May 2020

Automated Digit Recognition on Sound Pressure Level Meters Based on Deep Learning

Che-Wei Tung
University of Wisconsin-Milwaukee

AUTOMATED DIGIT RECOGNITION ON SOUND PRESSURE LEVEL METERS BASED ON DEEP LEARNING

by

Che-Wei Tung

A Thesis Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in Engineering

at

The University of Wisconsin-Milwaukee

May 2020

ABSTRACT

AUTOMATED DIGIT RECOGNITION ON SOUND PRESSURE LEVEL METERS BASED ON DEEP LEARNING

by

Che-Wei Tung

The University of Wisconsin-Milwaukee,2020
Under the Supervision of Professor Yi Hu

Sound pressure level (SPL) meter is one of the useful devices used for measuring the sound level pressure. The measurement device displays the SPL value in decibels (dB) on a standard LCD screen (no backlight). We could base on the digit number shown on the LCD screen to do some adjustments or evaluations. Thus, SPL has been widely used in several fields to quantify different noise, such as industrial, environmental and aircraft noise. However, in my basic knowledge, there is no previous study used machine learning to auto-recognize the digit on the SPL meter. This thesis presents a novel system that recognizes the digit number on the SPL meter automatically.

In this thesis, we present a novel approach to preprocess the image of SPL meter. This approach could help us to reduce the noise and amplify the number. Then, we train two machine learning models to auto-recognize the multi-digit on the SPL meter. In our experiment result, it could be efficient to detect the SPL meter under high accuracy. There are two main claims to our thesis. First, this is the original research that utilized the ML to auto-recognize the SPL meter. Second, we are the only researchers to set up the SPL meter dataset which includes one-digit and multi-digit images.

To

my parents,

my brother,

and my girlfriend

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. Introduction

In the United States, about 28.8 million adults could benefit from hearing device but not use. The main reason is that the current system of testing typically involves high costs due to a lack of insurance coverage for these diagnostic tests and the need for specialized instrumentation. Therefore, there is an idea that we could automate the process of sound calibration in smartphones that enables these devices to be used in hearing assessment applications. This approach could reduce the cost of the testing process significantly and hence greatly improve access and affordability of this technology to those who need it the most while maintaining high accuracy. To tackle this problem, an automated digit recognition on the sound pressure level meter system has been proposed in this thesis.

Due to different neural networks have been widely developed, such as convolution neural network (CNN) and deep neural network (DNN), the technique of automated digit recognition is a rapidly growing and development market. This technique has been widely used in different fields, such as license plate recognition [1], automatic meter reading [2] and street view imagery [3]. However, to our best knowledge, to automatically recognize the multi-digit shown on the SPL meter using deep neural networks has not been well studied.

Traditionally, multi-digit recognition would be separated up into three steps, which are localization, segmentation, and recognition. Thus, there are several different approaches have been proposed to segment and recognize the image. Optical Character Recognition (OCR) method is one of the conventional approaches [4], it would segment the individual characters and recognize these characters separately. Due to recognize each character individually ignores the relationship between the characters. However, this flow is useful in some digit recognition approaches, since it would not affect recognition when the number be segmented. And this flow is easy to combine

the neural network in the recognition step, such as [5] [6] using a conventional image detector algorithm to localized the digit and using a convolution neural network (CNN) to recognize digit. There are several recent studies [3] [7] proposed an approach that could recognize the multi-digit or text without segmentation. By training the probabilistic model of sequences given images, Goodfellow et al. [3] could recognize the multi-digit without segmenting the digit. Shi et al. [7] proposed a convolution recurrent neural network (CRNN) to integrate CNN and recurrent neural network (RNN) for text recognition. However, these models are impossible to implement on the portable device since the heavy network.

The digit number is generated by the seven-segment display shown on the SPL meter. And the numbers are surrounded by the rectangular box. Different from the other wording style, the digit number display on the seven-segment display consists of 7 segments. The number could not be written in a single stroke. There is some interspace in a single number. There is no existing dataset that we could use to train our model directly. Furthermore, the environment of the SPL meter could be messy and complicated, which could be used in different places under different lighting. Thus, we need to propose a uniform approach to segment the number from the SPL meter.

In this thesis, we propose an automated digit recognition on SPL based on a deep neural network that could be implemented on the portable device. This approach consists of digital region detection and recognition. To detect the digital region efficiently, we adopt Canny edge detection [8] in our image preprocessing step. Due to the portable device have its constraints, such as timing delay, limited computing resources. We propose two custom shallow networks to recognize the digit without high computation.

In this thesis, we make the following contribution:

- In our basic idea, we are the first one to build the SPL meter dataset. This dataset consisted of 1000 SPL meter images under different lighting and angle. And the label could be found in the file name. This dataset could be used to train the multi and single-digit recognition. Since each image consisted of multiple digits. And our dataset also provides every digit coordinate for cropping into a single digit.

- We build and train a custom neural network offline, achieving high validation and testing accuracy. Since the model is relatively shallow, it could be implemented on a portable device without a high-computing resource.

The remainder of this paper is organized as follows. In Section II, we present some related work that we would use in our approach. Subsequently, our proposed approaches are described in Section III. The experiment results and analysis are shown in Section IV. Finally, some concluding remarks are given in Section V.

II. Related Work

In this section, we provide a description of two main related work. One is a neural network, and the other one is a convolution neural network. In our method, we would use these two approaches to build our model to recognize the SPL meter. Thus, we would provide some previous works and describe some basic concepts in these two networks.

i. Neural network

The first concept of an artificial neural network was proposed in the 20th century [9]. This idea was inspired by the brain that several neurons are connected to process the data. The main feature of the brain is it can determine the answer by learning the experience from the previous result. Therefore, the neural network has been proposed to solve the problem in a similar way the human brain does but highly simplified. In recent years, the neural network has been widely developed and implemented in several areas. There are several reasons for this trend. First of all, there are several better hardware (e.g. GPU) or new architectures that have been proposed [10] which could be used to compute the network contain with lots of neurons efficiently. Second, there are more valuable dataset could be provided to train the model.

In a neural network, there are lots of kernels connected to compose a model. We refer to these kernels as neurons, as shown in Figure 1. The neuron receives one bias and inputs from m neurons. The strength of the connection from the first neuron to m neuron is denoted by w_0 to w_m. Then the neuron sums the weighted input and applied an activation function to produce the output. With this simple mechanism, the network could learn from the data simply. Thus, even the simplest network, two-layer fully connected network (i.e. one input layer and one output layer) can be served in several fields [11]. Our problem targets at recognizing digit shown on the SPL meter. Since the two-layer fully connected network has been shown it works well on handwritten digit

recognition. Therefore, we could suppose this approach could work well in our cases as well. However, to deal with a more complex problem, there are more networks have been proposed, either fully connected network or convolutional neural network. In the next section, we would discuss some related work on CNN, which is the most powerful model for object recognition.

