Predicting the Flexibility of Electric Cables using Robot Tactile Sensing and Push Primitives

Alison Bartsch Achu Wilson AmirPouya Hemmasian Avadh Patel

Abstract

As the prevalence of robot assembly continues to grow, so does the need for robotic methods that can dexterously manipulate flexible cables. However, cables will behave differently depending on their physical properties, changing how they should be modeled and controlled. In this paper, we create a dataset of GelSight images of different cables being manipulated for the purpose of classifying these cables. We perform significant data processing, extracting the optical flow vectors from the GelSight images to capture forces experienced during the manipulation task. We compare the classification ability of numerous shallow machine learning approaches with CNN . We also implement PCA to investigate how these methods change with lower dimensional data. We found our CNN approach had the best performance, with a classification accuracy of 95%*, though the shallow ML approaches also achieved good accuracies greater than* 80%*.*

1. Introduction

Advances in robotic manipulation are helping to automate numerous complicated assembly tasks. Wire harnesses are wire configurations for powering many different electrical devices. Currently, wire harness assembly is a time-intensive and manual process. One of the main tasks in wire harness assembly is routing a wire through brackets, tubes, holes and around mounts. The physical properties of the wire, specifically the flexibility (minimum bending radius) is an important parameter which determines the routing path. Hence, proper categorization/classification of cables must be done to ensure the wires are neither over stressed nor laid loose. We propose a method with which a robot manipulator equipped with a tactile sensing GelSight finger can sense the interaction forces on the cable, when it is pushed against the environment and learn to classify the cable based on its radius. Enhancing the flexibility of future applications to include assembly of bundles of wires of varying diameters.

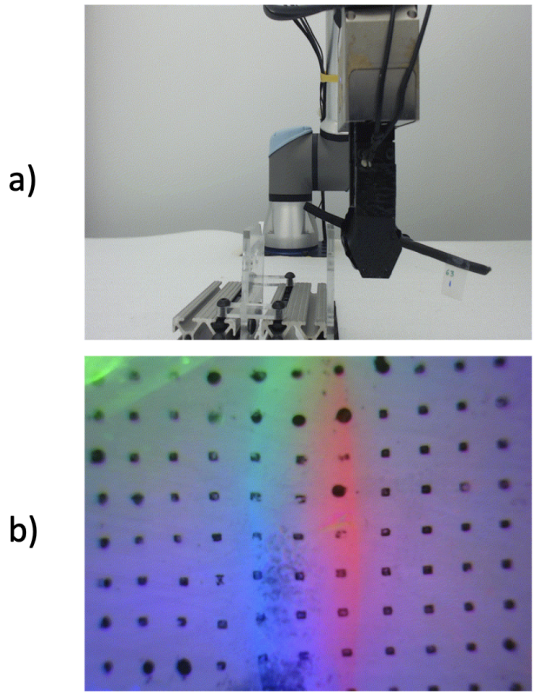

Figure 1. This figure illustrates the basic data collection process for this paper. a) Shows the 5 DOF robot arm equipped with a parallel gripper with a gelsight tactile sensor gripping a wire and pressing it against a wall. b) Shows the corresponding behavior of the gelsight sensor to this action.

2. Related Work

As the sector of autonomous robot assembly has grown, so has the research interest in developing approaches for flexible cable manipulation. Some of the earliest work presented a simple high-speed visual servoing system to manipulate the cables [4]. More recently, [7] proposed a framework for cable shape manipulation with a dual-arm robot, by parameterizing the wire shape by a Fourier series.

Similar to the growing interest in cable manipulation, once [6] presented a new sensor known as the GelSight tac-

tile sensor, there have been numerous creative approaches which highlight different ways these tactile sensors can be used. The GelSight is an image-based tactile sensor, with a piece of flexible elastomer that makes contact with surfaces. When the sensor makes contact with an object, the elastomer deforms and the camera records this distortion (shown in Figure 1b). The GelSight sensor has been used for object identification inside granular media [2], for exploring shape reconstruction with CNNs [1], and as a part of a robot hand design [5].

