Automating Reading Digital and Analog Meters using Machine Learning: a Gasmeter Implementation

By

Loubna El Meddah El Idrissi

Oberlin College

May 20, 2021

Department of Physics
Oberlin College

Abstract

Automating analog meters readings has been successfully attempted by Object Computing Inc. to read analog gauges as part of their Deep-Gauge Project. The Deep-Gauge Project uses a CNN-based Deep Learning model to read analog gauges. The goal of this project is to use the CNN part of the model to build a Gasmeter-Reader model that can read both the digits and dials displayed by a gasmeter. The digits are predicted to an accuracy of 92.92%, and the dials to an accuracy of 95.51%.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Project's Motivation

Natural gas is a fossil fuel used in buildings in the Unites States for heating purposes, and/or for generating electricity. There are multiple concerns about the efficiency of natural gas when it comes to its effect on the environment. Indeed, the combustion of natural gas emits carbon dioxide (CO_2), so having a sense of how much natural gas one consumes can help promote its efficient consumption. This project aims to test the CNN model from the Deep-Gauge-Project (by Object Computing Inc.) to see whether it can be used to detect other meters with analog dials, as well as digits, and more specifically if it could detect the readings of a gasmeter as seen in Fig. 1.1.

1.2 Project's Goal

The goal of this project is to automate the reading of digital and analog meters. This problem can be easily tackled if the problem at hand is reduced to a multiple image classification problems. Convolutional Neural Networks (CNNs) happen to be a commonly used algorithms for such a task. A meter with both digital and analog readings can be chosen to extract images of digits and dials to train two CNNs to read the extracted images.

1.2.1 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a neural network with some convolutional layers and some other layers, and it uses hierarchical vision is the process of going through the pixels of images to find some non-linear relationship between them. The usual structure of CNN is: Succession of convolution layers, pooling layers, Fully connected layers, and finally an Output layer.To implement the CNN models, more programs are designed to accomplish the pre-processing steps such as: image capture, image segmentation (cropping dial and digit images from gasmeter images), etc ... .

1.2.2 CNN Model Architecture

The CNN model used in this project has been originally developed as part of an Ensemble Model designed by Xiao Yang, OCI Director, of Object Computing Inc to read analog gauges (the article can be accessed here: $https://objectcomputing.com/resources/publications/sett/june-2019-using-machine-learning-to-read-analog-gauges$). To show that the CNN part is efficient in readings the dials and the digits of a meter, and that using an Ensemble Model can be skipped with enough images, the CNN part of the model is implemented in this project on another type of meter: a analog and digital gasmeter as shown in Fig. 1.1. Some of the advantages to using Machine Learning to accomplish this project's goal, and more specifically Convolutional Neural Networks are:

- Their ability to learn different features from different images and use them on input images once the model's training is done.
- Their ability to add images to make the CNN model even more efficient with different kind of images

The overall focus of this project is to build a program that can read the amount of gas consumed using images of the gasmeter's digits and dials.

1.3 Gasmeter Reader Program

1.3.1 Program Description

The Gasmeter Reader Program makes use of two CNN models from the Gauge-Deep-Reader project. Each model will be trained to detect either the segmented digit images or the segmented dials values.
The Gasmeter Reader Program will:

- Capture images of the gasmeter using a USB webcam
- Prompt the user to select the region where are located the two dials and the each of the four digits on the gasmeter
- Save the coordinates and use them to segment the images being captured by the webcam
- Crop all the images captured by the program and put each of the dial and digit to specific folder in the directory
- Pass the dial images to the Dial Reader model that will return the dials' readings
- Pass the digit images to the Digit Detector model that will return the digits' readings
- Save the images, the time and date of its capture, the CNN models outputed label categories as a file that can be opened using some application such as Excel.

The long-term goal of this project is to track the gas consumption in a building with gasmeters similar to the one used in this project. The two CNN models trained and tested in the Gasmeter-Reader program can also be trained with more images to read meters that track the electricity or water consumption.

1.3.2 Software used

All python scripts used in this project can be accessed at the github repository (username: , project name:).
It is always recommended to create a separate virtual environment to run any program with multiple libraries, and the Gasmeter-Reader program is no exception. The same virtual environment can be used to train and test the models in the future with new different images since no changes need to be done to the code as long as the libraries don't get deprecated. Miniconda is the fastest software to download and use to create a virtual environment, and does not take as much space as the larger version Anaconda.
The libraries required for the Gasmeter-Reader to run can be accessed in the requirements.txt file on the github repository mentioned above. The requirements.txt can be used to download the libraries and theneeded to be in the virtual environment. The most important libraries used are: OpenCV[], and Tensorflow version 1[].

1.4 Methods

1.4.1 Artificial Intelligence (AI)

In the Computer Science field, Artificial Intelligence (AI) refers to the study and creating of machines that seeks to human intelligence to replace human need of accomplishing some tasks. Artificial Intelligence has revolutionized a number of fields, and Computer Vision (CV) happens to be one of them. The goal of computer vision is to mimic the vision faculties of humans and this ability to computers through Deep Learning algorithms. One well known Deep Learning algorithms used in Computer Vision are Convolutional Neural Networks (also referred to as ConvNets or CNNs).

1.4.2 Advantages of Machine Learning for Computer Vision

At the beginning of the investigation of solving the problem of monitoring the gas consumption displayed by a gasmeter, OpenCV was first used to detect the line formed by the needle and the angle it made with a horizontal line. This method requires the user to manually try threshold values that can allow the program to pick on the needle's dark pixels. This method was unfortunately not accurate since the gasmeter images tend to differ because of the change of light throughout the day, and the potential scratches that can occur on the gas meter's protective box, etc... . Machine learning (ML) is thus preferred since ML models tend to pick up on all the image's pixels to determine a way to map out the input image to the input label. The other advantage of the Machine learning

way, it that the same ML model can be trained twice to detect the position of the dial's needle and the gasmeter's digits without the need to design two different algorithms that will only be successful on one type of dial and digit images.

Figure 1.1: Image of the gasmeter taken with a USB webcam and a python script.

Chapter 2

Background

2.1 Reading the Gasmeter

The gasmeter used in this project has a 4-digit counter that measures the gas consumption in 100 cf increments and is shown in Fig. 1.1. The gasmeter is generally set up between the incoming gas lines and the point of gas distribution of a building and measures the natural gas consumption in cubic foot (cf). In a given building, gas appliances consume gas at a rate that can be expressed through cfm. The gas flowing through the pipes drive the gasmeter's needles to rotate as a function of time proportionally to the force of the flowing gas. The number of time a gasmeter's needle rotates can be converted into a cf volume according to the following rules:

- One full revolution of the right dial's needle represents 0.5 cf of consumed natural gas,
- One full revolution of the left dial indicates that a volume of 2 cf of gas has been consumed.
- For every four full revolutions the right dial's needle completes, the left dial's needle rotates once.
- For every 50 revolutions the left dial's needle makes, the the "$\times$100" count gets incremented.

2.2 Computer Vision Technology

2.2.1 Machine Learning

One of the fields of research of Artificial Intelligence (AI) happens to be Computer Vision, which is a technology that would solve the problem of monitoring the gasmeter. To achieve

a computer vision that mimics the human vision abilities, AI uses Machine learning, which is a more specific branch of AI.
Machine learning (ML) involves using algorithms that can be trained to make an educated guess about a certain dataset. The ML technique used in this project is supervised deep learning.

2.2.2 Supervised Deep Learning

In supervised deep learning, the algorithm is presented with both training inputs and their target outputs, also called labels. Labels are used for the algorithm to determine the category where each input has to be mapped to.
Supervised deep learning is accomplished in two phases: a training phase, and a testing phase. During the training phase the images of the training set are used to "teach" the algorithm to detect the features and patterns of the input data. features mapped to category label. The training phase is then followed by the testing phase where the ML algorithm applies what it has "learned" from that previous phase to classify the input data to their respective category label. The larger the training set, the more data is available to the machine learning model to learn from. The testing set is used during this phase to evaluate the model's performance.
The accuracy of the model's classification of the testing set is an indicator on how well the model is capable of detecting the patterns it had learned during the training phase in new images to make guesses about the category label of each of the images of the testing set.

2.2.3 Convolutional Neural Networks (CNNs)

Convolutional Neural Network (CNN) is a class of supervised deep learning algorithms conceived to specifically receive and process pixel data making a great solution to many computer vision problems. CNNs have proven to be an efficient method to solve image classification problem. These neural networks are able to detect patterns in an image's pixels and make predictions about that image. After training two CNN models using gasmeter's images obtained from an USB webcam, the models can be deployed online and used to monitor the gas consumption and limit the human intervention in this process.