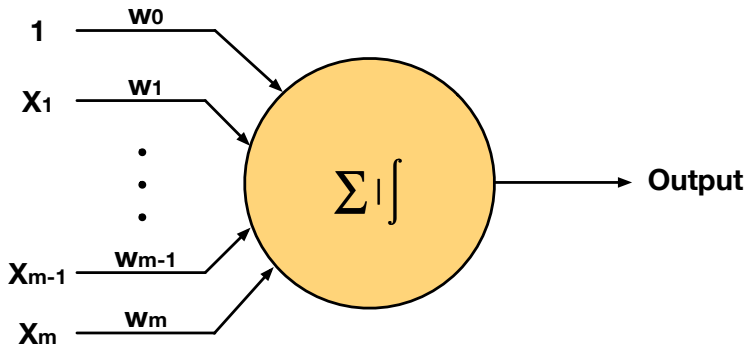


Figure 1. A diagrammatic representation of the perceptron

ii. Convolution Neural Network

Convolution neural network (CNN) has shown its strong capability in object recognition and pattern detection. CNN mainly contains several hierarchy convolutional layers which are the most computation heavy part. The convolutional layers are composed of several filters, which used to extract the feature from the input and create the feature map. The size of the filter should smaller than the input data to slide across the width and the height. And the dot products between the input and the filters are computed at every spatial position. Typically, the size of a 3x3 filter is used in practice. Apart from the convolution layer, CNN consisted of different types of layers. Normally, the pooling layer connects directly to the convolution layer. The pooling layer is used to reduce the spatial size of the convolved feature. Their task is to simplify the output of the convolution layer and to downsample the feature map for further processing. Subsequently, the fully connected would be connected at the end of the network to generate the output.

Due to its powerful computational ability in computer vision. Many CNN implementations have been proposed to focus on recognizing a single object [12]. To tackle with some complicated problems and improve the accuracy, there are a lot of networks based on the CNN have been proposed to recognize the object, such as Spatial Displacement Neural Network (SDNN) [13], recurrent CNN (RCNN) [14] and YOLO [15]. However, these networks suffer from the heavy computation workload. It is difficult to implement these networks on the portable device with limited computation resources, especially for cellphones. In our approach, we would segment the multi-digit into individual patches and then use CNN for digit recognition.

III. Technical Approach

i. System Pipelined

For machine learning application development, there are consisted of three main steps: data preparation, training model and making a prediction. As you can see in Figure 2, we would base on the data to train the model. After we build a model, the user could send the request for the system to predict the result. In our case, we want to develop an app for digit recognition. First, we need to gather enough images contain with different digit numbers that the system could recognition. Second, we would base on the dataset that we obtain from the last step to train a model. At last with the information provided by the user, the app should be able to recognize and show the result to the user.

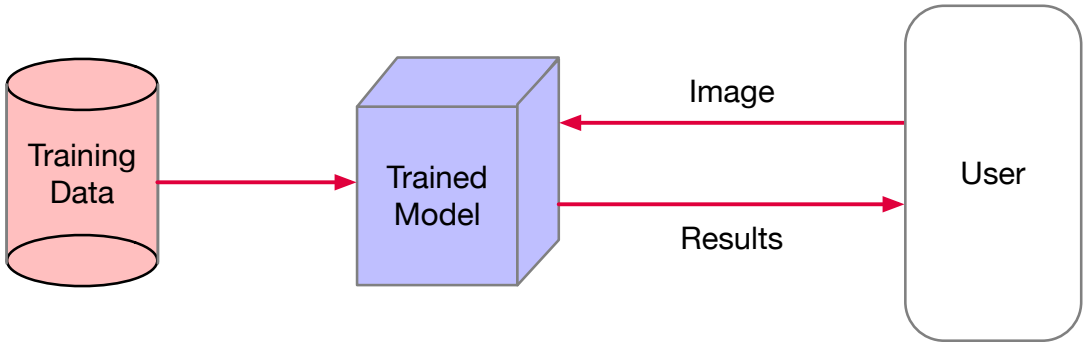


Figure 2. Architecture of how fully managed machine learning works

In machine learning application development, the approach you use to train the model could be varied. There are two main categories in the machine learning application. One is online learning and the other one is offline learning [16]. In offline learning, we would base on a certain dataset to train the model. Once you trained a model, you would use this model to make predication. However, in online learning, you would continuously upload your new batches of data to adjust your model. The model weight would base on the observation from a new dataset to adjust the

weight. Therefore, as an edged computing application, we would subject to the device's memory and clock delay latency. We would not available to retrain the data continuously. It would consume lots of memory and power consumption. In this case, we need to develop a system without heavy calculation. Thus, in this thesis, we propose several techniques to resolve those limitations.

As shown in Figure 3, is our system architecture. Our whole system consists of two parts. The first one is image preprocessing, when we received the input image, we would locate the digit and crop the image into a small size. This step would help us to reduce the noise and amplify the digit number. These could help us to recognize the digit more accurately. The second step is image recognition. In our proposed approach, the model is trained off-line. Since the training model takes a lot of resources and also takes time, if we could train the model off-line, it would help us to have a high-speed system.

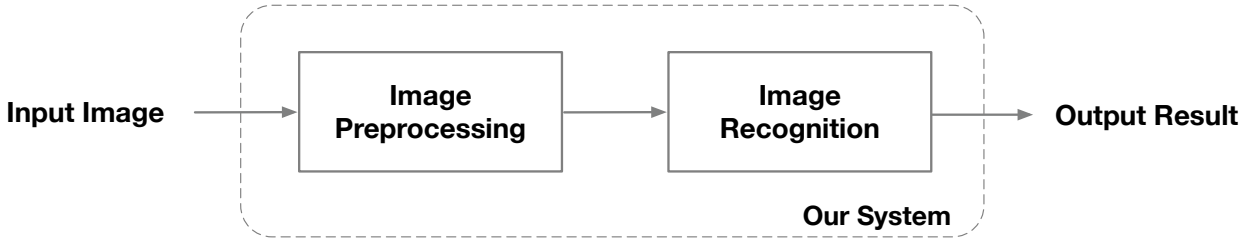


Figure 3. System architecture

ii. Data Collection

The data collection is one of the critical issues in machine learning. Since there are lots of novel applications have been proposed. The labeled data may not be enough or even not existed. Therefore, collecting the data would be the first step when we train the model. However, data collection usually is the most time-consuming and experience. If the dataset is too small, you need to go back and train again. Sometimes training with small data may hard to converge your model or cannot train a model. But collecting too much data may also cost too much money. Thus, how

to collect the data efficiently and also acquire a good dataset has been widely discussed. In [17], it discussed several kinds of approach to collect the data. There are largely three main methods of data collection. They are acquisition techniques, data labeling techniques and improving existing data.

