Viability of Image Classification with Introduction of Transparent Barriers

GUSTAV NORÉN
EMANUEL RAVEMYR
Viability of Image Classification with Introduction of Transparent Barriers

GUSTAV NORÉN
EMANUEL RAVEMYR
Abstract- In this thesis, image classification of difficult data through the use of machine learning algorithms is evaluated using a Kernel Machine. When trying to classify objects in images in real world situations the object in question might be behind some form of transparent obstruction or barrier. By implementing a simple machine learning algorithm this thesis aims to provide an approximate lower bound for the performance of machine learning algorithms in such circumstances.

Results show that machine learning is a viable option even though performance decrease with barrier complexity and thickness. In the best case performance dropped less than one percentage point when using a simple barrier compared to using no barrier and allowing the algorithm to train on images with objects behind said barrier. Performance is much worse when not allowing the algorithm to train on images with barriers. Furthermore, performance seems to be largely independent on image size despite the loss of information associated with introducing barriers.
Viability of Image Classification with Introduction of Transparent Barriers

Gustav Norén & Emanuel Ravemyr

Abstract—In this thesis, image classification of difficult data through the use of machine learning algorithms is evaluated using a Kernel Machine. When trying to classify objects in images in real world situations the object in question might be behind some form of transparent obstruction or barrier. By implementing a simple machine learning algorithm this thesis aims to provide an approximate lower bound for the performance of machine learning algorithms in such circumstances.

Results show that machine learning is a viable option even though performance decrease with barrier complexity and thickness. In the best case performance dropped less than one percentage point when using a simple barrier compared to using no barrier and allowing the algorithm to train on images with objects behind said barrier. Performance is much worse when not allowing the algorithm to train on images with barriers. Furthermore, performance seems to be largely independent on image size despite the loss of information associated with introducing barriers.

I. INTRODUCTION

Machine learning has garnered immense attention lately and is considered one of the largest growing fields in contemporary science. This is despite the fact that the machine learning concept is quite old. In other words something seems to have happened in the last couple of years making machine learning algorithms widely useful. Partly due to better training methods (machine learning algorithms) but mostly due to the growing production of data in today’s society and more powerful computers. As a direct result there is a necessity to identify the type of problems and/or datasets in which a machine learning solution is applicable.

Normal areas where machine learning is applied include financial applications, classifying pictures and helping doctors diagnose patients. Generally these problems can be divided in two categories i.e. regression problems and classification problems. A classification problem has the relatively simple goal of taking some input and placing it in a single category. An arch-typical example would be identifying which animal is in a picture. A regression problem takes some input and instead of mapping it to a single category it is placed in multiple. An example could be guessing house market trends where output parameters to guess could be house price, location, distance to services, paint color and so on.

This thesis will focus on image classification, the problem of determining what a picture depicts. The area has quickly grown in the hands of social media, which eagerly explores facial recognition, behavior etc. This area also has important applications in law enforcement and in the case of Amazon Go; retail [1]. In other words the possibilities of image classification are already being explored heavily. For a more thorough explanation of the possibilities and limitations of classification problems see for example [2].

However, most exploration is done using clear images whereas few experiments are done with physically obscured data. For instance a person seen behind the windshield of a car or an object being reflected in water. A person watching through the windshield of a car may be distracted by reflections, but would in most cases be able to determine the presence or absence of a person behind the wheel. For a computer, that may not be so easy. Taking it one step further, recognizing attributes of a person, may prove to be too much of a challenge. Coming up with a framework to solve these kinds of issues can be used in above applications and surely many more.

This thesis will aim to evaluate the performance of machine learning applied on a database of images where the object is hidden behind some transparent barrier. In such a case the image is not only less visible but can also be distorted. So far experimentation, as seen in [3] and [4], has been done by applying computer generated barriers such as Gaussian noise, blurring and contrast. In this thesis two different physical barriers will be used in varying thickness. If machine learning methods perform well on this more general problem it would suggest that machine learning methods could be more powerful than previously known. An example image of how the dataset looks is given below, see Figure 1.

II. TERMINOLOGY

In this thesis the physical obstructions used in images will be denoted barriers. Images taken of objects that are not placed behind a barrier will be denoted as clean or raw images while
all images with objects obstructed behind a transparent barrier will be denoted as barrier images or images with barrier.

III. Theory

The core concepts of Machine Learning and Pattern Recognition is outlined in [5]. Some further explanation of the more important concepts and algorithms are given below as well as a detailed explanation of the algorithm used in this thesis.