2.3 Visual Information

2.3.1 Digital Images

The input of CNN models are digital images. Digital images are used during both the training and testing phase. Therefore, the quality of the digital input images used in a supervised machine learning project is one of the most important factors when in comes to mimicking human vision. Unlike the visual cortex, CNNs have to learn the important

features presented by the pixels of an image, their position, color, etc..., and are used to make a decision to classify the input image to their predicted labels.

2.3.2 Pixels

A pixel is the smallest unit element in a digital image and they appear as small squares when the digital image is enlarged. A digital image is represented by pixels , the dimensions of this array determines the size of the digital image.

2.3.3 Pixels Coordinates

The coordinate system of these matrices is such that the x-axis is positive in the right direction, and the y-axis in the downwards direction. A digital image is known to be made of electron beam scanning pattern of televisions from the left to the right, subsequently the y-axis has an inverse direction than the mathematical convention.

2.3.4 Image Channels

Each of the CNN models used to read the gasmeter values can take in colorful images. A colored images has pixels with each of the following 3 colors: Red, Green, and Blue, from the darkest and lightest intensity. These 3 colors, and their different combinations and intensities make millions of different colors. The two CNN trained and tested both take in colorful images (3 channels).

2.4 Processing of Visual Information

2.4.1 Hierarchical Processing

Hierarchical Processing is the technique used by Convolutional Neural Networks to analyse visual information of the input images and classify them. In neuro-science, hierarchical vision is used to describe the way the brain processes the sensory information coming from the eye to create an image. Since the output information at each level of the process builds upon the input information coming from the previous level the vision is described to be hierarchical.

The concept of hierarchical processing of visual information was first brought to the field by two neuro-physiologists David Hubel and Torsten Wiesel. Hubel and Wiesel conducted experiments on the visual cortex of cats, the brain's region responsible for vision. In 1959, they published their paper "Receptive fields of single neurons in the cat's striate cortex"[3], which was crucial in the understand the visual system of humans.

Hubel and Wiesel experiment consisted on exposing cats to a screen with different dark and light patterns. They then observed what is well known now to be neurons of the visual cortex get stimulated through sensory input coming from the cats eyes. They have also observed that not all neurons respond similarly to different sensory inputs. From those

observations, Hubel and Wiesel have concluded that each neuron must be specialized in detecting a specific feature or characteristic of the input.
The concept of hierarchical vision is used today in Deep Learning to solve the problem of image classification through the implementation of Convolutional Neural Networks.

2.4.2 CNN and the Visual Cortex Analogy

The great discovery of the Visual Cortex made in the field of Neuroscience has inspired computer scientist to create Convolutional Neural Networks. A CNN essentially mimics the visual cortex by adopting the pattern formed by its neurons and their different roles, making CNNs artificially intelligent neural networks. Depending on the location of the neurons amongst the layers, they will detect simpler or more complex features. Just like the visual cortex, the neurons in the first layer of a CNN detect small features (i.e small features such as pixels of edges). The neurons in the following layers gradually build upon the previous smaller detected features to form bigger and more complex features (i.e increasingly abstract concepts such as a needle, its color, position, etc.) . The output of the neurons eventually is able to make an appropriate guess about the category label of the input.
The main difference between these two visual systems (CNN and the Visual Cortex) is that the convolutional neural network has to be trained before it is capable of 'seeing', or making guesses unlike the visual cortex. Table. 2.1 compares some of the characteristics of CNN and the Visual Cortex.

Convolutional Neural Networks	**Visual Cortex**
artificial neurons	biological neurons
edges	synapses
weight of neuron	number of connections of neuron
convolutional layer, max pool layer, FC layer	visual areas V2, V3, V4, and V5
non-linear activation function (see Ch3, Theory)	electrical impulse
training phase is crucial	no training phase needed
Layers' order, size, and number varies	visual areas depend on species

Table 2.1: Table comparing Convolutional Neural Networks (CNNs) and the Visual Cortex as two visual systems that process input images.

2.4.3 Artificial Neurons and Biological Neurons Analogy

CNN has adapted the Visual Cortex's use of neurons to create a network capable of processing the visual information coming images in a somewhat similar way than how the the sensory information coming from the photosensitive part of the eye's retina is analysed by the brain. CNN, inspired from biological neurons, makes use of artificial neurons to "see" an image and guess to classification. Artificial neurons are mathematical

function that take inputs of a specified threshold that varies from neuron to neuron. Due this difference in threshold, each neuron will focus some feature from the input image. Artificial neurons' threshold logic is similar to logic gates' threshold logic, and their outputs are called features.

2.5 The Evolution of Computer Vision

2.5.1 David Marr's Frameworks

The process of vision has long been a fascinating one. In 1982, David Marr, a neuroscientist, came up with a framework to solve computational neuroscience problems such as vision. Marr 's book, "Vision: A computational investigation into the human representation and processing of visual information", extended of the utility of Hubel and Wiesel's hierarchical vision concept in other fields such as computer science. His framework proposes 3 levels of analysis of visual information, where each level is considered to be independent of the other:

- *Level 1: Computational level.*
  This is an abstract level where the solution of the problem is suggested as well as the basic logic of how to solve it.
- *Level 2: Algorithmic and representational level.*
  This level consists of building a model capable of mapping out the input to an output.
- *Level 3: Hardware Implementation.*
  This is the level where the solution of the problem is outputted (ex. the brain)

The algorithmic and representational level is the level Marr contributed the most to in his book. Marr designed an algorithm that would be able to map the visual information to the output. Marr's suggested algorithm takes in a one-dimentional array that represents the primal sketch of the visual information. The 1-D input is then transformed into a 2-D sketch (edges from the 1-D sketch + depth of those edges). On the final step a 3-D model is constructed from the 2-D sketch.

2.5.2 Neocognitron algorithm

Another work that has made computer vision possible and efficient is the development of Neocognitron algorithm, the first neural network to make use the convolutional layers. Neocognitron was created in 1980 by the Japanese computer scientist Kunihiko Fukushima. Kunihiko Fukushima used the hierarchical vision approach to build the Neocognitron algorithm.

2.5.3 LeNet-5 and the Backpropagation algorithms

In 1989, Yann LeCun, a French scientist came up with a Backpropagation learning algorithm (see Ch3, Theory) and created LeNet-5 algorithm. The Backpropagation step increased the Neocognitron learning efficiency. Yann LeCun and other scientists published a paper describing a zip code reader that implements LeNet-5.

2.5.4 AlexNet algorithm

However, CNN's efficiency was not fully proven until 2012, when a team from the University of Toronto won the ImageNet Large Scale Visual Recognition Competition (ILSVRC) with an 8-layer CNN, AlexNet. The ImageNet competition consists of evaluating networds for object detection and image classification using large databases. AlexNet was trained using a million real-world images with the goal to output a category label for each input image from one thousand object categories. AlexNet's architecture builds upon LeNet and achieved an error rate of 16.4 % during the ImageNet competition in 2012. Other teams in the following years have succeeded in dropping the error rates and making CNNs a more and more popular network for image classification.

2.6 Today's CNN models

Following Marr's framework, Convolutional Neural Networks are classifying neural networks made of layers, both convolutional and non-convolutional. CNNs' architecture consist of an input layer, hidden layers, and a final output layer. The input images are first passed into the input layer of the model, and then to the next first hidden layer, and so on, until the output layer is reached. The 'hidden' layers constitute the part of the algorithm that maps the input to its targeted output. The first layer of the CNN detects the low-level features (i.e edges, etc.), hidden layers build upon these low-level features to detect high-level features (i.e shape, object). Chapter 3 (Theory), includes CNN layers definitions and their role in the neural network.
The entirety of the input images are passed to the CNN model in two runs, the completion of both runs constitute a pass. To say the model has completed *one epoch*, all images of the dataset should be passed once to the model. The two runs are referred to :

- *Feed-Forward Run or Propagation:* During this step, the algorithm passes the input images through all the layers of the CNN model
- *Backward Run or Back Propagation:* During this step, the algorithm updates the weights such as to minimize the value of the cost function (Eqn. 3.5).

Chapter 3

Theory

3.1 About CNNs.

3.1.1 What is a convolution?

A convolution is a mathematical operation that transforms two functions into a third function. One can look at the convolution operations undertaken by the convolution layers as matrix multiplication operations. The matrix operations performed by the convolutional layers have as input a matrix that represents the input of the layer (image, or output of the previous layer), and a matrix specific to the convolutional layer and the task it is performing, the latter is called a filter or kernel. When an image is transformed to a matrix, and passed to the first convoluted matrix, the layer performs a matrix multiplication of the image matrix and the filter to create an output matrix (these matrix entries stand for the image's pixels and what they represent in the image). In other words, the layer will slide the filter on the image , and we generate a new pixel in the output image. The filter usually has a smaller size, so the filter will slide on each section of the matrix (some region x of pixels). The total number of outputs of each slide constitute the entries of the output matrix . The difference between convolution operation in image processing and machine learning is that in CNNs, filters are not predefined: the value of each filter is learned during the training process.