As a new application for multi-digit recognition on SPL meter, the best way for us is to accumulate the data by ourselves. Though this way is relatively time-consuming, we need to collect the data and label it. But it is the best way for us to train the high-quality model with a small image dataset. Since the model quality depends on the quality of input data. By using this approach, we can make sure the image quality before we train the model. Therefore, our proposed approach of data collection consisted of three steps. These three steps are data acquisition, labeling and

First of all, we would take the picture for the SPL meter under different volumes, as shown in Figure 4. To acquire different digit numbers shown on the SPL meter. We would also collect the data under different conditions. To obtain the image with different light and shadow. The second step is labeling. In this step, we used the existing labeling tool 'LabelImg' to label the image. By using this tool, we would find out the boundary of each digit and store the coordinate in the xml file, as shown in Figure 5. However, it is not feasible to utilize the coordinate from the xml file directly. There are lots of redundant information in this file and also each image has one file. It's time-consuming to looking for the information by opening and closing the file several times. Instead of directly using the xml file, we would extract the coordination and label into a csv file. This information is only what we needed. Therefore, by using the csv file, we can crop the multi-digit image into a segmented digit patch efficiently. The last step is cropped the multi-digit image into a segmented digit patch. By referring MNIST dataset [18], the segmented digit would be resizing into 28x28 and convert the image into grayscale (i.e. convert the channel size into 2).

The reason why we crop the image size into 28 x 28 is that it may help us not only to find the feature more precise but also speed-up the training time. After the whole processing, I would store the new image into the new folder and rename my new image with the label. Then, our dataset would be set up and we could use this dataset as our input dataset in the machine learning. Some samples in our database are illustrated in Figure 6.

Figure 4.The data collected from the SPL meter

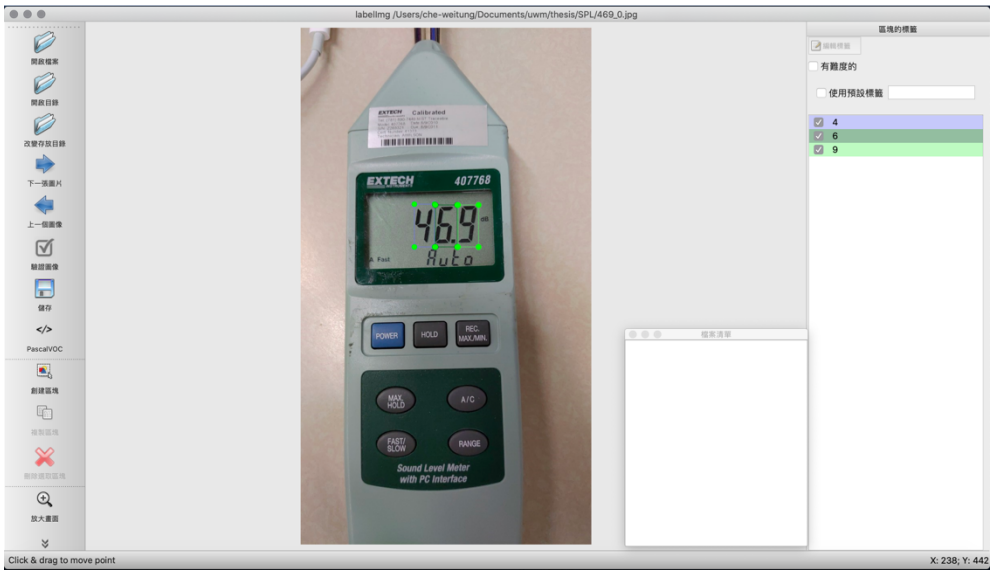


Figure 5. Labeling images using ‘LabelImg’ tool

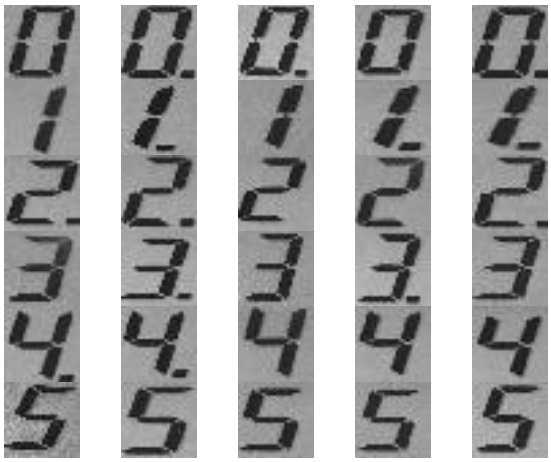


Figure 6. Sample images in our database

iii. Data Preprocessing

In our system, the users are required to take the picture on SPL meter. Since the data is from the natural world, it may have some noise, such as lightning, shadows, and specular highlight. These noises may affect our prediction accuracy. Thus, data preprocessing would be a good approach for us to reduce the noise and amplify the feature. There are a lot of approaches have been proposed to preprocess the data [19] [20]. In this thesis, I would use two main approaches to deal with my data. One is image detection, and the other one is image localization. In this section, I would present my data preprocessing design flow. The complete process of our preprocessing is demonstrated in Figure 7. The proposed SPL meter digit preprocessing algorithm includes the following steps:

1. *First canny edge detection:* On SPL meter, there are lots of function buttons and information on it. Canny edge detection algorithm has a good ability to eliminate the noise and find the main edge. Thus, we start to find the edge by using canny edge detection. Several edges would be found out and so further processing is applied.

2. *Contour selection:* By applying canny edge detection, there are lots of edges that have been found out. We could use the feature from the screen which is the biggest rectangular on the device. By setting the threshold, we could find out the bounding box for the next step.

3. *First cropping:* The best way to reduce noise is to eliminate it. Through the bounding box that we found from the previous step, we could easily crop the image into small size and save to grayscale.

4. *Second canny edge detection:* Canny edge detection algorithm performance a high response to detect the contour. The digit number shown on the SPL device is black. Thus, we could easily find out some digit candidates.

