8-2020

Object Orientation Detection and Correction using Computer Vision

Amila De Silva
amila.desilva20@gmail.com

Object Orientation Detection and Correction using Computer Vision

by

Amila De Silva

A Thesis

Submitted to the Graduate Faculty of

St. Cloud State University

In Partial Fulfilment of the Requirements

for the Degree of

Master of Science

in Computer Science

August, 2020

Thesis Committee:
Bryant Julstrom, Chairperson
Jie Hu Meichsner
John Sinko

Abstract

Product validation is a manufacturing step in which a product undergoes safety and functionality tests before it is released. Presently, human labor is used in product validation. Automating product validation can alleviate errors caused during the validation process by human error. However, an impediment to automated product validation is the orientation of the product. It is crucial that a product is correctly oriented before the artificial system can run the quality check. This experiment is designed to address the object orientation problem encountered during the object recognition step of the product validation process. Three techniques were examined to solve the object orientation problem. The first is Principal Component Analysis, and the other two are supervised machine learning techniques: Classification and Regression. The results of the three methods are surveyed, and the limitations of the methods are discussed. The machine learning models are better suited for solving the object-orientation problem in comparison to Principal Component Analysis. The classification model made better predictions twice as often on average than the regression model.

Acknowledgement

Reflecting on my academic career, I am grateful that I have made it up to this point. The journey has by no means been a lone woman's journey. I owe and extend many thanks to my family in Bahrain, and the family I have formed during my time in the United States, for their support and encouragement, my professors who have guided me during my undergraduate and graduate years at St. Cloud State, and my friends for making the journey twice as interesting. Finally, I extend my gratitude to all my well-wishers who took the time to read through my thesis, and offered constructive feedback. Thank you to all.

Table of Contents

Page

List of Tables

Table Page

List of Algorithms

Algorithm Page

List of Figures

Figure Page

Figure Page

CHAPTER 1: INTRODUCTION

Product validation is a manufacturing step in which a product undergoes safety and functionality tests before it is released into the market. Currently, humans perform product validation. The three main inconveniences for industries using human labor are the time taken to validate products, imprecision during validation due to boredom and repetition, and labor charges. If a machine is trained in the product validation process, industries can alleviate these bottlenecks. Although a variety of applications using digital cameras, such as image and face recognition, are prevalent in industry, product validation using artificial intelligence is just beginning.

The training process of product validation for artificial systems is cumbersome. The process may also vary depending on the product. For instance, the quality check of electronic goods is rigorous and detailed when compared to the quality checks of toys. Product validation includes the physical appearance of the product, its functionality, and the safety of the product. One impediment during the automated product validation of the physical appearance of the product is the orientation of the product. It is crucial that the product is correctly oriented before the artificial system can run a quality check. Object orientation is a crucial pre-processing step for object recognition, since the system needs to be able to identify multiple representations of the same object. This simple task for humans is challenging for machines. The goal, then, is to facilitate image recognition by identifying and correcting the product's orientation.

Humans use the context of the image to infer the orientation of an object, and in some cases, the orientation of the entire setting. In Figure 1, humans would immediately conclude that the mug in the image is upside down, since the rest of the objects in the image are standing up correctly. Similarly, in Figure 2, the image would be considered upside down since all the objects in the image are inverted. The context of the image can be exploited to train an artificial system

on the orientation problem. However, this may be a cyclical problem. Object orientation is necessary for image recognition, but at the same time, image recognition is necessary to determine an object's orientation. This problem is avoided by presenting multiple representations of the object during the training phase of the artificial system.

Figure 1

Using contextual information of the objects in the image to conclude that the mug is upside down

Figure 2

Using the context of the objects in the image to conclude that the image is upside down

Product validation is a multi-component process. This experiment is designed to address the object orientation problem encountered during the object recognition step. Chapter 4 describes three types of experiments that were conducted to solve the object orientation problem, one using Principal Component Analysis, and the other two using the supervised machine learning techniques of Classification and Regression. The results of the three methods are surveyed, and the limitations of the methods are discussed in Chapter 5. The machine learning models are better suited to solving the object-orientation problem in comparison to Principal Component Analysis. The classification model made about twice as many better predictions on average than did the regression model.

CHAPTER 2: PROBLEM STATEMENT

The goal is to define the base orientation of an object, detect the orientation of an object, and correct the object's orientation if it is identified as different from the base orientation. The accuracy of an object's orientation is subjective; it is determined by the developer by picking features that define the orientation. The use case of this experiment is to identify objects' orientations in a factory setting using a simple model that can be trained in under ten minutes, and still produce accurate results. The objects used to train, validate, and test the experiments are boxes. The images analyzed throughout the experiment are two-dimensional grayscale representations of three dimensional structures; each image contains an aerial view of a box. The aerial image of size 100 x 100 pixels is used to determine whether the box in the image is in the correct orientation. The base orientation of the object is also an aerial image of the box. If the snapshot of the box in the test image is incorrect, then the image is corrected to align the box to the correct orientation. Sections 4.1 through 4.4 describe methods to determine the angle of a box in an image. Section 4.5 considers how the orientation of the object can be corrected virtually.

CHAPTER 3: LITERATURE REVIEW

The problem described in Chapter 2 was solved using three methods: Principal Component Analysis (PCA), and two machine learning techniques: Classification and Regression. The first solution applied was Principal Component Analysis. PCA is a mathematical tool for feature extraction of a dataset. As the number of independent features increases, the dimensionality of the dataset also increases, which makes it harder to draw insights from the data. PCA accepts all the features of the dataset, but only considers a subset of features that it deems important to establish correlations between the selected features. The result of PCA is reduced dimensionality of the dataset (Song, Guo, & Mei, 2010). The process also presents eigenvectors that describe the directions along which the data vary most. The eigenvector has a unique value, or eigenvalue, associated with it. Using the eigenvalue and eigenvector, the angle of the new XY-axes can be determined.

Once an image is converted to grayscale, its matrix representation is two dimensional, as shown in Figure 3. The columns of this matrix can be viewed as features. When PCA is applied to this 100 x 100 matrix, the 100 columns are condensed down to two columns. This would seem to lose information, but this condensed representation makes it possible to find two directions along which the pixel values vary. Two directions are found, because the original representation of 100 columns is slimmed down to two columns. These two directions are perpendicular, and form the new axes (OpenCV, 2018b).

Principal Component Analysis is compatible with data that has similar clustered pixel values, as shown in Figure 4. However, this method is ineffective on images that have discontinuities in pixel values, such as the package in Figure 5. The package has varying pixel

intensities spread through the image; hence, drawing patterns from it is challenging for PCA. PCA optimizes the flow of pixel intensities to determine an orientation of the package.

