Flex sensor system for reading hand movements using machine learning models

Pratyush Kaware
EXTC Department
Sardar Patel Institute of Technology
Mumbai, India
pratyush.kaware@spit.ac.in

Vedant Kayande
EXTC Department
Sardar Patel Institute of Technology
Mumbai, India
vedant.kayande@spit.ac.in

Aditya Khambete
EXTC Department
Sardar Patel Institute of Technology
Mumbai, India
aditya.khambete@spit.ac.in

***Abstract*—The project was about a making a cost-effective sensor and read hand signals, by attaching the sensor to a finger, so as to classify them based on the degree of bent as well as the joint about which the finger was being bent. This was done by testing with various machine learning algorithms to get the most accurate and consistent classifier. Finally, we found that Support Vector Machine was the best algorithm suited to classify our data, using we were able predict live state of a finger, i.e. the degree of bent and the joints involved. The live voltage values from the sensor were transmitted using a NodeMCU micro-controller which we used to convert them to digital and upload them on a database.**

***Index Terms*—Machine Learning, flex sensor, support vector machine**

I. Introduction

The need for reading hand signals could be immense field of medicine or the chemical industry where it might be hazardous for humans to do on their own. These hand signals can then be used to emulate an artificial hand which would do the work in dangerous environments for us.

There are various ways to read hand signals mainly image processing, in which input is in the form of images, or using some form Electromyography(EMG) sensor that gives voltage readings based on muscle movements. Though each of these methods have their own flaws such as image processing is heavy on the CPU or the processor and EMG sensors are very expensive while also being prone to external noise.

In our method we used a sensor similar to a flex resistor which changes its value of resistance based on the amount the bend on the sensor. We attached this sensor to a finger and obtained the voltage values depending on the bend of the finger. This method can be used to get readings of all the fingers by attaching a sensor to each. We trained a machine learning model to create classifier which gave us the joints which were bent and the degree of rotation. This classifier can then classify the readings quickly unlike the image processing method and is cost-effective as well. Though in our method there is restriction in predicting the curl of a finger only towards and away from the palm direction.

II. Hardware Components

A. Flex Sensor

We made the flex sensor using two pieces of aluminum foil and graphite paper where, the graphite paper was sandwiched between the two pieces making sure they did not touch each other as shown in Fig. 1. The two pieces had the length to cover the three joints of the index finger: Metacarpophalangeal joint, Proximal Interphalangeal joint and the Distal Interphalangeal joint.

Fig. 1. Flex Resistor

When there is no bent on the sensor the value of resistance between the two outer aluminum foils is very high. The flex resistor changes its resistance depending on the bent of all the three layers. As the three layers get close to each other as shown in Fig. 2, the value of the resistance between the two conductors decreases.

Fig. 2. Bent Flex Resistor

If we apply a constant voltage to the flex resistor and connect the other end to an Analog to digital converter we can get changes in the voltage depending on the bent. Although, the bending of a finger is more complex than shown in Fig. 3, some pattern can be found in the readings depending on how much the finger is bent.

Fig. 3. Fabricated Flex Resistor

The above in Fig. 3 is the flex resistor we made. We substituted the graphite paper with a normal paper on which we scribbled on both sides with a pencil.

B. Micro-controller

We have used Node-MCU with ESP8266 WiFi chip (shown in Fig. 4) which also has a 10-bit Analog to Digital Converter(ADC).

Fig. 4. ESP8266 micro-controller

The the flex sensor was provided 3.3V voltage through the pins on the Node-MCU and the other end of the flex resistor was connected to the ADC pin of the micro-controller as shown in Fig. 5.

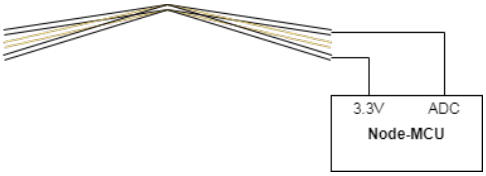


Fig. 5. Connections

The readings were then uploaded to a Real-time Database using Google's Firebase as the database. These values were then used to train the Machine Learning model.

III. About Data

The sensor was attached to the finger as shown in Fig. 6. The sensor needs to be attached tightly to the finger such as unwanted bends don't occur and when the finger is bent, the sensor also bends in a similar fashion. We also made markings as to where each joint is under the sensor so as to get consistent readings while re-attaching the sensor.

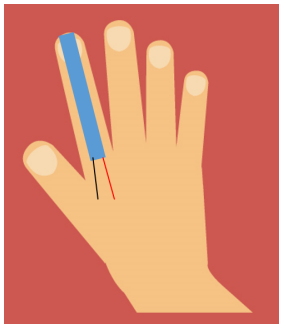
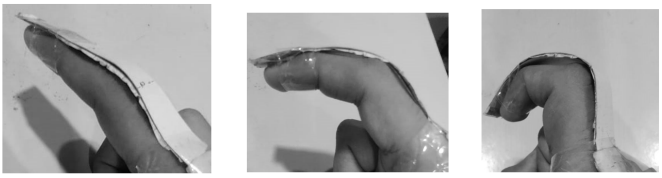


Fig. 6. Connections

The red terminal of the flex sensor needs to be attached to a constant voltage, then the other end could be connected to the ADC pin of the micro-controller. The hold and sample time for the ADC was 2ms which is enough to get an accurate reading. The values were then mapped from 0 to 1024. This was then uploaded to the database using the NodeMCU's Wi-Fi module. The database we used for this project was Google's Firebase real-time database. The value database received a new value every 100ms.