5. *Digit localization:* In our cases, there are three-digit numbers on the screen. By setting the threshold, we could obtain the coordinate of the digit number from the last step.

6. *Second cropping:* The coordinate that we found from the last step is the boundary for the digit. Thus, we could crop the image again to obtain the multi-digit image without any noise.

7. *Digit segmentation:* In our cases, the three-digit number septate equally on the screen. We could base on this feature to segment the three numbers into individual patches. And the whole preprocessing step would be done here.

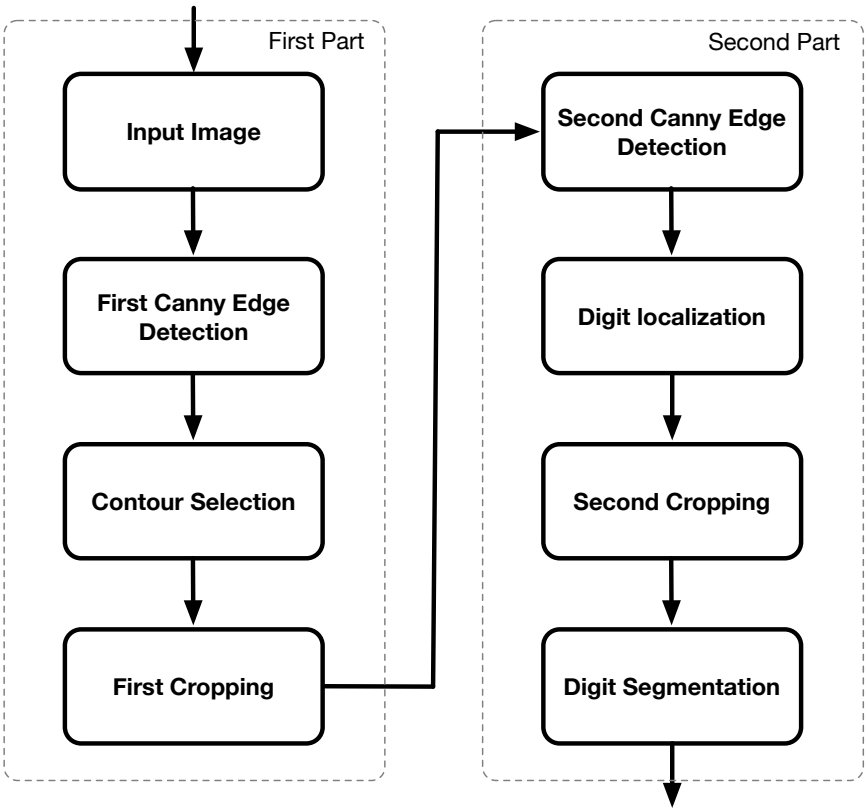


Figure 7. Step by step processing flowchart

In our approach, we address the issue by separating the processing into two parts, as I present in Figure 7. For the first part, we would crop the main screen into individual images. Since there are lots of bottoms and labels on the SPL measure device, as you can see in Figure. 8(a), the best way to reduce the noise is to remove that. Therefore, Canny edge detection has been used to detect the contour. By setting the threshold between 100 and 160, we could obtain several candidates for the screen, as you can see in the Figure. 8(b). There are lots of edge detection algorithm have been proposed [21]. However, Canny edge detection is one of the most efficient edge detection algorithms which has been widely used in image processing [8]. Canny edge detection consists of 4 steps [22]:

1. *Noise reduction:* The edge detection is highly sensitive to the noise. Therefore, the Gaussian filter has been used to remove the noise. The blurring effect could be

controlled by how big the kernel size that we want to use. In our case, since there are lots of reflections on the image, to remove those noise we would use 11 by 11 Gaussian kernel.

2. *Gradient calculation:* After filtering the image, we could detect the edge intensity by calculating the gradient of each pixel. We could almost get the edge after this step. However, the thickness of the edge is not uniform, we need to mitigate the thinker edge in the further step.

3. *Non-maximum suppression:* In this step, we would use a non-maximum suppression algorithm to find the pixels with the maximum value in the edge direction. It would help us to obtain a uniform edge (i.e. edge is one-pixel width).

4. *Double threshold:* In the above, we found out lots of edge candidate. To find the real edges, we would set two thresholds which are *maxval* and *minval* to make a decision. If the edge with the intensity gradient greater than *maxval*, we would set it as a real edge directly. If the edge with the intensity gradient between *maxval* and *minval*, we would tag it as a low-intensity gradient. Others would directly be ignored. Here, we set our *maxval* to 150 and *minval* to 100.

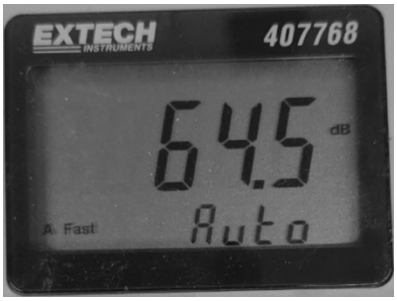
5. *Edge tracking by Hysteresis:* The edge that we tag it as a low-intensity gradient, we would check whether there is any confirmed edge surrounding nearby this pixel. If there is one confirmed edge surrounding it, we would also set it as a real edge. Thus, all the edge would be found.

After we obtain several candidates for the screen by utilizing Canny edge detection, we could use the screen feature which is the biggest square in the SPL meter to find out the edge of

the screen. By utilizing contour finder, we could obtain the coordination for the SPL meter's screen, as you can see in the Figure. 8(c). Therefore, we could base on this coordination to crop the image. This approach could help us to drop the noise and amplify the digit.

(a) Example test image (b) Canny edged feature (c) First bounding box

Figure 8. Input and intermediate image in finding first bounding box



(a) Cropped image after first bounding box

(b) Canny edge features

(c) Cropped image after second bounding box

(d) Segmented digit patch

Figure 9. Segmentation step

For the second part, we plan to segment the digits into individual patches and resize the image into 28 by 28. We would crop the image by using the final result in the first part. And filter the image into grayscale, as you can see in the Figure. 9(a).

By using the Canny edge algorithm again, we could obtain all the control in the image. However, the digit number shown on the SPL meter is the seven-segment display. That is to say we cannot find the digit by using contour finder directly. Since the digit number on the seven-segment display consists of seven segments. Although we utilized the GaussianBlur function to blur the digit, we cannot find out the individual contour for number "1" and "7". Thus, we set the threshold and boundary to find the digit, as shown in the Figure. 9(b). And then find the bounding box and crop into small size. In our cases, the three digits numbers in the bounding box are distributed equally. We used this feature to segment the digit patch. The digit patch is resized to 28 by 28 to fit in a simple convolution neural network (CNN) in the following step. The segmented digit patch is shown in Figure. 9(d).

iv. Digit Recognition

In this section we provide a description of two neural network models that we used to recognize the digit on the SPL meter. Since we want to build a system that could implement on the portable device. We build two shallow custom neural networks with fewer kernels and train the model only for the digit on the SPL meter. The first one is the model only consisted of two fully connected layers. The second one is the convolution neural network (CNN).

