Hand Gesture Recognition using Flex Sensor and Machine Learning Algorithms

Akash Kumar Panda
Department of Computer Science and Engineering, National Institute of Technology Meghalaya, Shillong, India.
(Email: akpanda734@gmail.com)

Rommel Chakravarty
Department of Computer Science and Engineering, National Institute of Technology Meghalaya, Shillong, India.
(Email: rommelchac@gmail.com)

Soumen Moulik
Department of Computer Science and Engineering, National Institute of Technology Meghalaya, Shillong, India.
(Email: mouliksoumen@gmail.com)

***Abstract*—In this paper, we have proposed a mechanism of hand gesture recognition using flex sensors and Arduino UNO. The data acquired from the sensors corresponding to different hand gestures are analyzed with the help of different traditional machine learning algorithms. We also proposed an adversarial learning approach that performs better classification in comparison with these traditional learning models. The results of the proposed approach records a commendable accuracy of 88.32% with a precision of 81.77%, a recall of 84.37% and F1-score of 82.78%. The proposed model yields better results in terms of all these four performance metrics.**

***Index Terms*—Hand Gesture Recognition, Flex Sensor, Adversarial Machine Learning, Neural Network.**

I. Introduction

In this world of continuously evolving technology, there is a need to interact with the technologies in a more improvised manner. As most of us have watched many science fiction movies, we must have noticed one of the most efficient ways of communicating with the technology is by using different fingers on our hands. The means of communicating with others with the help of nonverbal messages through the extensive use of finger motions is known as Hand Gestures. Such hand gestures come to us naturally and play a vital role in our life. By using it, one can express well and can maintain the flow of a conversation. Others perceive us more clearly when we use hand gestures while communicating with them. Even blind people use them automatically, like a reflex action.

Hand Gesture Recognition (HGR) is considered one of the main areas contributing a lot to the future of technology. One of the most advanced topics is HGR to speech conversion, which will help those with speech disabilities. In our work, we shall be using Flex sensors to classify different gestures. A Flex sensor, also known as a bend sensor is a low cost sensor which basically acts a variable resistor. On bending the surface of the sensor, the resistance is varied which gives necessary readings. When the sensor is bent, the resistance starts increasing; at 90 degrees, the resistance is maximum.

II. Related Works

In this section, we will be providing a brief idea of some previous works which have been very prominent in the field of HGR, mainly using Flex sensors and Electromyography (EMG) sensors.

In their works, Georgi et al. [1] have evaluated a wearable gesture recognition system's performance. The system captures the motions of the arm, hand, and finger. The fusion of wrist-worn Inertial Measurement Unit (IMU) and forearm's EMG signals was done to infer hand and finger movements. The classifiers used are Hidden Markov Models. With the above system's help, a recognition rate of 97.8% and 74.3% is observed in session-independent and person-independent recognition, respectively.

Mochammad et al. [2] have processed EMG signals, and they were classified using 16 time-domain feature extraction. Their study was used to classify the thumb, index, middle, ring, and little finger movement. The pattern recognition used Artificial Neural Network (ANN) with two layers of a feed-forward network resulting in a confusion matrix with the above finger movements' accuracy as 100%. The confusion index of 16.7%, an error of 3.3%, and a complete performance of 96.7% was observed.

In their works, Chuck and et al. [3] have used EMG signals and Electropalatogram (EPG) to recognize silent speech. In this case, the electrode signals were recorded from the larynx and it's sublingual areas, which are further filtered by noise and were transformed into features using complex dual quadtree wavelet transforms. The accuracy achieved in the method was 92% in classifying six sub acoustic words.

Xu Zhang et al. [4] have presented a hand gesture framework using a three-axis accelerometer and multi-channel EMG. To achieve the final results, a decision tree and multistream hidden Markov models are utilized. While recognizing 72 Chinese Sign Language, the experiment results on 95.3% and 96.3% accuracy for two subjects, i.e., ACC and EMG.