Figure 3

Pixel representation of an image converted to grayscale

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI
1	155	156	158	160	164	168	173	176	177	174	172	171	167	162	161	163	160	164	166	165	167	171	171	168	170	172	171	171	169	166	168	167	164	162	156
2	155	155	157	159	163	168	173	176	176	174	172	171	167	163	162	163	161	165	167	166	167	171	172	169	169	172	171	171	168	165	167	167	164	162	157
3	155	156	157	159	163	167	172	175	175	174	172	170	166	163	162	161	161	164	165	165	167	170	171	169	168	171	171	171	168	165	167	167	163	161	157
4	154	156	158	162	165	170	173	176	176	176	174	170	168	166	165	163	164	166	168	168	170	173	174	173	172	175	175	176	175	172	174	174	173	171	168
5	161	163	166	169	173	177	181	184	187	187	185	182	180	180	180	178	184	185	186	187	188	190	190	190	191	192	192	193	194	192	194	192	193	192	190
6	181	181	181	183	184	187	191	193	195	195	194	191	191	193	193	192	193	193	193	194	194	194	194	195	195	195	194	196	197	196	197	195	195	195	194
7	193	193	193	193	193	194	195	195	193	192	192	191	192	193	194	194	195	194	194	195	195	194	194	195	195	195	193	195	195	194	196	195	196	196	197
8	190	192	194	195	195	194	194	193	193	193	192	193	194	194	195	195	194	193	193	195	195	194	194	195	195	197	195	196	196	195	198	197	197	197	198
9	192	192	193	193	193	194	194	193	193	193	193	193	194	194	195	195	195	193	195	198	198	194	193	196	195	195	195	196	196	197	197	197	197	197	198
10	193	193	193	193	193	194	194	193	194	194	194	194	194	194	195	195	194	193	195	197	197	194	194	195	195	195	195	196	196	197	197	197	197	198	198
11	193	193	193	193	193	193	193	193	194	194	194	194	194	194	195	195	194	194	195	196	195	195	195	195	196	196	196	196	196	197	197	197	197	198	199
12	193	193	193	193	193	193	193	193	194	194	194	194	194	194	194	194	194	195	195	194	194	195	196	195	196	196	196	196	196	197	197	197	197	198	199
13	194	193	193	193	193	193	193	193	194	194	194	194	194	194	194	194	194	195	195	194	194	195	196	196	196	196	196	196	197	197	197	197	197	198	199
14	193	193	193	193	193	193	193	193	194	194	194	194	194	194	194	194	195	195	195	194	194	195	196	196	196	196	196	196	197	197	198	198	198	198	198
15	193	193	193	193	193	193	193	193	194	194	194	195	195	195	195	195	195	195	195	195	195	195	195	196	196	196	196	197	197	197	198	198	198	198	198
16	193	193	193	193	193	193	193	194	194	194	195	195	196	196	195	195	195	195	195	195	195	194	195	196	196	196	196	197	197	198	198	198	199	198	198
17	194	193	193	193	194	194	194	194	194	194	194	195	195	195	196	196	194	197	196	192	198	196	196	198	197	197	193	197	197	197	198	198	197	200	198
18	194	194	193	193	194	194	194	194	195	195	195	196	196	196	196	196	196	196	192	198	193	197	194	194	195	196	198	199	199	195	197	197	199	200	200
19	194	194	194	194	194	194	194	194	195	195	196	196	196	196	196	196	194	194	195	196	198	197	196	198	197	196	199	197	198	196	197	199	198	197	199
20	194	194	194	194	194	194	194	194	194	195	195	195	196	196	196	196	196	196	197	198	193	193	196	195	198	196	198	197	198	200	198	199	200	199	201
21	194	194	194	194	194	194	194	194	194	194	195	195	195	196	196	196	194	194	197	194	197	197	199	196	196	196	197	199	197	199	196	197	198	199	200
22	194	194	194	194	195	195	194	194	195	195	195	196	196	196	197	197	195	195	195	197	195	198	197	194	198	201	199	199	198	197	200	199	199	201	201
23	195	194	194	195	195	195	195	194	195	196	196	197	197	197	198	198	198	198	197	193	200	195	196	201	195	198	197	196	202	195	204	197	199	201	200
24	195	195	195	195	196	196	195	195	195	196	196	197	197	198	198	198	197	195	197	203	1	4	13	26	2	5	8	10	27	13	21	4	5	7	9
25	195	195	195	195	196	196	196	197	196	196	197	197	197	197	198	199	197	198	195	202	10	9	7	4	2	2	3	6	5	7	15	10	26	21	23
26	195	195	195	195	196	196	196	197	196	196	197	197	197	197	198	199	200	195	200	200	21	24	22	37	35	35	43	38	35	34	24	33	31	22	14
27	195	195	195	196	196	196	197	197	196	197	197	197	197	197	198	198	198	202	195	203	11	42	17	19	15	18	22	25	22	18	32	17	13	4	9

Figure 4

Image with similar pixel values clustered together. Each object has pixel values clustered in the one specific direction (OpenCV, 2018b)

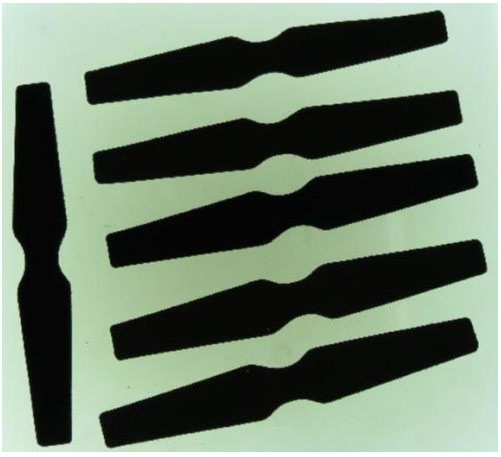


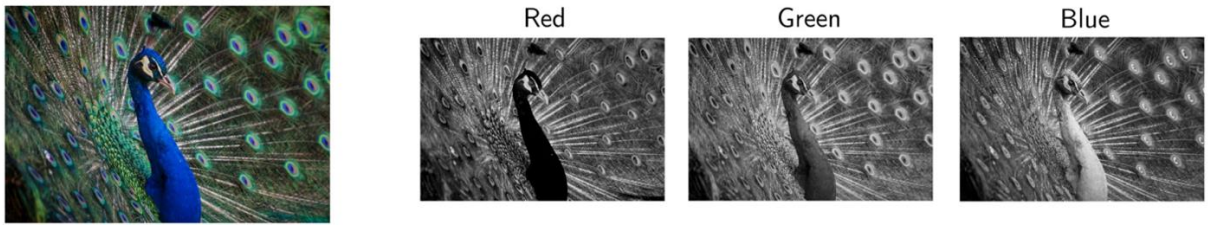
Figure 5

An image with discontinuities in pixel values

Another tool that was explored to solve the orientation problem was supervised machine learning in conjunction with Convolutional Neural Networks or CNNs. A neural network consists of inter-connected layers of neurons. Each neuron is a computational unit and is responsible for some task within the network, similar to biological neurons. Neural networks can learn patterns from examples, and then make predictions based on their prior experiences. The training process extracts features of the dataset and adjusts a network's weights to match training results. This process is repeated several times with many training samples. A CNN is a specific kind of neural network that is mainly used for image processing. CNNs use convolutions, which are specialized linear operations used in place of the general matrix multiplications typically used in regular neural networks (Goodfellow, Bengio, & Courvil, 2016). CNNs are beneficial in extracting features of a complex dataset. Images, the training data or input, are convolved as part of the feature extraction process (Aghdam & Heravi, 2017). For instance, CNNs can extract features from colored images containing red, green, and blue, or RGB channels, as shown in Figure 6. The features extracted in Figure 6 are three of many features that a CNN can extract.

Figure 6

Convolutional Neural Networks produces red, blue and green feature maps for the colored image on the left (Shafkat, 2018)



CNNs have been successful in image processing due to their ability to process large volumes of data, especially images. CNNs were preferred over regular neural networks for this experiment, because CNNs have the capability of learning from a collection of extracted features from images, as shown in Figure 8. A feature is extracted using a kernel, an N x N matrix that is slid across consecutive horizontal rows of the input data. The kernel averages out N x N regions of the input data into single values as it moves across the rows, and the result is a feature map. During the training phase, these kernels learn specific features within the images, and detect these features anywhere on the image. This flexibility allows CNNs to generalize patterns seen within images (Build your First CNN and Performance Optimization, 2018). The CNNs designed in this experiment accept input images of 100 x 100 pixels. This means that the total number of neurons required to handle incoming information is 10000, that is, one neuron per pixel. Regular neural networks are limited, since the number of neurons is pre-determined at the time the network is created, as shown in Figure 7. To avoid loss of information that may occur in regular neural networks, CNNs were preferred for this experiment.