3.1.2 Types of layers in a CNN.

3.1.3 Convolution layer

A convolutional layer has a number of filters that performs a convolution on a certain input of pixel values. Each one of the filters is applied to the input image (array) a number of times such that it creates a 2-D array called feature map, to which is passed to the ReLU function.

3.1.4 Pooling Layer

A pooling layer reduces the number of pixels to limit the number of needed pixels for the code to make an accurate prediction. The polling layers also uses strides, and they are usually seen between Convolution layers in most CNN architectures. Their role is to control overfitting by progressively reducing the spatial size of the network.

3.1.5 Fully connected Layer

Fully connected Layer is where all the neurons come together, and the highest value is chosen to be the prediction output. All the neurons in this layer have a complete connection to all the neurons in the previous layers. The new artificial neurons outputed are the product of a matrix multiplication followed by a bias offset (Eqn. 3.6). This is the last phase for a CNN network.

3.2 CNN Model Parameters

Some of the most important parameters are:

- **Strides size.**
  A stride can be compared to a window used by the algorithm to divide the input image into patches of adjacent pixels. (see Fig. **??**). The smaller the stride, the more information is lost from the input image, and the less accurate the predictions will be.

- **Filters size.**
  CNN use filters to get patches of the size of the filter of the input image. How big those patches are depends on the filters' size. Filters use strides to move along the images' pixels.

- **Pooling strides size.**
  The type of pooling layers used in the Dial Reader are max-pooling layers. A max-pool layer reduces the the dimensions of its input using strides of a certain size. The max-pool layer only keeps the most important pixels above a certain threshold, and by doing so, it prevent the model from generalizing. The role of the Pooling Layer is to control overfitting.

- **Choice of optimizer:**
  Adaptive Moment Estimation, or Adam Optimizer[] is a popular choice amongst optimizers in deep learning. Adam Optimizer is compared to adaptive learning-method algorithms, but implements that algorithm in a faster manner (see Adam Optimizer section below).

3.2.1 Back-propagation Algorithm

Back-propagation Algorithm controls the gradient descent of the CNN model. The CNN model has a gradient whose parameters needs to be updated until the global minimum is reached. The gradient descent is applied to the cost function and modifies the weights of the artificial neurons. The weights are changed according to a learning rate. The learning rate controls by how much the weights need to change to allow the model to make as accurate prediction as possible.

3.3 Adam Optimizer

3.3.1 Definition

Adam Optimizer is a adaptive gradient descent algorithm (Definition of gradient descent can be found in the section 3.5.1). Unlike the stochastic gradient descent (SGD) optimization algorithm that learns the learning rate of the CNN model as images are passed to it. It is used to control the rate of the gradient descent at finding the global minimum. The faster the gradient descent gets to the global minimum, and the less it oscillates throughout the process the more accurate the model will be once the training phase is done.

3.3.2 Adam's Parameters

Adam Optimizer controls the gradient descent using two parameters: momentum m_t and exponential moving average squared gradients v_t , where time t represents iterations or epochs in our implementation of CNN:

- m_t momentum, which controls the rate at which the model converges towards the minimum:

$$m_t = \beta m_{t-1} + (1 - \beta)[\frac{\partial C}{\partial w_t}] \tag{3.1}$$

m_t = aggregate of gradients at time t [current] (initially, $m_t = 0$)
m_{t-1} = aggregate of gradients at time $t - 1$ [previous]
w_t = weights at time t
w_{t+1} = weights at time $t + 1$
α_t = learning rate at time t
∂C = derivative of Cost Function
∂w_t = derivative of weights at time t
β = Moving average parameter (constant, 0.9)

This parameter is used to track and maximize the momentum used by the learning rate to reach a convergence faster.

- v_t is exponential moving average squared gradients:

$$v_t = \beta m_{t-1} + (1-\beta)[\frac{\partial C}{\partial w_t}] \exp 2, \tag{3.2}$$

w_t = weights at time t
w_{t+1} = weights at time t+1
α_t = learning rate at time t
∂C= derivative of Cost Function
∂w_t = derivative of weights at time t
v_t = sum of square of past gradients. [i.e sum($\frac{\partial L}{\partial w_t}$)] (initially, $v_t = 0$)
β = Moving average parameter (constant, 0.9)
ϵ = A small positive constant ($10 \exp -8$)
This algorithm makes it so that Adam Optimizer updates the weigh

3.3.3 Adam's algorithms

Adam Optimizer updates the weights of the artificial neurons in the CNN layers using two algorithms, each using one the parameters just mentioned above:

- a momentum algorithm that computes the rate of the gradient descent using the m_t parameter. This algorithm helps the gradient descent algorithm converge towards the minima faster.

$$w_{t+1} = w_t - \alpha m_t \tag{3.3}$$

- the Root Mean Square Propagation (RMSP), is also an adaptive learning algorithm that computes the exponential moving average squared gradients. RMSP is an adjustment of the older Adagrad algorithm that takes the cumulative sum of squared gradients. Adam itself is an upgrade from RMSP optimizer.

$$w_{t+1} = w_t - \frac{\alpha_t}{(v_t + \epsilon) \exp 1/2 \times [\frac{\partial C}{\partial w_t}]}, \tag{3.4}$$

3.4 CNN Model Hyperparameters

A CNN model has hyperparameters that control how the model is structured and how it is trained. The hyperparameters that have been tuned to get the dials predictions are stated below:

- **Activation functions.**
  An activation function is a non-linear function used to stimulate (activate) the artificial neurons.
  - The function used in the Dial Reader model is the ReLu (Rectified Linear Unit) function, and it is applied to every neuron in the convolution layer. The ReLu function is used for all layers, except the last one which uses a softmax activation function. It can be expressed as:

$$\text{f(x)}= \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

  - The Softmax function is an activation function is used for multi-class predictions. The softmax function converts its inputs to a normalized probability distribution.

- **Number of epochs.**
  The number of epochs is the number of times the input has been passed in the model and the weights have been updated.

- **Learning rate.**
  The learning rate controls how fast much the weights are updated in each pass. It is most important hyper-parameter of the training algorithm. The larger the learning rate, the faster the weights are updated, and thus the faster the training phase is.

3.5 CNN Model Evaluation Metrics

3.5.1 Cost function

A cost function is used to maximize the probability of the model making the right guess about the input images. It is a value that represents how well the guess of the CNN was. This value is calculated using variables such as the input of the layer, the weights and biases in the neural network of each artificial neurons, as well as the desired output.

During the back-propagation stage of the training, the model optimize the weights of the CNN.

This process is called back-propagation because the weights are updated from the last layer to the first: the opposite direction of how the weights are updated in the feed-forward.

The cost function used in this project is the cross-entropy function.
The cross entropy of a neuron in CNN is:

$$C = -\frac{1}{n}\sum[ylna + (1-y)ln(1-a)], \tag{3.5}$$

where n is the number of images in the training dataset.

In a CNN model, each neuron in the first layer is connected to a patch of pixel a_j. Each neuron has a weight w_j and a bias b. During the feed-forward phase, the following expression is applied to every a_j.
Let z be the weighted sum of the input images, such as:

$$z = \sum w_j a_j + b, \tag{3.6}$$

such as $a = \sigma(z)$, where σ is the non-linear function used in hidden layers of the network. In this project, the non-linear function σ used is ReLu (3.4). The gradient of the cost function (3.5) happens to be the rate at which the neurons in the layers learn. The gradient of the cost function is called the gradient descent. In other words, the rate at which the weights change such that the output of the cross entropy function is minimized. The gradient of the cost function with respect to the parameters (neuron's weight and bias) is calculated by the back propagation algorithm and can be expressed as such:

$$Cw_j = \frac{1}{n} \sum x_j(\sigma(z) - y) \tag{3.7}$$

3.5.2 Confusing matrix

Confusing matrix is used to evaluate the tendency of the model to confuse readings of the dials. Most errors are assumed to occur in dial images where the needle is in between two regions that are adjacent to each other. The values of the confusing matrix give us an idea on how well the model is capable of recognizing dials of different readings. A confusion matrix can be read as it follows:

- The rows of the confusion matrix represent the prediction labels.
- The columns of the confusion matrix represent the actual labels.
- The correct predictions are distributed along the diagonal values of the matrix.

3.5.3 Training Dataset and Validation Dataset

A well trained model would not memorize features from the training dataset images, but rather apply the "lesson" it learned during the training to make a best guess about the label category of new images that are passed to the model. A common technique used to avoid memorization consists of splitting the training dataset into two sets:

- A training set, used to evaluate how well the model fits the images in the training set.

