic grasping commonly rely on analytical or geometrical models that determine how a robot should grasp objects based on their shapes, feedback from contact forces, or certain prescriptive grasping techniques. These techniques depend on the features that are imposed by the expert and rules that define the points and angles for interacting with the object. For example, there is the Grasp Taxonomy discussed by Feix et al. [9], which classifies grips by the location of contact between the robot's tool and the

objects, and the Grasp Quality Metrics developed by Ferrari and Canny [10], which quantify the stability and feasibility of grips, using mathematical criteria.

Although there have been successes with past methods in certain circumstances, they can run into problems when attempting to apply the method to new objects or settings as they work from the prior knowledge of certain features and conditions. Furthermore, they may not make best use of all info which prevents them from adjusting to different object shapes and different environment conditions.

Many experiments and researches have been carried out to explore the effectiveness of robot grasping in the real environment and synthetic environments. Such investigations often focus on the comparison of various approaches and technologies based on shared or similar datasets or experiment scenarios, assessing variables such as the accuracy of object picking, time to completion, and processing complexity.

For example, Mahler et al. (2017) presented Dex-Net, a deep learning system geared towards converting point cloud models of objects into accurate grasp plans. Dex-Net uses large synthetic samples of grasp examples from which it learns synthetic point cloud data and analytic grasp metrics in training its neural networks. This allows robots to estimate secure grasp poses on novel objects in populated environments [11].

Varley et al. (2017) describe the architecture for robotic grasp planning based on shape completion with a 3D CNN. Using 440,000 3D exemplars as the training data, the CNN completes occlusions in two steps. Completed Objects: At runtime, 5D point clouds are utilized to plan grasps on the finished objects. Focus is given to computational cost as most calculations are done in the training phase. The effects of factor completion on quality are also investigated with regards to object familiarity, training set size, and test on new objects [12].

The study by Levine et al. (2018) seeks to understand how deep reinforcement learning can be used in improving hand-eye coordination for grasping tasks in robots. To do so the authors gather more than 800,000 grasp attempts from different environments and then train a neural network to make predictions on successful grasps based on visual input only. The outcomes demonstrate specific enhancements in grasp success rates, which aims to support the notion of using data-centric methodologies in intricate robotic applications [13].

Dmitry Kalashnikov, et al., fast forward to 2018, proposed QT-Opt, a scalable reinforcement learning technique for vision-aided robotic manipulation. With over 580k real-world grasp attempts, QT-Opt trains a deep Q-function neural network with 1.2M parameters. In this case, using RGB vision from an over-the-shoulder camera and achieving 96% success in perceiving previously unknown objects, the system is capable of fine-tuning the grasp strategies [14].

Likewise, Levine et al. (2019) also cover training legged robots on how to learn more motility skills effectively and efficiently. They created an approach based on deep reinforcement learning to facilitate the training of robots to pick objects based on raw sensor information. This helps the robots enhance their manipulative experience with various objects and the environment without relying on special designed features or prior grasping strategies [15].

Zhu, et al. (2019) present a robotic semantic grasping technique with the goal of generating accurate poses for the robotic manipulators to grasp objects in a six-degree-of-freedom (6DOF) perspective. Grasping pose estimation is the next step that involves using pixel mask along with the point cloud data of the target object

to estimate the 6DOF grasping pose of the identified graspable locations. It allows for a perpendicular contact with the surface of the object in consideration [16].

L. Bergamini (2020) propose an end-to-end deep learning solution for robotic object grasping using only vision in unstructured environments. They allow for autonomous grasp planning without any previous training or programming by introducing a new network, named grasp prediction network with an optimized loss function. The method reaches 75% accuracy to discern low-resolution 2D picture and has successful performance with the Baxter robot in the ROS environment [17].

Recent advancements in deep learning methodologies suggest great potential in improving methods through which objects are manipulated by robotic means. These approaches help enhance the grasping techniques, whereby proficient utilization of large datasets, statistic and neural networks enables the grasping to be proficient under several objects and environments, and opens a future of competent and versatile robotic systems. For example, in the Robotic Grasping and Manipulation Challenge (RoboCup) [18] authors staged the contest where participants needed to design the autonomous systems performing the most efficient robotic grasping and manipulation for the assigned tasks. Likewise, the YCB Object and Model Set provides an objective benchmark that reflects the performance of robust manipulation techniques in a variety of tasks and with a plethora of objects.

While there have been some developments in the methodologies for robot grasping [19], more advancements are still attainable and the current methods are far from perfect [20]. Non-parametric methods may even fail to generalize well for new objects or environments they were never trained on, due to their inherent set of defined features and assumptions. Deep learning methods also pose a great prospects, but they need a large amount of labeled data and computational resources for their work and therefore can be hardly applied in practice. Difficulty in addressing factors such as occlusions, clutter, and uncertainty continues to be considered major difficulties [21].

