ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 3201
WSCG 2022 Proceedings

# A Physical Device to Help the Visually Impaired Read Money Using AI/Machine Learning in Third World Countries

Nidhi Mathihalli
Saratoga High School
20300 Herriman Ave
USA 95070, Saratoga,
California
nidhi@mathihalli.com

## ABSTRACT

There are over 285 million blind people in the world, with approximately 87% of them living in developing countries. However, in the third world countries, there is currently very little technology to help the visually impaired, especially with financial independence. In this article we present the machine learning algorithms used to develop the device to help visually impaired distinguish between different forms of currency. Using the various currency images, we form a data set that is used to train the transfer learning model. Experimental results show over 94% accuracy with transfer learning model. The device is designed to be portable and hand-held. The device can distinguish between 1, 5, 10, and 20 dollar currency bills. Additionally, the model can work offline. Overall, the device is cost effective, portable, and can be used in the absence of internet connectivity.

## Keywords

Affordable Bill Detector, Effective Money Reader, Computer Vision-based Accurate Algorithms

## 1 INTRODUCTION

Currently, there are at least 285 million people in the world who are legally blind. Of these people, 87% of them live in third world countries. Thus, around 247 million people live in these developing countries. However, there is still very little technology to help them navigate their world.

Age-related vision loss such as with conditions like Macular Degeneration, are on a rising trend in developing countries. The numbers seem to parallel the trend in developed countries such as America. Specifically, their financial independence is restricted, since they are often not able to distinguish between different dollar bill denominations anymore. There is also the possibility of them being a victim of fraud. Cashiers and vendors may take advantage of their loss of sight and overcharge them by not giving them the correct change back.

Additionally, sudden loss of eyesight is proven to make one less confident and can result in the development of mental health issues, such as depression and anxiety. With cases on the rise, it is even more important to help the visually impaired become more financially independent. Online shopping is a far reach in third world countries where internet is a luxury. Everyday routine is drastically different from the status quo because of visual impairment.

Visually impaired people have trouble completing everyday tasks: reading the news, going to school/work, cooking meals, or making purchases at various stores. This solution assists the visually impaired by reading dollar bills, improving the shopping experience, in turn boosting social confidence with peers/friends.

## 2 BACKGROUND AND RELATED WORKS

Our initial research was conducted at the Shree Ramana Maharishi Academy for the Blind, in Bangalore, India. This school helps visually impaired orphans and kids from very poor families by giving them a free education. Meeting groups of students on a regular cadence, helped decipher what would be a practical design for the product.

The survey results from the students, who were aware of the purposes of these questions and responses, revealed that 100% wanted to have more financial capabilities through the use of this proposed device, and that 75% would rather have a physical device than a smartphone app. Most students expressed that a device such as a snap on device on sunglasses or a device hung by a lanyard over the neck would be a preferred choice. A physical device was clearly the choice, and there are two reasons for this - smartphones are expensive and there are challenges on using a device with a small screen for the visually impaired.

Over the course of a month, we gathered information on what types of technology should be considered keeping in mind easy of use and cost effectiveness. Researching on available current solutions revealed a few mobile

apps, which as stated previously, are not effective since their small screens make them very inaccessible by the visually impaired. Other solutions include the iBill, and the OrCam, however, these devices cost over $115. The iBill is a handheld device that identifies the denomination of the bill by taking a picture of any corner of the bill [1]. The OrCam is another handheld identifier, used for reading text, identifying products, etc [2].

According to a 2016 report by the National Sample Survey Office by the Government of India [3], the rural farmer's monthly salary in the third world country India, is 6,426 rupees per month, or $86.67. With these salaries, affordability for such physical devices costing over $120 is out of question.

Keeping all the above in mind, it is evident that the visually impaired in third world countries prefer an easy-to-use physical money detection device that is accurate yet cheap. Research on cost effective hardware materials led to a conclusion that the raspberry pi, (a mini computer priced at $20), an ESP32 Cam Module (priced at $4), and a Battery Breakout board (priced at $0.5) would be an ideal combination. This brings the total price to $25, which is a 79.16% decrease in price compared to the current cheapest device.