There have even been works that specifically leverage GelSight sensors for the purpose of cable manipulation. In [3], they present a perception and control framework that uses real-time tactile feedback from a GelSight sensor to manipulate freely moving cables. In this work, they primarily use the GelSight sensor for pose estimation as well as detecting slip forces. They achieved impressive results, with the robot able to lightly grip and slide along the length of the cable. However, this work focuses on how to manipulate these wires, whereas in this project we are interested in classifying the cables we are manipulating. As there are no works directly relating to this, we are collecting our own dataset.

3. Data

For this work, we collected our own dataset which consists of GelSight images and the corresponding label/class of the wire. To gather the samples, the robot picks up the cable, moves to the wall and then presses the wire down against the wall, recording GelSight images during this trajectory. We repeated this 35 times for each of the 3 wires. Resulting in a total dataset of 105 data points. This dataset is quite small due to the time-intensity of collecting real-world data, as well as due to the degradation of the prototype GelSight sensors. As the Gelsight prototype sensors were not designed with robustness in mind, the flexible elastomer as well as the reflective paint surfaces showed degradation on repeated and continuous usage. Due to the limited size of our dataset, the conventional CNN-based image classification approaches were infeasible. Thus, we needed to perform significant data processing, feature engineering and dimensionality reduction techniques to convert the image-based data into a different data type that shallow machine learning approaches could handle. On the other hand, it enabled us to try increasing the shallow machine learning performance through the application of reduced dimensions on the data.

3.1. Data Collection

In order to collect a large amount of data, with minimum human intervention we automated the data collection process. UR5e - a 6 degree of freedom robot manipulator from Universal Robots along with a parallel jaw gripper is used for data collection. The GelSight sensor is attached to one of the fingers of the gripper enabling us to collect the tactile imprints of the grasped object. In addition, a fixture with self resetting capabilities is designed and machined to hold the cables and reset it to a predefined pose after each iteration. The fixture also has a edge on which the cables can be pushed against to collect the tactile data and estimate the flexibility. Figure.1 illustrates the data collection setup that was used, as well as a raw sample of data obtained from the GelSight sensor. The data collection process is as follows:

1. Robot picks up the cable from its resting place.
2. Robot moves cable near to the edge of the fixture.
3. Data capturing of GelSight sensor and end effector pose is initialized.
4. Robot moves slowly downward, pushing the cable against the edge of the fixture.
5. Once the robot reaches the set z axis depth, data capturing is stopped, and the collected data is saved.
6. Robot moves back to the initial position and drops the cable into the wedge shaped resetting area.

These six steps are then repeated for cables of different diameters. During each iteration, the diameter of the corresponding cable is entered manually, from which the class label is generated and added to the dataset.

3.2. Data Processing/Feature Engineering

GelSight sensors can capture rich contact information which consists of high resolution surface geometry as well as the deformation information by tracking the black markers. The amount of marker deformation corresponds to the external contact force on the sensor. For the purpose of this project, we are only focused on these forces measured by the black markers (shown in Figure 1b). Focusing on the markers alone brings about two main advantages, the first one being making the system agnostic to the local geometry/shape/texture which will be very much prominent in the RGB raw image and could lead to over-fitting. The second advantage is the reduction in input feature size. Reducing the input feature size from 640x480x3 pixel values to an array of 11x9 vectors corresponding to the markers could greatly help the learning process time. As the GelSight sensor deforms when it presses against the object, these markers move based on the local forces they experience. The key intuition behind this project is cables of different bending radius or flexibility will produce different forces when pushed against an object.

The raw data collected consists of a time-indexed series of GelSight RGB images of size 640x480 each and robot poses corresponding to a single press motion. Once the raw images are captured, traditional computer vision techniques

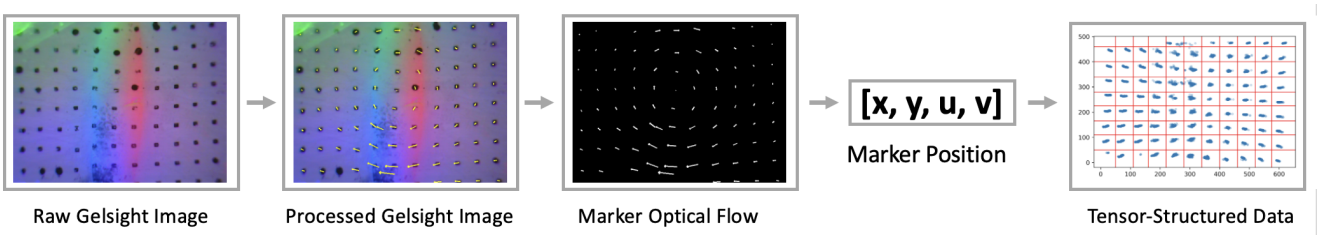

Figure 2. The data processing pipeline. The data is converted from the raw gelsight image to a processed gelsight image with vectors denoting the marker movement. This is then converted to a black and white image of the marker optical flow. This is then converted to an array format of marker position and vector information. Finally this is converted to a tensor data structure.