In their works, Shantanu K. Dixit and Nitin S. Shingi [5] have made an innovative use of Flex sensors, ultrasonic sensors, and a three-axis accelerometer to make a robotic hand. The Flex sensors were mounted on each joint of all five fingers. Motor 1, motor 2, and motor 3 of each finger in the robotic hand rotate according to the bending of Flex

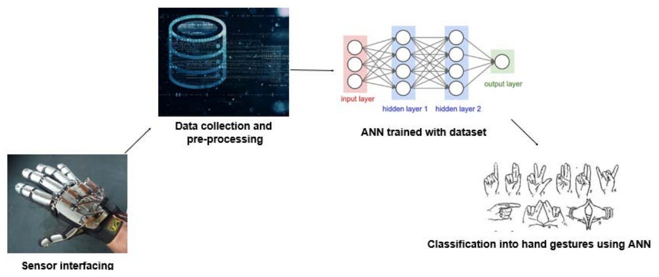


Fig. 1. Work Flow designed for the proposed project

sensors in the corresponding human finger.

Taking necessary cues from the above mentioned related works, we have developed a hand gesture recognition system by taking necessary readings from the Flex sensors and classify them using a suitable algorithm.

III. Proposed Work

The proposed work involves four primary sub-sections i.e., Sensor Interfacing, Data Collection and Pre-processing, Feature Extraction and Selection, and Gesture Recognition. The total work-flow is illustrated in Fig. 1.

A. Sensor Interfacing

Flex sensor (SEN - 08606) is the primary component used in this work. The hardware prototype with flex sensors mounted on fingers is represented in Fig. 2. All the used components are illustrated in Fig. 3. Flex sensors' terminal resistance changes when they are bent, and this helps in detecting the motion in a specific part of the body. The Flex sensor does not contain polarized terminals. So there are no positive and negative terminals. In general, the pin number P1 is connected to positive of power source and P2 is connected to ground, as illustrated in Fig. 4. With increase in bent/Flex in the Flex sensor, the resistance increases. Fig. 4 shows the connection followed in order to Interface the Arduino. After properly connecting the sensor with Arduino, the later is connected to PC. With the help of Arduino IDE, a specific designed code is exported into the Arduino based on which when there is a slight change in the bend of Flex sensor, there is a change observed in the readings obtained as output from the Arduino. Features and specifications of the used flex sensors are as follows:

- Flat Resistance: 10K Ohms
- Power Rating: 0.50 W
- Description: Spark Fun Flex Sensor
- Resistance Tolerance: ±30
- Length: 4.5 inches
- Bend Resistance Range: 60K to 110K Ohms
- Temperature Range: -35°C to +80°C
- Unit Weight: 2g

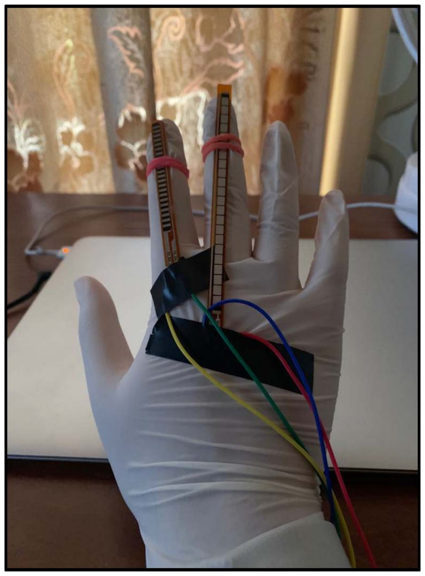
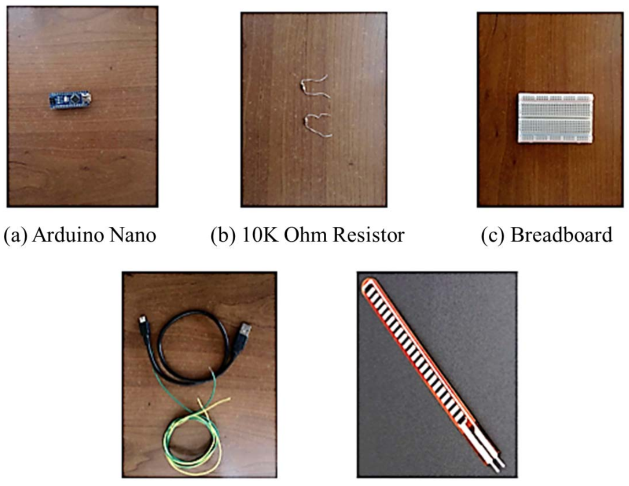