- A validation set, used to evaluate the capacity of the model to generalize and make predictions about images it has never seen before.

The accuracy of the model on both sets is calculated at each epoch. The training accuracy and validation accuracy are good metrics when it come to evaluating the overall performance of the model during the training phase. A good way to detect a case of memorization is to compare the model's training and validation accuracy:

- if the training accuracy is higher than the validation accuracy, then the model is *overfitting*.
  Overfitting happens when the neural network does not perform well on the testing dataset. It is an indication of memorization, the model fails to generalize the features it learned during the training.

- if the training accuracy is lower than the validation accuracy, the model is *underfitting*.
  Underfitting indicates that the model has failed to learn a way to map the input to the output, and that the model or the type of neural network used with the dataset is inappropriate.

- if the training accuracy is close enough to the validation accuracy (meaning). This means that the model is a rightfit and that it is capable to generalize to new input images. The model's accuracy is calculated using the following expression:

$$\text{Accuracy} = \text{correct predictions} \,/\, \text{total predictions}$$

Chapter 4

Implementation

4.1 Why Machine Learning (ML)?

In the early stage of the project, an attempt to automate gasmeter readings using OpenCV and other libraries such as Numpy to detect the needle and calculate the angle the needle is making with a horizontal line that goes through the origin of the circle represented by the dial. However, this technique turned out to require more involvement from the user, since the user will have to manually determine threshold value to only keep the needle's dark pixels. The threshold will also differ from image to image if some change in the lighting happens while the images are captured. The other disadvantage is that the digits will still require some kind of ML algorithm to be detected. For all these reasons, Machine Learning has been chosen as the core technique used to accomplish the goal of this project.

4.2 Overview on the Gasmeter-Reader

CNN is a neural network that classify images to certain number of labels. To put a limit on the number of labels used during the training, the dial is assumed to be divided into 10 regions (Fig. 4.10), such that each region represents one category, or value indicated by the dial (Table 4.2). This assumption governs how the images of the training and testing dataset are labeled.

The project's goal is then to classify the gasmeter dials' images into 10 categories. To accomplish that task, a program is designed to do the following:

- Capture images from the Gasmeter using a USB webcam
- Segment the Gasmeter images to get the dial and the digit images
- Create a labeled training and testing dataset to train the CNN model's to classify the dial images and test it
- Create a labeled training and testing dataset to train the CNN model's to classify the digit images and test it

- Train two CNN models:
  - a Dial Reader that classifies the dial images
  - a Digit Detector that classifies the digit images
- Evaluate the two models on the testing dataset

4.3 Limitations of the Gasmeter Reader

The CNN model used in the Gasmeter Reader has been originally built by to detect gauges. The github account for the Deep-Gauge program is $https://github.com/oci-labs/DeepGauge-ML-Demo$. The CNN model part of the Ensemble model from the Deep-Gauge project was adapted to train segmented images obtained from the gasmeter in Fig.1.1.
Even though the program described in this paper would only work on digits and/or dials that look like in Fig.4.8 , Fig.4.6 , the two CNN models can easily be trained with different labeled images to detect a new type of digit or needle.

4.4 Data Collection

4.4.1 Image Capture

The Gasmeter-Reader program collects data using a python script called $Image_capture.py$, the script contains a description of what the file does, and it should prompt the user to press the right keys. The images captured through the $Image_capture.py$ file are used to create the training, validation, and testing datasets for both the digits and the dials. $Image_capture.py$ can be found in the $Data_Prep$ folder.
The Gasmeter images are taken using a USB webcam at an interval determined by the user. The images are saved using the date and time at which they were taken as a filename. The program will prompt the user to determine the time interval between each two successive images taken by the webcam: the time count start when the user presses 's' and 'c' to stop.

4.4.2 Image Segmentation

The file *crop.py* in the $Data_Prep$ folder will prompt the user to segment the Gasmeter images into to multiple sub-images: one for each dial and digit (6 segments in total, see Fig.4.8). The goal of image segmentation is to restrict the number of pixels in the input images to only the pixels needed to read the gasmeter. The program will guide the user through selecting all the regions needed to make a reading from the gasmeter. The image segmentation is done manually, so it is crucial to keep the camera and gasmeter still. The user will have to crop the gasmeter images by selecting the region where the dials and digits are located. The coordinates of the selected regions are then used to extract the

dials from all the gasmeter images taken by the webcam. Each segment will be passed to either a trained the Dial Reader, or the Digit Detector that will return the segments' readings.

4.5 Training and Testing Datasets Preparation

4.5.1 Dial Reader Datasets Preparation

Since CNN is a supervised model, it requires from each of the images in the dataset to be accompanied with a label. Unfortunately, no consistent pattern has been found in the captured images in the 'Data acquisition' section to be able to label the images automatically.

To resolve this problem, the training and validating datasets were generated using one segmented dial images (Fig. 4.4). An online image application was used to separate the needle (see Fig.4.6) from the background of the image (see Fig.4.7). The needle's image was then rotated around the background image using OpenCV to create images where the needle is at different positions around the dial. The image generator code saves each generated image in a folder named after the image's label. This folder structure is required by the sklearn library used in the image generator code. Sklearn library is used to upload the data images and their corresponding labels.

The data images can be accessed in the directory: $DeepGauge - ML - Demo$. The image generator code can be found in the $generate_dial_images.py$ file in $Data_Prep$ folder. The images are using the intervals and labels shown in Fig.4.1. The needles are rotated around the dial background image by 1 degree starting from 0 degree representing the needle at pointing at 12 o'clock. Once the rotation angle reaches 37, the code will label the generated image according to the angle/label map in the table below, (Fig. 4.1). The images area created to be the same size 79×79, and it is worth noting that the CNN model has only been trained and tested on images with the same size. After the training phase, the Dial Reader model is be able to map each image from the testing dataset to a category. 20% of these images were used to make the validation dataset, and the rest for the training dataset. More images were later created after the training phase to test the CNN models decision making.

4.5.2 Digit Detector Datasets Preparation

To label the segmented digit images and use them to train and test the CNN model, a Stepper Motor Controller was used. The Stepper Motor Controller is used to control and manipulate the rate at which the gasmeter's needle rotates (pictured in Fig. 4.1) and updates its digits and dial readings. Turning the black knob in the anti-clockwise direction will make the cfm rate to negative values and safely drive the meter backwards. The Stepper Motor Controller simulates the use of 6 different household gas appliances that

labels	angle interval start	angle interval end
0.05	0	36
0.1	37	72
0.15	73	108
0.2	109	144
0.25	145	180
0.3	181	216
0.35	217	252
0.4	253	288
0.45	289	324
0.5	325	361

Table 4.1: Table of the angles used to label the 10 different regions of the gasmeter's dial.

consume natural gas at different rates in the following range: 0-9.9 cfm. The appliances that the Stepper Motor Controller allows us to simulate are the following:Boiler, Attic Heat, Den Heat, Water Heat, Dryer, Oven Range.

The Stepper Motor Controller can be used according to two modes:

- **Manual mode:**
  Click on the A steady cfm rate of 4.9 cfm for all appliances.
- **Crazy mode:**
  Click on the to switch on the appliances to be simulated on crazy mode: the selected appliances will turn themselves on and off following a programmed schedule. The rate of the appliances will randomly change for each appliance taking a value in the range between 0.0 to 9.9 cfm. This random selection is assumed to be closest to how house appliances are used throughout the day.

4.6 Dial Reader and Digit Detector Models Architecture

Reading the gasmeter is an image classification problem similar to the problem that the Deep Gauge project had solved in the past. The Deep Gauge project uses images of analog gauge images to train a CNN-based deep learning algorithm to read gauge dial readings. This project implements the same architecture of the CNN model used in the Deep Gauge project using Tensorflow. The CNN was trained to read the gasmeter's dials readings, as well as the digits on the gasmeter's face. Adam Optimizer and cross entropy (Eqn. 3.5) were used as respectively the optimizer and the cost function of our model. The architecture of the CNN is represented by Fig.4.9: the input images image are first passed to three convolutional layers, each with ReLU as the activation function:

- The first convolutional layers has 45 filters of size 17 with the following stride: [1, 7, 7, 1].

- The second convolutional layers has 17 filters of size 7 with the following stride: [1, 5, 5, 1].
- The third convolutional layers has 7 filters of size 1 with the following stride: [1, 1, 1, 1].

Two max-pooling layers are used in the CNN model, one after each of the first two convolutional layers. The first max pool layer uses a stride of size [1, 4, 4, 1], while the second one uses a a stride of size [1, 3, 3, 1]. The convolutional and max pool layers are followed by two Fully-Connected (FC) layers, each detecting 86 features. FC layers are one-dimentional non-convolutional layers that have the size of the number of categories. All neurons in an FC layer are connected to all neurons in the previous layer. Its role is to flatten the input information (features). The sofmax layer then applies the softmax function to the output of each category in the last FC layer, and comparing all the values of each category, it return a category label for the dial or digit image.

