Building And Evaluating A Skin-Like Sensor For Social Touch Gesture Classification

by

Tejas Umesh

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

ARIZONA STATE UNIVERSITY

May 2024

ABSTRACT

Socially assistive robots (SARs) can act as assistants and caregivers, interacting and communicating with people through touch gestures. There has been ongoing research on using them as companion robots for children with autism as therapy assistants and playmates. Building touch-perception systems for social robots has been a challenge. The sensors must be designed to ensure comfortable and natural user interaction while recording high-quality data. The sensor must be able to detect touch gestures. Accurate touch gesture classification is challenging as different users perform the same touch gesture in their own unique way. This study aims to build and evaluate a skin-like sensor by replicating a recent paper introducing a novel silicone-based sensor design. A touch gesture classification is performed using deep-learning models to classify touch gestures accurately. This study focuses on 8 gestures: Fistbump, Hitting, Holding, Poking, Squeezing, Stroking, Tapping, and Tickling. They were chosen based on previous research where specialists determined which gestures were essential to detect while interacting with children with autism. In this work, a user study data collection was conducted with 20 adult subjects, using the skin-like sensor to record gesture data and a load cell underneath to record the force. Three different types of input were used for the touch gesture classification: skin-like sensor & load cell data, only skin-like sensor data, and only load cell data. A Convolutional Neural Network-Long Short Term Memory (CNN-LSTM) neural network architecture was developed for inputs with skin-like sensor data, and an LSTM network for only load cell data. This work achieved an average accuracy of 94% with skin-like sensor & load cell data, 95% for only skin-like sensor data, and 45% for only load cell data after a stratified 10-fold validation. This work also performed subject-dependent splitting and achieved accuracies of 69% skin-like sensor & load cell data, 66% for only skin-like sensor data, and 31% for only load cell data, respectively.

ACKNOWLEDGMENTS

I would like to sincerely thank my advisor Dr Hasti Seifi for her guidance, support, and mentorship throughout this project. I would also like to thank Mr. Yatiraj Shetty for his extensive support in designing the sensor and for his feedback and inputs during this project. I would also like to thank my committee members, Dr. Pooyan Fazli and Dr. Nakul Gopalan for making this thesis a reality. I would like to thank my family and friends for their support throughout the project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Socially Assistive Robots (SARs) have gained in popularity in the field of Human-Robot Interaction. They represent a class of robots that is at the intersection of socially interactive robotics and assistive robots. The former interacts with the user through social interactions and the latter provides assistance to the user [1]. SARs have continued to grow in different sectors. The Haptic Creature Project developed a furry-like robot companion people can pet [2]. The robot Paro was developed as a robot therapy technique that was used at hospitals and facilities for elderly people as a new method for mental healthcare [3]. Inspired by animal therapy, the Huggable robot consists of a full-body skin-like material with temperature, force, and electric field sensors. Animal therapy has been shown to lower stress, reduce heart rate, and elevate mood, which improves the overall health of the subject [4],[5].

SARs have also been shown to be useful as assistants and caretakers. As assistants to the growing elderly population, SARs can be beneficial by integrating themselves into the healthcare system. Along with physical support, SARs can be beneficial in filling in for more complex types of support like social, emotional, and cognitive support [6]. Apart from the elderly, people with cognitive disorders like autism can greatly benefit from SARs. Research has shown that a long-term positive outcome is possible if people with autism are given extensive support and care early in their lives [7]. Prior work shows that robots have been used in pet therapy [8], as well as partners for children to learn from and interact with as they would with other children [9]. Recent works [10, 11] have shown that SARs can help children with autism as social mediators and therapy assistants. A key aspect of SARs in the context of children is that they respond better to robots that are child-sized and have exaggerated features. As part of this, the crux of this work focuses on sensors that are close to skin-like which can, in the future, be attached to SARs to make interactions more natural for the user.

When working as caretakers or assistants, natural interactions with subjects are crucial, and touch is an important factor in achieving this goal for SARs. Touch has been shown to play a key role in human interaction [12], and more specifically it has been a key component in communicating emotions and social messages [13]. Previous studies [14] have shown that touch can amplify stronger social emotions like trust and affection. Building touch-perception systems for SARs has some challenges. The main priority is to ensure that the sensors on the robots are as natural as possible, enabling natural interactions. These sensors need to be designed with the goal of recording high-quality readings during gesture performances. Furthermore, it is essential that the SAR can effectively identify the gesture being performed to interact with the subject freely.

This work focuses on developing a touch-perception system that can detect touch gestures for SARs. We build and evaluate a skin-like sensor and perform touch-gesture classification with the recorded data. Children with autism can benefit from SARs as therapy assistants and playmates. Our work can advance research in this field and serve as key sensors for SARs that are tasked with continuous interactions with their subjects.

In this thesis, we replicate a skin-like capacitive sensor and augment it with force sensing, to evaluate its performance in detecting social touch gestures. A notable development in skin-like sensors comes from a recent work that designs a novel human-like artificial sensor for robots [15]. This work makes the case for developing sensors where the comfort and likeliness of human skin is given equal importance as the other technical design characteristics of the sensor. Achieving skin-like sensors is vital in making robots more interactive and user-friendly [16], which is especially important in SARs. This thesis uses the methodologies in [15] to build a skin-like touch sensor as described in the paper. We also follow the methodology described in this paper to record data from a skin-like sensor and load cell underneath. The skin-like sensor can detect the spatiotemporal patterns of touch but it cannot measure the amount of force. The load cell, attached below the sensor, is used to measure the force when a user performs gestures on the skin-like sensor.

We ran a user study to record data from 20 subjects, using the skin-like sensor and a load cell. We focused on 8 gestures in our study– Fistbump, Hitting, Holding, Poking, Squeezing, Stroking, Tapping, and Tickling. These gestures were selected as they were the most commonly used gestures, as well as cover the space of different kinds of gestures, based on prior research [17]. Furthermore, we consulted a previous work [11] that identified the gestures needed for the SAR to detect while interacting with children with autism. There has been prior work in touch gesture classification using statistical machine learning methods, but little work has been done using deep learning techniques. To the best of our knowledge, there has not been any work where deep learning methods have been used for touch gesture classification from skin-like sensors. This work uses CNN-LSTM model architecture to perform touch gesture classification on the data collected from the skin-like sensor we developed, essentially using CNN to capture spatial features and LSTM for the temporal aspect. We implement the model on three different types of inputs: skin-like sensor & load cell data, only skin-like sensor data, and only load cell data, to observe how well the model can interpret the data and perform the touch gesture classification. We achieve an accuracy of 94% for skin-like sensor & load cell data, 95% for only skin-like sensor data, and 45% for only load cell data after a stratified 10-fold validation for each case, respectively. We also followed the leave one subject out (LOSO) scheme, wherein the splitting is subject-dependent, keeping one subject out as the test set and training the model on the remaining data. Owing to the variability in which users perform the same gesture, there is an overall decrease in accuracy results. We achieve an accuracy of 69% for skin-like sensor & load cell data, 66% for only skin-like sensor data, and 31% for only load cell data.

The thesis is organized as follows: in Chapter 2, we detail the related work in this field and the research that inspired this work. In Chapter 3, we detail the experimental setup of our thesis, including how we developed the skin-like sensors and set up the recording process to collect data from human subjects. Chapter 4 details the model architecture and the neural networks used in our model. We also detail the parameters in the model that were tuned during model development. Chapter 5 discusses data processing once it is collected from all the subjects. We detail how the data is pre-processed and made ready to input into the model. Finally, Chapter 6 discusses the results achieved and future steps one can take.

Chapter 2

RELATED WORK

2.1 Socially Assistive Robots

The first industrial robot was invented in 1954. Since then, there have been many developments in robotics and related technologies. In this work, we focus on the fifth generation of robots: Humanoid Robots. The first four generations were: Prototypes of Robotics, Robotics Arms, Walking Robots, and Behavior-Based Robots [18].

Robots have played a vital role in healthcare for decades and have primarily been created for personal support with daily living routines and physical training in rehabilitation [19]. Socially Assistive Robots, as part of the 5th generation of robots, make an effort to give the right emotional, cognitive, and social cues to support a person's growth, education, or rehabilitation [6]. These robots have served as caregivers, companions, and assistants [10]. They are made to take advantage of social and affective qualities in order to maintain motivation, boost engagement, and make coaching, monitoring, instruction, and communication easier [20]. Therefore, many socially assistive robots have been developed for various health care needs, including autism therapy [21], physical rehabilitation [22], and weight management [23].

Socially Assistive Robots like the Haptic Empathetic Robot Animal (HERA) has been shown to assist children with autism by being therapy assistants and social mediators, and help them develop emotion recognition and expression [24]. It has been shown that children with autism have a greater preference for robots with features that stand out and are child-sized [25, 26]. Another important caveat is that the robot should be controllable by the child and be predictable in its behavior [27].

Since tactile exploration and physical contact are crucial for a child's development, enhancing a robot's touch-detection abilities can significantly boost its potential for interaction and help children with autism [28].

A previous work [29, 30] shows how a child-like robot, KASPAR, uses force-sensing resistors (FSRs) as part of touch-sensing autism therapy in children. Another robot, named Keepon [31], was used in a 3-year study where children with autism interacted with the robot through various touch gestures. The conclusion of the study was that after interacting with the robot over a period of time, the children became friendlier and more comfortable with the robot.

In this thesis, our work in building and evaluating skin-like sensors could potentially improve the development of SARs that are easier to interact with and improve their functionality. SARs focusing on assisting children with autism place importance on comfortable interactions and accurate touch-detection abilities. Our work in touch-perception systems with skin-like sensors could greatly benefit them.

2.2 Touch Sensing

As touch is the main nonverbal means by which we express our most intimate emotions, it is vital to both our physical and mental health [32]. Our emotional well-being and social bonding are an essential part of life and are promoted by touch [33, 34].

Touch sensing has been an active area of research. Prior work shows that there have been different approaches to touch-sensing for tactile perception robots.

As mentioned previously, Kaspar [35] employs force-resistive sensors which are simple to implement and cost-efficient. Resistive sensors have also been used in the NAO Robot [10] where these sensors are proven effective in detecting social interaction. Resistive sensors have a drawback in that they provide only one-dimensional data output when a gesture is made. This makes it difficult to interpret the different positions of touch gestures and to identify a visual pattern of the performed gesture. Vision-based sensors like Gelsight [36] use soft sensor surfaces and high-resolution sensing of geometry for tactile perception. GelSight essentially measures a high spatial resolution geometry using a soft elastomer contact surface. The exact shape, tension, and contact force are inferred from sensor deformation. However, one major drawback is that the thickness of the elastomer limits the force sensing range and the sensing starts to saturate when the elastomer can no longer bend. If the Gelsight is designed with thinner elastomers, this becomes particularly noticeable. Furthermore, if the sensor happens to undergo uniform deformation, then the force cannot be measured [37].