Figure 7

Neural network with an input layer, hidden layer, and output layer

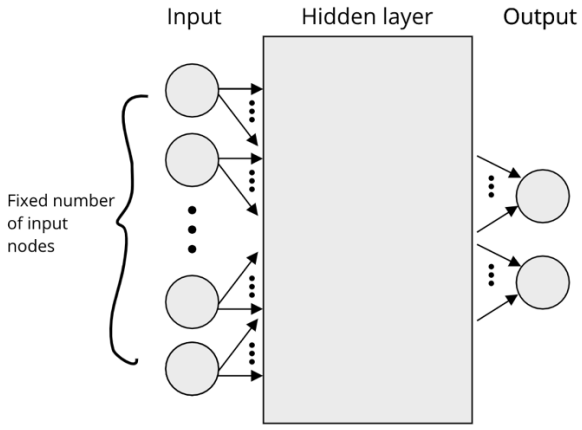
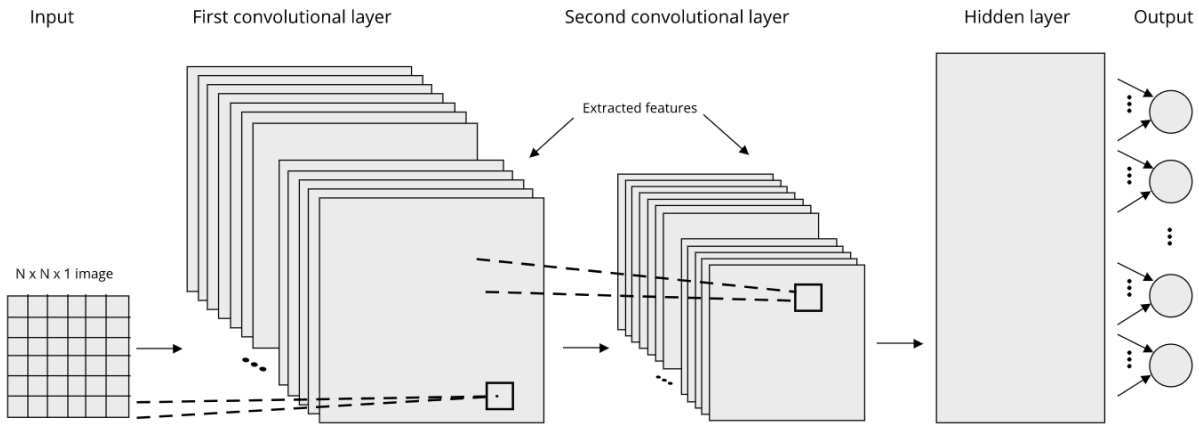


Figure 8

A simplified view of a CNN with an input image, two convolutional layers, hidden layer, and output layer



A major flaw in neural networks is their inability to distinguish symmetric object orientations. Saxena, Driemeyer, and Ng explored three-dimensional object orientation detection using Euler angles but described how Euler angles are prone to Gimble Lock, the loss of one degree

of freedom when two of the three axes align with one another. They explore Quaternions, which are another way to represent three-dimensional orientation. Quaternions are also limited in their ability to identify orientation for symmetric objects. For two identical orientations of an object, there would be two different and opposite quaternion representations. This makes the learning process inefficient. The authors implement an algorithm that dissects an image into four parts, extract angles of the edges from each section, and feeds these features into a supervised learning algorithm (Saxena, Driemeyer, & Ng, 2009).

CHAPTER 4: EXPERIMENT

The experiments include objects that are symmetric and asymmetric, as shown in Figures 9 and 10. For this experiment, asymmetry in objects is defined as no two angles having the same picture representation. In other words, if the object is asymmetric, then there is a unique picture representation for every angle the picture is rotated. More emphasis is given to asymmetric objects, since objects in industry are generally asymmetric, due to package labels, the packaging itself, and dents.

Figure 9

Symmetric object representation

Figure 10

Asymmetric object representation

4.1 Method using Principal Component Analysis

Principal Component Analysis is an iterative process. The PCA method reads an image and preprocesses it. The first preprocessing operation resizes the image to 100 x 100 pixels to

ensure all images are of uniform size. The second operation converts all images to grayscale to reduce the number of channels from three to one, that is, from RGB to grayscale. The Gaussian blur method blurs the resulting image using a 3 x 3 kernel to smooth the image and remove noise as shown in Figure 11. Binary thresholding finds a threshold of a grayscale image – the result of binary thresholding is shown in Figure 12. If a pixel is greater than the threshold *T*, then the pixel is assigned 255, the lightest pixel value; otherwise, the pixel is assigned 0, the darkest pixel value (OpenCV, 2018a). An optimal threshold *T* was determined for this experiment by trial and error over numerous trial runs. Using the OpenCV library's thresholding and binary thresholding functions, the upper and lower thresholds of the blurred image are calculated. The Canny operator uses these thresholds for edge detection (Ding & Goshtasby, 2001). The edges detected are used to extract contours. Eigenvectors are calculated for each contour.

Figure 11

Original image on the left is blurred by the Gaussian blur method using a 3 x 3 kernel

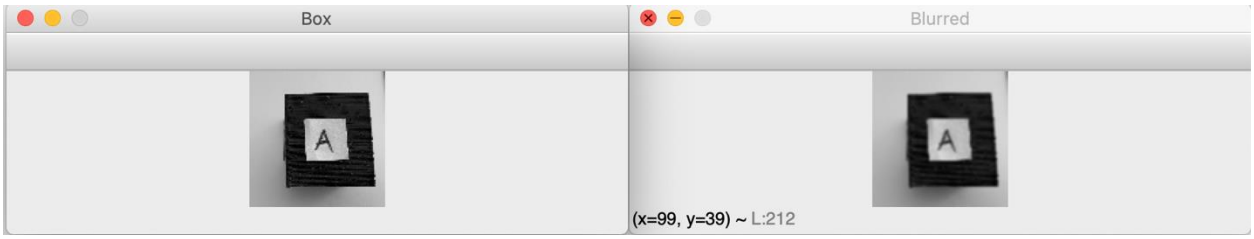
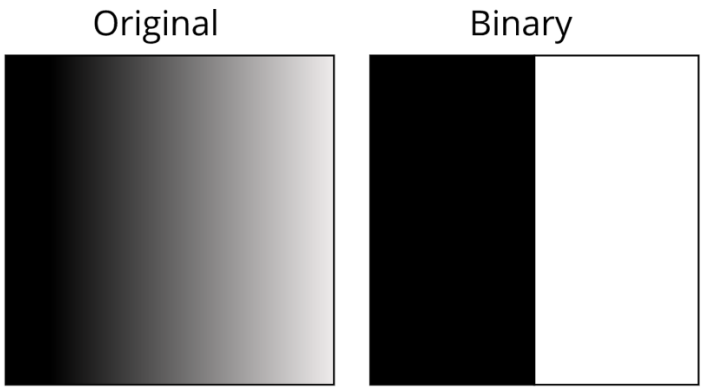


Figure 12

Original image on the left is converted to binary values 0 or 1 using binary thresholding



Algorithm 1: Principal Component Analysis.

1. *Read image*
2. *Convert to grayscale*
3. *Resize image*
4. *Apply Gaussian blur with a 3 x 3 kernel*
5. *Apply binary thresholding*
6. *Use these thresholds in the Canny operator*
7. *Find contours*
8. *For each contour c:*
    a. *Find the eigenvectors using PCA*

4.2 Model Description

For the solutions developed in Sections 4.3 and 4.4, two sets of images were used. The first held black and white clipart images that contained a distinguishable identifier. The second set held live data that was captured by a web-camera. Both input datasets contained images of size 100 by 100 pixels. This size was deemed appropriate, since it was able to contain crucial information within the image. Diminishing the image size by a factor of 25% to 50 by 50 resulted in loss of

crucial information. The live data were colored images following the RGB scale, which were converted to grayscale as part of the preprocessing step. As shown in Figures 13 and 14, the input data show the orientation of the object that the programmer deems the base orientation.

Figure 13

Base orientation of asymmetric clipart image

Figure 14

Base orientation of live test image chosen at random

For the live dataset, a variety of input data was used: images that were noisy, images with different backgrounds and lighting, and so on. These were augmented using an image data generator provided by the Keras preprocessing library (Keras, n.d.). The image data generator created new variations of the existing images by horizontally and vertically shifting, zooming in and out, stretching and shrinking, and altering the red, green, and blue intensities.

The training data set was created on-the-fly during the training phase. This involved randomly rotating each of the base images, and associating the new rotated image with the rotated

angle. An iteration of the training phase from the input layer till the output layer over a dataset is an epoch. All the models were trained for a total of 50 epochs, with training data generated in batches of size 128. The number of base images used for both input datasets was 100, so the total number of training samples for each model was 640,000. Separate regression and classification models were created for each input dataset.