In conclusion, the main goal was to create the money reader in an affordable, working, and effective way, with the key metrics being that the validity accuracy of the bill is over 90%, the time taken to predict the denomination is under 10 seconds, and the total cost of the device is under $30.

## 3 MATERIALS AND METHODS

Using the previously established goal for this project, we split the approach into multiple sections. Each section illustrates a different method we adopted to solve certain aspects dealt with creating the device. The sections either document the approaches we took to create/improve the machine learning model or build the hardware component. All code for this project can be found in this project's GitHub link [4].

### 3.1 Machine Learning

Please use a 10-point Times Roman font, or other Roman font with serifs, as close as possible in appearance to Times Roman in which these guidelines have been set. The goal is to have a 10-point text, as you see here. Please use sans-serif or non-proportional fonts only for special purposes, such as distinguishing source code text. If Times Roman is not available, try the font named Computer Modern Roman. On a Macintosh, use the font named Times. Right margins should be justified, not ragged.

Before creating each of our approaches, we decided on using machine learning as the identifier. Machine learning is an application of Artificial Intelligence that provides computers with the ability to automatically learn and improve from experience without having to program every possible case. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves. In Machine Learning, the primary aim is to allow the computers to learn automatically without human intervention or assistance and adjust actions accordingly. Machine Learning programs complete their tasks using algorithms. Algorithms are sets of rules that the computer follows in calculating operations. In machine learning, there are four types of algorithms: supervised, semi-supervised, reinforced training, and unsupervised.[5]

Supervised machine learning algorithms use past images to predict what each new image is. Using a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly. Supervised machine learning uses two main processes: Classification and Regression. Classification is the process of predicting what a group of images is representing. Regression is the measure of the relation between the values of one variable group to corresponding values of another variable group. Using these two procedures, computers use supervised machine learning to classify images.[5]

In Unsupervised Machine Learning, one trains images without any labels. People use unsupervised machine learning techniques for clustering, detecting anomalies, association mining, and for creating latent variable models. In clustering, a machine splits the images into groups, although they might be incorrect. Anomaly detection is used to figure out otherwise unrecognizable patterns or details. Therefore this type of machine learning is used a lot in finding out if a fraudulent transaction has occurred, or if there is an outlier among some data points. Association mining is when machine groups together certain objects that are similar or of the same use. This type of machine learning is used for retail marketing or on online shopping Websites in order to show users what is of a similar type as to the item the customer is currently looking at. Finally, unsupervised machine learning is used in creating latent variable models. Latent variable models are machine learning models that relate observable details to latent, or hidden, details.[5]

Semi-supervised machine learning combines both unsupervised machine learning as well as supervised machine learning. These datasets train using both labeled and unlabeled images, usually more unlabeled images. Programmers use this kind of machine learning in order to save time on labeling images. Furthermore, labeling too many images can impose a human bias on the machine. Therefore, by using semi-supervised ma-

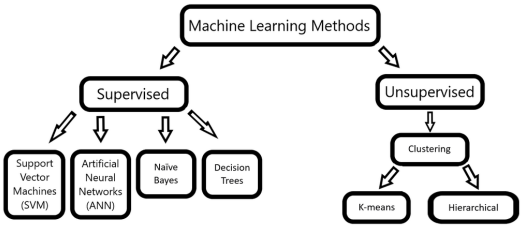


Figure 1: This image shows us the pipeline used for supervised versus unsupervised learning. Both of these algorithms were used throughout our report. [6]

chine learning, one saves time, doesnât create a human inclination, and makes sure that the images are being labeled properly.[5]

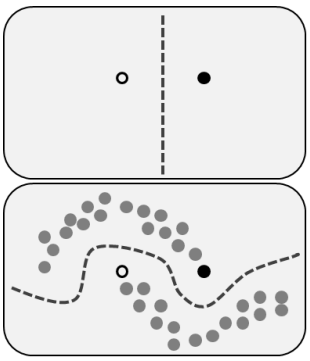

Figure 2: This figure shows us how semi-Supervised learning works, as there is both labeled and unlabeled data that helps the algorithm accurately sort the given data.[7]