After we preprocessed the image from the last step, these individual patches are the image that we want to recognize. One 28 by 28 patch consisted of one digit. Thus, these patches would be fed into the model that we built to recognize. Furthermore, we would use the dataset that we create by ourselves to train the model, as we mentioned in section ii. The training set consisted of

2686 grayscale images of size 28x28 pixels. Each image has a one-digit number. The images are divided into 10 classes (i.e. 0 to 9) and the label is shown on the file name.

The first model is consisted of two fully connected layers (FC1 and FC2), as shown in Figure 10. Since the image is two dimensions (i.e. 28x28), we need to flatten the image into a one-dimension vector. It transforms the bidimensional tensor into a monodimensional tensor. Thus, the input image would be flattened into a long vector (i.e. 28x28=784). After the flattening step, a long vector of input data would be fed into the artificial neural network to process further. The first fully connected layer consisted of 512 neurons. The second fully connected layer consisted of 10 neurons. And the output of FC2 is passed to SoftMax activation to generate the 10 probability distributions over 10 classes (i.e. label '0' to '9'). The class has the highest probability would be our final answer, the rest of the classes would be ignored. For instance, the probability of '1' is 0.7, the probability of '7' is 0.2, the probability of '9' is 0.1 and the rest of the label is '0'. The output result would be '1'.

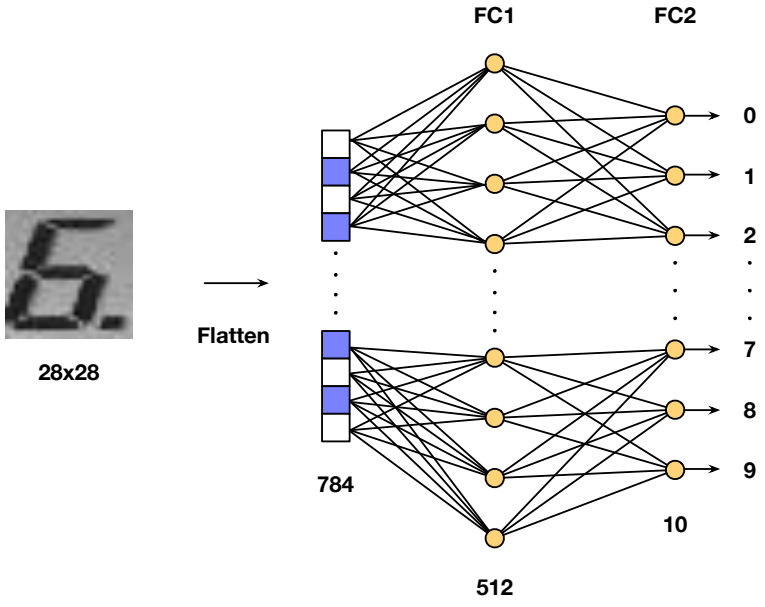


Figure 10. Fully connected neural network structure

In the second model, our architecture is similar to LeNet-5 [23] which is the most well-known CNN architecture for hand-written digit recognition. Our model comprised of three convolutional layers (C1, C2, and C3), two max-pooling layers (S1 and S2) and two fully connected layers (FC1 and FC2), as I showed in the Figure 11. The first convolution layer (C1) filters the input image (i.e. 28x28 grayscale single digit) with 32 kernels of 3x3 size. In the second convolutional layer (C2), it consisted of 64 kernels of 3x3 which used to filter the down-sampled 14x14x32 feature maps from S1. The third convolutional layer (C3) also consists of 64 kernels of 3x3 size while it filtered the down-sampled 7x7x64 feature maps from S2. The output of the C3 is fed into the fully connected layer (FC1). The two pooling layers (S1 and S2) implement the max-pooling function over a non-overlapping pooling window of size 2x2. Finally, we used the same configuration as the first model, two fully connected layer (FC1 and FC2) with 512 and 10 neurons have been used to calculate the distribution probability of ten classes and SoftMax activation has been used to generate the output result. Note that, we used ReLU [24] as our activation function which is used at the output of each convolutional layer and the first fully connected layer. Under the same accuracy, compare with other activation functions, such as sigmoid, hyperbolic tangent function, ReLU has been seen as the most efficient activation function [12].

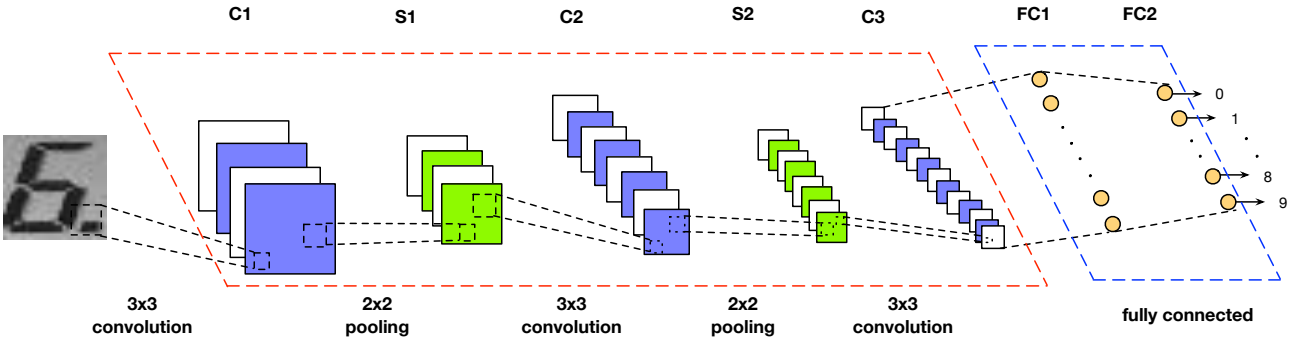


Figure 11. Convolution neural network structure

IV. Experimental Setup

In this section, we provide the description of our experimental setup including computational resources, datasets used for the experiments, training configuration, and the evaluation criteria to judge our system performance.

i. Build Method

The whole system pipeline includes image preprocessing and digit recognition is implemented in Python 3.7.5 and built using PyCharm Professional 2019. Building the model by importing Keras package to use TensorFlow backend. Experiments are worked on the laptop with a 2.3 GHz Intel Core i5 CPU.

ii. Data

In machine learning, the data play the most important part. Without high-quality data, it is impossible to acquire a high-accuracy model. Furthermore, the size of the dataset is also one of the main points. One of the main reasons that the machine learning would rapidly develop in recent years is because of the data booming. However, there is no existing dataset for the SPL meter device. Thus, there is one of our main contributions in this thesis, we create an SPL meter device dataset, as we mention in the previous section (III.ii). In this dataset, each image has a multi-digit in the single digit. We also provide the coordinate for every single digit. These could make a user for further processing, such as cropping or digit localization.