The models created in Sections 4.3 and 4.4 were evaluated by validation data that was created on-the-fly during the training phase. The two models use three convolutional layers. Convolutions are applied on input data using a kernel. The convention is that kernels are squares, so that the resulting output averages uniformly. Filters in convolutional layers are not to be confused with kernels in the Keras library. Filters are three-dimensional structures with multiple kernels piled together, whereas kernels are two-dimensional. In a convolutional layer, a filter extracts features from the input data.

Padding is of two types: valid and same. *Valid padding* shrinks images in size by excluding the pixels along the edges of an image, as they are passed through the convolutional layers. Shrinking the image sizes using valid padding is acceptable when the pixels in the middle portion of the image are more important than the pixels along edges of the images (Chollet, 2018). However, there is loss of information with valid padding, as the edges of the image are neglected. *Same padding* allows the pixels along edges of the images to be considered during the convolutions, and result in an output that is the same dimensions as the input (Dumoulin & Visin, 2018). The models in Sections 4.3 and 4.4 used same padding.

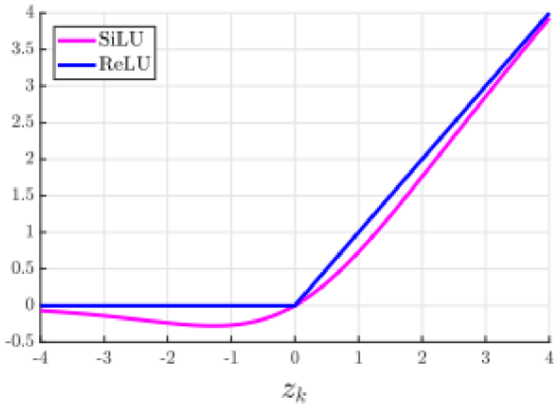
Strides are the number of pixels that a kernel skips as it is slid horizontally and vertically across the input data (Chollet, 2018). In the convolutional layers of the models described in Sections 4.3 and 4.4, strides of one pixel are made horizontally and vertically. The *RELU* activation

function, indicated by the blue line in Figure 15, is a non-linear function that activates if the input to the function is zero or greater. In other words, the function returns a value greater than zero when activated; otherwise, the output is zero. (Nwankpa, Ijomah, Gachagan, & Marshall, 2018). The two models used RELU activation which determine whether the output of a neuron is significant enough to be transmitted on to the subsequent neurons.

Error was indicated by the positive difference between the angle represented by the labeled image and the angle predicted by the model. Each model strived to minimize the angle error. Once the model was trained, it was tested on test data. Each model was tested on 100 samples, 100 times. Chapter 5.2 reports and discusses the mean of these 10,000 instances.

Figure 15

RELU activation function (Nwankpa, Ijomah, Gachagan, & Marshall, 2018)



4.3 Method using Regression

The regression model comprises three two-dimensional convolutions that contained 16 or 32 filters. All three convolutions involved a one-by-one kernel, same padding, and one-by-one strides. The third convolutional layer was followed by a flattening layer which converted the N x N feature map into a 1 x N array. In a dropout layer, the outputs of randomly selected neurons are not considered in the forward and backward passes of a neural network. The intuition behind this approach is to avoid any particular neuron being heavily considered during the training process, avoiding overfitting of training data. The dropout layer randomly dropped one quarter of the outputs from the flattening layer during each epoch of the training phase. The resulting outputs were then passed through a linear activation function. The result was a single output node that outputs a floating-point number between zero and 360 degrees.

Algorithm 2: Regression Model.

Training Phase:

For 50 epochs, do:

1. *Two dimensional convolution on input image with 16 filters, kernel size of 1, same padding, strides of 1, and a ReLU activation*
2. *Two dimensional convolution on input image with 32 filters, kernel size of 1, same padding, strides of 1, and a ReLU activation*
3. *Two dimensional convolution on input image with 32 filter, kernel size of 1, same padding, strides of 1, and a ReLU activation*
4. *Flatten layer*
5. *Dropout layer of 0.25*
6. *One output node with linear activation*

4.4 Method using Classification

The classification model comprises three two-dimensional convolutions, flattening, and dropout layers similar to those of the regression model. However, the output layer in this model consisted of 360 output nodes, each representing a single degree between zero and 360 degrees. Softmax activation ensured the vector of real numbers from the dropout layer was transformed into a probabilistic distribution (Nwankpa, Ijomah, Gachagan, & Marshall, 2018). The outputs from the Softmax layer summed to one or 100%.

Algorithm 3: Classification Model.

Training Phase:

For 50 epochs, do:

1. *Two dimensional convolution on input image with 16 filters, kernel size of 1, same padding, strides of 1, and a ReLU activation*
2. *Two dimensional convolution on input image with 32 filters, kernel size of 1, same padding, strides of 1, and a ReLU activation*
3. *Two dimensional convolution on input image with 32 filter, kernel size of 1, same padding, strides of 1, and a ReLU activation*
4. *Flatten layer*
5. *Dropout layer of 0.25*
6. *360 output nodes with Softmax activation*

4.5 Correcting Incorrect Orientations

A separate program was designed to correct incorrect box orientations for each of the methods described in Chapter 3. The base test data images were rotated at a random angle in degrees, an integer. The model then predicted the angle of the rotated image. The model's prediction was used to rotate the image back to the base orientation. For determining the accuracy of the models, this program was run 100 times on 100 random samples. The models were also tested in real time. Boxes were placed under a camera at random angles. The images that were captured using the real-time video feed were analyzed, predicted, and corrected almost instantaneously.

CHAPTER 5: RESULTS AND DISCUSSION

This chapter surveys the results of PCA, Regression and Classification performed on the input dataset as described in Chapter 4, and discusses the limitations of the three methods.

5.1 PCA

Figure 18 shows the outline of the box in Figure 16. The outline appears towards the top-right corner of the grid in Figure 18, and the XY-axes calculated by PCA appears in the bottom-left corner of the grid. This outline is reflected along the X-axis. As shown in Figure 17, the shadow of the box overlays the background; hence, edge detection is not continuous along the edges of the box in Figure 16. Figure 19 shows the noise detected by PCA in Figure 16 on the right-hand side of the grid. The outline of the noise in Figure 19 is also reflected along the X-axis. The noise in the background is considered an object; hence, PCA is applied to the noise.

Figure 16

Input image in grayscale

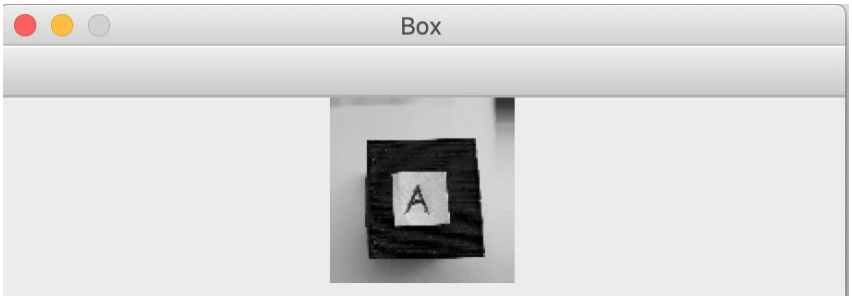


Figure 17

Edge detection results of image from Figure 16

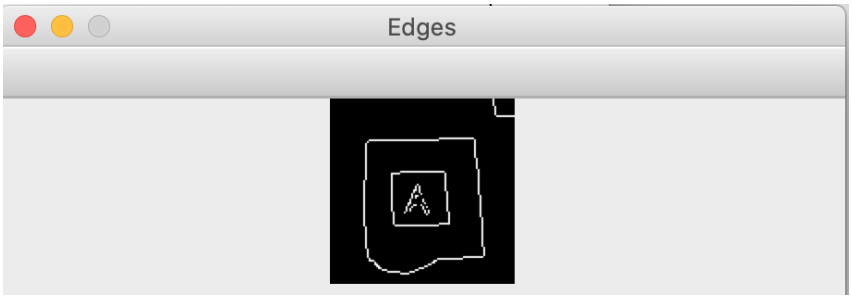


Figure 18

Main shape detected by PCA represented by the black outline on the top right, and the new XY-axes showing the shape's orientation represented by the red and blue lines