Reinforcement learning is when a machine learns from trial and error. In this form of machine learning, the machine gets feedback from its actions and experiences. This sort of machine learning can be analogized to a game. In this game, the creator gives the model no hints as to how to solve the game. The model has to 1) figure out how to play the game, and 2) sort each image correctly, in order to maximize the score of the game. Since reinforcement learning is a new idea, it is currently not being used as an application today. However, its creators are planning to use it in the future for assisting humans, as it is an AGI, artificial general intelligence, or as a method in figuring out the consequences of different strategies.[5]

## 3.2 Google Vision API

For our first machine learning attempt, we used the Google Vision's Auto ML API. The initial dataset had 450 images of various denominations. After training the model, we uploaded a few images in order to test if those images would be evaluated correctly. When we used the model to predict the images on Auto ML, the model predicted the images accurately. The next step was to create a programmatic version for prediction using AutoML. We programmed the software, building

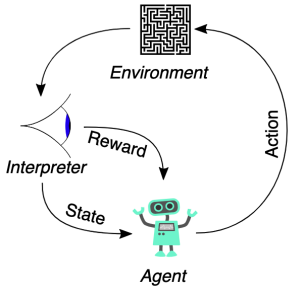


Figure 3: This figure shows us how reinforcement learning is able to analyze past results in order to learn from trial and error.[8]

off of the code outlined by AutoML API. To complete the code, we used the Auto ML modelâs id, the project id, and the images to create the final program. Additionally, we added the ability to programmatically capture the images of the bills.

## 3.3 Base Model

The base model is a native Tensorflow model. We used Tensorflow's "Convolutional Neural Network" Google Colab starter code [9]. This starter code looks at flower classification, and uses the CIFAR-10 data set. In order to use this starter code, we had to convert the images from the dataset into the format that the CIFAR-10 data set [10] uses. This was done by adding each image to one of two binary file, each respectively containing the training and testing data, and added information on the pixels and their colors. We then loaded the data into x_train, y_train, x_test, and y_test. We defined x_train and x_test as the image data, and y_train and y_test as the image labels.

### *3.3.1 Data Augmentation*

Data augmentation is a method of adding more training data to the model by slightly altering the images. This can include rotating, reflecting, and cropping the image. Our goal of using data augmentation was for the addition of data for our model through the different image variations. By adding more data, the model was able to identify and classify the image under different image conditions.

## 3.4 Binary Classification

Binary classification is used to identify whether an image represents a certain set object or not.

Using this type of model, we were be able to predict whether a bill is a 1 dollar bill or not. Similarly, we can do the same for figuring out whether it is a 5 dollar bill or not, 10 dollar bill or not, and if it is a 20 dollar bill or not and so on. The result is binary - either a yes or a no. Binary classification took less necessary images and layers per model, but leads to better results, since the output is one of 2 values rather than 4 ($1, $5, $10, and $20).

## 3.5 Filtering Data

Filtering Data was another approach we experimented with. We applied another layer of abstraction by applying different filters on the training data images before having the model train on them. Since dollar bills have distinct edges, we tried two cases, using the Box Filter [11], Contrast Filter, Sobel Filter, and Canny Filter. Figure 1 shows the original picture:

Figure 4: This figure is the original dollar bill upon which various filters were applied and contrasted in order to see if a specific filtered image gave a significant increase in accuracy

The Box Filter [11] is used to make images more blurry. It is a square array such that each element is $\frac{1}{numberoftotalelements}$. For a 5-by-5 Box Filter [11] array, each element will be $\frac{1}{25}$. Figure 2 shows the what Figure 1 looks like after the Box filter is applied to it:

Figure 5: The Box Filter [11] made the image more blurry, making it easier to find large features

As shown in the Figure 2, this image is blurrier than the original one. Box Filters [11] are often used to point out big features. The other filter we experimented with was the Contrast Filter. Doing the opposite of the Box Filter, this filter is used to sharpen certain smaller details in the image. As the name suggests, Contrast Filter makes the image darker/lighter in certain parts to make smaller details contrast more. Figure 3 shows what the dollar bill looks like after the Contrast filter is applied to it:

Figure 6: Contrast Filter slightly altered the image, but not very visibly

The third kind of filter is the Sobel Filter [12]. This filter is used to detect lines in an image. The Sobel Filter can be applied both vertically and horizontally. This brings out the lines in an image. Since lines are a big part of the features in dollar bills, we hoped that this would bring out the details that would help the model classify the denominations. Below are the pictures after applying the Sobel Filter. Figure 4 uses the Sobel Filter from the matrix we created while Figure 5 uses a Sobel Filter taken from the Skimage Library [12]:

Figure 7: Sobel Filter used from matrix gave a textured feel to the image

Figure 8: Sobel Filter used from Skimage Library gave similar results to that of the Canny Filter

The final filter we experimented with was the Canny Filter [13]. The Canny Filter [13] is another edge detection filter, which identifies a set of edges depending on a sigma value. The higher the sigma, the lower the resolution, make the features detected the larger ones, and vice versa. Figure 6 shows what the dollar bill looks like after the Canny Filter [13] is applied to it.

Figure 9: Canny Filter darkened the image, making the edges white thus greatly contrasting the two

## 3.6 KNN Feature Detection

The 5th algorithm we experimented with was the KNN algorithm [14]. The KNN algorithm [14] when applied to computer vision is was very useful when trying to find the most apparent features in the dollar bill.

The KNN algorithm [14] takes key points and values and finds where the key features are in each image and can easily translate one image to another. As shown in Figure 7, this is very useful in terms of dollar bill detection. This is because it can easily identify where these details are even if they're in different spots of the photo. For example if it sees an interesting curve at the top of a $1 bill, it will notice that same curve even if the dollar bill is flipped upside down and the curve is in a different location. Thus, KNN [14] enabled the classification of objects without having to feed the model augmented data. with few images, it is easy to identify if and how much an image is similar to another.

## 3.7 Transfer Learning

The final method of experimentation used transfer learning. Transfer learning is a broad field in machine

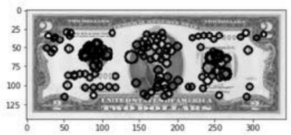

Figure 10: K-means feature detection using key points

learning and it is often used for all object classifications. This is because most transfer learning models work off of previously trained data heavy but very accurate models such as the Google Inception model [15], Microsoft ResNet model [16] and the MobileNet model [17]. Thus many applications will use these models to make an all object detection system. Since this was focused on dollar bills, we had to take off the last couple layers then feed in our own data and other specific information relevant to our application, but transfer learning helped create a really good model. Microsoft ResNet [16] is one of the most commonly used base models for transfer learning. Using ResNet [16] as the base, we removed the last couple of layers, added a Softmax layer with 4 labels, and trained the model. Since it was able to solely focus on dollar bills instead of other objects, this saved time, reducing the time it would normally take a ResNet model [16] to identify a dollar bill or any object.

## 3.8 Hardware Design

The initial design consisted of a Raspberry Pi and Pi Camera. The models were transferred to the Pi to see how much time it would take to recognize a picture that was taken with the Pi camera. The Raspberry Pi camera is very useful when working with the Raspberry Pi since a lot of the libraries that the Raspberry Pi camera uses are pre-installed with the Debian OS and hence this did not involve additional effort to install third party dependency libraries to process the picture or live stream.

However, we hit some roadblocks. We were originally using a Raspberry Pi 3. The downside of using transfer learning is that you have to import some libraries that are crucial to transfer learning. This included the Keras library and the Tensorflow library. These libraries take up a lot of memory. Additionally, these libraries cannot be installed in a 32-bit operating system.

Currently there are no known OS systems for the Raspberry Pi 3 that supports machine learning for the Raspberry Pi. However there is a beta version of an 64-bit operating system for the Raspberry Pi 4. This operating system did work, but because the OS was in its beta version, the libraries that the Raspberry Pi camera needed were not available. Thus we had to look for an alternate camera that was able to easily take pictures and send them to the Raspberry Pi.