For acquiring the training data, we first kept the finger in a Neutral state (as shown in Fig. 7-A) and took 100 readings of that state and then downloaded it from the database. For the next state, we moved the finger about the middle joint of the finger (as shown in Fig. 7-B) and then took 100 readings of this state. Similarly we did this with both the joints involved (as shown in Fig. 7-C) and took 100 readings of this state.



(A) Neutral position (B) Curl around middle joint (C) Curling around both joints

Fig. 7. Finger Positions

The motion about the uppermost joint near the tip of the finger was not recorded as this motion is highly constrained in an average individual and the change in values in that state was insignificant. A significant change is observed only when the middle joint is already somewhat curled. These readings were then downloaded from the database log in .json format. This data was then labelled for as N, M and F, where N refers to the Neutral readings while in state Fig. 7-A, M refers to readings taken while in states as shown in Fig. 7-B and F refers to readings take in states as shown in Fig. 7-C. All the values are integers i.e. no decimal point values. The data plotted on an axis based on the value of the state. This is shown in Fig. 8 below.

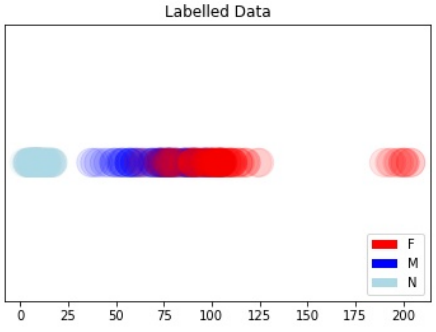


Fig. 8. Data plot

The minimum and maximum values for the data labelled N is 2 and 7. The minimum and maximum values for the data labelled M is 37 and 105. The minimum and maximum values for the data labelled M is 62 and 204. This shows that there exists an overlap in data labelled M and F in the range 62 to 105. The zoomed plot of this subset of data can be seen in Fig. 9.

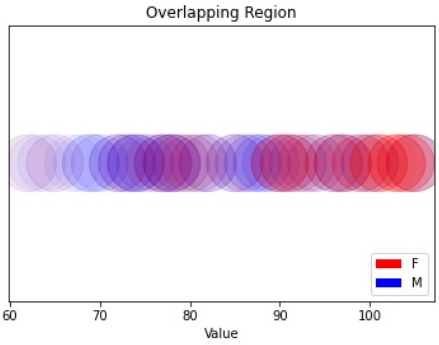


Fig. 9. Overlap plot

The data points are plotted with bigger symbols to emphasize the overlap, though even without them this proves that there is no definite range in which each label lies. This goes on the show us that the system is non-linear. So, to train based on these data we would need models that can classify non-linear systems.

IV. Selection of Model

The Area under the curve in Receiver Operating characteristics for neutral state N was 1 for the models. So, the models were mainly differentiated based on their ROCs for M and F states or middle joint of both joints state. From the ROCs in Fig. 10 we get a rough idea about which model is best suited for our data.

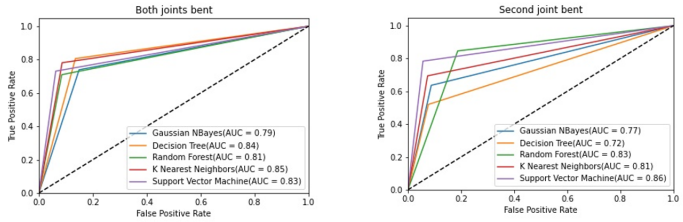


Fig. 10. ROC plots for various models

TABLE I
Analysis

Model Name	Results		
	Accuracy	*Misclassification*	*F1 score*
Random Forest	0.81	0.19	0.83
K Nearest Neighbor	0.84	0.16	0.84
Support Vector Machine	0.85	0.15	0.85

The following models give the best results for our data:

1) Random Forest
2) K-Nearest Neighbours
3) Support Vector Machine

The results by training with 10 fold cross validation with the best 3 models can be seen in Table I.

Our observations while training on these models and why we selected Support Vector Machine as our final model is explained below.

A. Random Forest

Random Forest is an algorithm which uses many decision trees in order to make a decision so as to minimize the number of misclassifications that may occur due to a decision trees not being consistent with new data. We used 10 estimators. This has a accuracy of 0.81 as can be seen in Table I.

Although this was an improvement over the Decision Tree model which had Accuracy of 0.79, still the misclassification rate was high in this. From the confusion matrix shown in Table II, below it can be seen that the misclassification took place between M and F in the overlap area.