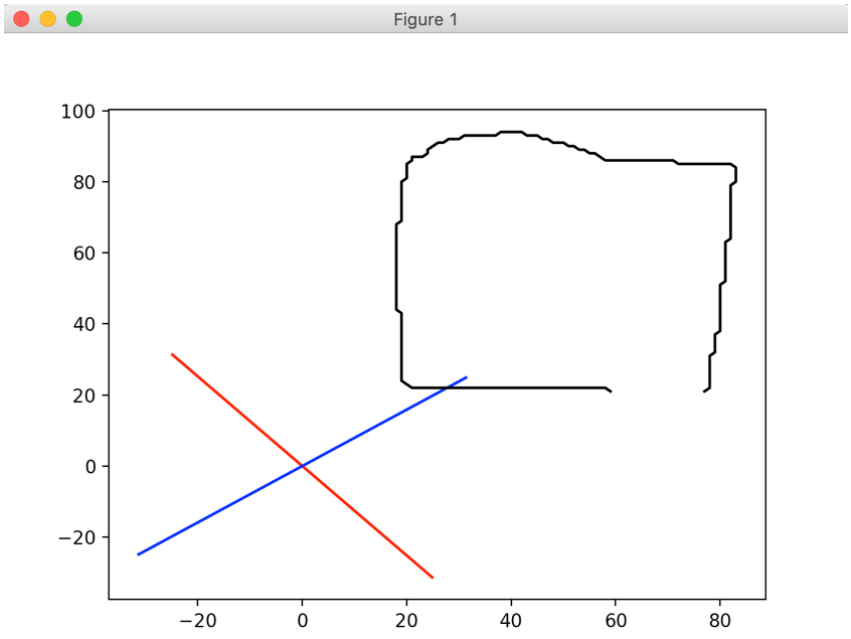


Figure 19

Noise detected by PCA represented by the black outline on the right, and the new XY-axes showing the shape's orientation represented by the red and blue lines

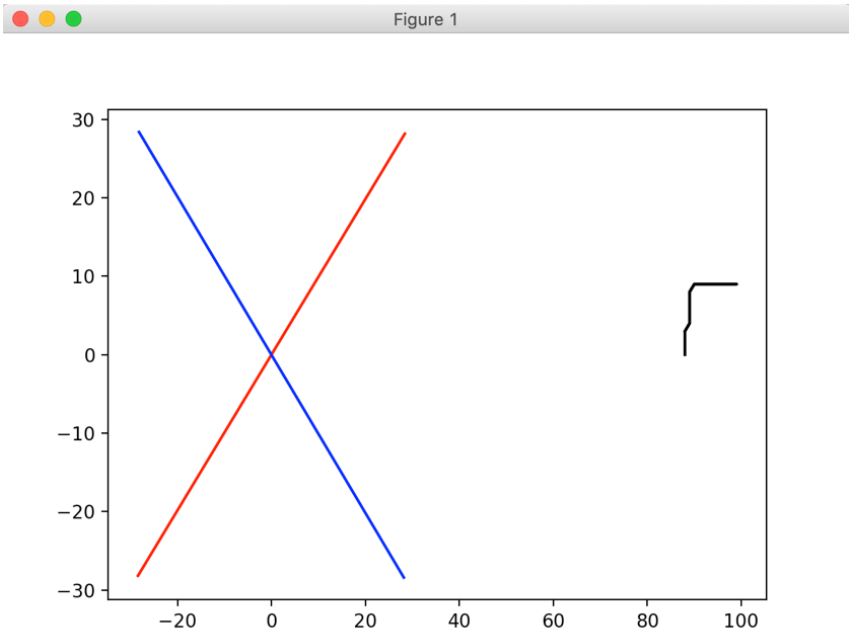


Figure 20 shows the new axes as determined by PCA for the box in Figure 16. The axes represented by the green and red lines in Figure 20 indicate the orientation of the box. The orientation of the axes is skewed 30 degrees counter-clockwise, since PCA considers the box's shadow as part of the box when calculating eigenvectors for each contour. However, the method can correctly determine the centroid of the contour as seen by the blue dot in Figure 20. The origin of the axes is the centroid of the object. PCA generates eigenvectors for the main contour of the box, and the noise in the background in Figure 16, as shown in Figure 21. The centroid and the calculated axes of the noise are also depicted in Figure 20, towards the top right-hand corner of the image.

Figure 20

Green and red lines showing orientation of image calculated by PCA

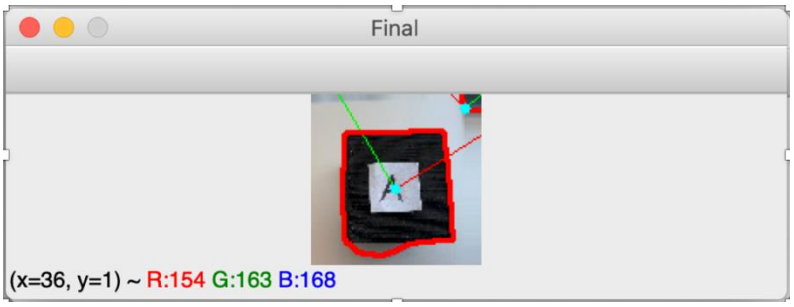


Figure 21

Two eigenvectors generated by PCA for image Figure 16

```
Mean:  [[48.83019  72.037735]]
Eigenvectors:
[[-0.62169623  0.7832584 ]
 [ 0.7832584   0.62169623]]

Mean:  [[89.9         5.7000003]]
Eigenvectors:
[[ 0.7098906   0.70431197]
 [-0.70431197  0.7098906 ]]
```

PCA is sensitive to the higher pixel values in an image, and is unable to distinguish between the noise in an image and the contour of interest. Hence, the method generates multiple eigenvectors for all contours in the image, including noise. When the noise in the background is identified as the dominant eigenvector, the method fails to identify the main object. An additional clause can be programmed to ignore the scenarios with noise by cropping the edges of the image, but this method does not generalize well, as important contextual information may be cropped off in the process of getting rid of noise.

Gaussian blur can be applied as part of the preprocessing step of the input data. However, Gaussian blur would not eliminate the noise completely, especially noise that occur in blobs as in Figure 16. Figures 22 through 24 show PCA performed on an input image with kernel sizes 3 x 3, 5 x 5, and 9 x 9. The kernel sizes used to blur the same input image result in different XY-axes, as seen in the bottom-right image of Figures 22 and 23. Figure 24 shows the results of a larger kernel: 9 x 9 used to blur the input image. Noise is not eliminated from the image, but contextual information from the main contour is lost. The result in Figure 24 generates three pairs of XY-axes. Sensitivity towards the pixel values makes PCA unable to provide a single orientation for the object of interest. Convolutional Neural Networks overcomes the problem of pixel sensitivity within an image.