## 3.9 WiFi Camera

The prognosis was that the Raspberry Pi unit will be a separate handheld device and will be a separate system from the camera itself. This was due to the fact that we didn't want something bulky on the camera, since it is meant to be wearable. Thus, we decided to make the camera and the minicomputer into separate systems.

The ESP32 camera is a Wi-Fi camera that is a part of the ESP32 modules. It is very useful for taking photos quickly, taking live streams and is often used with machine learning and AI projects. The ESP32 camera, however, works on Wi-Fi. Since the Raspberry Pi 4 can be made into an access point, we decided to make the minicomputer into an access point for the ESP32 camera. Thus, the final design comprised of the Raspberry Pi 4 and the ESP32. This allowed the ESP32 easily connect to the Wi-Fi on the Raspberry Pi and is able to take a picture then send it to the Raspberry Pi.

# 4 FIGURES/CAPTIONS

Place Tables/Figures/Images in text as close to the reference as possible (see Fig.**??**). It may extend across both columns to a maximum width of 16 cm (6.3"). Captions should be Times New Roman 10-points. They should be numbered (e.g., "Table 1" or "Figure 2"), please note that the word for Table and Figure are spelled out. Figure's and Table's captions should be centered beneath the image, picture or a table.

# 5 EXPERIMENTAL METHODOLOGY

While working with of our approaches, we needed to created a consistent training and testing plan in order to best compare the different approaches. To do so, we have noted below the experimental mythology that was used with each approach. We have defined below the 2 data sets that we created and defined our evaluation metrics.

## 5.1 Data set

In order to create accurate machine learning models, we created two data sets, each with varying levels of representative and accurate data.

### *5.1.1 Data set 1*

In order to create the base model, named "Data set 1", we first had to collect the necessary data. Unlike many other classification problems, there were few data sets that had pictures of dollar bills and their labels. Thus, we decided to create our own data set. For the first data set created, we took pictures on our own. Asking others to take a couple pictures from their different points of view as well, we ended up with over 100 pictures for each label. Although this data set worked relatively well, we decided to add more data, since many of the

| Approach Type | Accuracy Percent |
|---|---|
| Google Vision API | 95% |
| Base Model | 63.64% |
| Binary Classification | 80-85% |
| Filtering Data | 66-67% |
| KNN Feature Detection | 95+% |
| Transfer Learning | 94% |

Table 1: All results from the various models in a tabulated format

pictures were taking at similar angles and with similar backgrounds, which could severely bias the model. A sample for this dataset is in this project's GitHub link [4].

#### *5.1.2 Data set 2*

For the second data set, named "Data set 2" (which was created after the "Binary Classification" [18] step), we decided to add more data by data scrapping images off of the Internet using PyPi's "google_images_download" API [19]. Through this, we were able to get a data set with over 200 images. Additionally, the pictures scrapped from online resources were able to provide a contrast to the hand-taken pictures in terms of quality, background light, etc. Thus, the accuracy measures greatly increased when using this data set versus the previous one.

### 5.2 Evaluation Metrics

To measure how accurate the data set is, we used the following metric to assess the performance of the model: Accuracy. The accuracy of a model measures the amount of true positives and true negatives over all data (consisting of all positives and negatives). For example, if a one dollar bill has been predicted to have 8 true positives, 2 false negatives, and 3 false positives, and 12 true negatives, the accuracy would be 80%. The accuracies for all approaches can be seen in Table 1, with further discussions on these results in the following explanations.

## 6 RESULTS AND DISCUSSION

Using the previously stated experimental methodology, we tested each stated approach and have listed the results derived. We used these results to discuss which approaches should be implemented in the final design.

### 6.1 Google Vision API

The results of the first model were not promising. We first tried using "Data set 1" as our training and testing data. Although the testing measures were not bad, at 81.818% recall and 88.235% precision, the validity measures were still not acceptable. When testing this model against 20 images, only 7 images were recognized correctly, while the remaining 13 are recognized incorrectly.