TABLE II
Confusion Matrix RF

Predicted Labels	True Labels		
	F	*M*	*N*
F	18	10	0
M	11	25	0
N	0	0	36

The columns labels represent true label and the row labels represent false labels. Moreover because of the small areas of a state, the decision boundary plot shown in Fig. 11 shows that there might be some over-fitting.

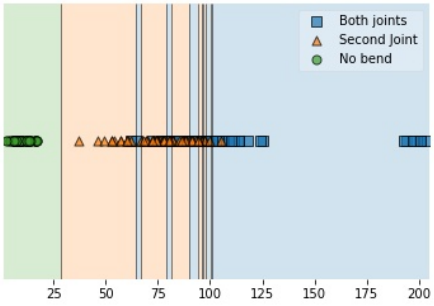


Fig. 11. Decision Boundary Plot obtained by Random Forest

B. K Nearest Neighbors

This model works by looking at the nearest values label to assign the label to a test value. This inherently works with non-linear data. We used 15 neighbors. This has a accuracy of 0.84 as can be seen in Table I. This has better accuracy than RF model. From the confusion matrix shown in Table

III, below it can be seen that the misclassification between M and F is very less than the RF model.

TABLE III
CONFUSION MATRIX KNN

Predicted Labels	True Labels		
	F	***M***	***N***
F	20	11	0
M	5	26	0
N	0	0	28

The columns labels represent true label and the row labels represent false labels. Although the accuracy is high, there is still a small region indicating some over-fitting in the decision boundary plot shown in Fig. 12. The overfitting goes away with increasing the number of neighbors but the accuracy goes down drastically.

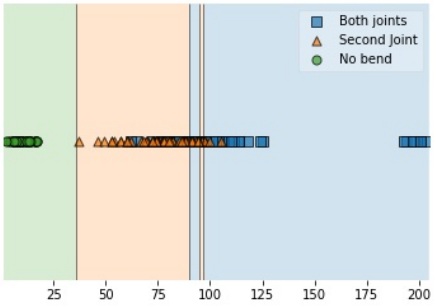


Fig. 12. Decision Boundary Plot obtained by KNN

C. Support Vector Machine

Support vector machine model works by creating hyperplanes that classify the data based on labels. But the linear kernel would not work for our data as there is some overlap. So we used Radial Basis Function kernel which can be used to classify non-linear data. This method works by creating a soft margin by calcultating the misclassifications around that margin. The RBF kernel is similar to a weighted KNN model. This has a accuracy of 0.85 as can be seen in Table I.

From the confusion matrix shown in Table IV, below it can be seen that the misclassification between M and F is very less similar to the KNN model.

TABLE IV
CONFUSION MATRIX SVM

Predicted Labels	True Labels		
	F	***M***	***N***
F	20	8	0
M	9	21	0
N	0	0	32

The columns labels represent true label and the row labels represent false labels. Although the accuracy is similar to the KNN model, this is a better model as their is no overfitting which can be observed from the boundary plot shown in Fig. 13 as it has a clean boundary. This shows that this model will work with novel data as well.

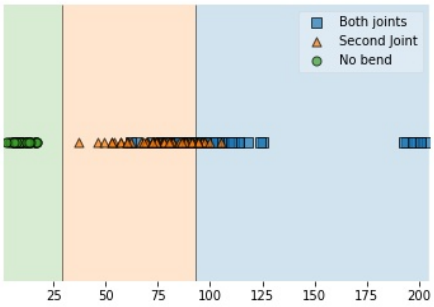


Fig. 13. Decision Boundary Plot obtained by SVM RBF kernel

V. RESULTS

We used Support Vector Machine model to train our data. The decision boundary plot shown in Fig. 13 shows that the states have a clean boundary between each other. This boundary helps us calculate the degree of rotation of the finger by first classifying the value in label and then dividing the value obtained by the range of the label to get the degree of rotation. For eg. the range of M or middle joint is 29 to 93 as obtained from the boundaries using the SVM model. So, for the value of 67 the degree of rotation would be 0.59 of state M.

VI. CONCLUSION

With the growing proliferation of object classification models in the world and the increasing capability of these adversarial attacks to deceive state-of-the-art classification models, the importance of adversarial defense is growing. Our project shows that finger movements can be recorded and predicted based only on the reads from a flex sensor accurately using a Support Vector Machine trained model. For future work, we intend to extend this project in several ways. Also we would try different neural networks as deep learning to see how it compares to our current model.

REFERENCES

[1] Y. Pititeeraphab and M. Sangworasil, "Design and construction of system to control the movement of the robot arm," 2015 8th Biomedical Engineering International Conference (BMEiCON), Pattaya, 2015, pp. 1-4

[2] Yi Tan and Guo-Ji Zhang, "The application of machine learning algorithm in underwriting process," 2005 International Conference on Machine Learning and Cybernetics, Guangzhou, China, 2005, pp. 3523-3527 Vol. 6

[3] H. Yigit, "A weighting approach for KNN classifier," 2013 International Conference on Electronics, Computer and Computation (ICECCO), Ankara, 2013, pp. 228-231

[4] NodeMCU documentation: https://nodemcu.readthedocs.io/en/master/