We use capacitive sensors in this work. Capacitive sensing is among the most widely used sensors [38] and its history, in the field of HCI, began in 1973 when CERN built a capacitive touch screen [39], followed by the first introduction of a multi-touch capacitive tablet in 1985 [40]. Over the past 20 years, the influence of capacitive sensing has increased dramatically due to the widespread usage of capacitive touch pads, screens, and other interactive devices on desktop, mobile, and wearable computers [41]. Capacitive sensors work on the principle of identifying touch by measuring the local change in capacitance on a grid array of electrodes. It enables high-resolution touch sensing and high-resolution sensing of multiple simultaneous touch contacts with low latency [42, 43, 41].

Another advantage of using capacitive sensors is the recent development of easy-to-build sensor designs. Multi-touch capacitive sensors previously required expensive development tools and were previously limited to companies and labs that could afford them, along with people with significant expertise in this area. However, recent work like [38, 15] have shown that it is possible to develop such sensors in a "do-it-yourself" manner with high-resolution touch sensing.

While there have been artificial skin fabrications proposed in the past [16, 44] there are very few research works that focus on creating sensors that are close to the properties of human skin. Previous works like [45, 46] focus on building cushioning layers and skin-like softness with elastic material. Researchers have experimented with robots using elastic materials on their fingers [47, 48] and a more recent work creates an elastic pad model that is sufficiently simple for analyses and real-time simulation, while also exhibiting behavior that is quite similar to interacting with real human skin [49, 50].

Recent work in skin-like sensor development [15] points out that sensor technology has prioritized functionality over comfort and human-like qualities, which are seen as an additional layer. The paper emphasizes that comfort should be prioritized equally with technical features in design, and achieving a sensor that is as human-like as possible helps achieve that goal. Natural skin-like sensors offer better sensor integration, increased robustness, and performance. Silicon-based materials have been used in recent works to develop this human-like feeling [51, 52]. While the previous work proposes a skin-like sensor, its performance in detecting touch gestures is still unknown. Our work aims to fill the gap by evaluating this skin-like sensor for touch gesture classification.

2.3 Gesture Classification

Touch gestures refer to intentional physical contact with a meaningful intention. A research paper [53] defined a commonly used touch dictionary of 30 gestures based on human interactions. A recent work [17] clustered these touch gestures into 9 clusters based on the semantic relationship between them. According to the study, people judge how similar touch gestures are based on their social and emotional meanings. The clusters are: social, romantic affection, caregiving affection, hand contact, aggression, forceful press, functional movement, nervous contact, and contact without movement.

In their study on touch-gesture classification for their NAO robot, [10] identified five gestures that children with autism deal with regularly, based on feedback from specialists. The data was collected through a user study with 15 participants. Using Random Forest, the collected gestures were classified with an average accuracy of 74.3%, which is approximately four times higher than chance.

In 2014, a paper [54] introduced the Corpus of Social Touch (CoST) dataset that consisted of 14 different touch gestures. These gestures were taken from the touch dictionary defined by [53], and were selected for interactions with a mannequin arm. The data was collected from 31 subjects with a total of 5,203 gesture captures. The work also performed touch gesture classification, wherein they used Bayesian and SVM (Support Vector Machine) classifiers to get accuracies of 57% and 60% respectively, which was 7 times higher than chance. The Human-Animal Affective Robot Touch (HAART) dataset [55] identified 7 gestures, out of the touch dictionary defined in [53], that were most commonly found in human-animal interactions. The data was collected from 10 subjects with a total of 829 gesture captures. A recent work [56], used CoST and HAART datasets to explore three deep neural network architectures: CNNs, CNN-RNNs, and Autoencoder-RNNs. The CNN resulted in a 42% accuracy for CoST and 56% for HAART. The CNN-RNN resulted in a 52% accuracy for CoST and 61% for HAART. The Autoencoder-RNN resulted in a 33% accuracy for CoST and 55% for HAART.

To the best of our knowledge, the most recent work using deep learning for touch classification is [57], where they use a CNN model while treating each touch gesture data point as a 3D image. Their dataset consists of 13 touch gestures, which were chosen from the touch dictionary in [53], based on whether the gesture was applicable to upper arm interactions and had no overlapping definitions. The paper got an accuracy of 66%, which is 8.6 times more than chance. This work also explores how considering shear force, along with normal force for the gestures, results in a higher accuracy of 74% which is 9.6 times more than chance.

To the best of our knowledge, there is no previous work in deep learning for touch-gesture classification apart from the two papers referenced above. To the best of our knowledge, there is also no previous work on touch-gesture classification using deep-learning methods for skin-like sensors. In this work, our goal is to establish touch-gesture classification using deep learning methods on a skin-like sensor. We believe that the skin-like sensor can detect the gestures accurately as it is more natural and receptive to touch-gesture data, closely mimicking human-to-human touch interactions.

Chapter 3

EXPERIMENTAL SETUP

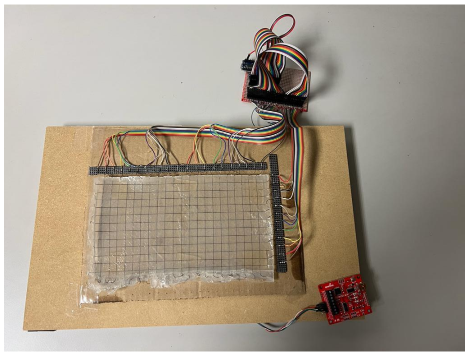
3.1 Setup

In this work, our goal is to build and evaluate skin-like sensors. We replicated the fabrication process described in [15]. This novel approach to designing skin-like sensors involved open-source tools that can be procured and designed to record touch gestures. When gestures are performed, the sensor outputs a sequence of images which can be processed by deep learning methods to extract relevant features for touch-gesture classification. We closely followed the fabrication process by using conductive yarn as electrodes instead of elastomers. This is because the electrical resistance of elastomers is frequently too high, which can impact touch detection. The electrodes are sandwiched between two layers of EcoFlex Gel, which is known to depict human fat-like properties [58] while being soft and flexible.

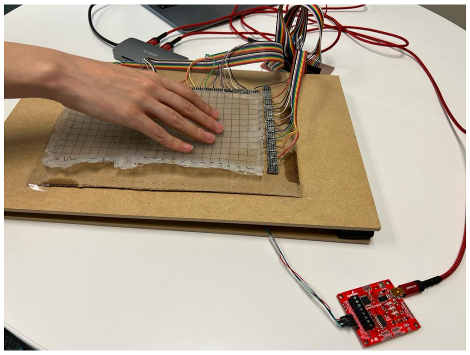
The electrodes are connected to a Mutual Capacitance (Muca) sensing development board with Arduino Nano to output images. As our work involves recording social touch gestures, the skin-like sensor must cover the entire palm. The Muca board has 21 transmission ports and 12 receiving ports. To ensure that we were building a sensor that covered the entire palm, we conducted trial-and-error experiments for its design. We conducted experiments for sensor design with 4mm and 8mm electrode spacing and determined that a 10mm spacing provided sufficient surface area. The sensors with smaller spacing gave better resolution of images when gestures were performed on them. However, we had to strike a balance between the resolution of images and hardware constraints.

Building the skin-like sensor came with several challenges. We had to manually attach the strings in a grid pattern as electrodes while maintaining their spacing. This was time-consuming, especially if a thread broke during setup and needed to be replaced. It was also key to ensure that the strings were placed in an orderly manner and with tightness intact. If the strings are held too tightly, there is a high possibility that they will curl up when trimmed. Conversely, if they are held too loosely, they will not remain in the grid pattern even with the Ecoflex sandwich holding them together.

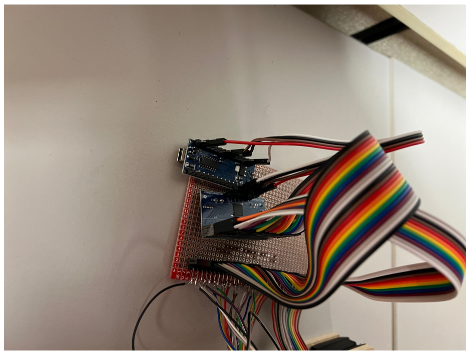
Figure 3.1 depicts the setup of our skin-like sensor, connected to the Muca board (the same board used in [15]) to read data. The Muca board, with Arduino Nano, is connected to the computer, through which we record and output the skin-like sensor data during the recording process. Below the skin-like sensor, we have a load cell that is connected to the SparkFun OpenScale Arduino [59], through which the weight values are recorded. Figure 3.2 shows how the load cell is attached. The load cell is placed directly beneath the sensor to measure the force applied by the user while performing gestures on the sensor. The skin-like sensor data is read as a sequence of frames with dimensions of $12X21$, signifying the 12 columns and 21 rows of the electrodes on the sensor. The load cell outputs weight values in kilograms, which increase with applied force. The average sampling rate across the subjects recorded was 9.7Hz for the skin-like sensor data and 15.7Hz for the load cell data. To ensure that the load cell accurately reads the weight, we calibrate it by setting a known weight and entering the value in the Arduino IDE. This way, the load cell knows the exact weight of the object being applied and is calibrated. We further place a varying number of coins as they are of standardized weights and set the known weight each time. Once a known weight was given as input, we tested the load cell with 10 different weights to confirm if it read each weight accurately.



(a) Skin-Like Sensor Setup



(b) Gesture On Skin-Like Sensor

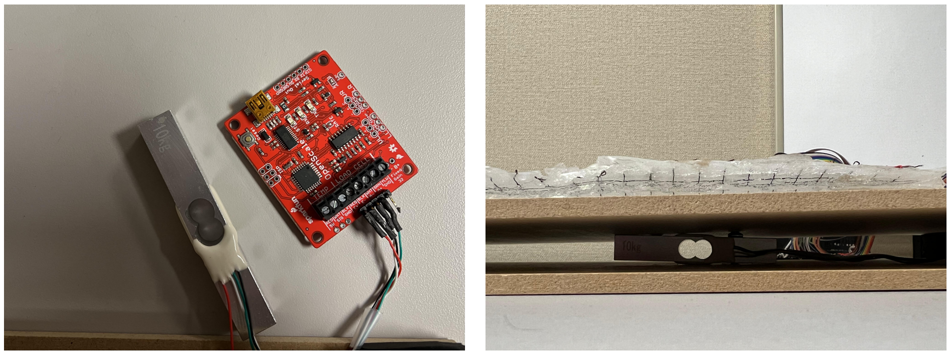


(c) The Muca Board

Figure 3.1: Setup Of Our Skin-Like Sensor

The computer records the skin-like sensor and load cell data into two different CSV files. The timestamps are also recorded for each start and stop cue, which is then later used to splice frames for the corresponding gesture. Python was used to code the setup and record the data. The skin-like sensor data and the load cell were recorded simultaneously using multi-processing in Python [60]. We use the PySerial library to read the data from the load cell through the OpenScale Arduino and the skin-like sensor data from the Muca board. We further use OpenCV to display and save the skin-like sensor data that are being read as frames. The load cell data is saved as a list of weight values in kilograms.