are used to extract the position and deformation of the black optical markers. The raw RGB image is first converted gray-scale and thresholded to extract the pixels corresponding to the markers. It is then cleaned up using a morphological closing operation with a circular kernel. Contours are then detected on the cleaned up image which are then filtered and sorted depending on their size. Finally the center positions of each markers are estimated by calculating the image moment on the filtered contours. The difference in marker positions during the start of data collection when the cable is not in contact with anything and the marker positions when the cable is fully pressed against the edge is calculated to estimate the marker displacements in x and y axes.

One of the issues with marker tracking at times, some of the markers may fail to be detected. This resulted in inconsistent number of marker vectors ranging from 87 markers to 93 markers. Inconsistent input shapes could be a bottleneck to the subsequent learning models. Markers were calculated on all images in the dataset and it is found that 85 markers were consistently detected in all frames. This the image processing system was modified to return only these robust markers.

Once the marker positions (x,y) and displacements (u,v) are calculated reliably, they can be converted into an 85x4 array. This is the data format that will be used for dimensionality reduction as well as all of the shallow machine learning approaches, except for CNN. To make this data usable for a CNN, we then convert this data into a tensor format, preserving the spatial component of the data without explicitly containing it. Figure 2 visualizes the data processing pipeline.

Even with the data processing, the array format of the data still has over 300 features. This is because there are 85 markers, each with a corresponding x, y, u and v. This is the perfect scenario to explore principal component analysis (PCA), as there is significant opportunity for reducing the feature dimensions. We found that 90% of variance is explained when reducing the features down to eight dimensions, 80% of variance is explained when reducing the features down to five dimensions, and 65% of variance is explained when reducing the features down to two dimensions. The two-dimensional features can be seen in Figure 3, where it is clear that each class is clustered.

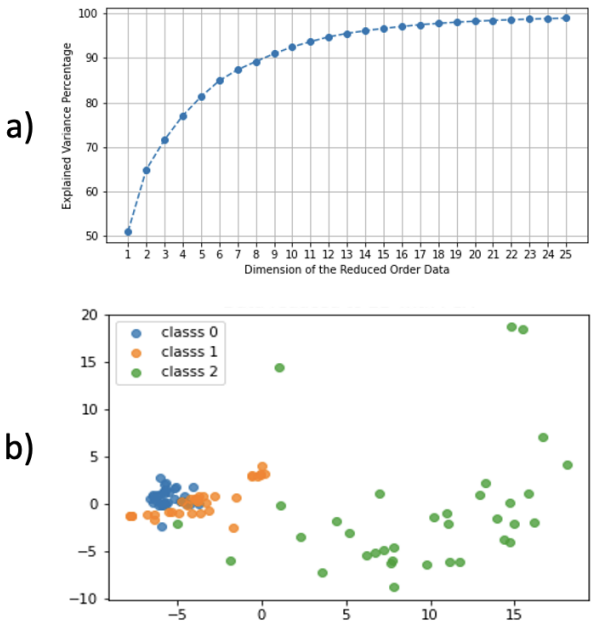

Figure 3. a) Shows how the reduced dimensions with PCA correspond to the percentage of variance explained. b) Shows the data reduced to 2D with PCA, where 65% of the variance is explained. Even still, visually the three wire classes are relatively distinct.