In this dataset, it contains with 1000 labeled high-resolution images. The images were collected from the two different SPL meters. We collect the image under different lightening, angle and distance by our cellphone camera. The noise is generated from the cellphone under constant frequency for SPL meter to capture and display on the screen. And labeled by humans using

LabelImg [25] labeling tool. To train a model, we need to have a constant input dimensionality. Therefore, we crop the image by using the coordination that we provide in the dataset and downsampled the image to a fixed resolution 28x28 in grayscale. Here, we divide the dataset into 7:3, 70 percent for training and 30 percent for testing.

iii. Training configuration

Many hyperparameters may affect the performance of training the model such as the optimize that we used, the number of the learning rate, batch size, and learning rate. In our research, we did not explore the best configuration. Instead, we use several parameter configurations to train the model and compare their performance. Thus, we would provide the further discussion in the below.

- *Optimizer:* The optimizer is one of the most critical hyperparameters for training the model. The optimizer that you choose would affect your algorithm converging or exploding. In [26], it discusses considerable optimizers. SGD (Stochastic Gradient Descent) has been widely used in practice. However, our model could not converge by using this optimizer. Therefore, we used a similar one, RMSprop optimizer, it has a better performance in our case. The only difference between RMSprop and SGD is the way how they calculated the gradient.

- *Learning rate:* The learning rate control how fast the model is adapted to the model. If the learning rate is too large, it would not find the optimal solution. Whereas the learning rate is too small can cause the process to get stuck. Furthermore, it requires more epochs for training updated. Since we used the RMSprop optimizer, we used its default value, 0.001, in our model to converge.

- *Batch size:* The batch size is a hyperparameter used to define the number of samples that will be propagated through the network once. Normally, we would set the size in power of 2 to fit the memory requirement, such as 32, 64, 128, and so on. The large batch size would cause the slowly converge processing but with the accurate results. In contract to small batch size, the learning processing converges quickly but with more noise. Due to our experimental result, the size of the batch size has the best performance when the number is 128 in CNN and 256 in the neural network.

- *Epoch:* Epoch is a parameter used to define how many times the learning algorithm will work through the whole training dataset. The number of the epoch should be integer. The size of the epoch would significantly affect the training time. In our case, the size of the epoch in 50 and 100 has a relatively better performance in our neural network and CNN model, respectively. We would have the further discussion in the next section.

- *Loss function:* Loss function is a function used to evaluate your model when you are training the model and the output value is referred to as loss. Since the loss function is directly connected to the activation function in the output layer of your neural network, we would base on your activation function to choose the loss function. In our case, we use SoftMax to categorize the 10 class. Thus, we used categorical_crossentropy as our loss function.

- *Metrics:* Matric is a function for you to evaluate your model after you trained the model. There are a lot of metrics we could use to evaluate your model. Base on different metrics, your result would also be different. We used the most

straightforward one, classification accuracy, as our metric to evaluate our machine learning algorithm.

iv. Performance Analysis

In this section, we analyze our performance by utilizing several approaches to explore our model. We plot the training curve to find the appropriate number of epochs. The training time has been recorded to analyze the network. And we also perform the accuracy of each class after the classification to demonstrate the dataset.

The training curve is a diagnostic tool that has been widely used to check the training is overfitting, underfitting or well-fit. By observing the plot, we could adjust our model for training faster and reduce overfitting. Thus, there are two matrices that we applied to plot the curve to evaluate the model. One is a performance learning curve that we based on the classification accuracy to make the plot which represents the accuracy over the epochs. The other one is the optimization learning curve which is based on cross-entropy loss to plot the curve for checking the model fitness. By dividing the data into training and validation dataset, the two training curves could be a plot to analyze our model, as I shown in the below.

Three common dynamics could be used to observant the behavior of the machine learning model, which are overfitting, underfitting and well-fit. By using this observation, we could adjust the configuration in the machine learning. For overfitting, the plot of the training loss continues to decrease with more experience. However, the plot of the validation loss decreases to the point and begin increasing again. For underfitting, the plot of the training loss has not been flattened at the end of the training. The plot of the training loss and the plot of validation loss has a small gap. Furthermore, the plot of the training loss has been flattened at the end of the training. This scenario refers to a well-fitting model.

In our neural network experiment, we allocated 2000 images for the training dataset, 379 images for the validation dataset and 307 images for testing. If the validation dataset is too small, it would be unstable. To analysis the model, the size of the epoch set to 150 in the beginning. Figure 12 shown the plot of the training and validation accuracy under our neural network training configuration. As we can see the plot of the training accuracy start to flatten when the epoch goes to nearly 100. At the same time, the accuracy is higher than 90%. Besides, the optimization learning curve shown in Figure 13. The curve of the training loss flattens when the size of the epoch is nearly 100. The curve of the validation loss decreases until the point of 90 and start to be unstable with more experiment. According to these two plots, it is observed that when the hyperparameter of the size of the epoch is 100 could avoid overfitting and reduce training time.