4. Preprocessing and Model Development

In this section, we will detail the data preparation phase where we explain how we train the model, and the specifics of designing a deep neural network to predict the grasping force.

4.1 Preprocessing Steps

After the data is preprocessed and analyzed, we will employ a variety of techniques to prepare the data for model training [5]. The first step is to calculate the average value in order to estimate the quality of grasping in each experiment. Then, we categorize the continuous data into quantiles to minimize the analysis process while revealing the patterns. Also, the ‘experiment_number’ and ‘measurement_number’ columns do not need to be included as they are not important features for the model.

At a later stage, we assess how well something is retained and this can be divided into broad categories with specific indices. This method eliminates the unnecessary processes and aids in training the model effectively. In order to keep the dimensionality similar and to avoid having to normalize the input features during training, the input features are normally scaled to have zero mean and unit variance. This process is carried out before training to reduce influence of features in different scales and thus provide consistent training.

Last but not the least, the preprocessed dataset is divided into training and testing data using the train_test_split function. This is beneficial in that it enables the testing of the generalization capabilities of the model with new data. This can be done by testing the model on a different set of data and comparing the grasp quality predicted by the model with the actual grasp quality on the test data. To recap, Table (2) is a summary of data preprocessing and transformation:

Table 2: Summary of Data Transformation and Preprocessing Steps

Step	Description
Calculate Mean Robustness	Aggregate grasp performance by computing mean robustness per experiment.
Categorize Continuous Data	Segment continuous data into discrete groups to facilitate analysis.
Drop Irrelevant Columns	Remove irrelevant columns ('experiment_number', 'measurement_number') to focus on relevant features.
Binarize Target Variable	Create binary labels for grasp quality based on a threshold value.
Apply Feature Scaling	Standardize feature values to ensure uniformity and aid algorithm convergence.
Split Dataset	Divide the dataset into training and testing subsets for model evaluation.

4.2 Model Architecture

The neural network model for predicting grasp robustness has dense layers followed by ReLU activation layers with four layers:

1. **First Layer**: This layer has 1024 units as well as batch normalization operation. It offers a computational shortcut known as batch normalization that aids to minimize internal covariate shift and hasten the training stage. The ReLU activation function is applied to allow the model to learn more quickly and minimize the feature maps dependency on initial weight values, which is particularly important in deep networks.
2. **Second Layer**: Likewise, the second layer has 1024 nodes, applies ReLU activation function to the input, and batch normalization is applied. It also helps to avoid overfitting because there are enough units within this model to be able to learn from the high-dimensional inputs fed into it.
3. **Third Layer**: This layer consists of 512 units, applies the ReLU activation function, and has the batch normalization option activated. Reduction in the number of units contributes in the gradual reduction of dimensionality and attention to critical features learned in the successive layers.
4. **Output Layer**: The last layer is an output layer that contains only one node because this is an unsupervised regression problem of predicting a single numerical value for grasp robustness.

The number of layers and units was chosen through careful planning and the model architecture was optimized to be somewhat complex yet capable of generalizing well across new data. The ReLU activation function was chosen due to its relative simplicity and remarkable propensity to not suffer from the vanishing gradient problem that hampers training in deep networks.

Table (3) provides the model architecture where the unit counts for each of the layers used, the activation functions used in this model and all other operation accepted in the model construction.

Table 3: Summary of the proposed model architecture

Layer	Description	Units	Activation Function	Additional Operation
First	Hidden layer with batch normalization	1024	ReLU	Batch Normalization
Second	Hidden layer with batch normalization	1024	ReLU	Batch Normalization
Third	Hidden layer with batch normalization	512	ReLU	Batch Normalization
Output	Output layer for grasp robustness prediction	1	None	-

4.3 Mathematical Representation of the Methods

In this subsection, we incorporate the mathematical expressions used in the analysis to offer more transparency. These representations provide accurate descriptions of preprocessing steps, architectural details of the neural networks used in the study, loss functions, and optimization algorithms used.

1) Preprocessing Steps

- **Mean Robustness Calculation:**

  For each experiment (i), the mean robustness R_i is calculated as [22]:

  $$R_i = \frac{1}{2}\sum_{j=1}^{n} r_{ij} \quad (1)$$

  where r_{ij} represents the robustness measure of the j trial in the i-th experiment, and n is the total number of trials..

- **Feature Scaling:**

  Each feature x is standardized to have a mean of 0 and a standard deviation of 1 [23]:

  $$x' = \frac{x-\mu}{\sigma} \quad (2)$$

  where x' is the scaled feature, x is the original feature, μ is the mean, and σ is the standard deviation.