4. Methods

Numerous approaches were explored for classifying the cables with the goal of comparing the performances of them, in this case decision trees, random forest, k nearest neighbors, support vector classification, naive bayes and logistic regression, with simple neural networks, in this case CNN. In each subsection, we will provide a very brief overview of each method, specifically the data it was trained

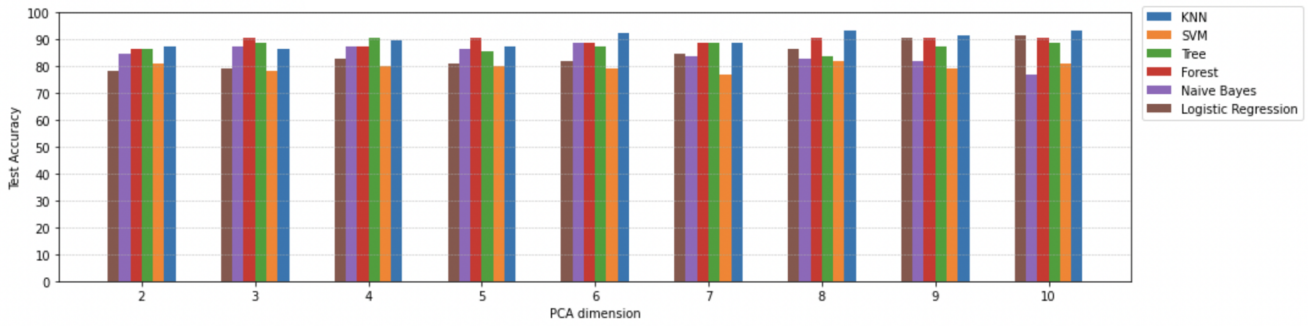

Figure 4. This figure shows a direct comparison of the models performance (that we ran with reduced dimensions of our dataset) on different PCA dimensions.

on, and why we chose to implement this method.

4.1. Decision Tree

Our decision tree model model was trained on the array format of the data. Specifically, we trained the decision tree on the high-dimensional dataset as well as reduced order data with PCA (shown in Figure 4). This method is very prone to overfitting, and we expect to see overfitting with our small dataset.

4.2. Random Forest

Random Forest is one of the most popular machine learning models which is basically an ensemble of decision trees. Because this model consists of many decision trees and classifies the data based on the majority vote of those trees, it is much less prone to overfitting than decision tree. Considering the low number of data points that we have, decision tree is highly prone to overfitting. Using ensemble methods like Random Forests can avoid overfitting in many scenarios, so this algorithm is a reasonable choice to try for our classification task.

4.3. Logistic Regression

We chose Logistic Regression as one of the candidates because of its simplicity and interpretability. We also used the default L2 regularization to avoid overfitting, which is probable due to the size of our dataset.

4.4. MLP

Neural Networks are universal function approximators, but they need large amount of data to train. Since the size of our dataset is not large, we had to avoid overfitting by reducing the complexity of the network, so we used a simple neural network with one hidden layer consisting of 16 hidden units in order to classify the data in low-dimensional spaces. The other hyperparameters of the MLP were the default settings of sklearn, except the `max_iter` parameter which was set to `2000`.

4.5. Naive Bayes

Despite its simplicity and simplifying assumptions, Naive Bayes model have shown very magnificent performance in many classification problems. We observed in the two-dimensional visualization of the dataset that each class indeed forms a cluster in the space with a rather simple shape and distribution. Naive Bayes algorithm assumes a Gaussian distribution for each feature in each class, therefore it makes sense to try this computationally cheap algorithm.

4.6. Support Vector Machine

The Support Vector Machine is considered to be a robust machine learning model for classification because it learns a decision boundary that maximizes the margin between classes. Looking at the two dimensional visualization of the data, we can see that the data clusters do not form complex manifolds, so SVM has a chance of performing well at classifying this dataset at some higher dimension.

4.7. K Nearest Neighbors

The K Nearest Neighbors Algorithm might be able to achieve very outstanding performance if the training dataset is small and is not too high-dimensional. Since the number of data points that we have is only 105, and we reduced the dimensionality of the data with PCA, KNN is definitely worth the try.