A. Classification problems

In the context of this thesis the classification problem concerns object recognition. The problem is supervised, as all objects given to the algorithm have already been classified by a human supervisor. Given some data for testing \( T \), a database \( D \), a classification \( C \) and some Classification algorithm \( A(p, q) \rightarrow R(q, C) \) the classification problem may be expressed as:

\[
A(D, T) \rightarrow R(T, C)
\]

Where \( R \) is a comparison of the Algorithm classification and the supervised classification \( C \). The general goal of the problem is to make the algorithm result as close as possible to the given classification, without making the algorithm aware of the existing classification.

B. Supervised learning algorithms

Since this thesis only concerns classification problems this section will cover how supervised learning algorithms (SLA) solve this problem in particular even though SLAs can be used on other problems. In essence SLAs are used to classify unknown input given some set of known data.

SLAs use a set of labeled training data

\[
D = \{(x_1, y_1), ..., (x_n, y_n)\}
\]

where each \( x_i \in \mathbb{R}^d \), \( y_i \in \{1, ..., C\} \), \( d \) is the dimension of an input vector \( x_i \) and \( C \) is the number of categories. The goal is to use \( D \) to find (learn) a classification function

\[
g : \mathbb{R}^d \rightarrow \{1, ..., C\}
\]

In other words to take an input vector of dimension \( d \) and return a suggested class for said vector. Here a typical example of how the classification function looks and functions will be given.

As seen in [5] and [6] the classification function generally has the form

\[
g(x, W, b) = \arg \max_{1 \leq j \leq C} f_j
\]

where \( f_j = w_j^T x + b_j \) and

\[
W = \begin{pmatrix} w_1^T \\ \vdots \\ w_C^T \end{pmatrix} \in \mathbb{R}^{C \times d}, \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_C \end{pmatrix} \in \mathbb{R}^{C \times 1}
\]

Furthermore [5] states that the function \( f = Wx + b \) in (1) is called the (linear) classifier and returns a vector with a score for each of the \( C \) categories. This score is how much the algorithm considers an input vector \( x \) to belong to each category respectively. Function \( g \) then classifies the input as the category with the greatest score.

To explain why this is possible consider the most important aspect of the function \( f \) i.e. the weight matrix \( W \). It is a matrix of real values called weights where each weight has the purpose of deciding how much a single input element in \( x \) contributes to the score of a single category. Another way to see it is that each class has some template that represents all inputs that belong to that class. Then if an input vector corresponds well to a template it will receive a high score and vice versa. The bias vector \( b \) is just the scores that will be output if no input vector (or a zero vector) is given and will not be considered in this thesis but is mentioned because of its frequent use.

There remains the problem of learning \( W \) and \( b \) so that unknown input vectors are classified as accurately as the algorithm is capable of. Commonly this is done by utilizing loss functions, again according to [5] and [6]. To simplify notation labels for the training data will be represented by one-hot target vectors since the function \( f \) (in equation (1)) is in vector form. One-hot target vectors \( t_i \) replaces the label \( y_i \) where each element in \( t_i \) is zero except element \( y_i \) which is set to one. In this way the classification will be correct if \( j \) in the classification function \( g \) (equation (1)) corresponds with element \( y_i \) in \( t_i \).

One basic loss function as seen in [5] is the squared error loss (equation (3)) which is very intuitive. Here the loss function is simply the squared distance from the target vector \( t_i \) and the vector \( f \). Since \( f \) is the score for each class the loss function has the meaning of distance from the target label to the suggested label (in one-hot representation).

\[
L(D, W, b) = \frac{1}{n} \sum_{i=1}^{n} ||t_i - (Wx_i + b)||_2^2
\]

To learn \( W \) and \( b \) one needs to find elements in \( W \) and \( b \) that minimize the loss. However, it is undesirable to find \( W \) and \( b \) so that the loss is zero. Zero loss would mean perfect classification of the training data but the networks purpose is to classify unknown data which is seldom a perfect match of a training example. Achieving too low loss is called over-fitting.

The most common way to mitigate this as seen in [5] and [6] is to use a regularization term.

\[
L(D, W, b) = \frac{1}{n} \sum_{i=1}^{n} ||t_i - (Wx_i + b)||_2^2 + \lambda ||W||_F^2
\]

Where

\[
||A||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} ||a_{i,j}||^2}
\]

for some matrix \( A \in \mathbb{R}^{m \times n} \) is called the Frobenius norm and is the matrix equivalent of the 2-norm for vectors. Lambda is called the regularization parameter, fulfills \( \lambda \geq 0 \) and determines the amount of regularization. There is no simple way to know what lambda should be for best performance on testing data.

In this thesis optimization of the loss function will be done with a well known trick called the kernel trick as shown in
C. Kernel Machine

As will be shown the kernel machine is an algorithm which is exceptionally fast at training. However, training time scales with the size of the dataset and usually substantial amounts of memory is required. In other words kernel machines are suitable for smaller datasets where each input vector is very large since training time mostly scales with the number of input vectors and not their size.