Next, we tried using "Data set 2". The results from this second model were better. Using the 20 test set pictures, this model had 19 images were recognized correctly, while only 1 was recognized incorrectly. Additionally, the recall and precision from this model were an improvement from the last model, as the recall was 97.895% and the precision was 96.875%, and both measurements of accuracy crossed 95%.

However, since working with AutoML requires Internet connectivity, if the user is not connected to the Internet, the money reader would stop working. Thus, we decided to abandon this approach since the device has to be connected to the internet in order to work or have its own public Wi-Fi.

### 6.2 Base Model

For this model, we trained the data on 2 Convolution 2D layers, and 3 Dense layers. We used data set "Data set 2" for our training and testing data. We also added a Softmax activation at the end. After training the model on 10 epochs, we got an accuracy of 63.64%. This was not acceptable for our purposes, so we had to find different ways to improve the model.

The data augmentation increased the accuracy, but not by enough. By doing this, we were able to increase the accuracy by 1-2%. This was mainly because there was not a lot of variety in the data, which led to the amount of data being increased, but the type of data staying relatively the same. Thus, the accuracy of the model barely increased.

However, the time that it took the Raspberry Pi to predict greatly increased to over a minute. Since the goal was to have a fast classification, specifically being that the Pi had predict the model within 10 seconds, this was not feasible.

Additionally, another source of error could be that the model was not specifically trained for the bill detection; rather, it was a generic model with a set type and number of layers.

### 6.3 Binary Classification

For the Binary Classification, we created four different models, each of which gave a binary response as to whether or not it was a certain type of denomination. "Data set 2" was used for the training and testing data, although it was formatted differently so to suit the different models. These 4 models had accuracy between 80-85%, which was an increase to the previous model. Additionally, by creating four threads and running them simultaneously, it would not take a lot of time to predict.

Although this worked initially, we were not able to transfer all 4 models to the Raspberry Pi. This was due to memory issues. Furthermore, when tested on a computer with enough computational power, the models would occasionally identify a certain dollar bill as both a 1 dollar bill and a 20 dollar bill. In these cases, it was very hard to figure out what the correct denomination was.

### 6.4 Filtering Data

We used “Data set 2" for our training and testing data. Although the filtered data was useful to the human eye since it was able to find many distinct features, it increased the accuracy by merely 2-3%. The main reason behind this was because many of the filters that we applied identified lines and features that were part of 2+ types of dollar bills rather than individual ones. Thus, although it was able to clearly identify where the dollar bill was in the photo, it had a hard time distinguishing between the types of dollar bills.

Additionally, if we were to feed this data into the model, we would have to apply these filters to the images while predicting. This could take a long time, especially with the Sobel and Canny filters, and since we want to reduce the amount of time taken to predict, this would not be feasible.

### 6.5 KNN Feature Detection

We used “Data set 2" for our training and testing data. Using KNN yielded the best results, with accuracies of over 95%. However, finding the features and comparing them takes a long time, often taking over 5 minutes per classification. Since the goal was to have the model predict the denomination under 10 seconds, this algorithm could not be used as well.

The source of error for this could be that a lot of the features that were recognized from the KNN feature detection were features that were common to all of the dollar bills. For example, it noticed the corners of each bill along with the words “Federal Reserve Bank." Although, this level of detail was useful, it also increased the time taken for determining the denomination.

### 6.6 Transfer Learning

We used “Data set 2" for our training and testing data. This model had an accuracy around 94% and this was only with three epochs. We decided against a higher epoch count because the more epochs we had would result in a model that was easily determine able to recognize its training and testing data, but would fail in the validation to data due to over fitting. Since ‘Data set 2" had only 100 images for each denomination, the amount of time for classification was greatly reduced. Currently, the model is able to perform in under 10 seconds to identify the dollar bill which is a vast contrast to the time the previous approaches offered. Due to the high accuracy and faster response time, this model was used for while testing the prototype.

### 6.7 Final Model and Results

We decided to use the transfer learning model. Through that, we were able to get a high accuracy prediction with little time taken. When testing 20 images taken, the model was able to get 100% accuracy. Additionally, the device did not need the use of Internet for it to work.