2) Neural Network Model

Our neural network model consists of several dense layers. The operations in each layer can be represented as follows:

- **Layer Operation:**

  For layer l with input $z^{(l-1)}$, weights $W^{(l)}$, biases $b^{(l)}$, and activation function $f(\cdot)$ [24]:

  $$\boldsymbol{z^{(l)}} = \boldsymbol{f}(W^l z^{(l-1)} + b^{(l)}) \quad (3)$$

  where $\boldsymbol{z^{(l)}}$ is the output of the layer.

- **Batch Normalization:**

  Batch normalization normalizes the input of the activation function [25]:

  $$\hat{\boldsymbol{z}}^{(\boldsymbol{l})} = \frac{\boldsymbol{z^{(l)}} - \boldsymbol{\mu_z}}{\boldsymbol{\sigma_z}} \quad (4)$$

  where $\boldsymbol{\mu_z}$ and $\boldsymbol{\sigma_z}$ are the mean and standard deviation of the batch.

- **Loss Function**

  The loss function used for regression in predicting grasp robustness is the Mean Squared Error (MSE) [26]:

  $$L(\hat{y}, y) = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 \quad (5)$$

  where $\hat{y}_i$ is the predicted robustness, y_i is the actual robustness, and m is the number of samples.

3) Optimization

The optimization algorithm used is Adam, which adapts the learning rate for each parameter:

- **Parameter Update:**

  Adam (Adaptive Moment Estimation) is an optimization algorithm that adjusts the learning rate for each parameter. The equation is as follows [27]:

  $$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \frac{\alpha}{\sqrt{\hat{\boldsymbol{v}}_t} + \in} \hat{\boldsymbol{m}}_t \quad (6)$$

  where $\boldsymbol{\theta_t}$ are the parameters at iteration t, α is the learning rate, $\hat{\boldsymbol{m}}_t$ and $\hat{\boldsymbol{v}}_t$ are bias-corrected first and second moment estimates, and $\in$ is a small constant.

4.4 Rationale behind Model Design

The formulation of the model was based on addressing complexity and nonlinearity of the problem at hand. Deep neural networks were chosen due to their ability to model sophisticated structure and dependencies in data by using hierarchies [28]. The ReLU function is selected as the activation function because it helps to avoid the vanishing gradient problem and speeds up the convergence in training.

To address the internal covariate shift issue, batch normalization was incorporated into each layer to help stabilize and speed up the training process. The "Adam" optimizer was chosen because of its inherent characteristic of having a learning rate that adapts for each parameter separately for improved speed and reliability when approaching convergence.

4.5 Hyperparameters

Hyperparameters are important in the results and speed of the deep learning models. In this paper, a vast amount of experimentation and a grid search strategy were used to determine the appropriate hyperparameters:

1. **Learning Rate**: 0.001. During iterations towards a minimum of the loss function, the learning rate defines the step size at each iteration. Therefore, 0.001 was chosen given it offers pretty good convergence speed and at the same time is fairly stable. The learning rate was chosen based on some initial tests that were conducted in order to find the best combination between the convergence rate and the best stability. However, it may cause occasional fluctuation because of more frequent updates but it helps in faster approach towards the solution in early iterations. To overcome such a problem, we used the learning rate scheduler that decreases the learning rate by a factor when the validation performance does not improve any more. This approach is used in fine-tuning of the model in the subsequent stages of training so as to minimize the occurrence as well as the extent of fluctuation in the process.
2. **Batch Size**: 32. The batch size means the number of examples is used for training in one iteration of the learning process. It was decided to use the batch size of 32 as it allows for reasonable usage of the CPU and GPU resources while providing a stable training process without significant fluctuations.
3. **Epochs**: 30. This means the number of epochs indicate how many times the process has traversed through the training set. Setting it to 30 gives the model enough chances to learn from the data it uses in the network. After 30 epochs, we noticed that validation loss stopped increasing further, meaning that the model is no longer performing better on the validation set. This plateau implies that the model has got the highest accuracy, as can be realized from the current data set and from the current hyperparameters setting.
4. **Optimizer**: The Adam optimizer was selected due to its proven ability to perform adaptive learning rate optimization and efficient computational speed.
5. **Activation Function Value**: ReLU. For the activation function of all hidden layers, the Rectified Linear Unit (ReLU) was employed due to its endorsement in minimizing the vanishing gradient problem.
6. **Batch Normalization**: Used after each hidden layer. To address internal covariate shift, batch normalization was added before each layer to normalize the inputs.

7. **Dropout Rate**: 0.5. The dropout rate remained at 0.5 was used to avoid or reduce overfitting of the model. During the training of dropout, it randomly sets a fraction of the input units in network to zero at each update, which allow the network to be less sensitive to the weights of individual neurons.
8. **Regularization**: L2 Regularization. L2 regularization was used so as to avoid cases where weights become very large hence leading to overfitting. This technique ensures that the weights learned by the model remain small thus enforcing the model to be simple and avoiding overfitting.