In the endeavor to optimize loss some new notation will be introduced. For simplicity the bias vector $b$ shall be omitted, this vector can be hidden inside the weight matrix $W$ if one so desires but is not necessary for the goal of this thesis. Generalize the multiplication of $W$ with $x_i$ by letting $W$ be multiplied by some function $\phi(x_i) \in \mathbb{R}^{d \times 1}$ and let $T = (t_1, \ldots, t_n) \in \mathbb{R}^{C \times n}$, $\Phi = (\phi(x_1), \ldots, \phi(x_n)) \in \mathbb{R}^{d \times n}$, this way the loss function can be written simpler as:

$$L(W) = \frac{1}{n} \left| \left| T - (W \Phi) \right| \right|^2_F + \lambda \left| \left| W \right| \right|^2_F.$$

Minimizing the loss over $W$ is equivalent to

$$\arg\min_W \left[ \frac{1}{n} \left| \left| T - (W \Phi) \right| \right|^2_F + \lambda \left| \left| W \right| \right|^2_F \right]$$

(4)

This can be done by taking the derivative of $L$ with regards to $W$ and finding where this derivative is zero. In other words fulfilling equation (5).

$$\frac{dL}{dW} = -\frac{1}{n} (T - W \Phi) \Phi^T + \lambda W = 0$$

(5)

Rearranging equation (5)

$$W = \frac{1}{\lambda n} (T - W \Phi) \Phi^T$$

and renaming part of it as

$$\psi = \frac{1}{\lambda n} (T - W \Phi)$$

one ends up with

$$W = \psi \Phi^T$$

(6)

Inserting equation (6) into (4) one ends up with an equation equivalent to (4) but that can be optimized over $\psi$:

$$\arg\min_\psi \left[ \frac{1}{n} \left| \left| T - (\psi \Phi^T \Phi) \right| \right|^2_F + \lambda \left| \left| W \right| \right|^2_F \right]$$

(7)

At this point it is starting to become evident why kernel machines can be trained quickly. Comparing equations (4) and (7) the striking difference is the fact that a matrix $W$ multiplied by vectors which are then summed has turned into matrix multiplication between $\Phi^T$ and $\Phi$ which is called the kernel function or rather the matrix variant of it.

The kernel function, sometimes referred to as the kernel trick as seen in [5], is simply the dot product of a function $\phi$ acting on two vectors:

$$k(x, x') = \phi(x)^T \phi(x')$$

Where $x$ and $x'$ are any two input vectors. As a side note the function $\phi(x)$ is called a feature space mapping, see [5] for more details on this.

There are several ways to construct a kernel function as seen in [5] but in this thesis a well known kernel will be used called the Gaussian Kernel, or a Radial Basis Function Kernel, which holds the property:

$$k(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right)$$

where any norm could be used instead of the 2-norm but the 2-norm is one of the most commonly used. Specifically the kernel function that will be used in this thesis is:

$$k(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right)$$

Using the kernel definition to substitute $\Phi^T \Phi$ in (7) one achieves

$$\arg\min_\psi \left[ \frac{1}{n} \left| \left| T - (\psi \Phi) \right| \right|^2_F + \lambda \left| \left| W \right| \right|^2_F \right]$$

(8)

Where $K = \Phi^T \Phi \in \mathbb{R}^{n \times n}$ is the kernel function applied to all training data.

To optimize for $\psi$ in equation (8) one can take the derivative with regards to $\psi$ and find where the derivative is zero. Doing so yields the following:

$$-\frac{1}{n} (T - \psi K) K^T + \lambda \psi K = 0$$

(9)

The matrix $K$ is symmetric and thus hold the property $K^T = K$ meaning equation (9) can be reduced to:

$$-\frac{1}{n} (T - \psi K) + \lambda \psi = 0$$

Solving for $\psi$ one gets:

$$\psi^* = T (K + n \lambda I)^{-1}$$

(10)

Where $I$ is the identity matrix and the resulting matrix $\psi^*$ gets denoted by a star to signify that it is the optimal $\psi$. In equation (6) it is stated how $\psi$ relates to $W$ and by using $\psi^*$ the appropriate weights for $W$ is found. To classify some unknown input $x$ first apply function $\phi(x)$ and then multiply $W$ with $\phi(x)$. In following the above procedure one ends up with a suggested target vector equivalent to the function $f$ in equation (1).

$$\hat{t} = T (K + n \lambda I)^{-1} k(x)$$

(11)

Here $\hat{t}$ denotes the vector equivalent to $f$ and $k(x) = \Phi^T \phi(x)$ is the kernel function applied on $x$ and all training data.