Figure 3.3 gives an example of how the data from someone performing the "holding" gesture is recorded on the skin-like sensor and load cell.



(a) Load Cell To OpenScale (b) Load Cell Placement

Figure 3.2: Setup Of Load Cell

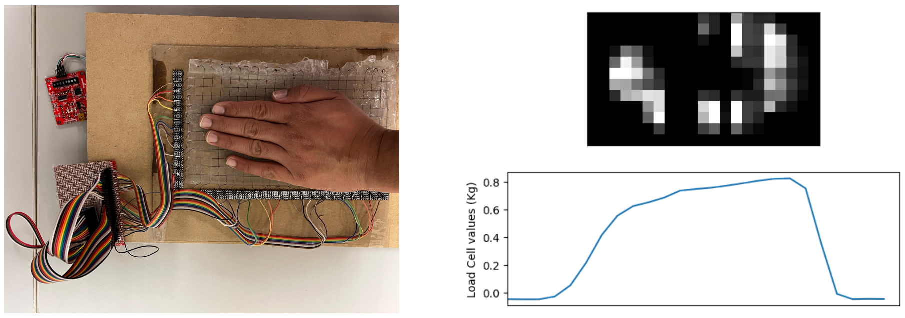


Figure 3.3: Holding Gesture On Skin-Like Sensor

We selected 8 touch gestures based on a recent work [17] which analyses the relationship between different touch gestures and how people perceive them. This paper groups the gestures into different clusters based on how the users group them by social and emotional features. We also took into account the recommended gestures, by specialists, for touch-perceiving robots to perceive while interacting with children with autism [11]. Table 3.1 shows the clusters from which we selected the 8 gestures used for this work.

Cluster	**Gesture**
Social	Fistbump
Romantic Affection	Stroking, Tickling
Caregiving Affection	Holding
Hand Contact	Tapping
Aggression	Poking, Hitting
Forceful Press	Squeezing

Table 3.1: The 8 Gestures Used In This Study

3.2 Data Collection User Study

After building the sensor, we ran a user study to collect data to evaluate the sensor and used the data to perform the touch gesture classification. The recording process involved 20 subjects, who were students of Arizona State University. There were 10 male and 10 female subjects in the age group of 19-35.

To be eligible, participants had to meet the following criteria:

1. An adult of at least 18 years of age
2. Have normal or corrected to normal vision and hearing
3. Have no sensory impairment in their hands
4. Have English proficiency at B2 level

We needed the participants to have normal or corrected to normal hearing and vision, along with no sensory impairment in their hands, to ensure that they could accurately perceive visual and audio feedback from our system and perform the touch gestures. Furthermore, as the instructions and cues to the participant during the recording were in English, we had to ensure that the participant could read and understand the detailed explanation for the gestures. Before starting the recording, the subjects had to complete an initial questionnaire wherein they filled out demographic information like age, gender, and whether they had previous experience with touch sensors. The user study was approved by Arizona State University's Institutional Review Board (IRB).

The total recording time for each subject in the user study was 45 minutes and received a compensation of 10$ for their efforts. The user study consisted of the following phases:

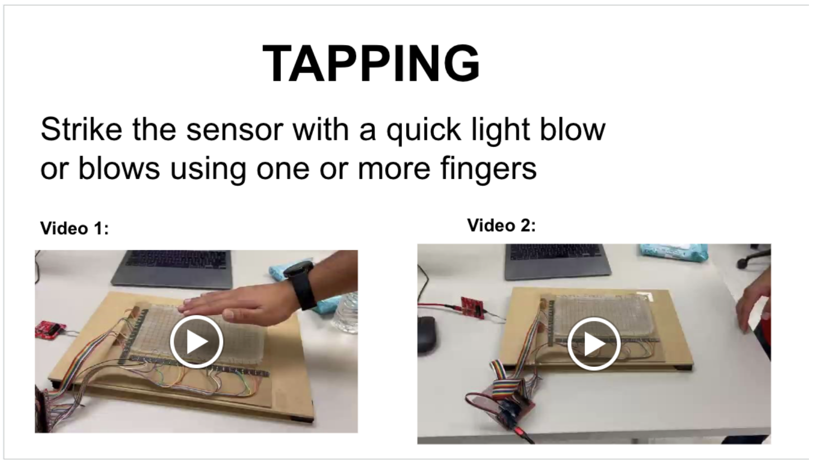
1. The first 5 minutes are given to the participant to complete the initial questionnaire.

2. The next 10 minutes are dedicated to showing the participant how to perform all the gestures, using text descriptions, and accompanying videos for each gesture performance. The participants are given time to practice the gestures during this training period, and the main recording starts only when they are confident that they are ready. Figure 3.4 shows an example of a slide from the PowerPoint presentation shown to the participant.

3. The next 25 minutes is the actual recording process. Upon the cue, the participant performs the gesture instructed. For each participant, the order in which they performed the gestures was randomized. The user receives both verbal and visual cues for the gesture being performed. Figure 3.5 shows what the subject sees on the display while performing the experiment. A start and stop timer is triggered when the participant initiates the gesture at the sight of the word "start" on the screen, along with the verbal cue for "start". The timer continues until the stop signal is called. 6 seconds are allotted for each gesture iteration. We ask the participant to repeat this 8 times for each gesture. For

the first 4 iterations, the participant performs exactly 3 times in each 6-second duration. Next, for each iteration in the next 4 times, the participant is allowed to perform the gesture as many times as they can within each 6-second duration. For every gesture, the skin-like sensor and load cell will record the user's gesture movement and force.

4. The last 5 minutes is for wrap-up where any questions the participant has would be answered, and the participant's information is collected for compensation.



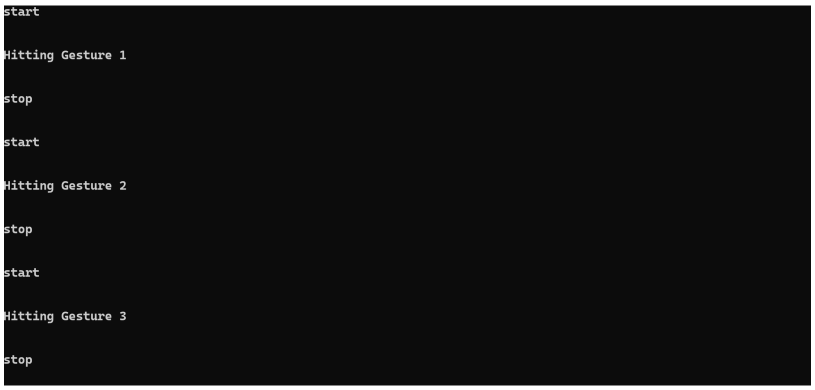
TAPPING

Strike the sensor with a quick light blow or blows using one or more fingers

Video 1:

Video 2:

Figure 3.4: Example Of A Slide Shown To Subject During Training Period



```
start

Hitting Gesture 1

stop

start

Hitting Gesture 2

stop

start

Hitting Gesture 3

stop
```

Figure 3.5: Subject's Screen During Recording Process

Chapter 4

DATA PRE-PROCESSING

In this section, we describe how the skin-like sensor and load cell data were retrieved and pre-processed after the user study was complete. Once the user study was complete, the data was stored in CSV files in their respective folder. Each gesture performed had its own folder with CSV files, identified by the user participation number. For example, the parent folder for the subject with participation number 05 was "Study_05". Figure 4.1 gives a visual description of how the data was saved.

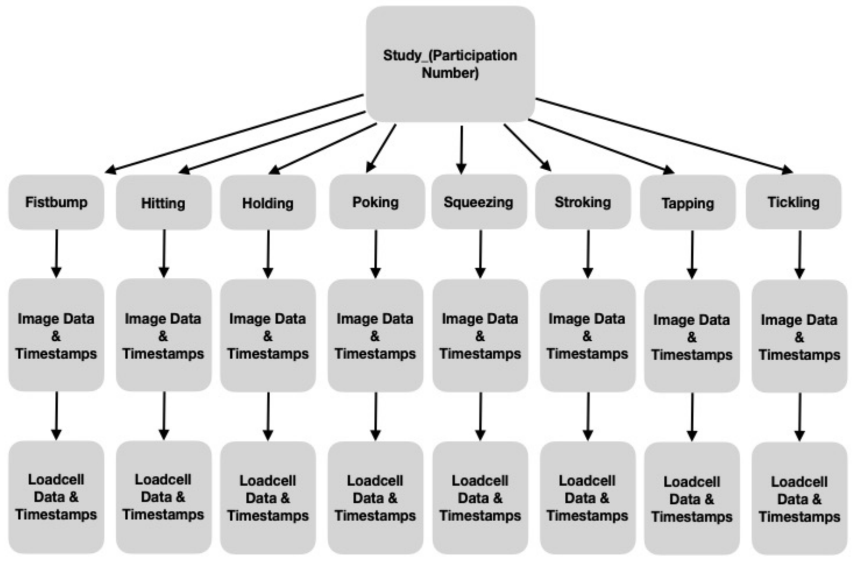


Figure 4.1: Folder Structure For Data Storage Across Subjects

We iterated through each gesture for each subject and got the timestamps as a list of tuples. Next, we iterated through the tuple list, spliced the skin-like sensor data accordingly, and saved it with the corresponding gesture label. Consider the gesture

"Tapping" performed by "User_05". During data collection, there were 8 iterations for each gesture, each lasting 6 seconds. Therefore, 8 timestamps were saved for each gesture, corresponding to the recorded iteration. The code extracts the frames, based on the beginning and end of each timestamp tuple, and assigns the label "Tapping". This is repeated for all subjects and their corresponding gestures. Similarly, the same was performed for the load cell data. Due to slight variations in the sampling rate during recording, the number of frames for the gestures was not exactly the same for all subjects. To ensure consistent input size for the model, we first calculated the average length of all gesture iterations for each subject. We padded zero matrices for gestures shorter than the average length to make them the same length as the average. We removed frames exceeding the average length from longer-than-average gestures. Similarly, the same process was performed for the load cell data. Furthermore, the load cell data was smoothened by performing a moving average windowing.

Each input data point corresponds to a labeled gesture. For touch-gesture classification, neural network algorithms cannot process text-based data and require numeric data [61]. Therefore, we transformed the labeled gestures (strings) into integers. One-hot encoding is a popular method for making this conversion. One-hot encoding transformed the output list of gestures to a list of vectors, each of length $1X8$, where one element is set to 1 and all other elements are set to 0 [61]. For example, for the gesture "Holding", the output datapoint for the model will be a vector with "Holding" set to 1 and all other gestures set to 0.

For the first 4 iterations, the subject performed the gesture exactly 3 times within 6 seconds for each one. This results in 3 gestures performed for 2 seconds each. We splice up each iteration into 3 equal parts. For the next 4 iterations, the subject naturally performed the gesture as many times as possible within 6 seconds for each one. We also splice each iteration into 3 parts. This ensures that we get naturally performed gestures within the 2-second window. This splicing is repeated for the load cell data. Figure 4.2 shows a flowchart that summarises this process. The dataset contains 3840 data points, representing all 20 subjects. Figure 4.3 shows what the raw sensor and load cell data look like for each gesture.