The choice of these hyperparameters, as well as their fine-tuning, were important for the achievement of the best result. This was done by conducting an iterative process of model tuning and validation that guarantees a highly accurate as well as a generalized model given the tested final test accuracy of roughly 96.77%.

4.6 Model Complexity

The size of our model can be measured by the number of parameters it has, which in turn determines its ability to fine-tune by data or computation needs. In our architecture, we used deep convolutional neural network, where the total parameters constituted of all the layers with different parameter values. In particular, the model architectures involved 4 convolutional layers and 2 fully connected layers with roughly 1. 2 million parameters. The reason for such a large number of parameters is to identify the finer details in the data set and make better predictions where necessary.

However, the large parameter count also implies having to handle computational resources and training time efficiently. The size of the model, in particular, the memory usage, amounts to some 50 MB, which by present-day standards is quite possible to host on existing hardware with the required amount of RAM. Although it has many hyperparameters, we used methods like overfitting prevention methods including regularization, dropout, and batch normalization to minimize the chances of the model ‘over-learning’ the data and therefore failing to perform well on unseen data. Thus, one aimed to balance model complexity and training strategies and achieve effective and efficient deep learning model used for high performance.

4.7 Simulation Environment

The simulation aspect of the project involves using the Gazebo software, which is a physics simulator that comes with the Robotic Operating System (ROS). It has a UR10 robot that is embedded with Shadow’s Smart Grasping System [29] and utilizes Microsoft Kinect for 3D vision sensing. The environment entails objects like cricket ball and a drill. The simulation is packaged into a Docker container that simplifies the deployment process across various environments and operating systems and allows for local testing as well as experimentation with cloud resources. Furthermore, it uses a total of two figures (the cricket ball and the drill) and makes use of the motion planning in ROS’s MoveIt! planning library for safe and efficient robot manipulation. Fig. (6) depicts the smart grasping sandbox in the construct.

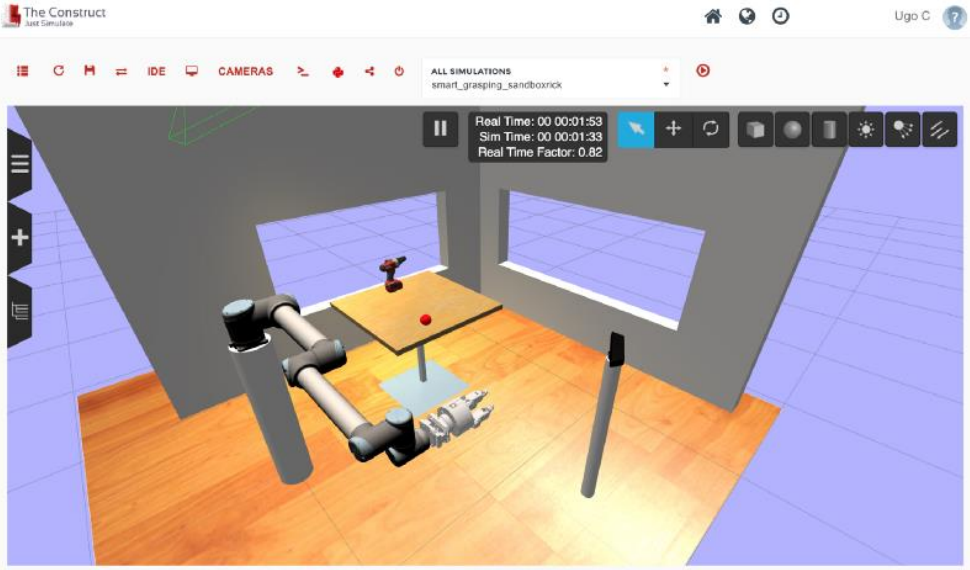

Fig. (6): The smart grasping sandbox in the construct.

5. Training and Evaluation

5.1 Training Procedure

While fine-tuning and testing our neural network model, we used TensorFlow, an often-used deep learning tool [30]. This process included fundamental stages relevant to model efficacy in predicting the quality of robotics grasping. Firstly, it was necessary to set critical hyperparameters like learning rate, maximum number of epochs to train, and the batch size ahead of model training. Another component of our architecture is the neural network which incorporates three fully-connected layers, all utilizing ReLU activation and batch normalization. This was done intentionally to capture the inferences of certain patterns in the set data to yield accurate prediction.

In the process of training during this exercise, the emphasis was made towards computation of the 'loss,' that is the disparities between the outputs made by the model and the corresponding labels. To make alterations to the model, we used the mean sparse softmax cross-entropy loss function as this type of loss function is used for classification problems. In training our model we used the Adam optimizer that minimizes this loss during training through updating the parameters of the model. This approach of optimization provides a continuous update of the parameter and allows for a proper convergence that can help the model to determine and understand the necessary patterns for grasping quality in a robotic system.