Figure 22

Results of PCA using Gaussian blur performed with a 3 x 3 kernel

Figure 23

Results of PCA using Gaussian blur performed with a 5 x 5 kernel

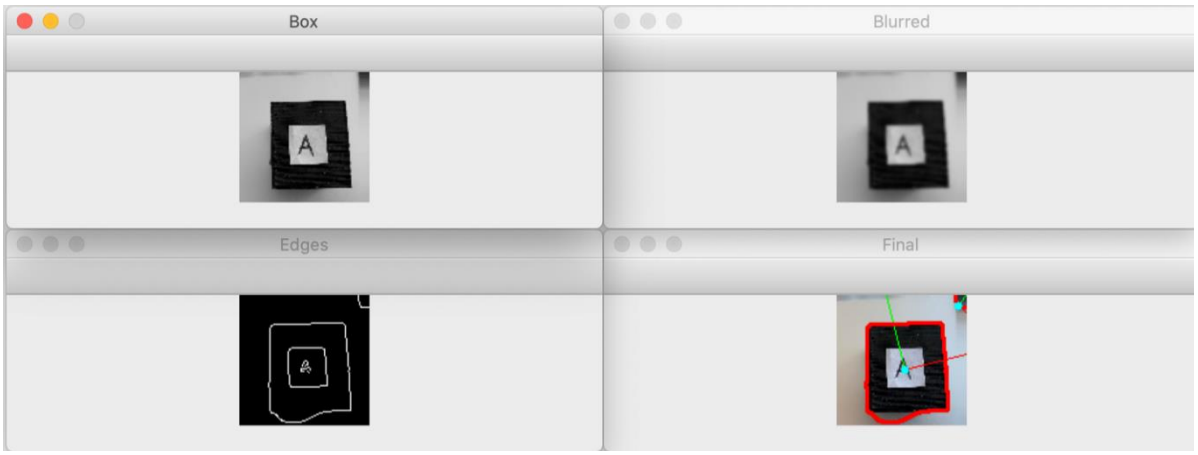
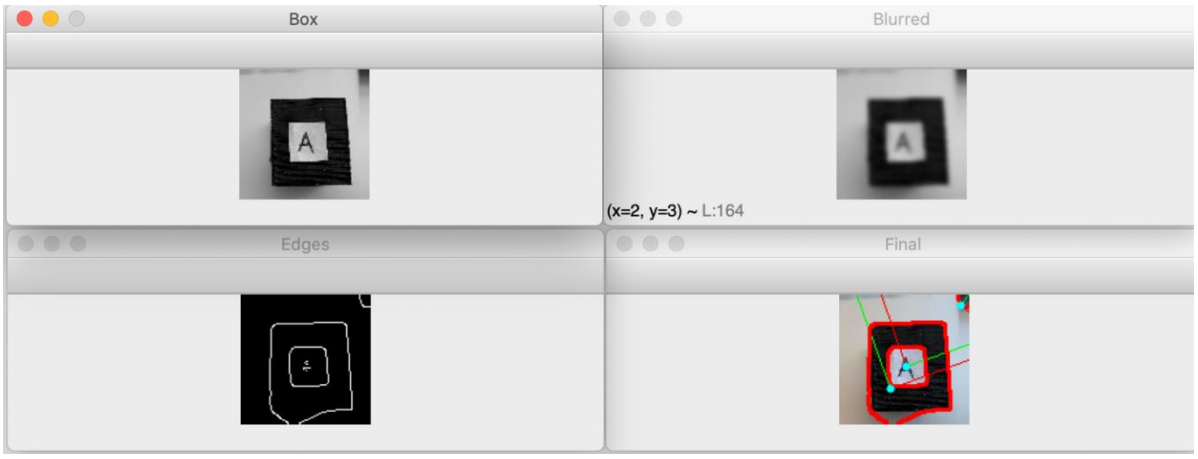


Figure 24

Results of PCA using Gaussian blur performed with a 9 x 9 kernel



5.2 Regression and Classification

Compared to PCA, the regression and classification CNNs adapt to the pixel values in images. They generalize, and learn important features within a set of training data by extracting features of interest. Feature extraction occurs within the convolutional layers of the two models.

The models are robust to noise, since the training phase of these models involved seeing the object in different conditions, as described in Section 4.2. Skipping pixel values using strides in the convolutional layers helps the models be less sensitive to any one pixel value within the image.

Moreover, the augmented data included images in which the box lay closer to the ends of the image. To keep these edges from being ignored, the CNNs were designed to have all images padded during the convolutions. This helped convolutional layers consider information along the edges of the training images. At the same time, this would be costly for the CNNs, as noise along the edges was picked up. A balance is achieved by implementing padding and strides that pick crucial information as well as noise from the edges of the image. This balance helps the CNNs further generalize the training data. Furthermore, the dropout layer gets rid of connections at random during each epoch of the training phase. This ensures that no one node is heavily relied on during the training phase, allowing the CNNs to generalize the training data.

PCA produces multiple orientations of the image as discussed in Section 5.1. The method produced eigenvectors for noise in the image as well as the box. The maximum eigenvector determined by PCA could either be the orientation of the noise or the box. On the other hand, the regression and classification models gives only one orientation or angle per prediction. Figure 25 shows a sample run from a trained classification model. As shown in Figure 25, there are 360 bins, which are denoted in an array consisting mainly of zeros. The bin holding a one, highlighted by the red box, is the bin that the classification model predicts as the likely angle of rotation. In Figure 25, the predicted angle of rotation is 266°. The base rotation of the box in this case was 269°, so the model's error of prediction is 3°. In a factory setting, 3° would be a negligible error. This shows that the model is able to come close to the actual angle of rotation, equivalent to a human's best guess. In contrast to PCA, the machine learning models consider the entire input image when

determining the object's orientation; hence, returning its best prediction for the object's angle of rotation.

Figure 25

A sample test run of the classification model built on asymmetric clipart images

```
Classification model using 360 bins on asymmetric images.
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
Base image is rotated 269 degrees.
Bin no: 266
Predicted Angle: 266
Error: 3
```

The current classification model can be enhanced by dividing the output bins into 72 bins instead of 360 bins. Each bin would account for five degrees, that is 0-5°, 5-10°, and so on. Reducing the number of output bins would improve the mean error rate of the classification model, since negligible errors can be avoided. The accuracy of the model depends on the use case. Detecting package orientations in a factory setting can do with lower accuracy, but detecting the orientation of a pacemaker in a human subject would demand a higher level of accuracy. Computing power in industries can be leveraged to train complex models with higher accuracy; for instance, by increasing the number of convolutional layers in the network. Sophisticated models like AlexNet, which contains 60 million parameters, can be trained with better image

quality and image augmentation to yield improved accuracy during the testing phase (Krizhevsky, Sutskever, & Hinton, 2017).

A problem encountered during the training phase of the classification model was the symmetry of the object. If the object or the identifier on the object is symmetric, then the model fares poorly during the training phase, due to confusion. For instance, Figure 26 is the base orientation of the object, and Figure 27 is the object rotated 180 degrees. Due to the symmetry of the object, the figures look exactly the same, but the true angles of rotation are different. In this scenario, the classification model provides a Softmax array of four values representing the four angles, but since the training phase requires a single prediction, the maximum probability of the four angles is output. Hence, there is a 25% chance that the prediction is correct, but as a justification, it is humanly impossible to accurately predict the angle of rotation in Figure 26.

Figure 26

Base orientation of an image

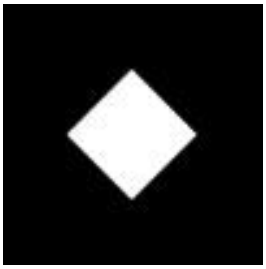
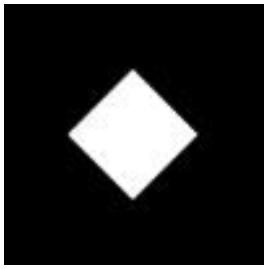


Figure 27

Image in Figure 26 rotated 180 degrees



This was a frequent problem in the clipart dataset, where the image representations were "clean," and had no noise in the background to help the classification model distinguish similar picture representations of the object. In Table 1, the mean error of the classification model run on a live dataset, shown on row 5, is lower than the mean error of the model run on a clipart dataset in row 2, for symmetric images. The mean indicates the average error rate over a series of 10000 runs. This result is significant and surprising. Although clearing out noise in PCA was beneficial for detecting the object's orientation, the CNN models consider noise as important. Noise helps the models during the training process to learn from the patterns, and generalize to the surrounding contextual information of the image. In the live dataset, contextual information includes the shadow of the box, the background scene, the lighting, and any other noise caused by camera quality. This contextual information is not available in the clipart dataset; the images are "clean" as shown in Figures 26 and 27.

The quality of a 100 x 100 pixel image is poor; hence, the edges of the box in live data appear pixelated. Since the output bins in the classification model are one degree per bin, the classification model has to be precise in its predictions. The mean error rate for live data, as shown in Table 1, is more than 50°. Condensing the number of output bins from 360 to 72 bins would

minimize the mean error, since the model will have flexibility to predict close to the true angle of rotation without its prediction being considered incorrect.

Table 1

Mean for 100 runs performed on 100 samples per run using classification.