Fig. 2. Hardware prototype developed using Flex Sensor



(a) Arduino Nano (b) 10K Ohm Resistor (c) Breadboard

(d) Connecting Wires (e) Flex Sensor

Fig. 3. Components that were used in the Sensor Interfacing process

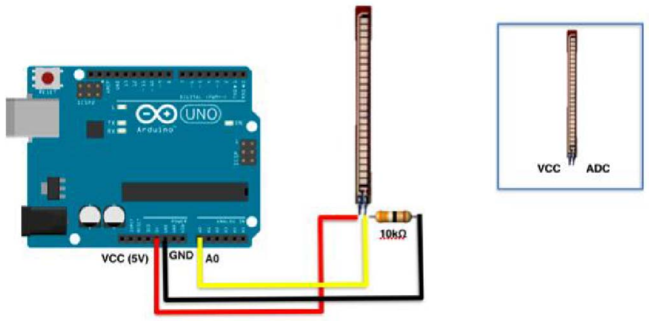


Fig. 4. Flex Sensor Circuit Diagram

B. Data Collection and Pre-processing

After the above fundamental work, we proceed to next level that is taking the data from actual conditions of the flex sensors, i.e., when they are attached on fingers and the readings actually measure the bending of each finger they are attached to. The setup was prepared in the following steps:

- In our case, white surgical gloves were worn on the hand. It is done to check the effect of perspiration on the Flex sensors.
- The Flex sensors were then tied on index finger and middle finger of right hand. Elastic bands were used to keep the sensors in place.
- Long copper wires were used to connect the flex sensors to Arduino and resistor as per the circuit diagram.

When the setup is ready, as illustrated in Fig. 2, the connection was checked again to ensure no loose connection and the work then proceeded to the next stage, i.e., data pre-processing.

Data Pre-processing: In this phase, the collected data were categorized on four basic class labels, i.e., Class label 0, Class label 1, Class label 2 and Class label 3. The Gestures signified by the Class labels are described here-under:

- *Class label 0:* This class label indicated the hand gesture in which both of the index finger and middle finger have curled inside.
- *Class label 1:* This class label indicated the hand gesture in which both of the index finger and middle finger are fully stretched and kept in a manner that represented 'V' symbol.
- *Class label 2:* For this class, the middle finger is curled inside while the index finger remains as it is in the case before and the readings are noted.
- *Class label 3:* This is the last class label we are considering. The hand gesture in this class label is derived from the hand gesture considered in class label 1. The difference is that the index finger is curled inside in this case and all the other finger position remaining same.

The data recorded from each sensor is an integer value. In the interval of 0.25 seconds, a new value was obtained based on the change done in bending of the sensor. The two data that is obtained is labelled as x and y in our dataset. The x and y represented the value obtained from the flex sensor tied to index finger and middle finger, respectively. Before the use of the above dataset, shuffling was carefully done by using random library of python. The shuffling ensured sufficient availability of various label data to the machine learning model for training, testing and validation purposes. Some out-layer physical noises were also removed from the dataset.