### 6.8 WiFi Camera

We decided to stick with the final product consisting of the Raspberry Pi and the ESP32 and a battery breakout board. First the ESP32 takes a picture and sends it to the Raspberry Pi. The Raspberry Pi then uses the transfer learning model, to predict what the denomination of the dollar bill is. After the prediction results are communicated through headphones. It is a very quick process, and due to the affordability of the materials, can be bought and used by anyone. Additionally, the accuracy of the model is over 94% and runs under the 10 second limitation , allowing users to quickly identify the denomination of the bill.

## 7 CONCLUSION

Overall, this application for determining the denomination of dollar bills is has high accuracy and is of need to the visually impaired, specifically in third world countries. However, there are some limitations. Due to the 94% accuracy, there is a chance that the wrong denomination could be identified. However, upon further analysis, we uncovered that this was only a problem in dark lighting conditions. However, this problem was mitigated when we programmed the transfer learning model to use colored images, the model was able to identify the portion of dollar bills we ran in the low light conditions correctly.

The next step is to to improve the accuracy of readings, especially at different angles. Additionally, we are working with the Shree Maharishi Academy for the Blind ( Bangalore, India) in order to get 10 of these devices to them. Effort is also underway to make this device available on APH ( American Print House) - a portal for products for the visually impaired.

We also plan to extend the use of this application to assist blind students in reading. Since the product can indeed read numbers on price tags and bills, we can also expand the use into reading words. This will help young, elementary school students to participate in class reading activities to build relationships with peers and teachers. By extending the use of reading, visually impaired adults will be able to read signs and other papers.

## 8 ACKNOWLEDGMENTS

Our thanks to Professor Anagha Kulkarni at UCSF and Makers Lab's Scott Drapeau for their mentorship on this project. We would also like to thank the Shree Ramana Maharishi Academy for letting us conduct our initial research there.

## 9 REFERENCES

[1] Research, O. iBill US Bank Note Reader. http://www.orbitresearch.com/product/ibill-talkingbanknote-identifier/.

[2] OrCam OrCam Read. https://www.orcam.com/en/

[3] Office, N. S. S. Income, expenditure, productive assets and indebtedness of agricultural households in india.

[4] Mathihalli, N. Github link for money reader project., https://github.com/nidhimath/moneyReader.

[5] Brownlee, J. 14 Different Types of Learning in Machine Learning. https://machinelearningmastery.com/typesof-learning-in-machine-learning/.

[6] CreightonMA, SarkarSaha Figure1A.png. https://upload.wikimedia.org/wikipedia/commons/5/52/Sarkar%26Saha_Figure1A.png

[7] Techerin, Example of unlabeled data in semisupervised learning.png

[8] Megajuice Reinforcement learning diagram.svg.

[9] authors, T. Convolutional neural network (cnn)., https://colab.research.google.com/github/ tensorflow/docs/blob/master/site/en/tutorials/ images/cnn.ipynb.

[10] Alex Krizhevsky, V. N. and Hinton, G. The cifar-10 dataset., https://www.cs.toronto.edu/kriz/cifar.html.

[11] Bernardo Rodrigues Pires, J. M. F. M., Karanhaar Singh Approximating image filters with box filters.

[12] Image, S. Scikit Filters Library. https://scikitimage.org/docs/stable/api/skimage filtershtml.

[13] OpenCV Canny Edge Detection. https://docs.opencv.org/3.4/da/d22/tutorial py cannyhtml.

[14] Wikipedia k-nearest neighbors algorithm., https://en.wikipedia.org/wiki/Knearest neighbors algorithm.

[15] Google Advanced Guide to Inception v3 on Cloud TPU. https://cloud.google.com/tpu/docs/ inceptionv3advanced.

[16] Microsoft ResNet. https://docs.microsoft.com/enus/azure/machinelearning /componentreference/resnet.

[17] Keras MobileNet and MobileNetV2. https://keras.io/api/applications/mobilenet/.

[18] Wikipedia Binary classification. , https://en.wikipedia.org/wiki/Binary classification.

[19] PyPi google images download 2.8.0. https://pypi.org/project/google images download/.