During the training, our model was trained iteratively and passed through several training epochs where it encountered various parts of the dataset. In every training round, the system precisely assessed the loss and then used backpropagation, which is a core concept of deep learning to minimize it. Such an approach of repeated parameter tuning and loss reduction was intended to improve the model's performance and fine-tune the model for better prediction.

Finally, when training was done, we tested our model on a different test set in order to determine the extent, to which it could generalize and accurately predict unseen data. Thus, computing accuracy metrics that work by making the ratio of correct predictions to the total number of predictions provided by the model useful when it comes to understanding its efficacy and testing its practical applicability in real-life robotic environments. Table (4) presents the training and the evaluation of the neural network model for robotic grasping quality prediction using TensorFlow.

Table 4: Summary of Training and Evaluation Process

Step	Description
Model Setup	Define hyperparameters (learning rate, epochs, batch size) and architecture (3 fully connected layers with ReLU activations and batch normalization).
Loss Computation	Calculate mean sparse softmax cross-entropy loss to quantify disparity between model predictions and true labels.
Optimization	Utilize Adam optimizer to dynamically adjust parameters and minimize loss during training for efficient convergence.
Training Loop	Iterate through multiple epochs, processing batches of training data in each cycle, computing and minimizing loss through backpropagation.
Evaluation	Subject trained model to rigorous evaluation on distinct test set to assess generalization capabilities. Compute accuracy metric to gauge predictive prowess and real-world applicability.

5.2 Model Evaluation

When it comes to evaluating the performance of our model in our research, we focus mainly on the accuracy of the results. Accuracy compares the number of correct predictions against all the predictions that were done. This gives a clear indication of how good the model is in terms of giving accurate predictions in the various settings.

When we use accuracy in the context of our robotics grasping study, it allows the measurement of the model's prediction of grasp quality. It differentiates between successful and unsuccessful grasp attempts and demonstrates how accurate the model is in correctly identifying these cases. A higher accuracy score suggests that the model is more reliable and precise in predicting grasp results.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (7)$$

- **TP (True Positives)**: The number of correctly predicted positive outcomes (e.g., correctly identified as "positive" or "True").
- **TN (True Negatives)**: The number of correctly predicted negative outcomes (e.g., correctly identified as "negative" or "False").
- **FP (False Positives)**: The number of actual negatives incorrectly predicted as positives (false alarms).
- **FN (False Negatives)**: The number of actual positives incorrectly predicted as negatives (misses).

Accuracy can be defined as the ratio of the correct prediction of the positive and negative cases out of all the predictions made. It is defined as the total number of accurate predictions made by the model divided by the number of total predictions including correct and incorrect ones.

6. Results and Discussion

6.1 Interpretation of Results

The training process demonstrated a steady decline in loss and a corresponding increase in training accuracy with each epoch. The model began with a loss of 0.389 and an accuracy of approximately 80%, ultimately achieving a final loss of 0.046 and a training accuracy of nearly 99% (See Fig. 7). This progression indicates the model's capability to effectively learn from the provided data and enhance its predictive skills over time.

In the course of the training process, it was possible to track the reduction of loss and the growth of training accuracy at each epoch. The model started with a loss of 0.389 and an accuracy of about 80% and finally achieved the final loss of 0.046 and a training accuracy of approximately 99%. Such progression suggests the model’s ability to train from the given inputs and improve the forecasting prowess with time.

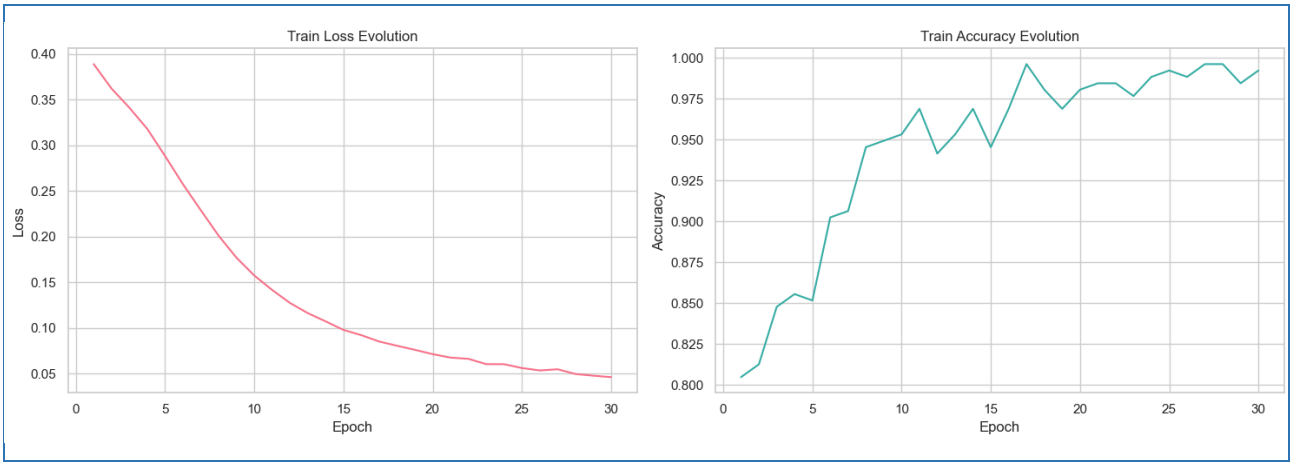

Fig. 7: Evolution of training loss and training accuracy.