D. Classification Accuracy and Error

In the testing part of this thesis, two types of errors are tested. First is the classification error which is the measure of misclassification in an absolute sense. Namely, the error is given as the number of incorrectly classified images divided by total number of tested images. The classification error is mostly presented by its inverse, the classification accuracy and is a very intuitive way to present algorithm performance.

The absolute classification error is a more deliberate measure of classification performance, as it measures the norm of...
the vector distance between the algorithm’s classification and the correct classification. This is a very good measure for comparison between classification on different sets. The value does however not give any additional understanding. See equation (12) for the error function used in this thesis.

\[
\text{error} = 20 \times \log \left( \frac{||T - \hat{T}||_F}{||T||_F} \right)
\] (12)

IV. METHODOLOGY

The project follows a very causal time line. First, a dataset must be created with some images without barriers. These form the basic training set. In this case, the objects are fruits. Due to fruit being prone to change colour or shape with time and environmental impact, the images were taken over the course of one weekend. Then images with barriers must be added into the dataset, in order to do varied testing and training. The same objects will be used in the barrier images as the non-barrier images. The barriers in this case was plastic bags and plastic sheets. In order to test the limitations of the method, one, two and three layers of barriers were applied to the objects, see section IV-A. When the dataset is completed, an algorithm must be devised to process the data. In this case, a Kernel Machine was used, see section IV-B. Finally, the algorithm is used to test the dataset, which will provide the results of this thesis. How this is done is described in section IV-C.

A. Generating the new dataset

The dataset consists of ten categories of fruit and at least five specimen for each category. Usage of at least five specimen was considered enough since it is possible to orient the same specimen in different ways so that it is difficult to tell if an image is of the same specimen or not.

There is always a trade off between size and quality when creating a dataset. For this thesis it was decided that the quality and diversity of the dataset was the most important goal to achieve. Therefore, the datasets are very diverse but rather small. This decision was made mainly for time concerns since the creation of data was not very costly or advanced, but very time consuming.

For barriers, two types were used to examine different ways in which images might be occluded. One is transparent grocery bags and the other is transparent plastic sheets. Grocery bags soften corners but somewhat take the shape of the object whereas plastic sheets merely serves as a filter. In total three levels of bags were used by placing the objects inside one, two or three bags at a time. For foil four levels were used by placing objects behind one, two, three or four plastic sheets at a time. Based upon the assumption that humans have a harder time correctly classifying objects inside many layers of plastic bags than behind sheets more levels of sheets was used. The assumption was made on the authors observation while collecting fruits.

Per category of fruit 25 clean images (without barrier) were collected and 15 barrier images per level of barrier totaling to a dataset of 1300 images. However, the largest dataset to be used in a single test consists of 400 images. This is because most images belong to different types of barrier images. Each image can contain either one or more specimen of the same category. In order to reduce misclassification due to outside factors, all images were taken in the same location, with similar light conditions and background.

For images with barrier, two different approaches were adapted. When the barrier consisted of plastic bags, the items were simply placed into the bags and then photographed. Efforts were made to ensure that there was minimal folding of the plastic bag over the objects, so that the number of layers would be consistent. When the plastic foil was used as the barrier, a sheet of foil was fixated over the table at an angle to give some constant reflection from light. When applying more layers, these were placed upon the already fixated foil sheet. The properties of the plastic foil lead to sheets sticking together. Due to this some ripples were caused in the surface. Since the same sheet was used for all images, the effect of the ripples are similar throughout.

All images were taken with a Canon EOS 400 D in raw format, cropped to center the object(s) in question and then downsampled to .png files. Downsampling was done to different image sizes for the purpose of examining how performance is dependent on image size which is important in real world applications. Pixel sizes for images used was $339 \times 250$ and $15 \times 11$ and each pixel contains RGB values.

B. Algorithm

In this thesis an algorithm called a Kernel Machine is used. For a full explanation on how the Kernel Machine works see chapter III. The primary reason for using a Kernel Machine is because it is much faster to train in comparison to other methods for the specific dataset used. In general networks have one weight matrix (or more) $W$ which is filled with $C \times d$ number of weights, see equation (2), which have to be learned. Where $C$ is the number of classes to identify and $d$ the dimension of an input vector. If the dimension $d$ is very large training time will be great since the amount of weights to learn is also great. However, by using Kernel Machines the amount of weights to learn is essentially reduced to $n \times n$ where $n$ is the number of training examples. In other words for the relatively small dataset ($n$ never exceeds 350) in this thesis with extremely large input vectors ($d = 339 \times 250 \times 3 = 254250$) a Kernel Machine is appreciated.

The algorithm was implemented using MATLAB since the creation, manipulation and usage of large matrices is central to using the Kernel method. First, a database generating function was written which converted the image files into vectors holding data for the RGB-values of each point. Other functions were then written to generate the kernel matrix, finding the optimum value for $\lambda$ and calculate the classification accuracy. All inner details of how the Kernel Machine works are in chapter III. There are some variations to how Kernel Machines can be done but only the one used in this thesis is covered in chapter III.