Figure 4.1: Stepper Motor Controller

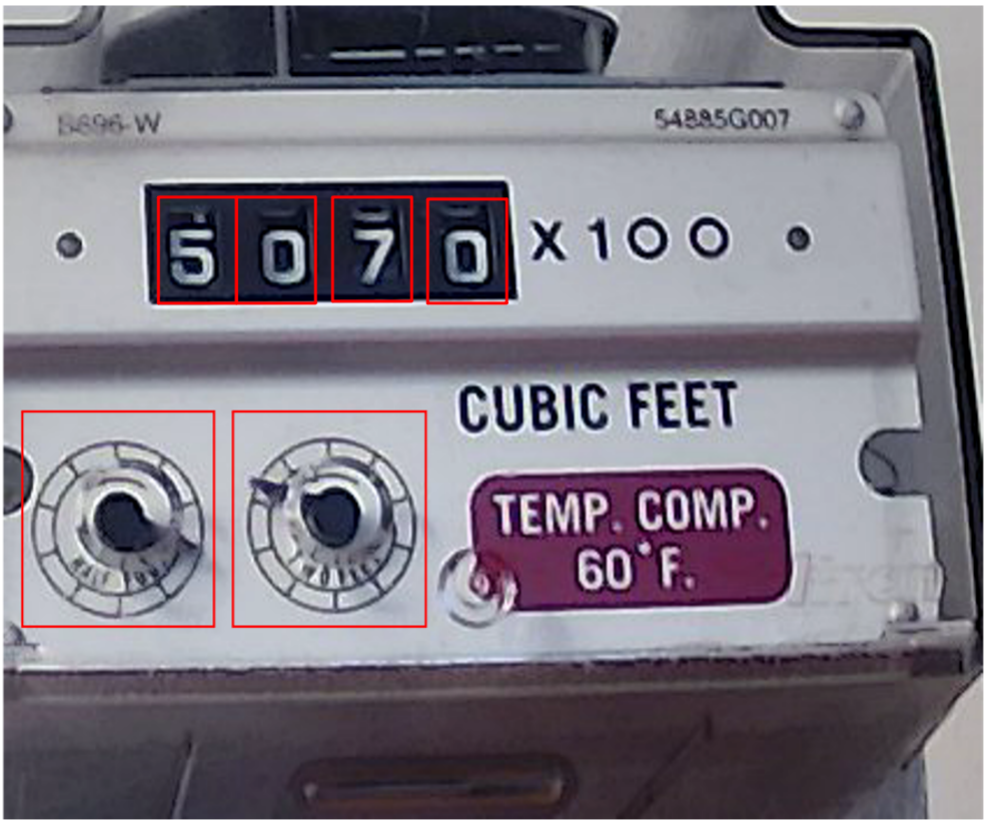

Figure 4.2: Gasmeter images taken by an usb webcam before segmentation, with the to-be-cropped region of interest selected in red

Figure 4.3: Dial images after image segmentation step

Figure 4.4: Digit images after image segmentation step

Figure 4.5: Example images from the digit training and validation datasets

Figure 4.6: Example of needle images used to generate the training, validation, and testing datasets.

Figure 4.7: Example images of the background images used to generate the training, validation, and testing datasets.

Figure 4.8: Example images from the generated the training and validation datasets

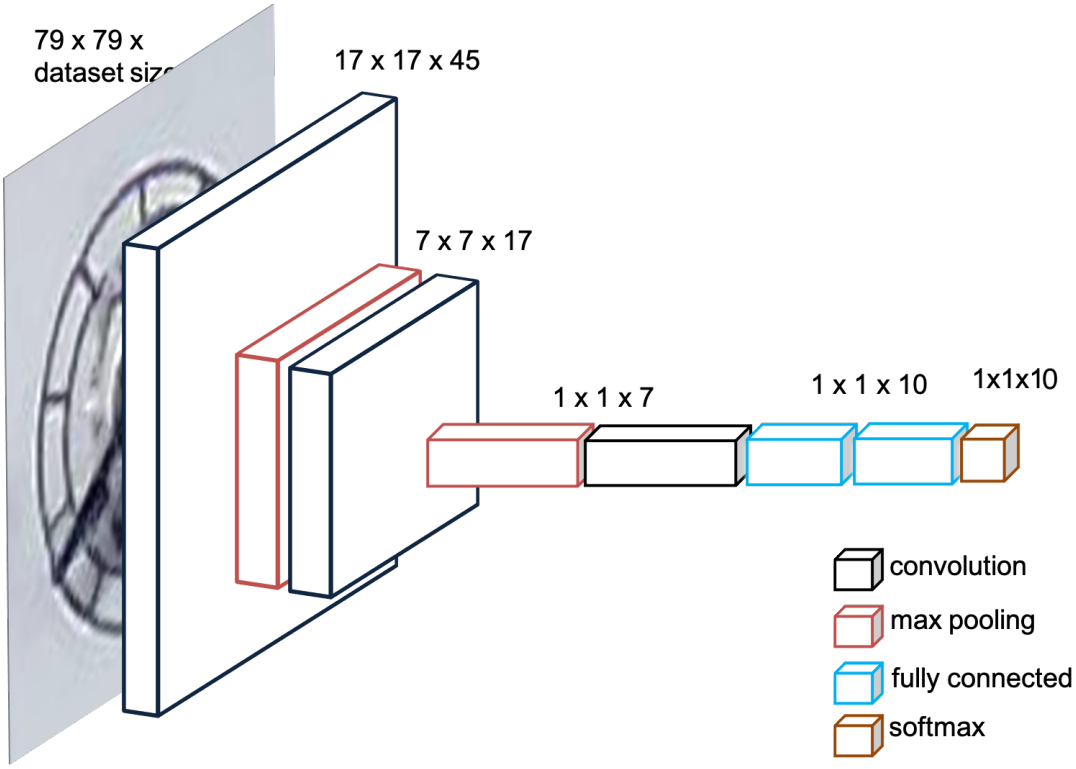

Figure 4.9: Image representing the CNN model architecture.

Figure 4.10: The goal of the software is to classify the dials depending on the location of the needle. Assuming that each increment of 0.05 represents a category, the dials will have 10 different regions that will determine the value of the needle.

Category	**Label**
0	0.05
1	0.1
2	0.15
3	0.2
4	0.25
5	0.3
6	0.35
7	0.4
8	0.45
9	0.5

Table 4.2: Table of the assigned categories and their target labels. The target labels are the values that the software is expecting to

Chapter 5

Results and Discussion

5.1 Dial Reader Model Analysis

5.1.1 Dial Reader Training Evaluation

The initial training dataset is generated using Python and OpenCV to create dials with different type of needles and background dials. The background and needle images used to create this dataset are diverse in shape, color, etc. This is done so to make sure that the model picks up on those same features when presented with new images extracted from the images captured from the usb webcam.
The image generated during the training phase are split into two sub-datasets: 20 % of the images are assigned to the validation set, the rest is assigned to the training set. The training dataset comprises 6,828 dial images, while the validation dataset includes 1,708 images.
The Dial Reader model was trained at a learning rate of 0.0005 across 187 epochs.

Cost Minimization

One of the main goals of the training phase is to minimize the value of the cost function as the number of epochs increases. The values of the cost function for both the training and validation datasets are calculated at each epoch during the training phase and are plotted as shown in Fig.5.1. The curve obtained from cost as a function of epochs during the model's training gives important details about the learning process of the model: the stepper the slope of the curve the more learning the model does.
As the number of epochs increases, the model's cost value drops almost exponentially. The value of the cost function seems to stabilize around the 187th epoch.
Different hyper-parameters were tested on the datasets of different sizes. With a learning rate of 0.0005 and a training dataset of 6,828 images, the cost value starts slowly increasing back after the 187th epoch. To prevent the cost vs epochs curve to grow up after 187 epochs which would lead the model to overfit on the training model, early-stopping is used to interrupt the training around the 187th epoch as seen in Fig.5.1. As the number

of epochs approaches the 187th epoch, the curve starts to stabilize and doesn't not show any sign of a potential exponential growth that could lead to the model overfitting on the training dataset. It can be inferred that the model was trained for the optimal number of epochs.

Optimization of the Validation Accuracy .