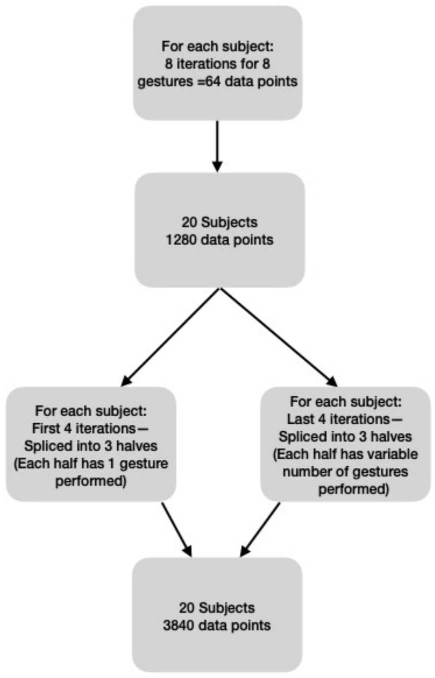
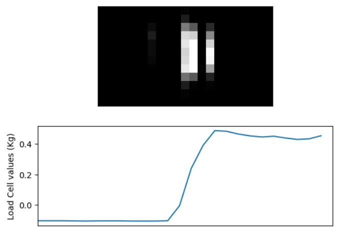
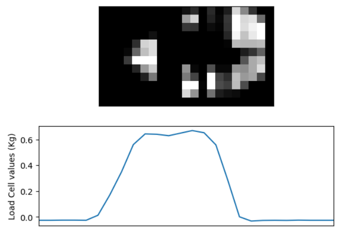


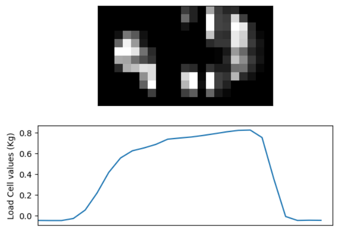
Figure 4.2: Flowchart Of Data Retrieval



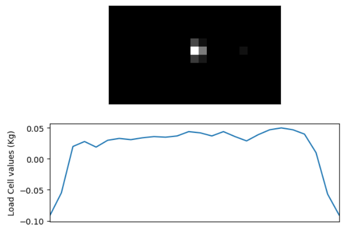
(a) Fistbump



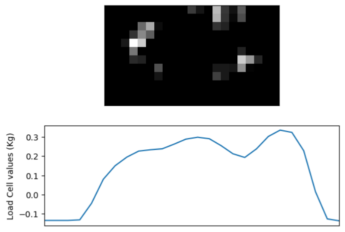
(b) Hitting



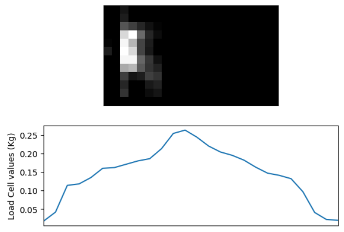
(c) Holding



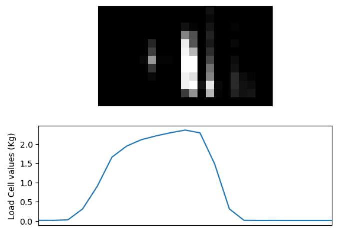
(d) Poking



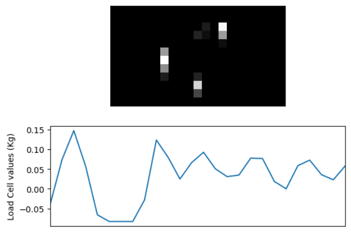
(e) Squeezing



(f) Stroking



(g) Tapping



(h) Tickling

Figure 4.3: Output Of Skin-Like Sensor And Load Cell For Each Gesture

Chapter 5

MODEL ARCHITECTURE

In our work, when a gesture is performed on the skin-like sensor, it is recorded as a series of frames. The load cell data is recorded as a continuous array of values that capture the force of the performed gesture. It is essential to develop a model that captures both spatial and temporal information from the load cell and sensor data to classify touch gestures.

Convolutional Neural Networks (CNNs) have been widely used to interpret spatial information and are known to be very successful in image recognition and classification problems. Advancements in CNN models [62, 63, 64, 65] have been crucial in achieving accurate results in these fields.

Long Short-Term Memory (LSTM) networks have been widely known to be successful in tasks involving temporal information, such as machine translation [66], generating natural language captions for images [67], and video classification [68].

Combining the strengths of both, CNNs and LSTMs, can be beneficial when the model needs to learn both spatial and temporal information from the input data. This model architecture is called the CNN-LSTM and it has proven to be an effective solution. Previous works have shown that CNN-LSTMs can be used to achieve high accuracy–facial expression recognition [69], lung ultrasound video classification [70], and action recognition in video sequences [71].

5.1 Convolutional Neural Network (CNN)

The CNN model was first introduced and developed by [72, 73] wherein they created the LeNet-5 multi-layer artificial neural network, which was capable of classifying handwritten numbers. This model could provide accurate image representation with very little preprocessing [74]. When it comes to interpreting images, CNNs are a big improvement over traditional ANNs for spatial data. ANNs consist of inter-connected nodes that are structured as input layers, then hidden layers, and finally output. The ANN receives input data which is then processed through the hidden layers of the network. The hidden layers then make decisions and learn to make accurate decisions for the output [75].

CNNs can extract features from images more efficiently than ANNs due to the use of convolutions, resulting in fewer parameters [76]. In fact, one of the biggest disadvantages of ANNs when it comes to dealing with image data is the computational complexity owing to the large number of parameters required [75]. The main feature of CNNs is the use of convolution filters. The convolution filter can be thought of as a sliding window that moves across the image and extracts features from the input data [77]. Each CNN layer consists of hyper-parameters that must be fine-tuned to get the best results. The parameters are as follows [77]:

5.1.1 Kernel Size

The kernel size in convolution layers refers to defining the size of the NXN matrix which acts as the convolution filter. The kernel matrix is multiplied with the corresponding values in the input matrix. As the matrix is a sliding window, it slides to the next set of values and so on. Figure 5.1 visually depicts how a kernel interacts with the input image. The input given to this hyper-parameter is an integer, or a tuple representing the dimensions of the kernel matrix.

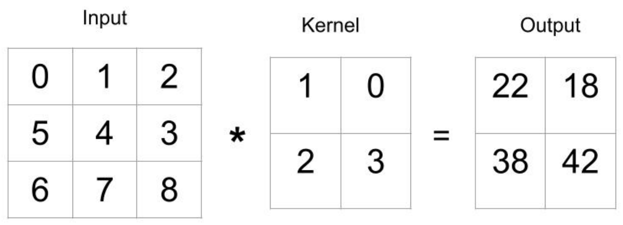


Figure 5.1: Example Of Kernel Filter On Input Matrix

5.1.2 Stride

As mentioned previously, the convolution filter, with the defined kernel size, is a sliding window through the input matrix. As the sliding window moves, it inevitably leads to overlap. By defining the stride hyper-parameter, we can control the amount of overlap. Figure 5.2 shows how a filter with a stride value of 1 and 2 would operate, respectively. As we can see in the figure, there's a $5X5$ matrix with a filter size of $3X3$. With a stride of 1, the output will result in a $3X3$ matrix. With a stride of 2, the output will be a $2X2$ matrix.

5.1.3 Padding

After the convolution process, the output matrix becomes progressively smaller due to filtering. There's a possibility that the information on the image borders might be lost as the filter slides through the matrix. This loss in information adds up through multiple convolution layers in the model.

Padding helps solve this issue by adding zero pixels around the border of the image, which causes the dimensions of the input matrix to increase. However, after filtering, the output size remains the same as the input matrix thus resolving the

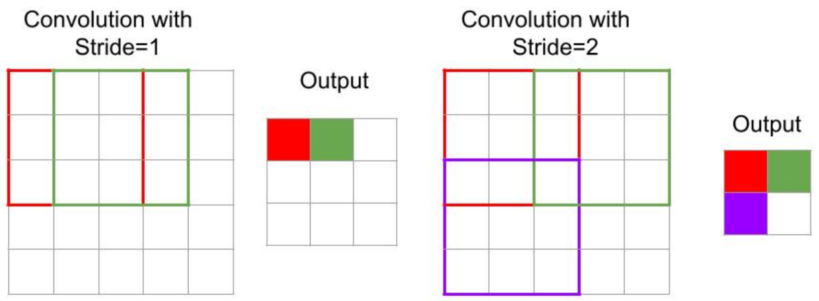


Figure 5.2: Visual Representation Of Stride=1 & Stride=2 On Input Matrix

issue of information loss across the border. As the number of convolution layers can be increased without a constant decreasing output size, we can use any number of layers in our model [78]. Figure 5.3 how the output size remains the same as the input matrix with a padding layer.

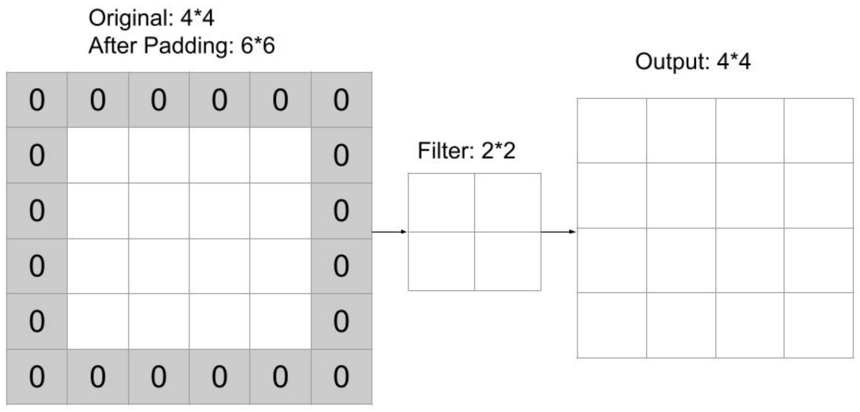


Figure 5.3: Visual Representation Of Padding Layer

5.1.4 Activation Function

Activation functions add non-linearity to the network. Without a non-linear activation function, the output would pass through a linear function where it wouldn't be able to handle complex interpretations of the data. To accurately interpret complex and high-dimensional data, non-linear activation functions are used to achieve non-linear mappings from inputs to outputs [79].

We use the Rectified Linear Units (ReLU) activation function for the hidden layers in our model. ReLU, introduced by [80], is one of the most successful and widely used activation functions in neural networks [81]. Mathematically, ReLU can be defined simply as [82]:

$$f(x) = max(0, x)$$

This equation means that when $x < 0$, then the output is 0, and when $x >= 0$ it's a linear function.