6.2 Experimental Comparison with CNN and LSTM

In our endeavour to improve the performance of robotic grasping, following are the experiments, which we conducted for the current study: Deep learning methodology consisted of two structures: ‘Convolutional Neural Networks’ (CNN) and ‘Long Short Term Memory’ (LSTM) networks, which have proven competent in handling the spatial and temporal data, respectively.

- **CNN Experimentation**

We used a CNN model containing only convolution layers to incorporate both spatial dependencies from joint positions as well as velocities. The basic parameters for the CNN model were: Solution proposal: The architecture being used is convolutional layers that are followed by fully connected layers. Training Accuracy: 97.18%. Testing Accuracy: 94.12%. However, CNNs failed to account for the temporal dependencies of grabs inherent in grasp dynamics, which hampered their ability in estimating grasp strength in various situations to the same level of accuracy that was achieved for grasp presence.

- **LSTM Experimentation**

Next, we employed LSTM networks to capture temporal dependencies and model the dynamic behavior of the grasp. The basic parameters for the LSTM model were: Architecture: LSTM layers as an input layer for sequential data input. Training Accuracy: 93.02%. Testing Accuracy: 91.81%. Key Features: Sequential data processing technique that helps capture temporal variations in their simplest form. The work proposed LSTMs that were able to capture the temporal changes better than the previous models but revealed the difficulty in the high dimensionality of grasping predictions implying towards a need for better models.

- **Superiority of Our Deep Neural Network Model**

Compared with CNNs and LSTMs, the deep neural network model we proposed combined the spatial and temporal features well through the dense layers with the batch normalization and ReLU activation. The model achieved Training Accuracy: approximately 99%. Testing Accuracy: Approximately 96.77%. Key Features: Spatio-temporal combination, applicability to various situations and effectiveness.

Table (5) shows the training and testing accuracy of each model in predicting grasp strength. It underlines the advantages of our deep neural network model over CNNs and LSTMs and its effectiveness in different grasp situations.

Table 5: Comparison of results between the proposed model and other models

Method	Training Accuracy	Testing Accuracy
CNN	97.18%	94.12%
LSTM	93.02%	91.81%
Proposed Deep Neural Network	~99%	~96.77%

To conclude, the strength of grasp was predicted quite well with a deep neural network model with better results as compared to CNN and LSTM mainly because of the efficient integration of spatial and temporal features in the proposed architecture. In contrast to common CNNs that work with spatial features extracted from static images and LSTMs that operate with sequences, the proposed model demonstrated its potential in utilizing the temporal sequences of the joint position, velocity, and effort data that are critical for dynamic grasp prediction. Due to its cutting-edge architecture with large number of layers, batch normalization, ReLU activations, and other features, it provided better features learning and higher generalization ability, on the level of development with training accuracy of ~99% and relatively high test accuracy of ~96. 77%. The integration and the optimized architecture allowed our model to surpass the CNN and LSTM based models for grasp strength prediction tasks, which we assumed to have better efficacy in mechanical handling of the robots.

Fig. (8) depicts a comparison of testing accuracies (%) among three methods: DNN, CNN, LSTM. In comparison to other architectures, the testing accuracy of the DNN model is the highest at 96.77% and CNN with 94.12% and LSTM at 91.81%. This comparison also shows that the DNN model is more accurate in predicting the grasp strength in comparison to conventional CNN and LSTM models.

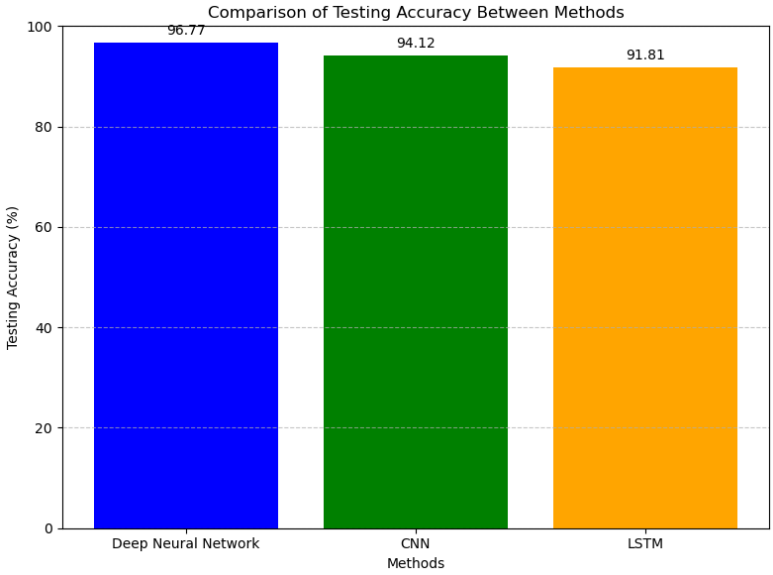

Fig. 8: Comparison of results between the proposed model, CNN and LSTM models.