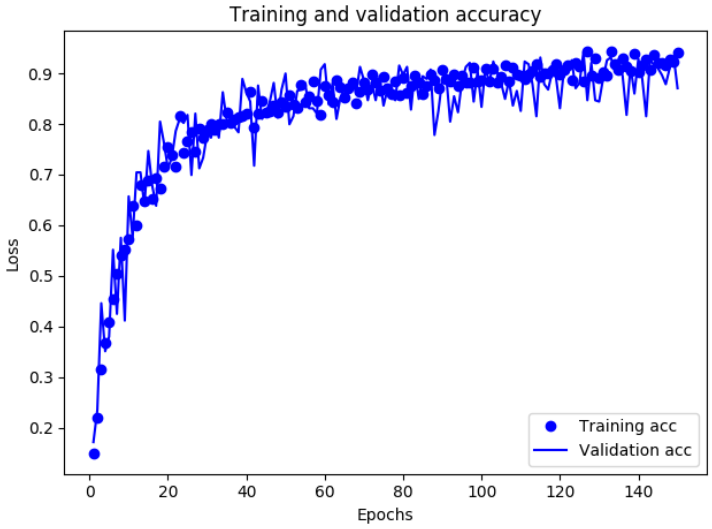


Figure 12. Training and validation accuracy in our NN model

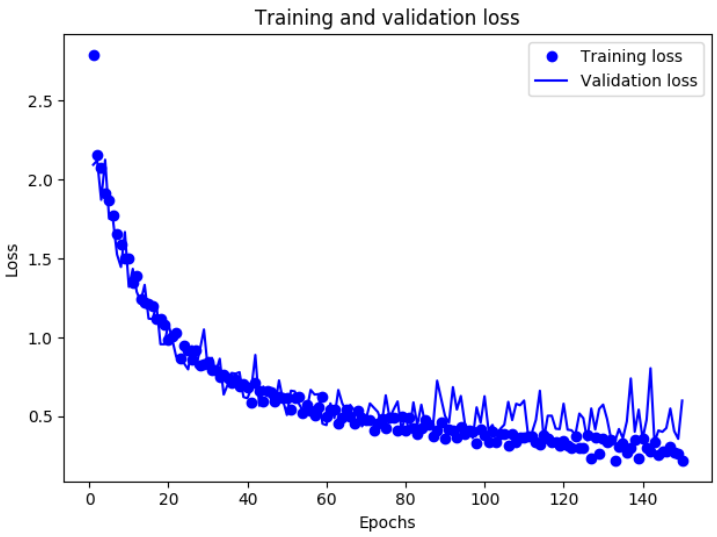


Figure 13. Training and validation loss in our NN model

For our CNN model, we allocated 2055 images for training, 324 images for validation and 307 for testing. We also set the size of the epoch to 150 initially for analysis. The plot of the training and the validation accuracy is shown in Figure 14. When the size of the epoch is 50, both of the training and validation curves start to flatten, and the accuracy is nearly 100% at the same time. Figure 15 illustrated the learning performance over experience. As we can see, the plot of the training loss decrease until the size of the epoch is 50. Furthermore, the smallest gap between the plot of the training loss and validation loss is also at 50. However, after the size of the epoch is 50, the plot of the validation loss starts to increase which means the training is overfitting. By these observations, we set the size of the epoch at 50 when we train our CNN model.

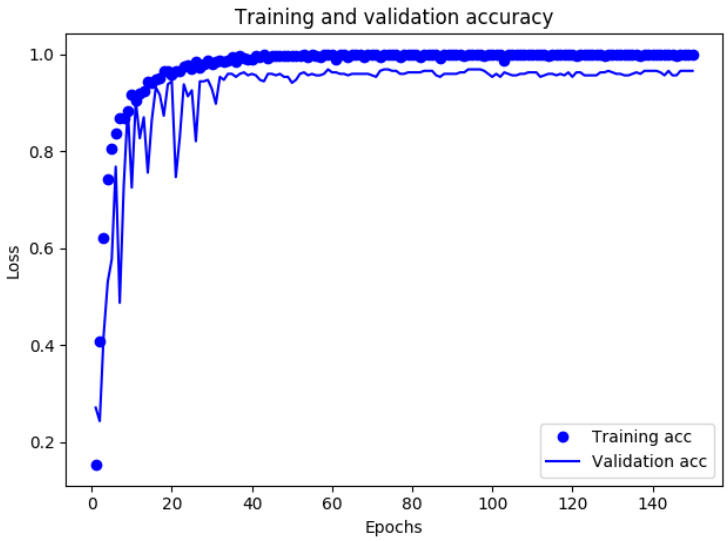


Figure 14. Training and validation accuracy in our CNN model

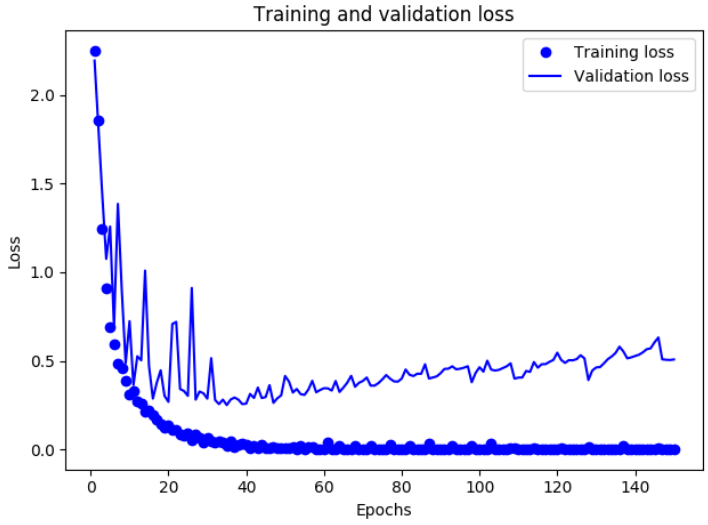


Figure 15. Training and validation loss in our CNN model

We train and build the model offline in python. The dataset that we used is mentioned in the section IV. ii. The performance of the different models is represented in Table 1. We also train the state-of-the-art object detection model, tiny YOLO model [15] on the workstation with two GeForce RTX 2080 Ti GPU to make a comparison. As you can see the accuracy in CNN and YOLO could achieve 100%. However, due to the tiny-yolov3 model which contains 19 layers instead of 7 layers in our CNN model, it takes at least 6 hours to train this heavy network in two

GPUs. According to our experimental results, CNN would be the best model to implement on the SPL automated recognition. Since the system that we proposed would be used on the sound calibration for hearing aid. There is no error-tolerant for medical device. Therefore, the system could not make any mistakes. Furthermore, to implement the system on the portable devices, the size of the system is one of the critical issues. Due to the memory limitation and the computational capability, it is infeasible to contain too much convolutional layers in the model. The reason is that the convolutional layer is the most computationally intensive in machine learning. Table 2 shows the CPU time breakdown in the stage of preprocessing and recognition under two models. The main difference between the neural network and the convolutional neural network is CNN composed of several convolutional layers. In our CNN model, it consisted of three convolutional layers which make the timing increase at the stage of recognition.