The activation function for the output layer of our model is a softmax activation function. The softmax function can be described as a combination of several sigmoid functions. Given that a sigmoid function yields values between 0 and 1, we can interpret the outputs for each class in a multi-class classification as the probabilities of the respective class. While sigmoid functions are used for binary classification, the softmax function can be used for multi-class classification like our task. For each class, it results in a probability value and the total adds up to 1 [79]. Using a threshold, usually 0.5, any probability above that is set to 1 and the rest to 0. In an output vector, 1 signifies the gesture exists with the rest being 0.

The formula for the softmax function is as follows:

$$\sigma(n_i) = \frac{e^{n_i}}{\sum_{j=1}^{N} e^{n_j}} \quad for\ i = 1, 2, \ldots, N$$

where N is the number of classes and n is the data points.

5.1.5 Pooling

Pooling can be thought of as another filter that slides through the input matrix to down-sample the image and reduce the complexity. Max-pooling is a type of pooling that is commonly used. The idea of max-pooling is to take the maximum value from each filter position and repeat. Figure 5.4 is an example of the max-pool filter being $2X2$, with a stride of 2.

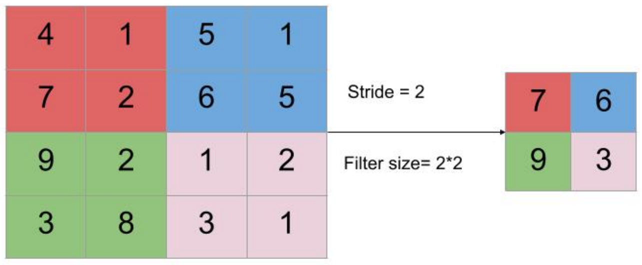


Figure 5.4: Max-Pooling With Dimensions $2X2$ & Stride=2

5.2 Long Short-Term Memory (LSTM)

LSTM, first introduced by [83], improves upon the traditional recurrent neural network (RNN) [84]. The traditional RNN structure is widely used for learning temporal information but struggles with the vanishing and exploding gradient problem [85],[86]. The vanishing gradient is when it goes to zero, resulting in the inability of the model to learn long-term dependencies. The exploding gradient is when it becomes extremely large, resulting in instability [87]. The LSTM model solves these issues and creates more accurate predictions [88].

Figure 5.5 gives a visual representation of the LSTM cell. The red circle depicts the memory cell. The input is the known data and the output is the result. The green represents the three gates in the memory unit: input, output, and forget gate. The input gate decides how much of the input is added to the memory cell. The output gate decides what data from the memory cell affects the output and the forget gate controls whether to keep the data in the memory cell or remove it [89]. The dashed lines indicate the function of the previous state, and the blue depicts the multiplications inside the unit. Due to the function of all these gates, LSTM memory units can interpret complex data and features from temporal data [90].

5.3 Model Architecture

In our work, we perform touch-based classification with three different types of input: skin-like sensor & load cell data, only skin-like sensor data, and only load cell data. We use The CNN-LSTM model architecture for the first two types of inputs and an LSTM neural network for only load cell data.

Each CNN layer consists of 64 as the number of convolution filters and 5 as the kernel size. The activation function for all the CNN layers is ReLU. We add a 2D

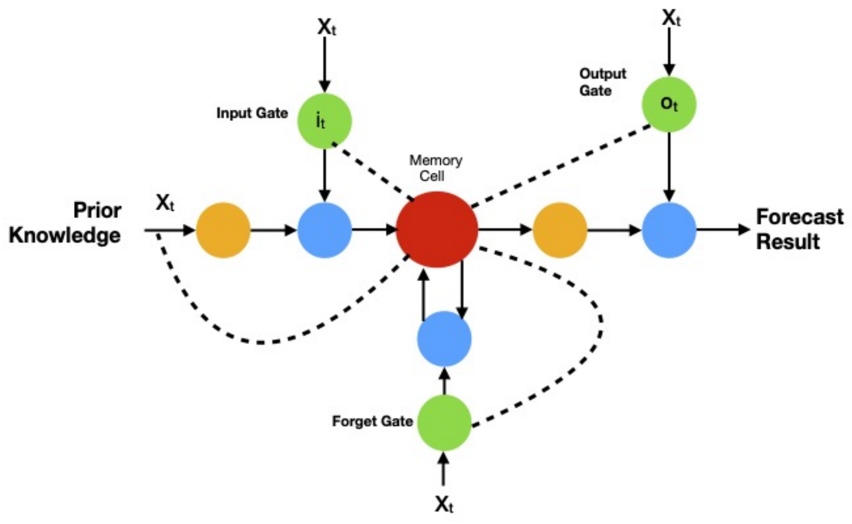


Figure 5.5: The LSTM Memory Unit

MaxPooling layer along with each CNN layer, with $(2, 2)$ as the dimensions for both the pool size and stride. The LSTM layers have 128 units and a dropout value of 0.2 when used in hidden layers. The final LSTM layer in each model has 256 units and a 0.5 dropout rate before the Dense layer. Dropout [91] is used to prevent over-fitting by dropping a percentage of random units during training. The fully-connected layer at the end of each model is the Dense layer. It has the number of output classes equal to the number of gestures. The activation function used is softmax. The loss function of the model is categorical cross-entropy. The formula for categorical cross-entropy loss is as follows [92]:

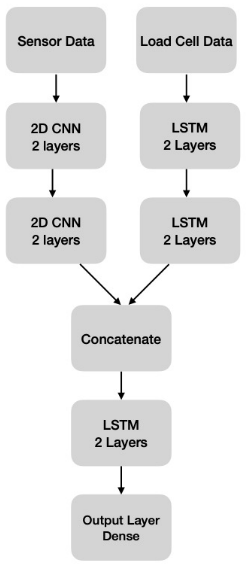
$$L(m, n) = -\sum_{k=1}^{N} m_k * log(n_k)$$

where "m" is the test set and "n" is the predicted set, with N being the number of gestures to classify.

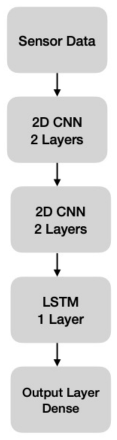
The optimizer used in our model is Adam (Adaptive Moment Estimation). During neural network training, the Adam optimizer follows an iterative optimization approach that minimizes the loss function. More details on its inner workings can be found in [93]. We use Adam since it outperforms the other optimizers by a significant margin when it comes to getting high performance with low training costs.

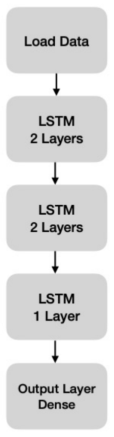
We added an EarlyStopping callback to our model which monitors the validation loss to decide when to stop the training. This helps prevent over-fitting. Choosing a large number of epochs could result in over-fitting while a smaller number could cause the model to underfit. With EarlyStopping, the model stops training when the validation loss begins to increase while the training loss is still decreasing [94]. This prevents over-fitting and helps achieve the highest accuracy. In our work, the batch size is set to 32 and the number of epochs to 30.

Our model architectures were determined through empirical experiments and parameter fine-tuning. Figure 5.6 gives a visual representation of the model architecture for the three input types.



(a) Skin-Like Sensor & Load Cell





(b) Only Skin-Like Sensor (c) Only Load Cell

Figure 5.6: Model Architectures Used In This Work. Each CNN Layer: 64 Filters, Kernel Size=5, & ReLU Activation. Each LSTM Layer: 128 Units & Dropout=0.2

Chapter 6

RESULTS & CONCLUSION

In our work, we aim to classify touch gestures using the skin-like sensor with a load cell underneath. We used deep learning methods to perform this classification task. Given the input sensor and load data, the model classifies it as one out of the eight gestures recorded. We used the accuracy metric to calculate how well our model performs. The method to calculate accuracy is as follows:

1. The test and prediction sets are represented as lists containing one-hot encoded vectors. Each vector in the prediction list is the model prediction of the corresponding vector in the test list.

2. We iterate through the test and prediction sets simultaneously. For each case of the two vectors being equal, the counter adds by 1. For example, where i is the positional index while iterating through the sets:

$$TestSet[i] = [0, 1, 0, 0, 0, 0, 0, 0], PredictionSet[i] = [0, 1, 0, 0, 0, 0, 0, 0]$$

3. If the vectors are not equal, then the counter is unchanged. This continues until we have iterated through the whole set. We then divide the counter value by the total length of the test set to determine the accuracy.

Our input dataset consists of 3840 data points for the skin-like sensor and load cell data, respectively. We perform a stratified 10-fold cross-validation across the dataset. In 10-fold cross-validation, the dataset is divided into 10 folds. In each iteration, one fold is used as testing data, while the rest of the folds are used as training data. This process repeats across all 10 folds, thus ensuring that the entire dataset is covered

[95]. The stratified 10-fold works the same as the normal 10-fold cross-validation, with the difference being that the percentage of samples for each class is the same across all the folds. This helps us make meaningful comparisons across the folds.

We perform the touch gesture classification using the following inputs:

1. Skin-Like Sensor & Load Cell Data
2. Only Skin-Like Sensor Data
3. Only Load Cell Data

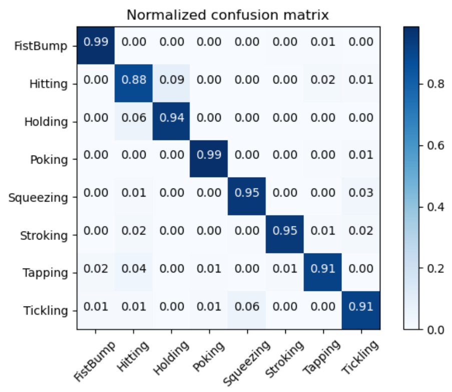
6.1 Model Results

For each set of inputs, our results consist of the accuracy values and the normalized confusion matrices. For each fold, the remaining dataset is split into training and validation sets with a split of 90% and 10% respectively. The fold, which is unseen during training, is the test set.

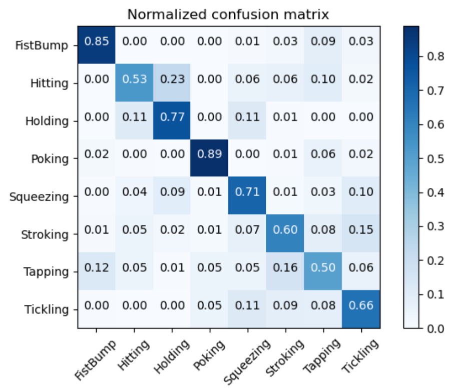
We also perform subject-dependent splitting of the data by keeping each subject separately as the test set and taking the rest of the subjects together as training and validation sets.