After each epoch, the training and validation accuracy are calculated then plotted as shown in Fig.5.3. The validation accuracy is usually expect to be higher than the training accuracy, since the validation indicates how to model will perform on real data outside of the training phase. The training dataset accuracy indicates how well the model is picking on features (pixels) from the input images, while the validation dataset is used to compare how well the model will perform on the testing dataset. Fig. 5.3 shows the evolution of the training accuracy compared to the validation accuracy. The model start to immediately learn and pick on some features from the input dataset, it then stabilises for around 10 epochs. Overall, the model seems to pick up on the features and the validation accuracy seems to be higher than that of the training. After the 187th epoch, the training accuracy reaches 97.95 % and the testing accuracy 99.24%. Based on these values, it can be concluded that the validation accuracy is higher than that of the training. Therefore, the model is capable of generalizing what it has learned to during the training phase to make predictions with an expected accuracy of 99.24% on a new dataset. Since the training accuracy and the validation accuracy are close enough, and it can be inferred that the model is a rightfit to the training dataset.

5.1.2 Dial Reader Performance on Testing dataset.

The model predicted the images category labels of the testing dataset with an accuracy of 95.51%. The prediction accuracy of the model can be calculated from the confusing matrix diagonal input divided by the total number of prediction made.
The Dial Reader's confusion matrix in shown in Fig.5.5 visualise the performance, and more specifically the errors made by the model. Most errors seem to occur between adjacent regions, just as expected. The Dial Reader model has predicted the labels of 500 '0.05' images right, and confused 2 of them for 0.1' images. The model has also predicted 17 images to be in the 0.05 category, while they belong to the 0.5 category. The highest confusion error happens in the '0.4' region: the model seem to have confused 25 '0.4' images for being '0.45' images, and 11 '0.4' images for '0.35' images. These results prove that CNN can successfully classify the dials images in their right category. Therefore, it can be concluded that the Dial Reader can detect the dial's readings to an accuracy of 95.51%.

5.2 Digit Detector Model Analysis

5.2.1 Digit Detector Training Evaluation

Just like with the Dial Reader 20 percent of the set images created during the training phase was used to create a validation set. The testing set images is created after the training, they are passed to the CNN model for the first time during the testing phase. The training dataset comprises $INSERT_NUM$ images, and the validation dataset $INSERT_NUM$ images.
The Digit Detector model was trained at a learning rate of 0.0003 across 200 epochs.

Cost Minimization

Fig. 5.1

Optimization of the Validation Accuracy

Fig. 5.3

5.2.2 Digit Detector Performance on Testing dataset.

The model predicted the digits category labels with an accuracy of 92.92% .
The Digit Detector's confusion matrix in shown in Fig.5.6.
Therefore, it can be concluded that the Dial Reader can detect the dial's readings to an accuracy of 92.92% .

5.3 CNN Models Predicted Accuracies

It can be concluded that the Gasmeter Reader can detect the digits from the gasmeter's face to an accuracy of 95.51%, it can detect the with an accuracy of 92.92%.