Table 1. Model performance

Model	NN	CNN	tiny-yolov3
Training time	9.69 seconds	50.25 seconds	6 hours
Best accuracy	99.34%	100%	100%

Table 2. Runtime breakdown between stage

	Preprocessing	Recognition
NN	0.0289	4.9307
CNN	0.0314	7.1791

A confusion matrix is a table that often used to evaluate the result of a machine learning model on a set of testing data for which the true values are known. Furthermore, there are some performance measures applied to analyses the classifier performance, such as precision, recall and

F-factor. These three performance measures are computed from the confusion matrix. The classification accuracy could also be calculated by the confusion matrix as follow:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Precision is defined as the ratio of the total number of correctly classified positive examples (i.e. true positive) over the total number of predicted positive examples (i.e. the number of true positives plus the number of false positive) as follow:

$$precision = \frac{TP}{TP + FP} \quad (2)$$

Recall is defined as the ratio of the total number of correctly classified positive examples divided to the total number of positive examples (i.e. the number of true positives plus the number of false negatives) as follow:

$$recall = \frac{TP}{TP + FN} \quad (3)$$

F-factor is also called F-measure which is the performance measure represents the harmonic mean of precision and recall, as shown in (4).

$$f1 = \frac{2 * Precision * Recall}{Recall + Precision} \quad (4)$$

Table 3 presents the classification report from our neural network model. Although the accuracy could achieve 100% in our CNN model, we still could though the classification report to analyze our result to strengthen our SPL meter dataset.

Table 3. Classification report

Class	Precision	Recall	F1-score
0	0.96	1.00	0.94
1	0.88	0.92	0.94
2	0.50	1.00	0.65
3	0.95	0.93	0.97
4	1.00	1.00	0.99
5	1.00	0.97	0.98
6	1.00	0.96	0.99
7	1.00	0.98	0.99
8	0.90	1.00	0.95
9	1.00	0.64	0.78

In our cases, class 2 has a high recall but low precision. It illustrates the number of the testing data is not 2 but the model recognizes it as 2 is large. For class 9, it has a high precision but low recall. This presents that there are a lot of numbers are 9 but the model recognizes it as a different number. The rest of the classes have high recall and high precision which means all results return correctly in that class. According to this observation, most of the class 2 has been recognized correctly but some of the class 9 has been recognized as 2. Thus, we could improve our dataset by increasing more class 9 to enhance class 9 identification rate.

V. Conclusion

In this thesis, we proposed a novel system that could be applied to recognize the multi-digit on the SPL meter automatically. In our basic knowledge, digit recognition for SPL meter by using a machine learning approach has not been well study. We are the first group to propose this idea. From our experimental results, our system achieves high accuracy under low computational resource and memory requirements. We also proposed a new approach to preprocess the input image for machine learning. By utilizing this algorithm, we could obtain the individual digit patches from the original input image efficiently. Nevertheless, we built the first SPL meter dataset which is the first complete dataset for SPL meter. This dataset could be widely used for future applications.

References

[1] C. J. S.M. Silva, "License Plate Detection and Recognition in Unconstrained Scenarios," in *ECCV*, 2018.

[2] V. B. ,. M. A. D. ,. G. R. G. ,. W. R. S. ,. D. M. Rayson Larocaa, "Convolutional Neural Networks for Automatic Meter Reading," *Journal of Electronic Imaging ,* vol. 28, no. 1, 2019.

[3] Y. B. J. I. S. A. V. S. Ian J. Goodfellow, "Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks," in *International Conference on Learning Representations*, 2014.

[4] M. C. Y. N. a. H. N. A. Bissacco, "PhotoOCR: Reading Text in Uncontrolled Conditions," in *IEEE International Conference on Computer Vision*, Sydney, 2013.

[5] M. D. S. F. F. K. R. S. Michael Opitz, "End-to-End Text Recognition using Local Ternary Patterns, MSER and Deep Convolutional Nets," in *11th IAPR International Workshop on Document Analysis Systems*, 2014.

[6] J. P. Xuan Yang, "MDig: Multi-digit Recognition using Convolutional Nerual Network on Mobile," in *Proc. Yang2015 MDigMR*, 2015.

[7] B. a. B. X. a. Y. Shi, "An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2016.

[8] V. H. a. R. B. Milan Sonka, Image Processing, Analysis, and Machine Vision (Second Edition), Beijing: Posts & Telecom Press, Sep 2003.

[9] P. W. Sonali. B. Maind, "Research Paper on Basic of Artificial Neural Network," *International Journal on Recent and Innovation Trends in Computing and Communication ,* vol. 2, no. 1, pp. 96-100, 2014.

[10] Y. C. J. E. A. S. a. Z. Z. V. Sze, "Hardware for machine learning: Challenges and opportunities," in *IEEE Custom Integrated Circuits Conference (CICC)*, Austin, 2017 .

[11] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Transactions on Information Theory,* vol. 39, pp. 930-945, 1993.

[12] I. S. a. G. E. H. Alex Krizhevsky, "Imagenet classification with deep convolutional neural networks," in *dvances in Neural Information Processing Systems 25* , 2012.

[13] L. B. Y. B. a. P. H. Y. Lecun, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[14] J. D. T. D. a. J. M. R. Girshick, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CVPR,* 2014.

[15] S. K. D. R. B. G. A. F. J. Redmon, "You Only Look Once: Unified, Real-Time Object Detection," *CoRR,* 2015.

[16] S. K. E. &. M. Y. Ben-David, "Online Learning versus Offline Learning," *Machine Learning 29,* pp. 45-63, Oct. 1997.

[17] G. H. S. E. W. Y. Roh, " A survey on data collection for machine learning: a big data-ai integration perspective".*IEEE Transactions on Knowledge and Data Engineering.*

[18] "The MNIST Database of Handwritten Digits," 〈http://yann.lecun.com/exdb/mnist/〉.

[19] L. Canchen, "Preprocessing Methods and Pipelines of Data Mining: An Overview," *arXiv:1906.08510 [cs.LG],* Jun 2019.

[20] M. S. H. T. L. a. M. W. T. N. Minh, "Automated Image Data Preprocessing with Deep Reinforcement Learning," *Computing Research Repository (CoRR),* 2018.

[21] C. Zhan, X. Duan, S. Xu, Z. Song and M. Luo, "An Improved Moving Object Detection Algorithm Based on Frame Difference and Edge Detection," in *Fourth International Conference on Image and Graphics*, 2007.

[22] J. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vols. PAMI-8, no. 6, pp. 679-698, Nov. 1986.

[23] L.́. B. Y. B. a. P. H. Yann LeCun, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* 1998.

[24] V. N. a. G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *In Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010.

[25] Tzutalin, "LabelImg," 2015. [Online]. Available: https://github.com/tzutalin/labelImg.

[26] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747,* 2016.