6.1.1 Skin-like Sensor & Load Cell Data

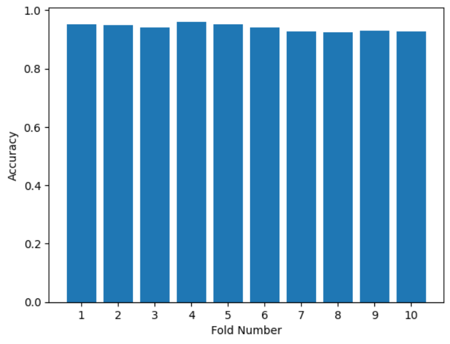
For an input combination of sensor data and load cell, we get an average of 94% accuracy after the stratified 10-fold cross-validation which is 7.5 times greater than chance. For the subject-dependent splitting, we get an average of 69% accuracy across all 20 subjects which is 5.5 times greater than chance. Figure 6.1 shows the normalized confusion matrices, bar plots for accuracies, and training vs validation loss for 10-fold validation and subject-dependent splitting respectively. Both training and validation losses decrease uniformly through the epochs which suggests that the



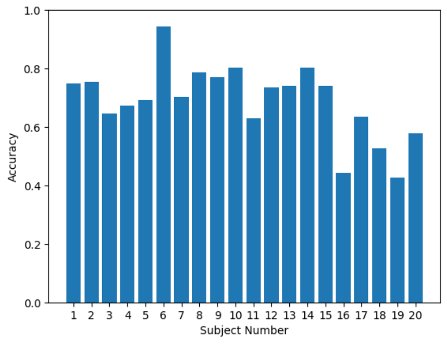
(a) 10-Fold Cross-Validation



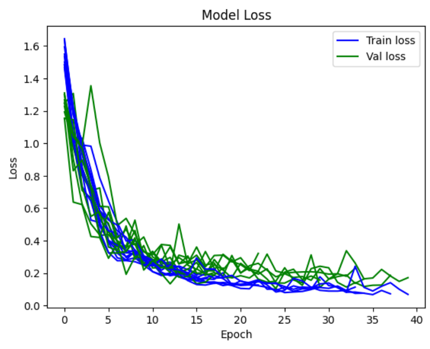
(b) Subject-Dependent Splitting



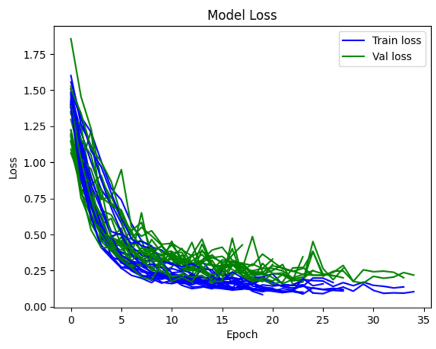
(c) 10-Fold Cross-Validation



(d) Subject-Dependent Splitting



(e) 10-Fold Cross-Validation



(f) Subject-Dependent Splitting

Figure 6.1: Normalised Confusion Matrices, Bar Plot of Accuracies, & Train vs Validation Loss Plots for Skin-Like Sensor & Load Cell Data

model fits well with the input data.

6.1.2 Only Skin-Like Sensor Data

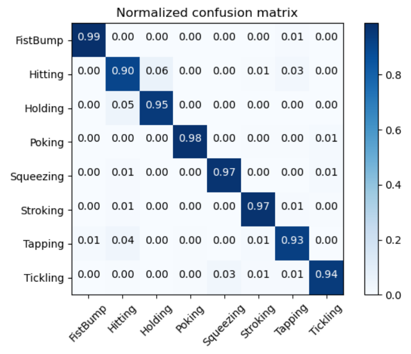
For the input being only the skin-like sensor data, we get an average of 95% accuracy after the stratified 10-fold cross-validation which is 7.5 times greater than chance. For the subject-dependent splitting, we get an average of 66% accuracy across all 20 subjects which is 5.5 times greater than chance.

Figure 6.2 shows the normalized confusion matrices, bar plots for accuracies, and training vs validation loss for 10-fold validation and subject-dependent splitting respectively. Both training and validation losses decrease uniformly through the epochs which suggests that the model fits well with the input data.

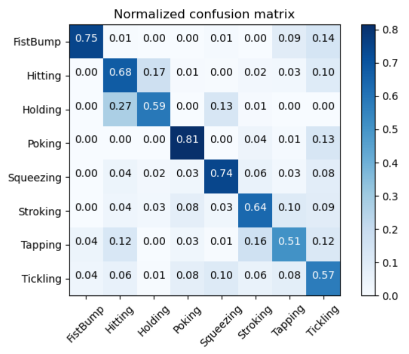
6.1.3 Only Load Cell Data

For the input being only the load cell data, we get an average of 45% accuracy after the stratified 10-fold cross-validation which is 3.5 times greater than chance. For the subject-dependent splitting, we get an average of 31% accuracy across all 20 subjects which is 2.5 times greater than chance.

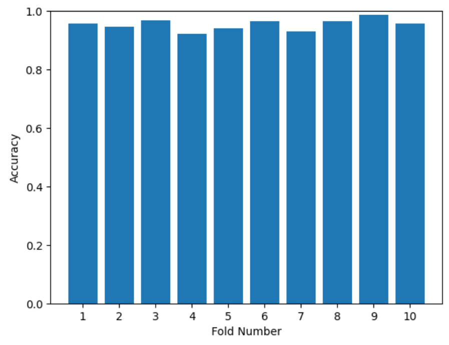
Figure 6.3 shows the normalized confusion matrices, bar plots for accuracies, and training vs validation loss for 10-fold validation and subject-dependent splitting respectively. The loss graphs suggest that the model is not able to learn much from the load cell data to perform the classification.



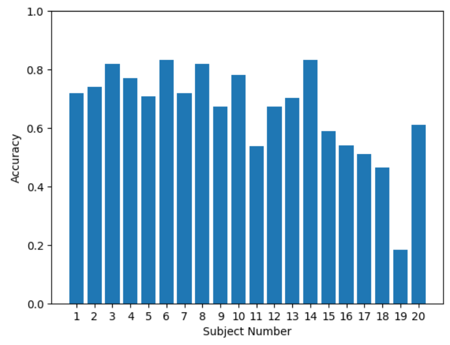
(a) 10-Fold Cross-Validation



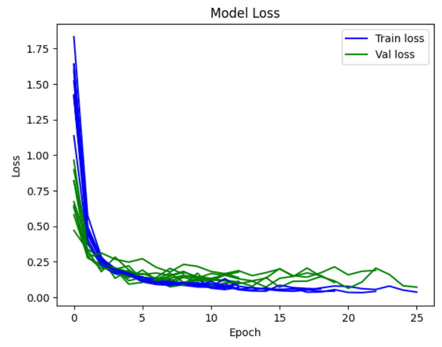
(b) Subject-Dependent Splitting



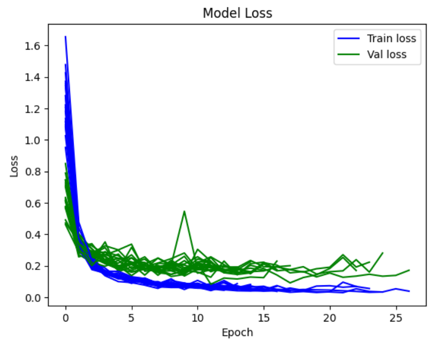
(c) 10-Fold Cross-Validation



(d) Subject-Dependent Splitting



(e) 10-Fold Cross-Validation

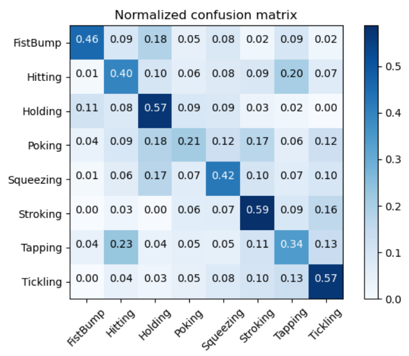


(f) Subject-Dependent Splitting

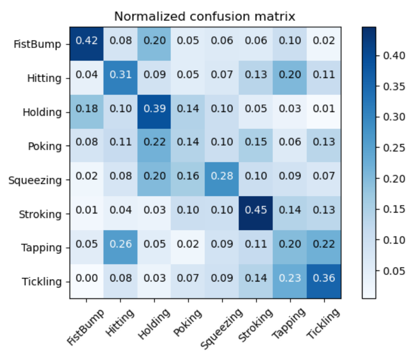
Figure 6.2: Normalised Confusion Matrices, Bar Plot of Accuracies, & Train vs Validation Loss Plots For Only Skin-Like Sensor Data

6.2 Discussion

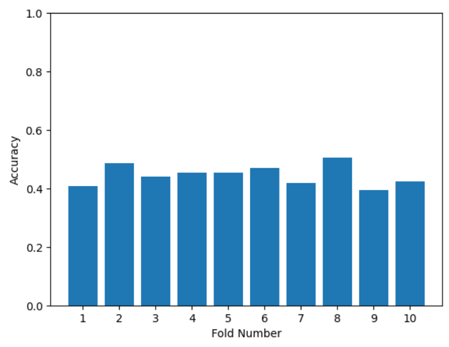
Our study demonstrates that the CNN-LSTM model achieves high accuracy in identifying eight touch gestures. Specifically, the accuracies are high when the model uses the first two types of input. This shows that the CNN-LSTM is able to capture



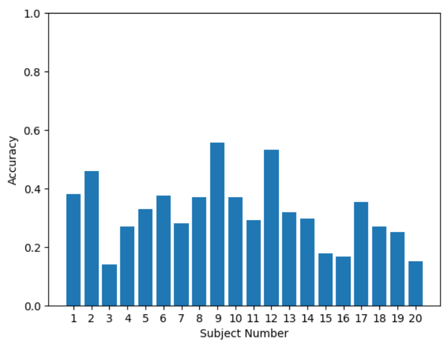
(a) 10-Fold Cross-Validation



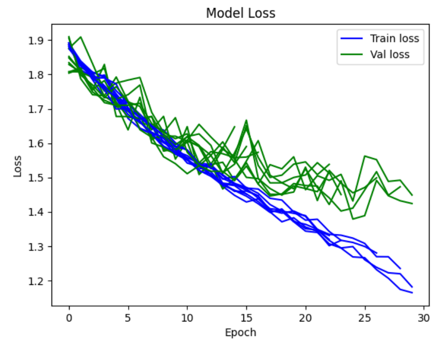
(b) Subject-Dependent Splitting



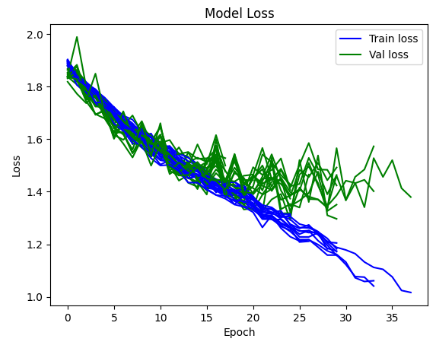
(c) 10-Fold Cross-Validation



(d) Subject-Dependent Splitting



(e) 10-Fold Cross-Validation



(f) Subject-Dependent Splitting

Figure 6.3: Normalised Confusion Matrices, Bar Plot of Accuracies, & Train vs Validation Loss Plots For Only Load Cell Data

both the spatial and temporal features from the input data. The LSTM model, with the load cell data as input, performs poorly comparatively. This suggests that spatial data is important for accurate predictions. Table 6.1 summarises the average accuracy for all 3 input types in this study.