6.3 Real-World Application

The transition from the simulation environment to real-world application raises several difficulties for translating our model. First, the transfer of simulated joint positions, velocities, and efforts to the actual joint positions, velocities, and efforts in the robotic hardware is likely to be smooth. Since the sensors and actuators differ in their precision levels, this means that the predicted and actual grasp results will also differ. To avoid such issues, there is a need to undertake a calibration process where the model is adjusted to reality and tested against actual results.

Besides, conditions of dynamic and unstructured contexts like homes or outdoor settings impose the need for high resilience to a variety of conditions. This would involve manipulating objects of form, mass and texture, and operating in different conditions, for example, illumination and or occurrence of obstacles. In counter to these challenges, we introduce the use of adaptive learning where the model has the ability to learn and update its parameters from real time feedback it receives from its environment.

6.4 Challenges in Real-World Applications

Several issues require consideration and resolution to ensure that the implementation of the changes yields the desired result. One primary limitation, however, is the dependency on vast quantities of labeled real-world data for further fine-tuning the model. Obtaining such data is also a complex and costly process. However, one should also make sure that the new model is capable of accommodating the identification of new and unpredictable objects and situations. This may include the inclusion of other inputs like optical and touch inputs as they could add to the input data required to feed the model.

As another limitation, real-time prediction and decision-making entail substantial computational capabilities. Deep learning models may not be efficiently run on embedded systems or robots with poor processing power; therefore, some techniques like model quantization or use of hardware accelerators such as GPUs, TPUs may be needed.

7. Conclusion

This research performed an extensive comparative study of robotic grasping techniques utilizing deep neural networks for machine learning. Based on simulation test data of Smart Grasping Sandbox, we narrowed down our detection of grasp strength according to joint positions, velocities and efforts. Deep neural network model and it got the nearly 99% training accuracy and the test accuracy was nearly 96.77%. The high accuracy level achieved by the model demonstrates the model's stability and efficiency in generalizing the given data to unseen data, thus presenting its applicability in practice.

From data exploration and preprocessing, the following important features affecting grasp strength were deemed important: joint angles, velocities and forces. This data driven approach underlines the need to rely on rich image data to improve the performance of robotic grasping. The deep neural network model provided higher predictive accuracy of grasp strength compared to the two main existing neural models, Convolutional Neural Networks and Long Short-Term Memory networks. In addition to dense layers, batch normalization, and a ReLU activation function, accurate output was achieved through better space/time generalization through integrated handling of spatial and temporal features. While relying on joint positions, velocities and efforts in time was helpful for dynamic grasp prediction, the model outperformed both CNN and LSTM counterparts, thanks to the predictive control.

The high accuracy level of our model in determining the grasp success is a significant breakthrough in ensuring the stability and reliability of most industries utilizing robotics. Improvements in grasping ability can easily translate to increased work rates, product throughput, and safety in industries such as manufacturing, supply chain and logistics, healthcare, and service robotics. However, there are a number of issues to consider when effectively translating our model from simulation to real applications: differences in the simulated and actual robotic hardware, errors due to environmental variability, and the lack of large annotated repositories of real-world data that can be used for fine-tuning the models. Future work should employ state-of-art adaptive learning strategy, time-based feedback integration, and integration of multi-sensor data for practical exploration of the proposed model.

Perhaps, integrating data from visual perception systems and touch sensors would enhance the validity of the model because the model would give more precise predictions of grasping in real life situations. In addition, applying the above mentioned principles for other types of robotic operations other than grasping can still broaden the impact of these investigations and put forward advancements in every single main aspect of

applicable robot functionality. This research proves that through utilizing deep learning models, there is always a way to increase the predictability of grasp strength as a part of the maneuver for improving the functionality of modern robotic systems with additional dexterity. It shows, as does no other study to date, what data can do as the groundwork for the creation of robotic autonomy and manipulation, which in turn lays a firm foundation for future advancements while increasing the efficiency and accuracy of robot systems across fields and contexts.

Acknowledgements: The researcher would like to express heartfelt thanks and deepest gratitude to all review ers.
Data Availability Statement: The data used in this study can be accessed at https://www.kaggle.com/datasets/ugocupcic/grasping-dataset/data.