Row Number	Symmetric / Asymmetric	Live Data / Clipart	Mean Error
1	Asymmetric	Clipart	9.47
2	Symmetric	Clipart	138.80
3	Both	Clipart	66.35
4	Asymmetric	Live Data	68.70
5	Symmetric	Live Data	111.49
6	Both	Live Data	76.26

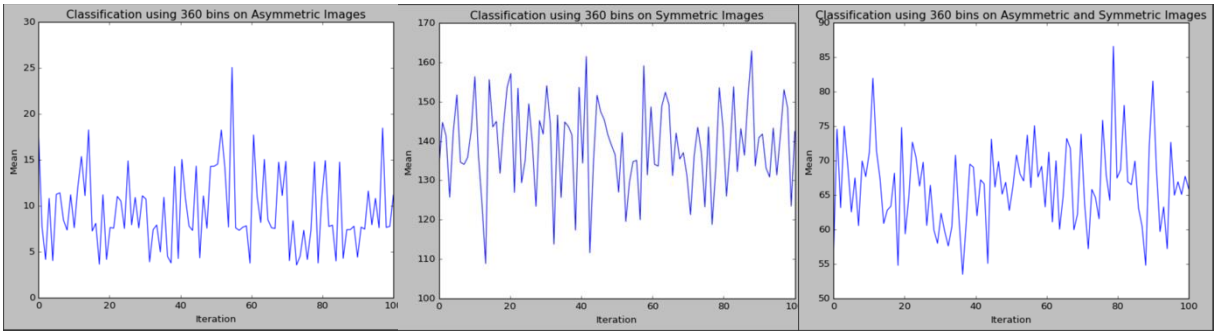
Figure 28 shows the mean errors oscillating per iteration for the classification model's predictions on each category of clipart data. Rows 3 and 6 in Table 2 show the ranges of mean errors oscillating per run for asymmetric and symmetric clipart and live data using classification which are 32.80 and 52.40 respectively. In comparison to the individual input datasets, that is either symmetric or asymmetric data, the classification model shows a lower range in mean oscillations when using a combination of asymmetric and symmetric input data. This shows that the classification model makes better predictions when using a combination of symmetric and asymmetric data.

Moreover, a lower range of mean oscillations indicated a low number of inconsistent or anomalous predictions made per iteration. Overall, the symmetric input dataset consisted of higher mean error ranges as in Table 2, similar to the high mean error rates shown in Table 1, indicating

that the model tends to get confused when predicting symmetric data. Accuracy, the measure of total error, was calculated using the mean of the error rates from trial runs (Bruce & Bruce, 2017a). Extreme values in the raw data collected during trial runs contributed to a higher mean, since the mean is sensitive to outliers (Bruce & Bruce, 2017b). The mean was used as a metric to measure accuracy to understand the effects extreme values had on the model's predictions. A trimmed mean or median can be used to disregard the extreme values in future work.

Figure 28

Mean error per iteration classification models trained on asymmetric data alone, symmetric data alone, and combined asymmetric and symmetric data.



Note. The input dataset used was the clipart dataset.

Table 2

Range of the mean of 100 runs performed on 100 samples per run using classification

Row No.	Input Type	Upper Limit	Lower Limit	Range
1	Asymmetric Clipart	25.00	3.51	21.49
2	Symmetric Clipart	163.00	109.10	53.90
3	Asymmetric & Symmetric Clipart	86.50	53.70	32.80
4	Asymmetric Live Data	92.30	44.80	47.50
5	Symmetric Live Data	139.10	84.90	54.20
6	Asymmetric & Symmetric Live Data	101.60	49.20	52.40

Precision was another metric used to measure the performance of the classification model. Precision measures the accuracy of positive predictions (Bruce & Bruce, 2017a). The classification model's precision on asymmetric data is 61.1%. This implies that more than half of the model's predictions were true positives. On the other hand, the precision of the model's predictions on symmetric data is 36%, indicating that the model fared poorly when predicting symmetric data. The precision of the model's predictions on asymmetric and symmetric data is 46.1%, which still indicates that the model is unable to make correct predictions when symmetric data are involved. Reducing 360 output bins to 72 in the classification model provides a tolerance of up to five degrees during the test phase. The model does not have to predict the exact angle of rotation; a neighboring angle within the tolerance of up to five degrees is acceptable. This enhancement will deliberately boost precision when the model is tested on all three categories of data.

Similar to the classification model, the regression model has higher mean errors when run on symmetric images when compared to the asymmetric input data, and a combination of

asymmetric and symmetric data, as seen in rows 2 and 5 respectively, in Table 3. The mean error for symmetric clipart data is 180.07, slightly higher than the mean error of symmetric live data, which is 179.43 using the regression model. A similar situation arises when comparing the mean errors of symmetric live data and a combination of symmetric and asymmetric data. The differences in both cases are not significant, indicating the regression model's inability to generalize and predict correctly regardless of the type of input. In comparison, the classification model has a significant difference in mean errors between the symmetric input data and the symmetric-asymmetric input data combination, as shown in Table 1.

Table 3 shows the delta mean error of the two machine learning models on the six categories. It is interesting to see that the delta mean error of the symmetric datasets, shown in rows 2 and 5 of Table 3, has increased less in comparison to the other categories. Since regression predicts a value based on prior knowledge, and does not follow the binning principle like classification, the regression model does not get confused with orientations of symmetric data. Nevertheless, the model still fails to perform better than the classification model.

Rows 3 and 6 in Table 4 show the ranges of the mean error for asymmetric and symmetric clipart and live data using regression. In comparison to the individual input datasets, the regression model shows a lower range in mean error oscillations when using a combination of asymmetric and symmetric input data similar to the classification model. In Table 4, the range of mean errors for asymmetric clipart data is 52.30. This is higher than the range of mean errors of asymmetric live data which is 49.30. This suggests the predictions per iteration for asymmetric clipart data have more fluctuations than asymmetric live data. The observation may seem intuitive, since asymmetric clipart data contains no noise in comparison to asymmetric live data. However, the

regression model uses the noise in the live data to better predict the orientations of the objects. The same reasoning applies to the range of mean errors of asymmetric-symmetric clipart data, 44.10, in comparison to asymmetric clipart data.

Table 3

Mean for 100 runs performed on 100 samples per run using regression.

Row Number	**Symmetric / Asymmetric**	**Live Data / Clipart**	**Mean Error**	**Delta (regression mean error – classification mean error)**
1	Asymmetric	Clipart	177.91	168.44
2	Symmetric	Clipart	180.07	41.27
3	Both	Clipart	179.99	113.64
4	Asymmetric	Live Data	178.87	110.17
5	Symmetric	Live Data	179.43	67.94
6	Both	Live Data	181.14	104.88

Table 4

Range of the mean error of 100 runs performed on 100 samples per run using regression

Row No.	**Input Type**	**Upper Limit**	**Lower Limit**	**Range**
1	Asymmetric Clipart	203.20	150.90	52.30
2	Symmetric Clipart	201.60	153.40	48.20
3	Asymmetric & Symmetric Clipart	201.90	157.80	44.10
4	Asymmetric Live Data	205.70	156.40	49.30
5	Symmetric Live Data	208.30	151.80	56.50
6	Asymmetric & Symmetric Live Data	197.50	141.60	55.90

In comparison to the regression results in Table 3, the classification results in Table 1 have lower mean errors for each input data category. The classifier predicts by identifying an image's angle of rotation to one or more of the 360 bins. The regressor, on the other hand, tries to approximate the angle of rotation after seeing multiple images during the training phase. These two scenarios are analogous to a child during its early years of learning. A classification problem presented to a child, such as identifying an apple from five options of fruit would fare better than having the child name the fruit. This intuition justifies the lower mean errors for each category in the classification results in Table 1 in comparison to its equivalent regression results in Table 3. Moreover, classification predictions can be evaluated using accuracy, whereas regression predictions can be evaluated using root mean squared error (Goodfellow, Bengio, & Courvil, 2016). Regression suits this problem because the features of the input data are correlated to the orientation of the object. However, it is ineffective when compared to the classifier, because it does not generalize over the input data as well as the classifier does.