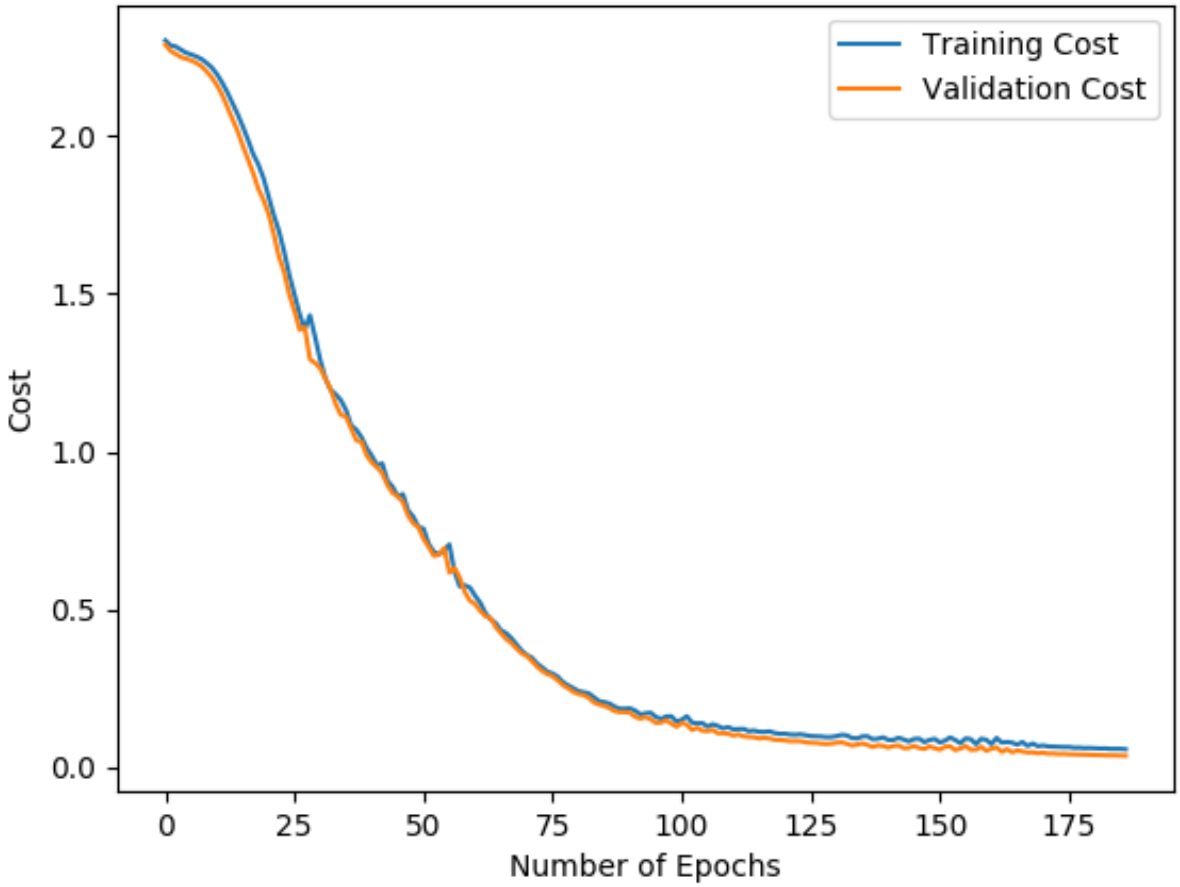

Figure 5.1: Dial Reader's cost value as a function of the number of epochs

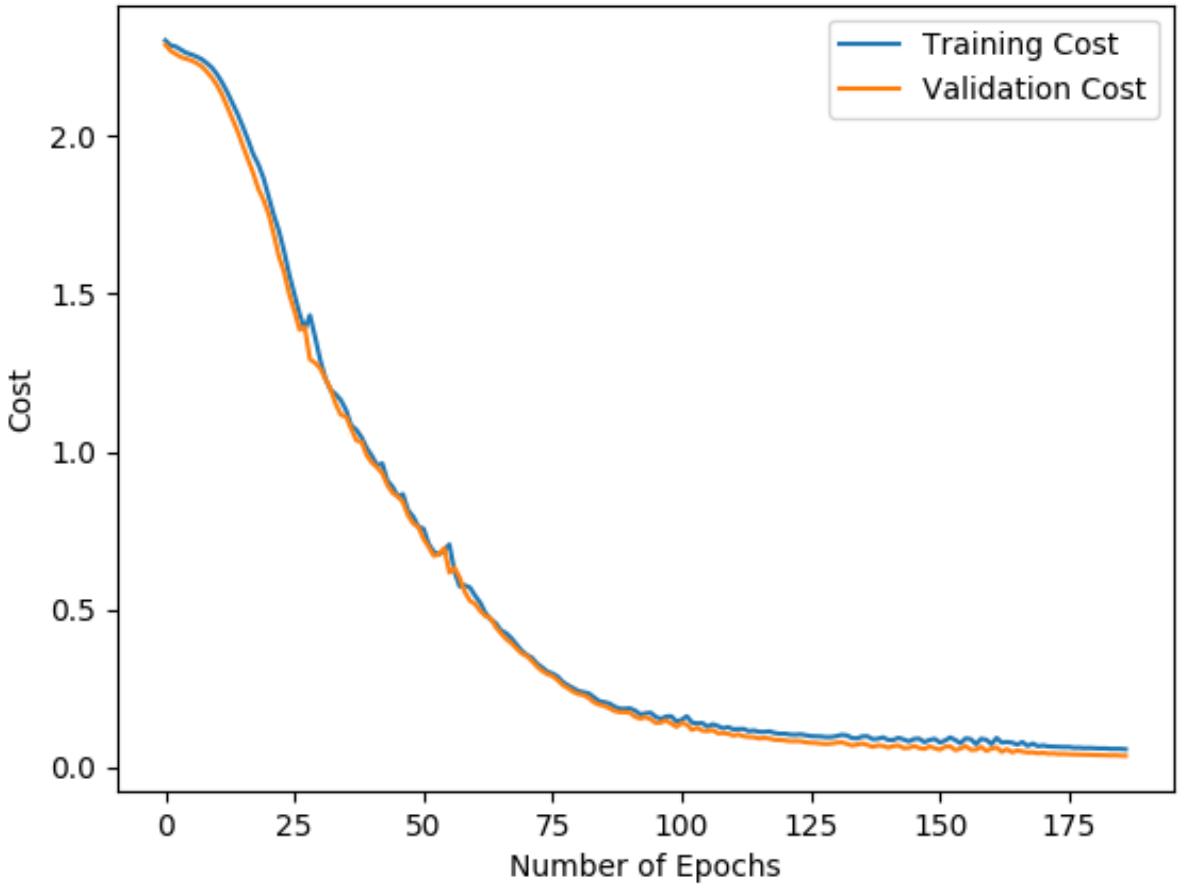

Figure 5.2: Digit Detector's cost value as a function of the number of epochs

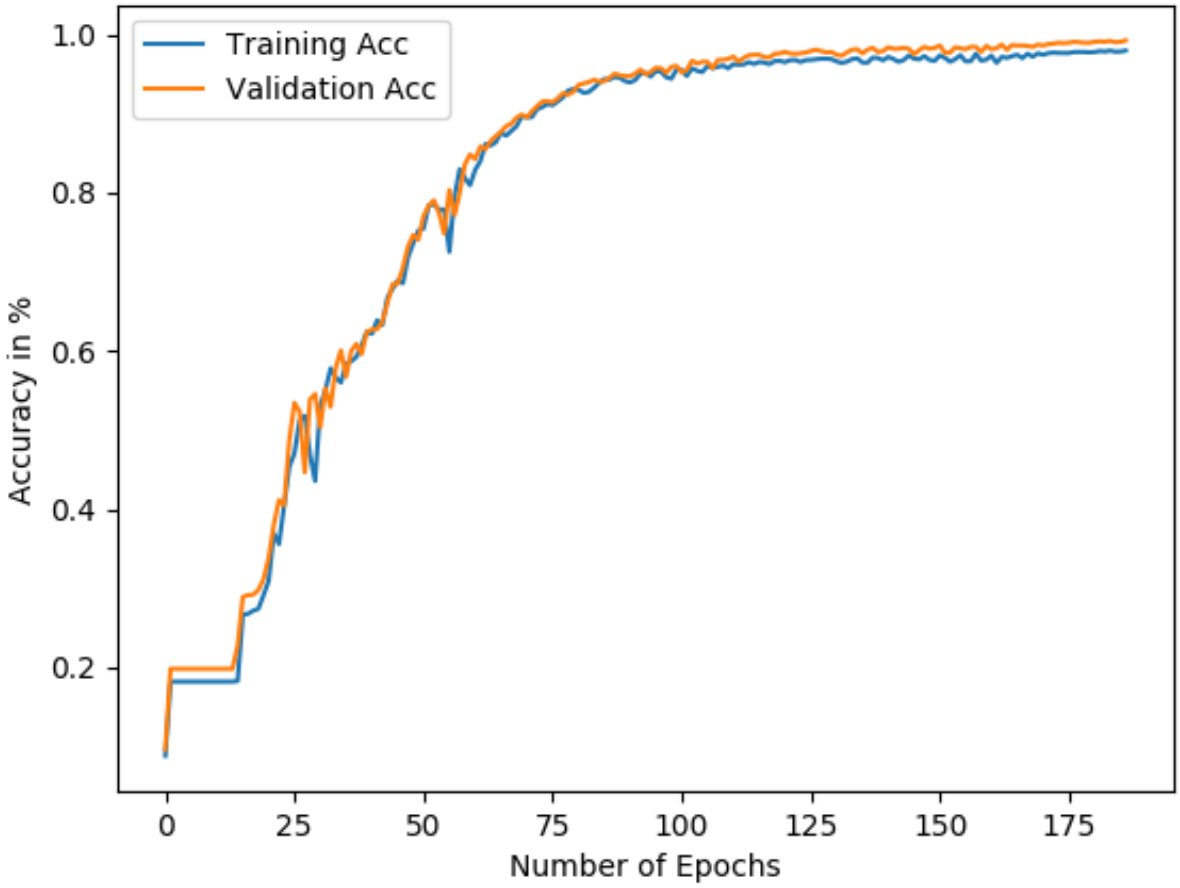

Figure 5.3: Dial Reader's training and validation accuracy as a function of the number of epochs

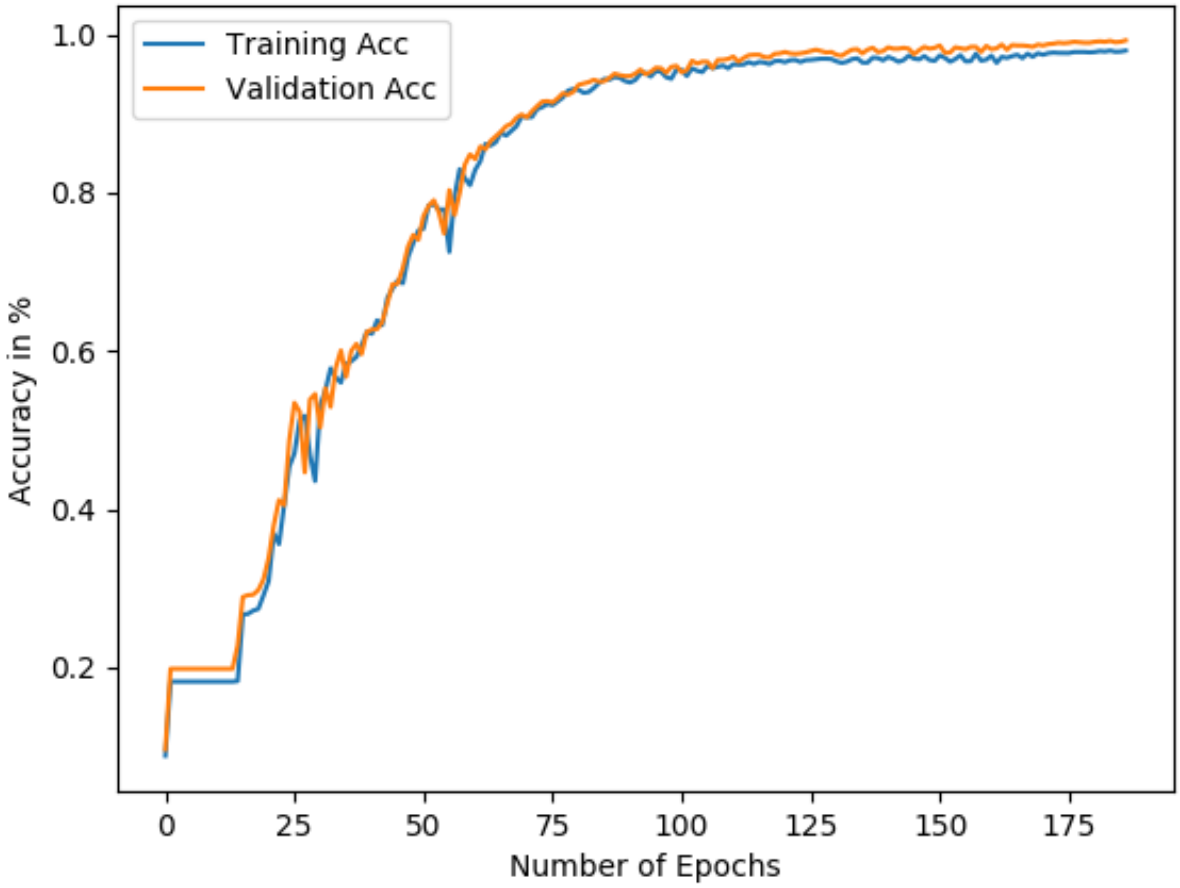

Figure 5.4: Digit Detector's training and validation accuracy as a function of the number of epochs