4.8. CNN

The use of domain knowledge can be of high importance in developing machine learning models. When dealing with spatial data, where nearby locations and pixels might have meaningful relations, Convolutional Neural Networks are one of the best performing models. These kinds of models revolutionized Machine Learning for images and generally tensor-structured data. Due to the nature of our data which is a two-dimensional distribution of marker move-

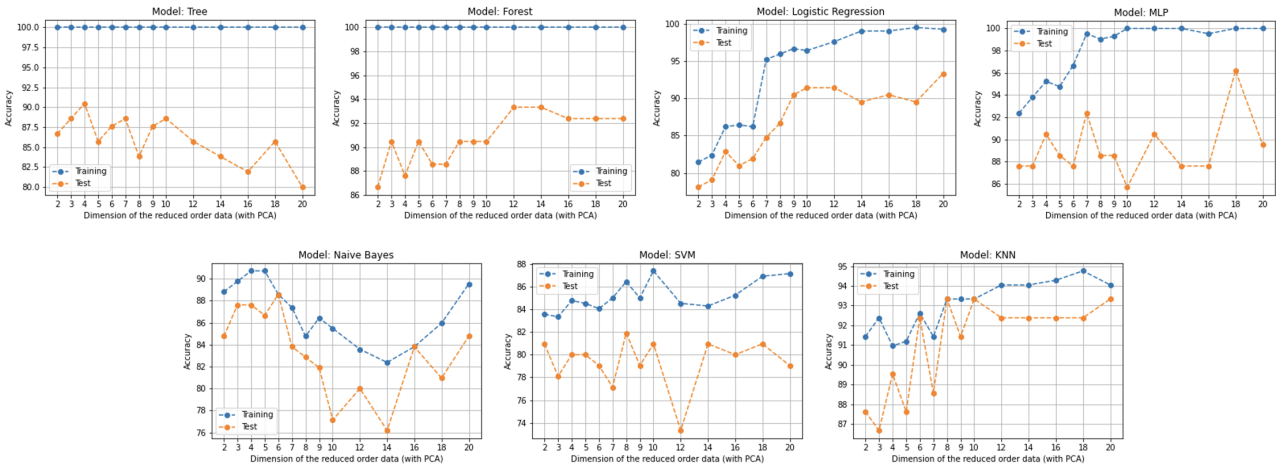

Figure 5. This figure shows how the decision tree, random forest, logistic regression, Naive Bayes, SVM and KNN models' accuracies change when given various reduced order data, ranging from 2 to 20 dimensions. The decision tree, and Naive Bayes models had the best test performances with the lower dimensions of data. Whereas random forest, SVM, and KNN models all had the best test performances with the data between 8 and 12 dimensions. Finally, the logistic regression and MLP performed best at test time with the larger dimensional data.

ments, we chose CNN as one of the promising models to try. We started with a very simple CNN with only two convolutional layers, each with only one filter, and the model still managed to learn the pattern of the data with a very high accuracy. The model begins by applying a 3x3 convolution, followed by a 2x2 max pooling, then a 2x2 convolution, and finally flattening the result and using a dense layer with softmax activation for the final prediction. This CNN consists of only 45 parameters which is very impressive considering the dimensionality of the tensor-shaped data is almost four times this number. Since the trained model highly depends on the initialization of its parameters, we trained this architecture with 30 random initialization and chose the best one based on the test accuracy.

5. Experiments

Each of the six shallow machine learning models was trained on 80% of the data, and tested on the other 20%. For each algorithm, we performed a 5-fold cross-validation and reported the average test accuracy across the five folds for each model. We tried training the models with different number of PCA dimensions for the dataset, ranging from 2 to 20, and reported the average test accuracy. The results can be found in figure 5. We can see that the ML models show different behaviours when we increase the dimension of the data. For all models, the default settings of the hyperparameters in sklearn were chosen.

We observe that the decision tree noticeably overfits to the training data, and we see that the overfitting gets worse with the increase of the number of dimensions. The Random Forest however, avoids this amount of overfitting because it takes into account many decision trees. When the dimensionality increases, the random forest leverages the new information provided by the higher dimensions pretty well and shows an increasing test accuracy, up to 92%. Since the default setting of sklearn does not include any regularization technique for these models, the training accuracy for these models are 100%, meaning that the trees will continue to grow until the training data is fully pure in each leaf.

The logistic Regression model uses L2 Regularization, which reduces overfitting. we can see that the trend for training accuracy and test accuracy is similar, but the gap widens for larger dimensions, indicating more overfitting. The results indicate that the model uses the new information added in the high dimensions to improve its performance and achieving a test accuracy up to 90% with just 10 dimensions.