A more advanced algorithm than the Kernel Machine would probably perform better but would only lengthen training times and is not important for the purpose of this thesis.
C. Testing the dataset

The testing procedure is relatively simple but is tailored to yield the easily comparable results with a small dataset. First what is tested will be outlined for clarity followed by the actual procedure of how the testing was done.

The main purpose is to compare performance between using barriers and not using barriers (clean images). As such tests were done on clean images, barrier images trained on only clean images and barrier images trained on both clean and barrier images. Furthermore tests were done on high pixel images (339 × 250 pixels) and low pixel images (15 × 11 pixels).

Since the dataset is small how one selects which data will be used for training the network and which will be used for testing it has a large impact on performance. As such the testing set was randomly generated and evaluated many times for each experiment i.e. run for many iterations. In the case of tests only on clean images 50 out of 250 images were used for testing and the remaining 200 for training. For barrier experiments 50 out of 150 barrier images were used for testing and either all clean images were used for training or all clean images and the remaining 100 barrier images. In each iteration the testing set was picked at random by taking five images from each class. This ensured that training was equal for all types of images. In total, 500 iterations were done for each test.

After all iterations are done, the saved accuracies are used to get a total average accuracy, as well as to find the standard deviation of the accuracies. This data is which is presented in the results section. By running many iterations statistical fluctuations cancel out and what remains is easily comparable.

Testing is done on images without any barrier, as well as images up to three bags or four foils. No image contains both barriers.

V. Results

The results of testing will be presented with graphs and tables. They are divided into tests on bags and tests on foil with tests done on clean images added to both for comparison. All details are presented in tables that include the image size, average accuracy across all iterations, average absolute error across all iterations and standard deviations for average accuracy and error. Then graphs are shown of the average accuracies for easy comparison.

Since many iterations are run for each test a visualization of accuracies for all iterations is given in Figure 2.

A. Results on plastic bags as barrier

In Table I and II are results of tests done when using plastic bags as barrier. The case of zero layers of barrier indicate testing on clean images and in this case training data are 200 clean images and testing data is 50 clean images. In the case of one or more layers all 250 clean images are used for training. For Tabel I the Kernel Machine only uses clean images as training data and for Tabel II it additionally uses 100 barrier images.

To get a more clear overview of how the accuracy depends on the number of barriers, see Figure 3 and 4.

B. Results on plastic foil as barrier

For the accuracy on the datasets tested on images with plastic sheets, see Table III and IV.

Once more, the accuracies are presented in graphs for the reader’s convenience, see Figure 5 and 6.

VI. Discussion

A. Creating a dataset

The creation of proper datasets is one of the most important parts in machine learning since a poor dataset might not yield accurate results. The creation of the dataset was more time demanding than expected, and therefore the creation took place over the course of two days. It is entirely possible that the fruit may have changed colour enough to reduce classification performance. Another aspect of the time demanding process is the natural changing of light. Despite choosing a room with sufficient lighting, natural light may have changed during the creation of data. This may also bear an adverse effect on the results.

This changing of light may have had an even larger effect on the images where the reflection of light played a large part,
namely the images with the sheet barrier. Since images of the same class were taken within a small time interval the algorithm may learn to recognize the light signature of a class, rather than the feature that a person might recognize fruit from. This is especially bad as images belonging to the same class will not typically have similar light signatures in another setting.

Another point that may impact the classification performance is the aspects of focus and centering. For most images, efforts were made to ensure the camera was focused on the objects but sometimes the barrier came into focus instead. Furthermore, each object was not centered when taking the picture but centered by cropping resulting in some irregularities in size, form and background. Images are however quite diverse so this should not contribute much to bad performance but it is worth pointing out.

B. Choosing an algorithm

When choosing the Kernel Machine an assumption on the necessary image size was made. The idea was that large amounts of pixels would counteract the reduced visibility and as such let performance drop less than for small images. However, results indicate that severely reducing image size has very little effect on performance. In other words a more complex algorithm could be used and training time would still be low. One expects that in such a case performance would be higher but that is of little import to the result’s viability since the main concern is the drop in accuracy from clean images to images with barriers.

It is also important to note that the results of a specific network may not generalize to results of other networks. Since the basic idea is the same in most networks one expects that results do generalize but it is unclear how changing a network would impact say performance on images with many layers of barrier. Deeper networks with more than the one layer used in this thesis may be better at disregarding the barriers but could also overfit more to the specific barrier. In essence more research is needed.

C. Tests

Since the dataset is very diverse, it’s small size may cause issues. In this case, accuracy may depend heavily on what images get chosen for testing. To avoid this issue, testing
TABLE IV
RESULTS WHEN TRAINING ON CLEAN DATA AS WELL AS THE PLASTIC FOIL(S) AND TESTING ON PLASTIC FOIL(S).