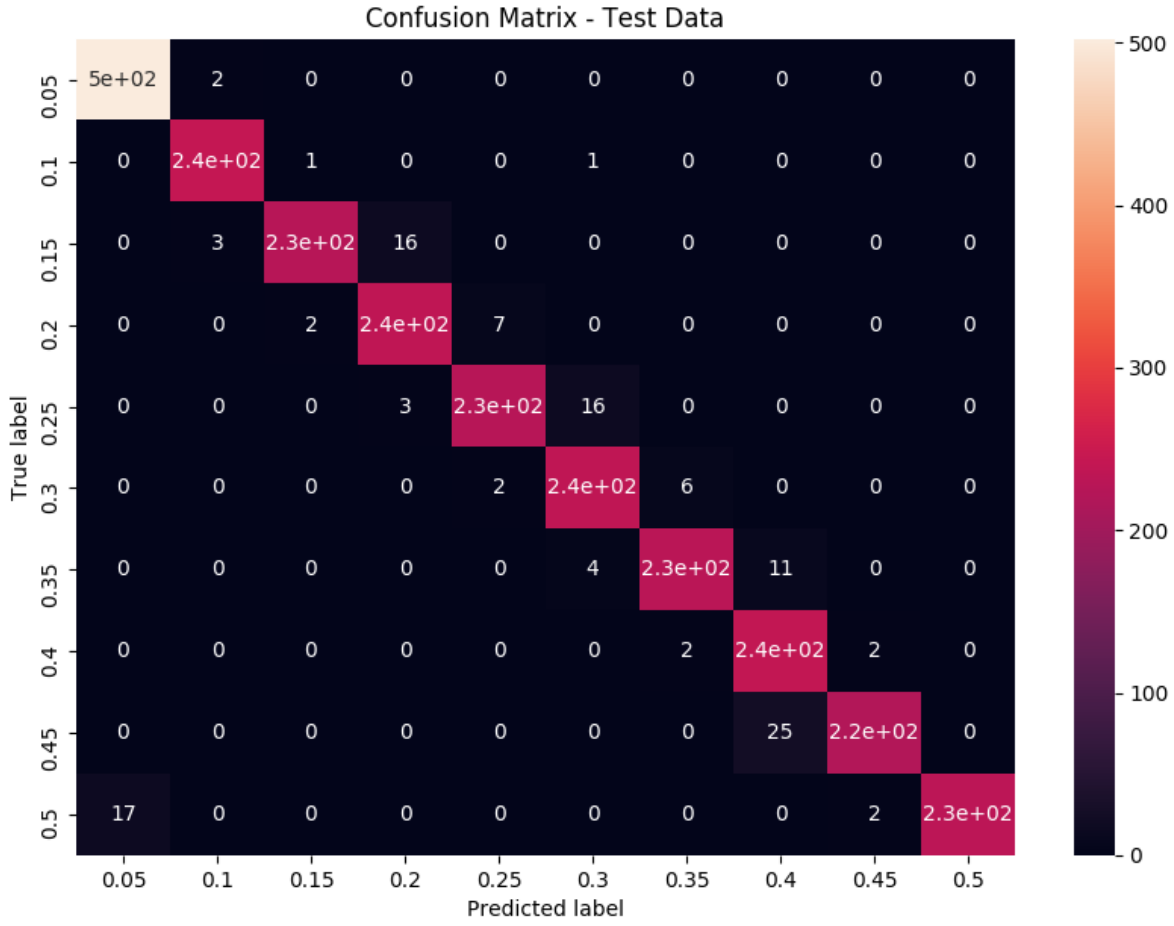

Figure 5.5: Dial Reader's confusion matrix

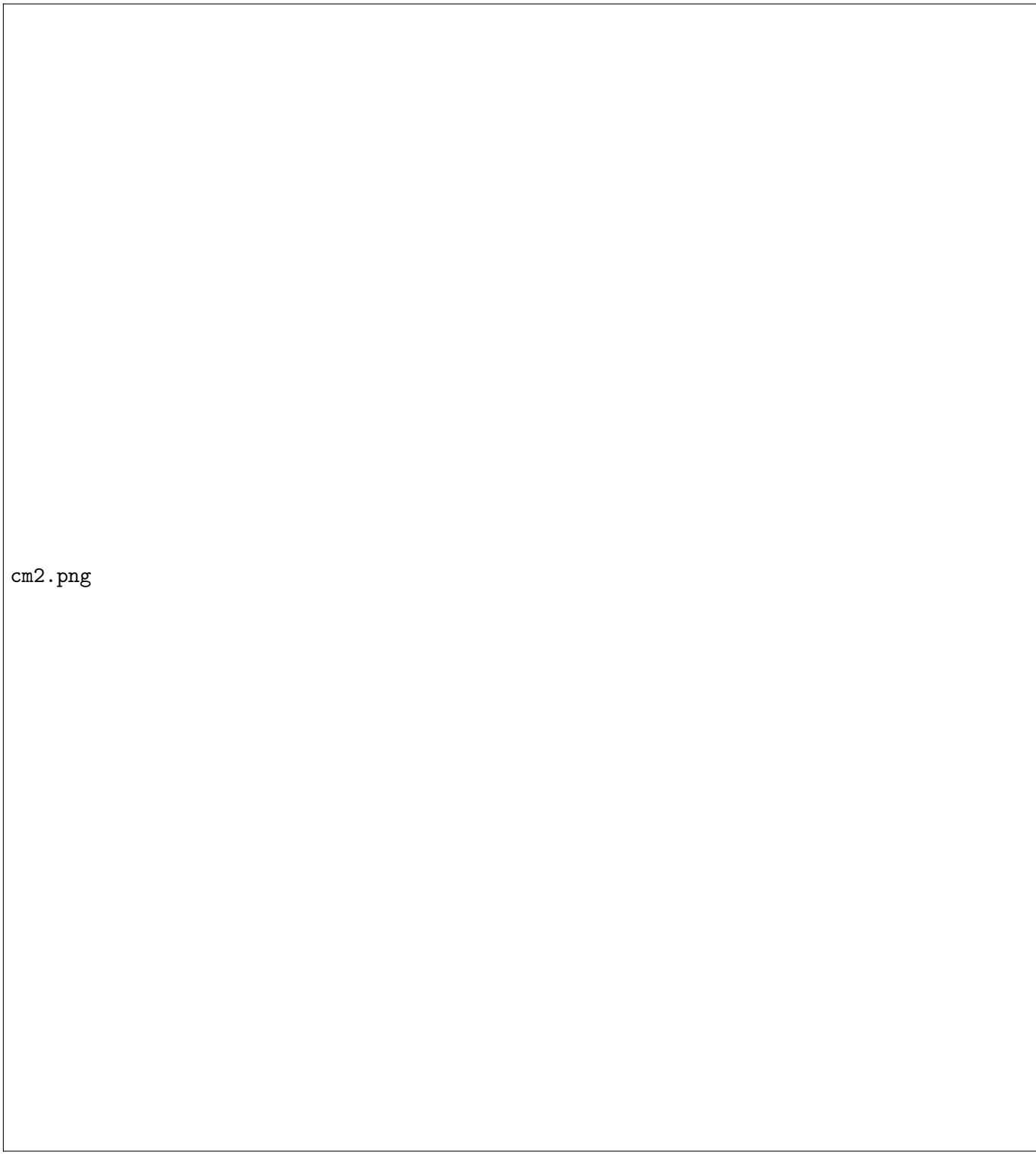

Figure 5.6: Digit Detector's Confusion matrix.

Chapter 6

Conclusion

CNN has been proven to be efficient in reading the gasmeter dials. The model trained in this project has achieved a high accuracy of 95.51 % in detecting the digits and an accuracy of 92.92% in detecting the reading of dial. Since the two models have proven to successfully read analog gauges, and have successfully read both the digital and analog display of the gasmeter, then it can be assumed that the two CNN models can be used to read any type of meter as long as they are trained with images similar to the ones they will be tested on.

Chapter 7

Future improvements

- Add more images to the files in the folder data in the following directory:$DeepGauge-ML-Demo$ to train the CNN models to read a water meter, or electric meter, etc ... .
- Create a program that can retrieve the readings outputted by the CNN models and use them to display graphs of the gas consumption over a period of time.

References

[1] OpenCV Bradski, G. (2000). The OpenCV Library. Dr. Dobbx27;s Journal of Software Tools.

[2] TensorFlow version 1
TensorFlow Abadi A., et al. 2016. TensorFlow: Large- Scale Machine Learning on Heterogeneous Distributed Systems. arXiv:1603.04467 [cs] (March 2016). Retrieved October 28, 2019 from http://arxiv.org/abs/1603.04467 Keras Chollet, F., et al. 2015. Keras. Retrieved from https://keras.io

[3] HUBEL, D. H., WIESEL, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. The Journal of physiology, 148(3), 574{591. https://doi.org/10.1113/jphysiol.1959.sp006308

[4] Adam Optimizer,
Adam Kingma, D. P., Ba, J. 2015. Adam: A Method for Stochastic Optimization. (2015). Retrieved October 28, 2019 from https://dare.uva.nl/search?identifier=a20791d3-1aff-464a-8544-268383c33a75
https://www.geeksforgeeks.org/intuition-of-adam-optimizer/

[5] Rostyslav Demush: A brief history of computer vision and convolutional neural networks,
https://medium.com/hackernoon/a-brief-history-of-computer-vision-and-convolutional-neu

[6] https://www.webopedia.com/definitions/pixel/: :text=Pixel.

References