C. Feature Extraction and Selection

For better training and prediction, some extra features were extracted from the dataset. The extra features can be categorized as follows:

- Squares: In this kind of feature extraction, there are two features obtained, i.e., square(x) and square(y), where square(x) = x^2 and square(y) = y^2
- First Difference: This kind of feature ex-traction takes into account the values of previous row as well and finds the deviation occurred from it, i.e., X_{fd} and Y_{fd}. The absolute value is considered. Therefore, $X_{fd} = abs\ (x_i – x_{i-1})$ and $Y_{fd} = abs\ (y_i – y_{i-1})$.
- Second Difference: This kind of feature extraction takes into account the values of previous row and the upcoming row and finds the mean deviation of the three, i.e., X_{sd} and Y_{sd}. The absolute value is considered. Therefore, $X_{sd} = abs\ (x_{i+1} – 2 * x_i + x_{i-1})$ and $Y_s d = abs\ (y_{i+1} – 2 * y_i + y_{i-1})$

A total of 6 features were extracted from the pre-existing data and were then used for Machine Learning of the models. It is observed that the models were able to perform better when the extra features were present. The data included the values of x and y when the gestures were prominent and the data that is recorded while the transition from one gesture to another or when the gesture is ambiguous is ignored.

D. Gesture Recognition

In order to interact with different electronic devices, we can give hand gestures as input. This way of taking the input can bring a vast change in each one of our life. For the people who are incapable of speaking can utilise the concept of converting hand gestures to speech which will be done by a machine at real time. Very soon the application can be seen in various fields. This greatly inspired us to come to our final and main part of our work, i.e, recognition of hand gestures. All the processed data were trained, validated and tested on different machine learning models. For our classification task, we have used a modern machine learning technique – Adversarial Machine Learning.

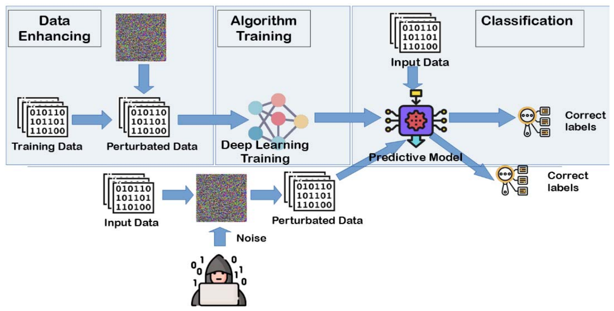


Fig. 5. Architecture of Adversarial Learning Model Used.[9]

The main motivation behind using this model is to reach to a maximum possible accuracy even if there is some alteration which has occurred in the data for training or testing phase by external factors. The external factors can be anything or anyone who is trying to change the data and expecting a wrong output, intentionally. Such people may be termed as hackers. The various attacks that are more likely to occur for our deep learning models include fast-gradient sign method (FGSM), basic iterative method (BIM) or momentum iterative method (MIM). The earlier mentioned attacks are some pure form of many gradient-based evading techniques that attackers use for evading the classification model. Adversarial attacks take place when noise is added to the data which in turn, while validating the already trained model may result in the classification of false labels. The proposed model is used to prevent such attacks in order to develop an efficient and robust ANN model which can rightfully classify labels despite being feed with noisy data. The architecture of Adversarial Learning Model is illustrated in Fig. 5.

IV. Result

The entire code for the experiment was developed using Keras, a Python framework which is used commonly for developing deep learning models. Additionally, other libraries used consisted of Scikit-learn, pandas and NumPy. The experiment required the use of a computer server with dedicated Graphical Processing Unit (GPU) as a result of which a machine with AMD Ryzen 5 3550H processor and NVidia GeForce GTX 1050 graphics card was used.

The Adversarial Learning was found to have outperformed standard classifiers. The comparisons were done by running all the machine learning algorithms on the dataset obtained from our initial work. We have tried our best to keep all the necessary conditions as similar as possible in order to have an ideal comparison of all of the models being compared. The below table shows the performance matrix of ANN used before the application of Adversarial Learning on our dataset:

TABLE I
Performance Matrix for Simple ANN without Adversarial Learning

Training Acc.	Training Loss	Validation Acc.	Validation Loss
82.42%	0.3519	78.83%	0.3640

Considering Model Accuracy graph in Fig.6, it is evident from the figure that the model is performing well. Initially, the Accuracy values are less for both Testing and Training but after 20 epochs, a sporadic increase in their value is observed. The values keep on increasing and decreasing for the next 8-10 epochs. After 30 epochs, a pretty high accuracy values are observed with not much change in them. This phenomenon continues up to next 70 epochs.