<table>
<thead>
<tr>
<th>Size (px)</th>
<th>No. of Layers</th>
<th>Average accuracy (%)</th>
<th>$\sigma$ of average accuracy (pp)</th>
<th>Average absolute error</th>
<th>$\sigma$ of Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>15x11</td>
<td>0</td>
<td>69.9</td>
<td>0.228</td>
<td>1.5</td>
<td>0.8</td>
</tr>
<tr>
<td>339x250</td>
<td>0</td>
<td>70.2</td>
<td>0.223</td>
<td>2.34</td>
<td>0.45</td>
</tr>
<tr>
<td>15x11</td>
<td>1</td>
<td>69.4</td>
<td>0.237</td>
<td>1.77</td>
<td>0.54</td>
</tr>
<tr>
<td>339x250</td>
<td>1</td>
<td>57.6</td>
<td>0.237</td>
<td>1.62</td>
<td>0.28</td>
</tr>
<tr>
<td>15x11</td>
<td>2</td>
<td>60.2</td>
<td>0.268</td>
<td>1.28</td>
<td>0.49</td>
</tr>
<tr>
<td>339x250</td>
<td>2</td>
<td>46.7</td>
<td>0.219</td>
<td>1.14</td>
<td>0.32</td>
</tr>
<tr>
<td>15x11</td>
<td>3</td>
<td>65.7</td>
<td>0.246</td>
<td>1.30</td>
<td>0.52</td>
</tr>
<tr>
<td>339x250</td>
<td>3</td>
<td>47.9</td>
<td>0.232</td>
<td>1.20</td>
<td>0.24</td>
</tr>
<tr>
<td>15x11</td>
<td>4</td>
<td>61.1</td>
<td>0.246</td>
<td>1.12</td>
<td>0.59</td>
</tr>
<tr>
<td>339x250</td>
<td>4</td>
<td>50.3</td>
<td>0.219</td>
<td>1.37</td>
<td>0.19</td>
</tr>
</tbody>
</table>

Fig. 5. How accuracy depends on the number of layers of foil when training on clean and/or layer images for images of size 15x11.

is done in several iterations. According to the central limit theorem in statistics, this will cause the mean of the calculated values to reach a proper mean of the distribution. This should lead to results that are fairly statistically secure. However, tests were made with 500 iterations, which while a large number, may not be sufficiently large to remove all statistical flux. Because of this, the deviation of both accuracy and classification error is given, alongside the actual performance of both these measurements. Since the values of these deviations are small throughout the results (See Table I to IV) one can verify that the number of iterations is indeed large enough to produce satisfactory results.

D. Accuracies depending on number of layers and training set

Consider the results from training only on raw images, see Tables I and III and Figures 3, 4, 5 and 6. It is clear that performance drops significantly with increasing number of layers which is in line with the expectation. Each introduced layer of barrier increases the difference between tested images and trained images. Worth noting is that performance is still better than random guessing even for high number of layers and much better than random guessing on few layers.

For results when training on raw images and barriers (of the same layer as testing) see Tables II and IV and Figures 3, 4, 5 and 6. Except for anomalies with higher accuracies for few pixels (which is discussed in section VI-E) the same pattern as before is apparent with decreasing accuracies with increasing number of layers. However, in this case accuracies are in general much higher with the lowest accuracies (high number of layers) are higher or about the same as training only on clean and testing on a single layer. These results are also within expectation because of two reasons. One is that introducing barriers in training data gives the network opportunity to learn how barriers distort the image. The other is that the number of training data has increased which can be a source of unreliable results. It is unknown how large performance would be in the case of training only on raw images but using as many raw images as the total amount of training data used in when training on both raw and barrier images. In other words these results are not entirely comparable but given the small dataset this approach is still preferable since otherwise the training set would have a very high ratio of barrier images.

As a side note an interesting phenomena can be observed in Figures 3, 4, 5 and 6. For some cases performance increase when the number of layers increase. This is contrary to the expectation.

E. Accuracies depending on image size

The expected result of higher performance on images with more pixels seems to be refuted, see in particular Figure 5 compared to 6 but also Figure 3 compared to 4. In the case of plastic sheets this could be related to the fact that the same sheets are used for all images and taken in a series (see section VI-A). It could also be related to how down-sampling works and the fact that it relates to some average over a number of pixels. If distortions cancel out on average down-sampling
would result in images that seem less obstructed by a barrier. However, more research is needed to draw any conclusions.