Input Type	**Average Accuracy (%)**	
	10-Fold Validation	**Subject-Dependent Splitting**
Skin-Like Sensor & Load Cell Data	94	69
Only Skin-Like Sensor Data	95	66
Only Load Cell Data	45	31

Table 6.1: Average Accuracy For Each Input Type

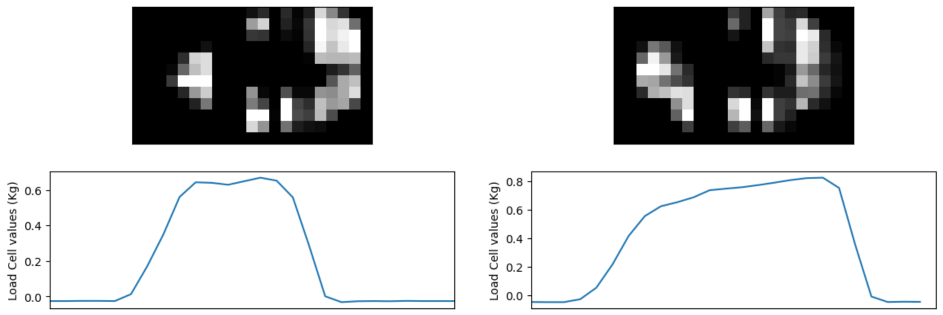
One consistent pattern across all normalized confusion matrices is that the model struggles to distinguish between holding and hitting gestures. It could be due to the similarity in the way these two gestures are performed on the sensor, as seen in Figure 6.4. The only noticeable difference between the two gestures is the duration of force applied on the load sensor. The similarity between how the gestures are recorded could explain the error in classifying them. Subject 19 performs poorly with only skin-like sensor data as input compared to skin-like sensor & load cell data combined. A potential reason could be that the subject performed more reserved gesture actions.

The model's performance is consistently higher across all the folds when stratified 10-fold cross-validation is performed. Its performance decreases for subject-dependent splitting. This is consistent for each input type. The performance of the model also varies between each subject. This could be due to the different ways in which each subject performs the gestures. Due to the variations in how each subject performs the same gesture, the model's performance is expected to decrease when the classification is subject-dependent. The accuracies from stratified 10-fold could be a good representation in real-world applications where SARs are personal or at-home robots. Such robots would receive touch gestures from the same person or group of people and can use an initial calibration phase to accurately detect the gesture performed. The accuracies from subject-dependent splitting could be a good representation for SARs in public spaces where they would receive touch gestures from a consistently large variety of people without any initial calibration.

The paper [56], used CoST and HAART datasets, with 14 and 7 gestures performed respectively, for touch gesture classification using deep learning methods. They performed subject-dependent splitting for their classification results. The CNN-RNN resulted in a 52% accuracy for CoST and 61% for HAART, which translates to 7.3 times more accurate than chance and 4.3 times more accurate than chance respectively. Our CNN-LSTM model achieved a 69% accuracy with a skin-like sensor and load cell. It is 5.5 times more accurate than chance, with 8 gestures.

The paper [57] uses a dataset with 13 touch gestures. In this work, the data was split into training, validation, and testing sets evenly across all the users. The paper got an accuracy of 66%, which is 8.6 higher than chance. This work also explores how considering shear force, along with normal force for the gestures, results in a higher accuracy of 74% which is 9.6 higher than chance. We can compare our stratified 10-fold validation results with this paper. We achieved a higher accuracy of 94% with 8 gestures. it is 7.5 higher than chance.

For a more standardized comparison, it might be better to analyze how the model predicts when compared to chance since the number of recorded gestures differs across papers. Based on this metric, our work has marginally lower accuracies compared to previous studies. However, our average accuracy results are higher than previous works.



(a) Hitting (b) Holding

Figure 6.4: Comparison Of Hitting & Holding Gestures

6.3 Conclusion

In this work, we built and evaluated a skin-like sensor, replicating a recent work [15] that introduces a novel design. Using the skin-like sensor and a load cell attached below to record force measurement, we performed a user study data collection with 20 subjects. We selected 8 gestures out of 30 commonly used ones, based on prior work with children with autism. Specialists identified these as important gestures for the robot to detect during interactions. We also referred to a prior work that clustered them based on their social and emotional meanings.

Using the data collected, we evaluated the skin-like sensor by performing touch-gesture classification using deep learning methods. We performed the classification with three types of input–skin-like sensor & load cell data, only skin-like sensor data, and only load cell data. We built a CNN-LSTM model for the first two types of input and an LSTM model for the third. The CNN-LSTM model performed well and was able to classify the data for 8 gestures with a high accuracy. The LSTM model, with only load cell data, performed poorly by comparison.

The user study data collection and the touch gesture classification performed in this study show that skin-like touch sensors could potentially be used on SARs in the future. Our research demonstrates that skin-like sensors can detect and interpret hand movements and can also serve as a natural interface for users interacting with them. This is particularly useful for SARs employed as assistants and caregivers, where natural interaction is crucial.

6.4 Future Work

In the future, a larger number of gestures can be added to the classification set and more subjects can be recorded to increase the dataset. The skin-like sensors also have the potential to be applied to curved surfaces in the future. With advances in deep learning models, touch gesture classification can also potentially be improved by using models like Transformers [96] and other state-of-the-art methods. Finally, this study can be extended to recording data from children with autism; using skin-like sensors in a real-world application.

REFERENCES

[1] David Feil-Seifer and Maja J Matarić. Socially assistive robotics. *IEEE Robotics & Automation Magazine*, 18(1):24–31, 2011.

[2] Steve Yohanan and Karon E MacLean. The haptic creature project: Social human-robot interaction through affective touch. In *Proceedings of the AISB 2008 Symposium on the Reign of Catz & Dogs: The Second AISB Symposium on the Role of Virtual Creatures in a Computerised Society*, volume 1, pages 7–11. Citeseer, 2008.

[3] Takanori Shibata and Kazuyoshi Wada. Robot therapy: a new approach for mental healthcare of the elderly–a mini-review. *Gerontology*, 57(4):378–386, 2011.

[4] Walter Dan Stiehl, Jeff Lieberman, Cynthia Breazeal, Louis Basel, Levi Lalla, and Michael Wolf. The design of the huggable: A therapeutic robotic companion for relational, affective touch. 2005.

[5] Walter Dan Stiehl, Cynthia Breazeal, Kuk-Hyun Han, Jeff Lieberman, Levi Lalla, Allan Maymin, Jonathan Salinas, Daniel Fuentes, Robert Toscano, Cheng Hau Tong, et al. The huggable: a therapeutic robotic companion for relational, affective touch. In *ACM SIGGRAPH 2006 emerging technologies*, pages 15–es. 2006.

[6] Maja J Matarić and Brian Scassellati. Socially assistive robotics. *Springer handbook of robotics*, pages 1973–1994, 2016.

[7] Fred R Volkmar, Catherine Lord, Anthony Bailey, Robert T Schultz, and Ami Klin. Autism and pervasive developmental disorders. *Journal of child psychology and psychiatry*, 45(1):135–170, 2004.

[8] Debra Phillips Parshall. Research and reflection: Animal-assisted therapy in mental health settings. *Counseling and Values*, 48(1):47–56, 2003.

[9] Takayuki Kanda, Takayuki Hirano, Daniel Eaton, and Hiroshi Ishiguro. Interactive robots as social partners and peer tutors for children: A field trial. *Human–Computer Interaction*, 19(1-2):61–84, 2004.

[10] Rachael Bevill Burns, Hyosang Lee, Hasti Seifi, Robert Faulkner, and Katherine J Kuchenbecker. Endowing a nao robot with practical social-touch perception. *Frontiers in Robotics and AI*, 9:840335, 2022.

[11] Rachael Bevill Burns, Hasti Seifi, Hyosang Lee, and Katherine J Kuchenbecker. Getting in touch with children with autism: Specialist guidelines for a touch-perceiving robot. *Paladyn, Journal of Behavioral Robotics*, 12(1):115–135, 2020.

[12] Tiffany Field. *Touch*. MIT press, 2014.

[13] Alberto Gallace and Charles Spence. The science of interpersonal touch: an overview. *Neuroscience & Biobehavioral Reviews*, 34(2):246–259, 2010.

[14] Judee K Burgoon. Relational message interpretations of touch, conversational distance, and posture. *Journal of Nonverbal behavior*, 15:233–259, 1991.

[15] Marc Teyssier, Brice Parilusyan, Anne Roudaut, and Jürgen Steimle. Human-like artificial skin sensor for physical human-robot interaction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3626–3633. IEEE, 2021.

[16] David Silvera-Tawil, David Rye, and Mari Velonaki. Artificial skin and tactile sensing for socially interactive robots: A review. *Robotics and Autonomous Systems*, 63:230–243, 2015.

[17] Ramzi Abou Chahine, Steven Vasquez, Pooyan Fazli, and Hasti Seifi. Clustering social touch gestures for human-robot interaction. *arXiv preprint arXiv:2304.01334*, 2023.

[18] Adam Robaczewski, Julie Bouchard, Kevin Bouchard, and Sébastien Gaboury. Socially assistive robots: The specific case of the nao. *International Journal of Social Robotics*, 13:795–831, 2021.

[19] M Butter, Boxtel Jv, S Kalisingh, et al. Robotics for healthcare, state of the art report. *TNO, commissioned by the European Commission, DG Information Society*, 2007.

[20] Adriana Tapus, Mataric Maja, and Brian Scassellatti. The grand challenges in socially assistive robotics. *IEEE Robotics and Automation Magazine*, 14(1):N–A, 2007.

[21] ES Kim, E Newland, R Paul, and B Scassellati. A robotic therapist for positive, affective prosody in high-functioning autistic children. In *Poster Pres. at the Intl. Meeting for Autism Research*, 2008.

[22] Maja J Matarić, Jon Eriksson, David J Feil-Seifer, and Carolee J Winstein. Socially assistive robotics for post-stroke rehabilitation. *Journal of neuroengineering and rehabilitation*, 4:1–9, 2007.

[23] Cory D Kidd and Cynthia Breazeal. Robots at home: Understanding long-term human-robot interaction. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3230–3235. IEEE, 2008.

[24] Rachael Bevill Burns, Hasti Seifi, Hyosang Lee, and Katherine J Kuchenbecker. A haptic empathetic robot animal for children with autism. In *Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, pages 583–585, 2021.

[25] John-John Cabibihan, Hifza Javed, Marcelo Ang, and Sharifah Mariam Aljunied. Why robots? a survey on the roles and benefits of social robots in the therapy of children with autism. *International journal of social robotics*, 5:593–618, 2013.

[26] Ben Robins, Nuno Otero, Ester Ferrari, and Kerstin Dautenhahn. Eliciting requirements for a robotic toy for children with autism-results from user panels. In *RO-MAN 2007-The 16th IEEE International Symposium on Robot and Human Interactive Communication*, pages 101–106. IEEE, 2007.