Considering Model Loss graph in Fig. 7, we can observe that Training Loss and Testing loss are very much similar by value throughout the execution. This indicates us that the chances of over-fitting and underfitting is extremely less.

From the graph given in Fig. 8, it is observed that the testing process records a commendable accuracy of 88.32%. For other performance matrices also the performance of

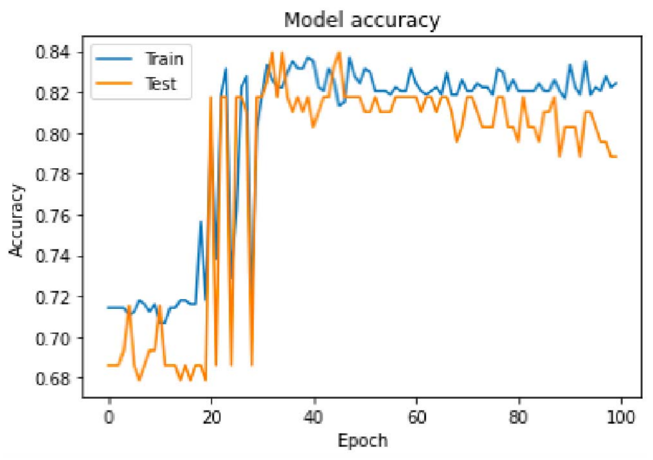


Fig. 6. Model Accuracy

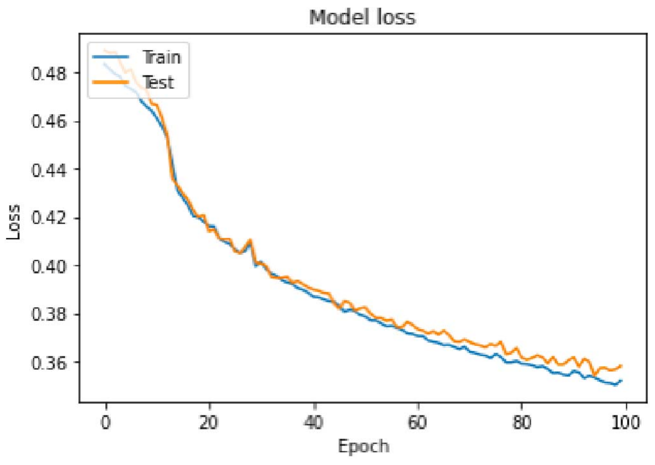


Fig. 7. Model Loss

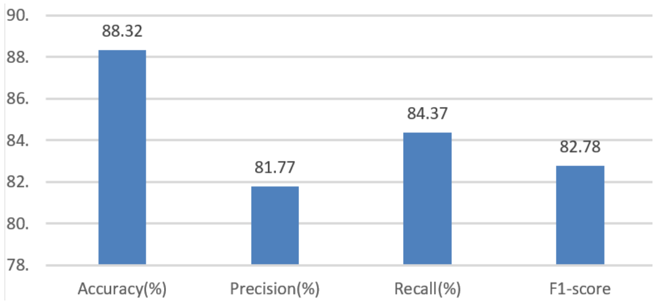


Fig. 8. Observations recorded from data testing after application of Adversarial Learning

the proposed model is better than traditional approaches, as illustrated in Fig. 9. This implies that the outcome of using the adversarial learning process was fruitful. However, it becomes more important to compare the results of this approach with some other standard classifiers in order to draw some definite comparisons.