F. Comparison of barrier type

In Figure 1 images using both barriers are shown and for a human onlooker it is possible to surmount that the case of plastic sheets is easier to classify. The statement is based upon the fact that plastic bags reduce the visibility of edges much more than the plastic sheet. However, comparing Figures 3 and 4 to Figures 5 and 6 (trained on both clean and barrier) it is not easy to draw such a conclusion. On one hand Figures 4 and 6 (high number of pixels) seem to indicate that the Kernel Machine had a harder time classifying images using plastic sheets as barriers. On the other hand Figures 3 and 5 (low number of pixels) seem for the most part indicate that images using plastic bags are harder to classify except for the case of two layers. As to why this might occur is difficult to say but the phenomena is worth pointing out for further study.

G. Classification error and standard deviations

The highly diverse dataset presented a problem of accuracies being highly dependent upon the choice of training data, see Figure 2. By randomly selecting training data and running the same experiment many times and calculating the mean value this was counteracted as can be seen in the case of the standard deviations (for the mean) being around 0.2pp. On the other hand calculating mean for the accuracies in Figure 2 or standard deviations (for the mean) being around 0.2pp. On the other hand calculating mean for the accuracies in Figure 2 or similar yields a value around 5pp.

H. Applicability to real world settings

The dataset used is very diverse. This was however a conscious decision by the authors, as diversity of data is a very natural thing. In an applied setting, data would in many cases be more uniform than the used data set. But since the algorithm is functional already with very diverse data, it makes sense that performance would not be adversely affected if testing- and training were data more similar to one another. An appropriate next step before implementing this would be to create a larger dataset which would be less diverse but contain even more classes and data points. This dataset could then be run on a more advanced network, as the results of this thesis seems to suggest that performance is not too sensitive in regard to the size of the trained images. This would enable the use of neural networks, etc. These types of networks typically perform very well and could bring the project to a state where it is prepared for applications.

VII. CONCLUSION

The results indicate that the usage of machine learning is applicable when trying to classify images behind barriers, as long as training is done on data with similar barriers. The performance drop in these cases were less than 8% for all instances when testing on one barrier layer, and for some datasets, around 1%. Given the similarities between many machine learning algorithms it is reasonable to expect similar performance differences for other situations and networks.

In other words, results suggest that using a more advanced network and larger datasets performance may become sufficiently high for real world applications. The results do however suggest that training only on clean images would not be sufficient when trying to achieve acceptable performance. Here the performance was shown to drop between 20% to 40% on one layer and only performing slightly better than random choice on several layers. This would most likely not be reversed by a larger dataset or a more powerful network.

APPENDIX A
MATLAB CODE

```
function [accVec, errVec] = runTests(barrier, nrTests)
    %testSets = {'Raw', 'Bag1', 'Bag2', 'Bag3'};
    testSets = {'Raw', 'Foil1', 'Foil2', 'Foil3', 'Foil4'};
    testSets = {'Raw'};
    nrTests = 500;
    testVec = 0:length(testSets) - 1;
    meanAccVec = zeros(1, length(testSets));
    meanErrVec = zeros(1, length(testSets));
    for i = 1:length(testSets)
        [accVec, errVec] = runTests(testSets(i), nrTests);
        meanAcc = mean(accVec);
        meanErr = mean(errVec);
        stdAcc = std(accVec);
        stdErr = std(errVec);
        save([testSets(i), 'accVec', 'errVec', 'meanAcc',
              'meanErr', 'stdAcc', 'stdErr']);
    end
    plot(testVec, meanAccVec);
    xlabel('Number of barrier layers');
    ylabel('Accuracy');

function [X_Train, T_Train, X_Test, T_Test] = LoadTrain();
for i = 1:nrTests
    if strcmp(barrier, 'Raw')
        [X_Train, T_Train, X_Test, T_Test] = loadRaw();
    else
        [X_Train, T_Train, X_Test, T_Test] = loadBarrier(barrier);
    end
    acc = mean(accVec);
    err = mean(errVec);
    stdAcc = std(accVec);
    stdErr = std(errVec);
    save([testSets{i}, 'accVec', 'errVec', 'meanAcc',
          'meanErr', 'stdAcc', 'stdErr']);
end
plot(testVec, meanAccVec);
xlabel('Number of barrier layers');
ylabel('Accuracy');

MATLAB CODE

function [X_Raw, T_Raw] = LoadRaw()
filepath = '...';
classes = {'apple', 'banana', 'carrot', 'lemon', 'onion',
    'orange', 'pear', 'paprika', 'potato', 'tomato'};
X_Raw = [];
T_Raw = [];
for i = 1:length(classes)
    t = zeros(10, 1);
    for j = 1:25
        image = imread(fullfile(filepath, classes{i}, num2str(j), '.png'));
        X_Raw = [X_Raw, image(:)];
        T_Raw = [T_Raw, t];
    end
end
X_Raw = im2double(X_Raw);
```
F6A: VIABILITY OF IMAGE CLASSIFICATION