The Naive Bayes algorithm shows an interesting behavior which is not consistent across different number of dimensions. In the low dimension regime, the performance of the model is at its best, but when we increase the dimensionality to numbers around 10 to 15, the performance gets worse. However, the model begins to improve its performance again by increasing the dimensionality from the range 10-15 to higher dimensions. The reason behind this behavior is that Naive Bayes assumes conditional independence of features within each class, and a gaussian distribution for each feature in a class. The bad performance of this model in the range of 7 to 15 might be due to the fact that the assumptions made by naive bayes do not hold at all,

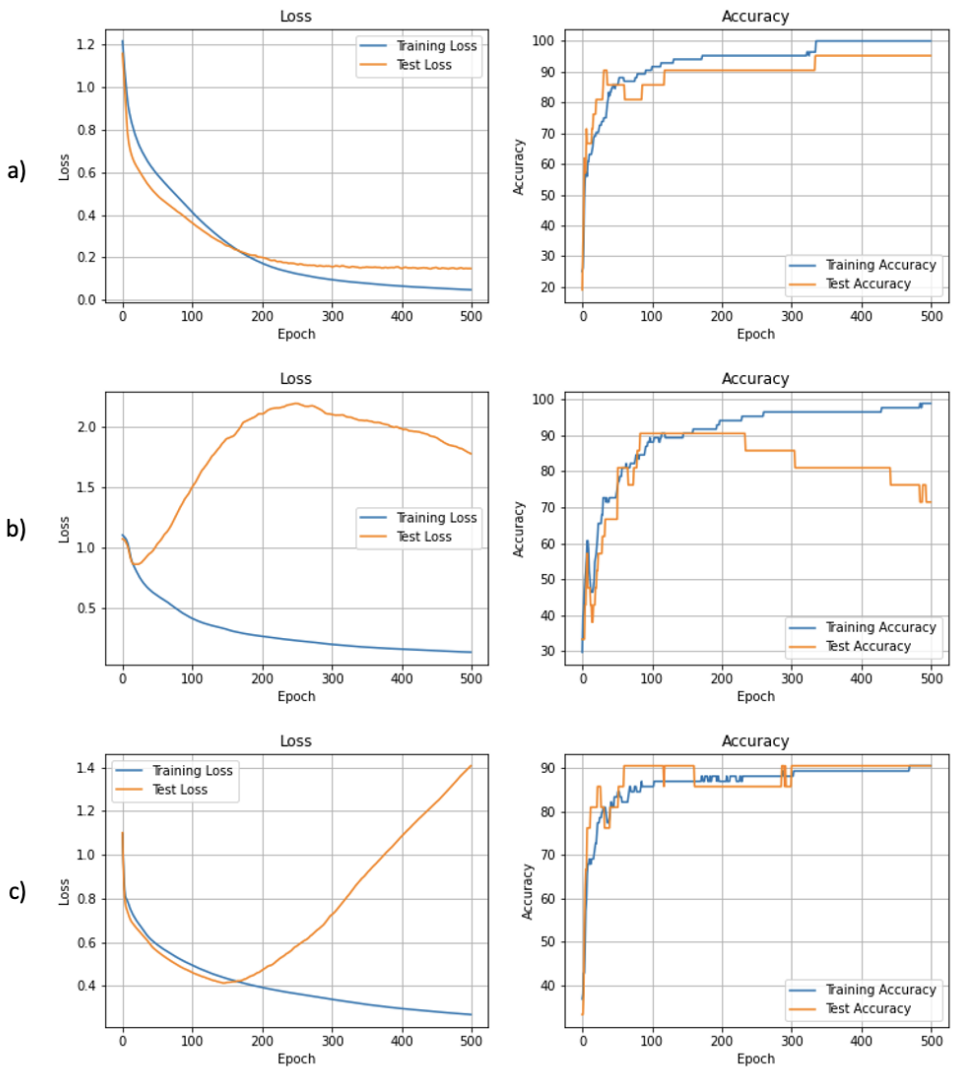

Figure 6. These figures show the loss and accuracy vs epoch for our CNN. a) Is the final model, which reaches a test accuracy of 95%. b) Shows a case of overfitting. c) Is an example of how an outlier in the test set makes the cross-entropy loss diverge, despite high accuracy.

while in other regimes for the dimensionality, the features can be represented with gaussian distribution within each class.