Figure 29

Base orientation of asymmetric clipart image

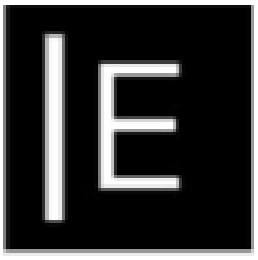


Figure 30

Figure 29 rotated 269 degrees by the model

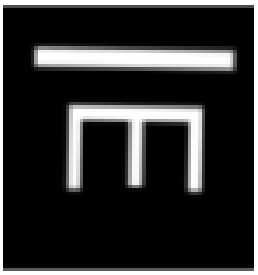
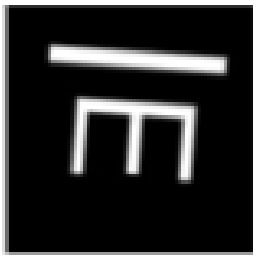


Figure 31

Predicted angle of 266 degrees by model



Symmetric objects were made asymmetric by placing an identifier on the object in the clipart data, as shown in Figures 29 to 31, and marked with the letter A in the live data, as shown in Figures 32 to 34. In industry, objects typically have a unique characteristic that would make its representation asymmetric, such as a label containing a bar code. Hence, it is logical to mimic this observation in the live data. The mean errors for the machine learning models trained with asymmetric identifiers show an improvement in performance over to the models trained with only symmetric data, as seen in Tables 1 and 3. When asymmetric and symmetric data were used, the mean error fell roughly around the midpoint of the mean errors of the models trained on asymmetric and symmetric data individually. For instance, the mean error of the classification model trained on asymmetric clipart data is 9.47, and the mean error of the model trained on symmetric clipart data is 138.80. The mean error of the model trained on asymmetric and

symmetric clipart data is 66.35 while the midpoint of the former mean errors is 64.66. However, the synergy of symmetric and asymmetric data is crucial, because if the models are not trained on symmetric data, they will not be able to make an educated guess during the test phase.

Figure 32

Base orientation of live test image chosen at random.

Figure 33

Figure 32 rotated 37 degrees by the model

Figure 34

Predicted angle of 55 degrees by model

The mean errors generated by the machine learning models can be improved. In an industrial setting, if the mean errors tend to increase over time, the existing model can be re-trained with new data. One limitation observed during the experiment was that the rotate method provided by the OpenCV library created an extra white background underneath the rotated image. This works well for images that have a white or similar background, but fares poorly for backgrounds of other colors. In the latter case, this may skew the predictions of the model. Ideally, the rotated image should have a transparent background to receive accurate predictions from the model. A sanity check was performed during the test phase on the two machine learning models to determine if the models were consistent in their predictions for every input angle of rotation. The models were 100% consistent in their predictions.

Hyperparameters are elements in a machine learning model that can be adjusted to prevent it from being over-trained. The hyperparameters of a model are also tuned to balance a model's accuracy and computational complexity (Bruce & Bruce, 2017c). Throughout this experiment, different hyperparameters, such as the number of epochs, were tested on the machine learning models. The intent behind this experiment was to create a simple model that could be trained in under ten minutes, and still produce accurate results, replicating the conditions of industry. As the models grow complex with many layers, and the number of training images grows, the training process is computationally more expensive and time-consuming. Hence, the models presented in Algorithms 2 and 3 only consist of three convolutional layers. The experiment emphasized having more training data run through a less complex model. These models are less complex when compared to CNNs like AlexNet which works with 650,000 neurons spread across five convolutional layers and other neural layers (Krizhevsky, Sutskever, & Hinton, 2017).

5.3 Lessons learned

The initial versions of the regression and classification models were trained using raw RGB images. This was a computationally expensive process as the CNN had to do computations on data that were not normalized, that is, in the range of 0 to 255. Moreover, RGB images involve three channels: red, green, and blue. When these raw data were fed into the CNN, it was doing redundant work: that is, it was extracting almost the same features in the three different channels. The results produced from the models trained with raw data were no better than the models trained with preprocessed data. The training time and computations done by the CNN were significantly reduced by normalizing the RGB values to grayscale.

The time taken to create the machine learning models was primarily dedicated to data pre-processing and cleaning. The model performs as well as the data that they are trained on. The more training data the CNNs were fed, the more accurate the models' predictions were. These models can be further improved by training the model with more augmented data. Training samples of various rotations can also be fed into the network during training to improve the models' ability to predict accurately.

CHAPTER 6: CONCLUSION

The goal of this experiment was to define the base orientation of an object, detect orientations of an object, and correct the object's orientation if it is identified as different from the base orientation. The capabilities and limits of three models: PCA, Regression and Classification were explored in this experiment. PCA is a mathematical tool used for feature extraction of a dataset. However, PCA is highly sensitive to the noise in an image, and includes noise as an important part of the analysis. It is also limited by discontinuities in pixel values within the image. Regression and Classification using Convolutional Neural Networks helped overcome the two limitations of PCA. These models are adapt to the pixel values in images, generalize, and learn the important features within the set of training data by a process of extracting the important features. The Regression model fared poorly in comparison to the Classification model, because it does not generalize over the input data as well as the classifier does. When tested on live data, the classifier had a mean error of 76.26, and the regressor scored a mean error of 181.14 for 100 random samples run 100 times. This shows that the classifier's predictions were about twice as good on average as the regressor's. The two models encountered difficulty when tested only on symmetric data. The confusion in angles of orientation due to symmetry can be overlooked since it is a similar issue for humans. The error rate can be decreased by using more training examples during the training process, and using transparent backgrounds during image rotations. Using this approach, a simple model with a wide collection of training examples makes it possible to train, use, and re-train a model in an industrial setting to automate the detection and correction of incorrect object orientations.

REFERENCES

Aghdam, H. H., & Heravi, E. J. (2017). In Guide to Convolutional Neural Networks (pp. 61-70). Springer International Publishing.

Bruce, P., & Bruce, A. (2017a). Classification. In Practical Statistics for Data Scientists (pp. 173-208). Sebastopol: O'Reilly Media Inc.

Bruce, P., & Bruce, A. (2017b). Exploratory Data Analysis. In Practical Statistics for Data Scientists (pp. 1-42). Sebastopol: O'Reilly Media Inc.

Bruce, P., & Bruce, A. (2017c). Statistical Machine Learning. In Practical Statistics for Data Scientists (pp. 208-248). Sebastopol: O'Reilly Media Inc.

Build your First CNN and Performance Optimization. (2018). In M. Sewak, M. Karim, & P. Pujari Practical Convolutional Neural Networks (pp. 47-77). Packt Publishing.

Chollet, F. (2018). Deep Learning for Computer Vision. In Deep Learning with Python (pp. 119-177). New York: Manning Publications Co.

Ding, L., & Goshtasby, A. (2001). On the Canny edge detector. Pattern Recognition, 721-725.

Dumoulin, V., & Visin, F. (2018). A guide to convolution arithmetic for deep learning. arXiv, 12-17.

Goodfellow, I., Bengio, Y., & Courvil, A. (2016). Deep Learning. In Machine Learning Basics (pp. 96-161). Cambridge: MIT Press.

Keras. (n.d.). Image data preprocessing. Retrieved April 26, 2020, from Keras: https://keras.io/api/preprocessing/image/

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet Classification with Deep Convolutional Neural Networks. Communications of the ACM.

Nwankpa, C. E., Ijomah, W., Gachagan, A., & Marshall, S. (2018). Activation Functions: Comparison of Trends in Practice and Research for Deep Learnin. arXiv.

OpenCV. (2018a). Thresholding. Retrieved April 03, 2020, from Open Source Computer Vision: https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html

OpenCV. (2018b). Introduction to Principal Component Analysis (PCA). Retrieved April 03, 2020, from Open Source Computer Vision: https://docs.opencv.org/3.4/d1/dee/tutorial_introduction_to_pca.html

Saxena, A., Driemeyer, J., & Ng, A. Y. (2009). Learning 3-D Object Orientation from Images. 2009 IEEE International Conference on Robotics and Automation, 794-800.

Shafkat, I. (2018, June 1). Towards Data Science. Retrieved April 10, 2020, from Intuitively Understanding Convolutions for Deep Learning: https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Song, F., Guo, Z., & Mei, D. (2010). Feature selection using principal component analysis. 2010 International Conference on System Science, Engineering Design and Manufacturing Informatization, 27-30.