function([X_Barrier,T_Barrier])=LoadDataBarrier(barrier);
filepath = '...
classes = {'apple','banana','carrot','lemon','onion','orange','pear','paprika','potato','tomato'};
X_Barrier = [ ];
T_Barrier = [ ];
for i=1:length(classes)
t = zeros(10, 1);
t(i) = 1;
for j=1:15
    image = imread(fullfile,barrier, classes(i), num2str(j),'.png');
    X_Barrier = [X_Barrier, image(:)];
    T_Barrier = [T_Barrier, t];
end
end
X_Barrier = im2double(X_Barrier);
end

function [X_Train,T_Train,X_Test,T_Test] = RandomizeDataRaw(X_Raw, T_Raw)
X_Train = [ ];
T_Train = [ ];
X_Test = [ ];
T_Test = [ ];
for i=1:10
    testImages = randperm(25, 5);
    lia = ismember(1:1:25, testImages);
    for j=1+(i-1)*25:25+(i-1)*25
        if lia(j) == 1
            X_Test=[X_Test,X_Raw(:,j)];
            T_Test=[T_Test,T_Raw(:,j)];
        else
            X_Train=[X_Train,X_Raw(:,j)];
            T_Train=[T_Train,T_Raw(:,j)];
        end
    end
end
end

function [X_Train,T_Train,X_Test,T_Test] = RandomizeDataBarrier(X_Raw, T_Raw, X_Barrier, T_Barrier)
X_Train = X_Raw;
T_Train = T_Raw;
X_Test = [ ];
T_Test = [ ];
for i=1:10
    testImages = randperm(15, 5);
    lia = ismember(1:1:15, testImages);
    t = zeros(10, 1);
t(i) = 1;
for j=1+(i-1)*15:15+(i-1)*15
    if lia(j) == 1
        X_Test=[X_Test,X_Barrier(:,j)];
        T_Test=[T_Test,T_Barrier(:,j)];
    else
        %Comment these lines out to remove training on barriers
        X_Train=[X_Train,X_Barrier(:,j)];
        T_Train=[T_Train,T_Barrier(:,j)];
    end
end
end
end

function [acc, err] = KernelCodeFunction(X_Train, T_Train, X_Test, T_Test)
J = length(X_Train(1,:));
K = my_kernel_fast(X_Train,X_Train);
K_test = my_kernel_fast(X_Train,X_Test);
[bestLambda,lambdaVector,accVec,T_hat] = find_best_lambda(T_train,T_test,J,K,K1);
acc = Calculate_accuracy(T_Test,T_hat);
err = abs(Calculate_error(T_Test,T_hat));
edn

function K = my_kernel_fast(X,Y)
N1 = size(X,2);
N2 = size(Y,2);
n1sq = sum(X.^2,1);
n2sq = sum(Y.^2,1);
D = (ones(N1,1)*n1sq)’ + ones(N1,1)*n1sq -2*(X’*X);
temp = 2*sum(sum(D))/N1/N1;
D = (ones(N2,1)*n2sq)’ + ones(N1,1)*n2sq -2*X’*Y;
K = exp(-D/temp);
return
end

function [useLambda, lambdaVector, accVec, T_hat] = find_best_lambda(T_train,T_test,J,K,K1)
maximal = 0;
bestLambda = 0;
lambdaVector = [ ];
accVec = [ ];
lambdaRange = 10.^( -10:10);
for lambda = lambdaRange
    I = eye(J);
    M = T_train*((K +lambda*J*I)^(-1));
    thatt = M*K1;
    count = Calculate_accuracy(T_test,thatt);
    lambdaVector = [lambdaVector,lambda];
    accVec = [accVec,count];
    if count > maximal
        maximal = count;
        bestLambda = lambda;
    end
end
useLambda = bestLambda;
M = T_train*((K +useLambda*J*I)^(-1));
T_hat = M*K1;
end

function accuracy = Calculate_accuracy(T,T_hat)
errnum = 0;
N = size(T,2);
for i=1:N
    score_est = T_hat(:,i);
    score_gt = T(:,i);
    [~, maxind_est] = max(score_est);
    [~, maxind_gt] = max(score_gt);
    if(maxind_est~=maxind_gt)
        errnum = errnum + 1;
    end
end
accuracy = (N-errnum)/N;
end

function error = Calculate_error(T,T_hat)
error = 20*log10(norm(T-T_hat,’fro’)/norm(T,’fro’));
end

ACKNOWLEDGMENT

The authors would like to thank their Thesis advisor Saikat Chatterjee for challenging us intellectually and giving us a great foundation for this project. We would also like to thank Alireza Javid for helping us with complicated concepts, and providing efficient code for us to use.

REFERENCES