The SVM algorithm has the worst performance on this problem, and it does not seem to show much difference in its performance with changing the dimensionality of the dataset. This means the nature of our data is not compatible with this algorithm, and the patterns cannot be detected as well as the other ML algorithms.

Next, we can see that the performance of the KNN model is pretty impressive considering its simplicity. In the 20 dimensional space, the data points seem to form nice clusters so that the class of each point can be predicted from its neighbors pretty well.

The MLP also shows an overfitting behaviour in high dimensions because the number of parameters and the input dimensions increase while the size of the dataset remain the same low number. However, it can still achiece a test accuracy up to 92% if provided with the right input dimension.

Finally, we have the CNN model which is the most justifiable model because of the structure of our data. The most important takeaways of our experiments with the CNNs will be discussed in this paragraph. First of all, we mentioned in the previous section that the final model highly depends on the initialization of the parameters of the model. If a model starts from a good location in the loss landscape, the final model will be a well performing model. We trained the same architecture 30 times with different parameter ini-

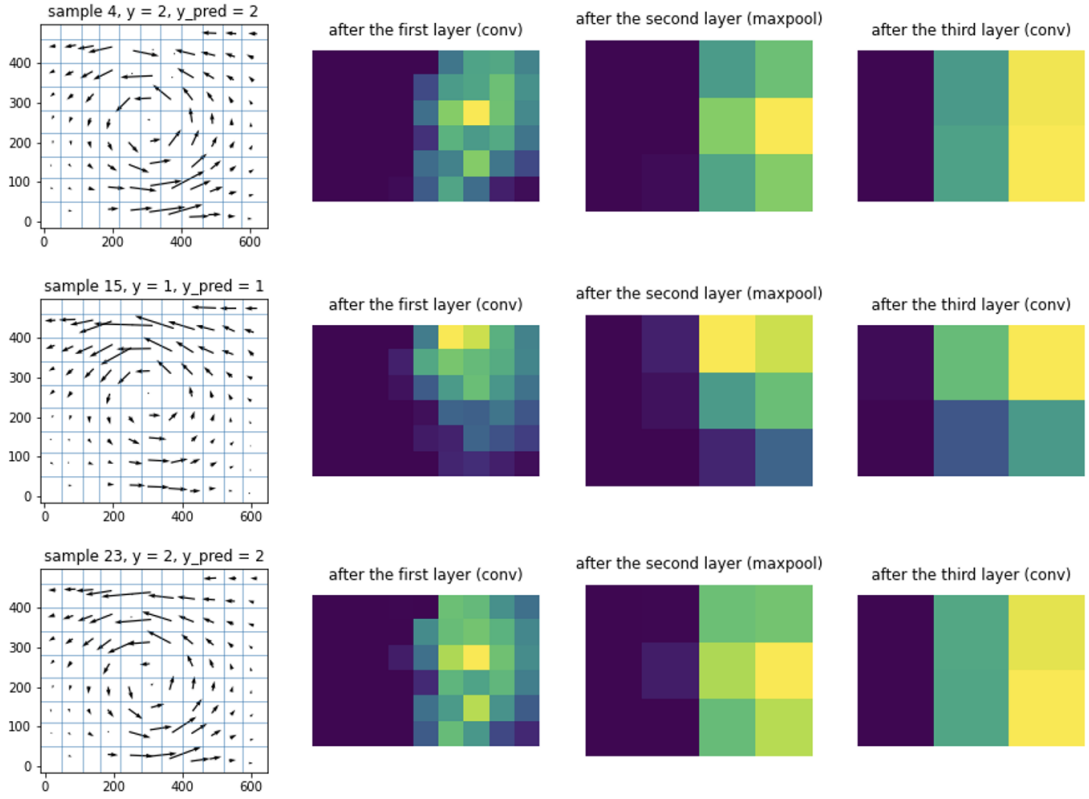

Figure 7. This figure shows the convolution layers of our CNN for different samples. Each row represents a different sample from our dataset. The graph at the left shows the movement of each marker of the GelSight sensor for that sample. The 3 figures on the right show the convolutions layers. Looking at these convolutions, it appears that the CNN is learning where the wire is.