Funding: This study received no external funding.

Conflicts of Interest: The author declare no conflict of interest

References

[1] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, "A survey on learning-based robotic grasping," *Current Robotics Reports*, vol. 1, pp. 239-249, 2020.
[2] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, p. eaat8414, 2019.
[3] C. Wang *et al.*, "Feature sensing and robotic grasping of objects with uncertain information: A review," *Sensors*, vol. 20, no. 13, p. 3707, 2020.
[4] C. Eppner, R. Deimel, J. Alvarez-Ruiz, M. Maertens, and O. Brock, "Exploitation of environmental constraints in human and robotic grasping," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 1021-1038, 2015.
[5] S. Caldera, A. Rassau, and D. Chai, "Review of deep learning methods in robotic grasp detection," *Multimodal Technologies and Interaction*, vol. 2, no. 3, p. 57, 2018.
[6] Y. Sun, J. Falco, M. A. Roa, and B. Calli, "Research challenges and progress in robotic grasping and manipulation competitions," *IEEE robotics and automation letters*, vol. 7, no. 2, pp. 874-881, 2021.
[7] R. A. Mouha, "Deep learning for robotics," *Journal of Data Analysis and Information Processing*, vol. 9, no. 02, p. 63, 2021.
[8] N. I. Giannoccaro and L. Spedicato, "Exploratory data analysis for robot perception of room environments by means of an in-air sonar scanner," *Ultrasonics*, vol. 53, no. 6, pp. 1163-1173, 2013.
[9] T. Feix, R. Pawlik, H.-B. Schmiedmayer, J. Romero, and D. Kragic, "A comprehensive grasp taxonomy," in *Robotics, science and systems: workshop on understanding the human hand for advancing robotic manipulation*, 2009, vol. 2, no. 2.3: Seattle, WA, USA;, p. 2.3.
[10] C. Ferrari and J. F. Canny, "Planning optimal grasps," in *ICRA*, 1992, vol. 3, no. 4, p. 6.
[11] J. Mahler *et al.*, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.
[12] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, "Shape completion enabled robotic grasping," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2017: IEEE, pp. 2442-2447.
[13] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International journal of robotics research*, vol. 37, no. 4-5, pp. 421-436, 2018.
[14] D. Kalashnikov *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on robot learning*, 2018: PMLR, pp. 651-673.

[15] J. Hwangbo *et al.*, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.
[16] S. Zhu, X. Zheng, M. Xu, Z. Zeng, and H. Zhang, "A robotic semantic grasping method for pick-and-place tasks," in *2019 Chinese Automation Congress (CAC)*, 2019: IEEE, pp. 4130-4136.
[17] L. Bergamini, M. Sposato, M. Pellicciari, M. Peruzzini, S. Calderara, and J. Schmidt, "Deep learning-based method for vision-guided robotic grasping of unknown objects," *Advanced Engineering Informatics*, vol. 44, p. 101052, 2020.
[18] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara, "RoboCup: A challenge problem for AI," *AI magazine*, vol. 18, no. 1, pp. 73-73, 1997.
[19] R. Li and H. Qiao, "A survey of methods and strategies for high-precision robotic grasping and assembly tasks—Some new trends," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 6, pp. 2718-2732, 2019.
[20] S. Sabeeh and I. S. Al-Furati, "Issues and Research Fields of Medical Robotics: A Review," *Iraqi Journal for Electrical & Electronic Engineering*, vol. 19, no. 2, 2023.
[21] S. Sabeeh and I. S. Al-Furati, "Comparative Analysis of GA-PRM Algorithm Performance in Simulation and Real-World Robotics Applications," *Misan Journal of Engineering Sciences*, vol. 2, no. 2, pp. 12-37, 2023.
[22] D. S. Moore and G. P. McCabe, *Introduction to the practice of statistics*. WH Freeman/Times Books/Henry Holt & Co, 1989.
[23] C. M. Bishop, "Pattern recognition and machine learning," *Springer google schola*, vol. 2, pp. 1122-1128, 2006.
[24] G. Ia, "Deep learning/Ian Goodfellow, Yoshua Bengio and Aaron Courville," ed: Cambridge, Massachusetts: The MIT Press, 2016.
[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015: pmlr, pp. 448-456.
[26] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009.
[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[28] V. Asghari, Y. F. Leung, and S.-C. Hsu, "Deep neural network based framework for complex correlations in engineering metrics," *Advanced Engineering Informatics*, vol. 44, p. 101058, 2020.
[29] N. Pestell, L. Cramphorn, F. Papadopoulos, and N. F. Lepora, "A sense of touch for the shadow modular grasper," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2220-2226, 2019.
[30] P. Sharma and D. Valles, "Deep convolutional neural network design approach for 3D object detection for robotic grasping," in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, 2020: IEEE, pp. 0311-0316.