From the Table II, it is quite evident that our proposed model has recorded the best performance matrices as com-

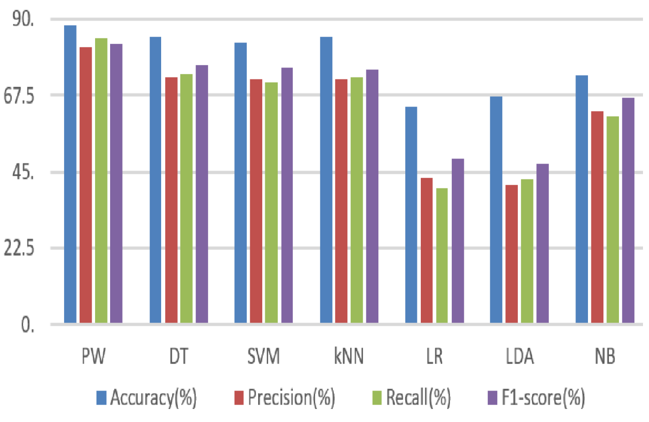


Fig. 9. Graph depicting performance comparison

TABLE II
PERFORMANCE COMPARISON OF PROPOSED WORK WITH OTHER MODELS

Models	Accuracy(%)	Precision(%)	Recall(%)	F1-score(%)
PW	88.32	81.77	84.37	82.78
DT	84.79	72.84	73.74	76.32
SVM	83.04	72.39	71.26	75.70
kNN	84.79	72.41	72.76	75.13
LR	64.32	42.98	40.20	48.81
LDA	67.25	41.26	42.70	47.25
NB	73.68	62.90	61.50	66.97

pared to the other standard classifiers. We also observe that the results for Logistic Regression and Linear Discriminant Analysis are the worst in the lot. This is primarily because of the fact that they are inconsistent when it comes to classifying multi-class data set. Other classifiers' performance was very mediocre.

V. CONCLUSION

Hand Gesture Recognition finds its impact on healthcare to be absolute and unprecedented. This motivates us to develop a classification model for the same which shall be beneficial for societal needs. Conclusively, we are looking forward to improve the results of the model by increasing the sample size of the dataset extracted. Also, we attempt to improve the existing algorithm by fine-tuning the existing model and its related constraints. An app will be developed that will perform the conversion of hand gestures to speech. We sincerely feel that this will be beneficial for those with speech disabilities.

REFERENCES

[1] Georgi, Marcus, Christoph Amma, and Tanja Schultz. ”Recognizing Hand and Finger Ges-tures with IMU based Motion and EMG based Muscle Activity Sensing.” In *Biosignals*, pp. 99-108. 2015.

[2] Ariyanto, Mochammad, Wahyu Caesarendra, Khusnul A. Mustaqim, Mohamad Irfan, Jonny A. Pakpahan, Joga D. Setiawan, and Andri R. Winoto. ”Finger movement pattern recogni-tion method using artificial neural network based on electromyography (EMG) sensor.” In 2015 *International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology (ICACOMIT)*, pp. 12-17. IEEE, 2015.

[3] Chuck, Jorgensen, D. Lee, and Shane Aga-bon. ”Sub Auditory Speech Recognition Based on EMG/EPG Signals.” In Proceedings of the *International Joint Conference on Neural Networks*, pp. 1098-7576. 2003.

[4] Zhang, Xu, Xiang Chen, Yun Li, Vuokko Lantz, Kongqiao Wang, and Jihai Yang. ”A framework for hand gesture recognition based on accelerometer and EMG sensors.” *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 41, no. 6 (2011): 1064-1076.

[5] Dixit, Dr Shantanu K., and Mr Nitin S. Shin-gi. ”Implementation of flex sensor and electronic compass for hand gesture based wireless automation of material handling robot.” *International Journal of Scientific and Re-search Publications* 2, no. 12 (2012): 1-3.

[6] SparkFun Flex Sensor Documentation.

[7] https://sites.google.com/site/ardunitydoc/getting-started/run-examples/flex-sensor

[8] https://towardsdatascience.com/adversarial-machine-learning-mitigation-adversarial-learning-9ae04133c137