tializations and picked the best performing one based on the test accuracy. We defined and trained our CNN with TensorFlow, and used the Adam optimizer and the cross-entropy loss. Looking at Figure 6, we can see different scenarios that might happen while trying to train a CNN. In some scenarios, the training process might lead the model to a bad local optima which causes overfitting and a bad performance on the test set, as can be seen in Figure 6.b. Since the formula for cross-entropoy loss contains logarithms, values close to zero in the logarithm can make the loss diverge, so outliers can have a negative impact on the loss and the general training. Here in Figure 6.c, there is an outlier in the validation dataset, which makes the test loss diverge, but the test accuracy is not affected as much because it is only one or few outliers. Another interesting observation is that CNNs learn meaningful features. We visualized the output of the layers of our final CNN in Figure 7, and the filter in the first layer seems to learn the area in the sensor which is experiencing the most tension and that is where the cable is being held.

To summarize, the CNN achieves the best performance with 95% test accuracy, and that is due to the fact that this model takes advantange of the positional information of the data, and learns translation-invariant features. However, other ML models also can achieve good performance of around 90% accuracy, but are more prone to overfitting and lose information in the process of dimensionality reduction. CNN on the other hand, includes an automatic feature detection and is seen to learn meaningful features by its parameterized filters in the convolution layers.

6. Conclusion

Through this project, we learned the value of data processing and feature engineering. Though we have a small dataset, with only 105 samples, we were able to achieve good classification accuracy with a number of shallow and deep machine learning methods. We were able to reach this accuracy by transforming the raw GelSight images into simpler data that was easier to handle and learn from.

Though we are able to demonstrate good results with our dataset, the dataset itself is quite small and only includes three different cables. Thus, in order to improve this work, in the future we need to collect significantly more data and

demonstrate stronger, more reliable and generalizable results. In this work, we only focused on cable classification, specifically classifying the cables based on their diameters. However, with more data, we hope to explore regression as well. This way, we could show our model an unseen cable, and correctly identify its properties. The results of this work would be useful for cable manipulation methods. The robot would be able to perform the simple action of pressing the cable against a surface to identify its properties, then by simply modifying parameters in the controller, best perform the task with that given cable. This has the potential to improve robustness of cable manipulation approaches as well as generalizablity to new, unseen cables.

7. Contribution

Achu worked with the hardware setup and data collection and data prepocessing, computing marker optical flow and converting the gelsight images to the array format, as well as initial implementations of kmeans and random forest. For the writeup, Achu worked on the data section. Alison worked with the hardware setup and data collection, as well as initial implementations of knn and SVM. For the writeup, Alison worked on the introduction, literature review, data, the figures and conclusion. AmirPouya implemented the final versions of the shallow ML algorithms as well as the CNN and MLP. In doing so, AmirPouya reformatted the data to the tensor structure to work with the CNN. For the writeup, AmirPouya focused on the methods and experiments sections. Avadh created an initial implementation of an MLP, and kmeans and for the writeup helped with the methods section and data collection.

8. Code

Our code and dataset can be accessed with this Google Drive link: `https://drive.google.com/drive/folders/1mmT6ZaLZNHHHePVQzkeQzmKcWhH2K-fg?usp=sharing`

References

[1] J. Li, S. Dong, and E. H. Adelson. End-to-end pixelwise surface normal estimation with convolutional neural networks and shape reconstruction using gelsight sensor. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1292–1297, 2018.

[2] R. Patel, R. Ouyang, B. Romero, and E. Adelson. Digger finger: Gelsight tactile sensor for object identification inside granular media, 2021.

[3] Y. She, S. Wang, S. Dong, N. Sunil, A. Rodriguez, and E. Adelson. Cable manipulation with a tactile-reactive gripper, 2020.

[4] T. Tamada, Y. Yamakawa, T. Senoo, and M. Ishikawa. High-speed manipulation of cable connector using a high-speed robot hand. In *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1598–1604, 2013.

[5] A. Wilson, S. Wang, B. Romero, and E. H. Adelson. Design of a fully actuated robotic hand with multiple gelsight tactile sensors. *CoRR*, abs/2002.02474, 2020.

[6] W. Yuan, S. Dong, and E. H. Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12), 2017.

[7] J. Zhu, B. Navarro, P. Fraisse, A. Crosnier, and A. Cherubini. Dual-arm robotic manipulation of flexible cables. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 479–484, 2018.