[27] Andreea Peca, Ramona Simut, Sebastian Pintea, Cristina Costescu, and Bram Vanderborght. How do typically developing children and children with autism perceive different social robots? *Computers in Human Behavior*, 41:268–277, 2014.

[28] Carissa J Cascio, Estephan J Moana-Filho, Steve Guest, Mary Beth Nebel, Jonathan Weisner, Grace T Baranek, and Gregory K Essick. Perceptual and neural response to affective tactile texture stimulation in adults with autism spectrum disorders. *Autism Research*, 5(4):231–244, 2012.

[29] Ben Robins, Farshid Amirabdollahian, Ze Ji, and Kerstin Dautenhahn. Tactile interaction with a humanoid robot for children with autism: A case study analysis involving user requirements and results of an initial implementation. In *19th International Symposium in Robot and Human Interactive Communication*, pages 704–711. IEEE, 2010.

[30] Ben Robins and Kerstin Dautenhahn. Tactile interactions with a humanoid robot: novel play scenario implementations with children with autism. *International journal of social robotics*, 6:397–415, 2014.

[31] Hideki Kozima, Marek P Michalowski, and Cocoro Nakagawa. Keepon: A playful robot for research, therapy, and entertainment. *International Journal of social robotics*, 1:3–18, 2009.

[32] Jan BF Van Erp and Alexander Toet. Social touch in human–computer interaction. *Frontiers in digital humanities*, 2:2, 2015.

[33] Carissa J Cascio, David Moore, and Francis McGlone. Social touch and human development. *Developmental cognitive neuroscience*, 35:5–11, 2019.

[34] Harry F Harlow and Robert R Zimmermann. Affectional response in the infant monkey: Orphaned baby monkeys develop a strong and persistent attachment to inanimate surrogate mothers. *Science*, 130(3373):421–432, 1959.

[35] Luke J Wood, Abolfazl Zaraki, Ben Robins, and Kerstin Dautenhahn. Developing kaspar: a humanoid robot for children with autism. *International Journal of Social Robotics*, 13(3):491–508, 2021.

[36] Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017.

[37] Wanlin Li, Meng Wang, Jiarui Li, Yao Su, Devesh K Jha, Xinyuan Qian, Kaspar Althoefer, and Hangxin Liu. L ˆ{3} f-touch: A wireless gelsight with decoupled tactile and three-axis force sensing. *IEEE Robotics and Automation Letters*, 2023.

[38] Narjes Pourjafarian, Anusha Withana, Joseph A Paradiso, and Jürgen Steimle. Multi-touch kit: A do-it-yourself technique for capacitive multi-touch sensing using a commodity microcontroller. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, pages 1071–1083, 2019.

[39] Frank Beck and Bent Stumpe. Two devices for operator interaction in the central control of the new cern accelerator. Technical report, European Organization for Nuclear Research, 1973.

[40] SK Lee, William Buxton, and Kenneth C Smith. A multi-touch three dimensional touch-sensitive tablet. *Acm Sigchi Bulletin*, 16(4):21–25, 1985.

[41] Tobias Grosse-Puppendahl, Christian Holz, Gabe Cohn, Raphael Wimmer, Oskar Bechtold, Steve Hodges, Matthew S Reynolds, and Joshua R Smith. Finding common ground: A survey of capacitive sensing in human-computer interaction. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pages 3293–3315, 2017.

[42] Jun Rekimoto. Smartskin: an infrastructure for freehand manipulation on interactive surfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 113–120, 2002.

[43] Darren Leigh, Clifton Forlines, Ricardo Jota, Steven Sanders, and Daniel Wigdor. High rate, low-latency multi-touch sensing with simultaneous orthogonal multiplexing. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 355–364, 2014.

[44] Humza Akhtar, Qian Kemao, and Ramakrishna Kakarala. A review of sensing technologies for small and large-scale touch panels. In *Fifth International Conference on Optical and Photonics Engineering*, volume 10449, pages 209–221. SPIE, 2017.

[45] Toshiharu Mukai, Masaki Onishi, Tadashi Odashima, Shinya Hirano, and Zhiwei Luo. Development of the tactile sensor system of a human-interactive robot “ri-man”. *IEEE Transactions on robotics*, 24(2):505–512, 2008.

[46] Markus Fritzsche, Norbert Elkmann, and Erik Schulenburg. Tactile sensing: A key technology for safe physical human robot interaction. In *Proceedings of the 6th International Conference on Human-robot Interaction*, pages 139–140, 2011.

[47] Nicholas Xydas and Imin Kao. Modeling of contact mechanics and friction limit surfaces for soft fingers in robotics, with experimental results. *The International Journal of Robotics Research*, 18(9):941–950, 1999.

[48] Dianne TV Pawluk and Robert D Howe. Dynamic lumped element response of the human fingerpad. 1999.

[49] Luigi Biagiotti, Claudio Melchiorri, Paolo Tiezzi, and Gabriele Vassura. Modelling and identification of soft pads for robotic hands. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2786–2791. IEEE, 2005.

[50] B.B Edin, L Beccai, L Ascari, S Roccella, J.J Cabibihan, and M.C Carrozza. Bio-inspired approach for the design and characterization of a tactile sensory system for a cybernetic prosthetic hand. In *Proceedings ICRA 2006.*, pages 1354–1358. IEEE, 2006.

[51] Alexander Schmitz, Perla Maiolino, Marco Maggiali, Lorenzo Natale, Giorgio Cannata, and Giorgio Metta. Methods and technologies for the implementation of large-scale robot tactile sensors. *IEEE Transactions on Robotics*, 27(3):389–400, 2011.

[52] Takashi Minato, Yuichiro Yoshikawa, Tomoyuki Noda, Shuhei Ikemoto, Hiroshi Ishiguro, and Minoru Asada. Cb2: A child robot with biomimetic body for cognitive developmental robotics. In *2007 7th IEEE-RAS International Conference on Humanoid Robots*, pages 557–562. IEEE, 2007.

[53] Steve Yohanan and Karon E MacLean. The role of affective touch in human-robot interaction: Human intent and expectations in touching the haptic creature. *International Journal of Social Robotics*, 4:163–180, 2012.

[54] Merel M Jung, Ronald Poppe, Mannes Poel, and Dirk KJ Heylen. Touching the void–introducing cost: corpus of social touch. In *Proceedings of the 16th International Conference on Multimodal Interaction*, pages 120–127, 2014.

[55] Xi Laura Cang, Paul Bucci, Andrew Strang, Jeff Allen, Karon MacLean, and HY Sean Liu. Different strokes and different folks: Economical dynamic surface sensing and affect-related touch recognition. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pages 147–154, 2015.

[56] Dana Hughes, Alon Krauthammer, and Nikolaus Correll. Recognizing social touch gestures using recurrent and convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2315–2321. IEEE, 2017.

[57] Hojung Choi, Dane Brouwer, Michael A Lin, Kyle T Yoshida, Carine Rognon, Benjamin Stephens-Fripp, Allison M Okamura, and Mark R Cutkosky. Deep learning classification of touch gestures using distributed normal and shear force. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3659–3665. IEEE, 2022.

[58] Yue Wang, Cherry Gregory, and Mark A Minor. Improving mechanical properties of molded silicone rubber for soft robotics through fabric compositing. *Soft robotics*, 5(3):272–290, 2018.

[59] SARAH Al-Mutlaq and A Wende. Load cell amplifier hx711 breakout hookup guide. *Retrieved from Sparkfun Start Something website: https://learn. sparkfun. com/tutorials/load-cell-amplifier-hx711-breakout-hookupguide/introduction*, 2016.

[60] Neftali Watkinson, Aniket Shivam, Alexandru Nicolau, and Alexander Veidenbaum. Teaching parallel computing and dependence analysis with python. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 320–325. IEEE, 2019.

[61] Sikha Bagui, Debarghya Nandi, Subhash Bagui, and Robert Jamie White. Machine learning and deep learning for phishing email classification using one-hot encoding. *Journal of Computer Science*, 17:610–623, 2021.

[62] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[63] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[64] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[65] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[66] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[67] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[68] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015.

[69] Muhammad Abdullah, Mobeen Ahmad, and Dongil Han. Facial expression recognition in videos: An cnn-lstm based model for video classification. In *2020 International Conference on Electronics, Information, and Communication (ICEIC)*, pages 1–3. IEEE, 2020.

[70] Bruno Barros, Paulo Lacerda, Celio Albuquerque, and Aura Conci. Pulmonary covid-19: learning spatiotemporal features combining cnn and lstm networks for lung ultrasound video classification. *Sensors*, 21(16):5486, 2021.

[71] Amin Ullah, Jamil Ahmad, Khan Muhammad, Muhammad Sajjad, and Sung Wook Baik. Action recognition in video sequences using deep bi-directional lstm with cnn features. *IEEE access*, 6:1155–1166, 2017.

[72] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.

[73] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[74] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern recognition*, 77:354–377, 2018.

[75] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[76] Mohammad Mustafa Taye. Theoretical understanding of convolutional neural network: concepts, architectures, applications, future directions. *Computation*, 11(3):52, 2023.

[77] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

[78] K Teilo. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[79] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316, 2017.

[80] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *nature*, 405(6789):947–951, 2000.

[81] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[82] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[83] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[84] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[85] Van-Dai Ta, CHUAN-MING Liu, and Direselign Addis Tadesse. Portfolio optimization-based stock prediction using long-short term memory network in quantitative trading. *Applied Sciences*, 10(2):437, 2020.

[86] Ons Zarrad, Mohamed Ali Hajjaji, and Mohamed Nejib Mansouri. Hardware implementation of hybrid wind-solar energy system for pumping water based on artificial neural network controller. *Studies in Informatics and Control*, 28(1):35–44, 2019.

[87] Boris Hanin. Which neural net architectures give rise to exploding and vanishing gradients? *Advances in neural information processing systems*, 31, 2018.

[88] Wenjie Lu, Jiazheng Li, Yifan Li, Aijun Sun, and Jingyang Wang. A cnn-lstm-based model to forecast stock prices. *Complexity*, 2020:1–10, 2020.

[89] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. Cambridge University Press, 2023. `https://D2L.ai`.

[90] Zheng Zhao, Weihai Chen, Xingming Wu, Peter CY Chen, and Jingmeng Liu. Lstm network: a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, 11(2):68–75, 2017.

[91] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[92] Elliott Gordon-Rodriguez, Gabriel Loaiza-Ganem, Geoff Pleiss, and John Patrick Cunningham. Uses and abuses of the cross-entropy loss: Case studies in modern deep learning. 2020.

[93] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[94] Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In *International conference on machine learning*, pages 8093–8104. PMLR, 2020.

[95] Isaac Kofi Nti, Owusu Nyarko-Boateng, Justice Aning, et al. Performance of machine learning algorithms with different k values in k-fold cross-validation. *International Journal of Information Technology and Computer Science*, 13(6):61–71, 2021.

[96] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.