POLYTECHNIC UNIVERSITY OF BUCHAREST

FACULTY OF AUTOMATIC CONTROL AND COMPUTERS

AUTOMATION AND INDUSTRIAL INFORMATICS DEPARTMENT

MASTER RESEARCH THESIS

Machine learning multi-class classification: dog breed identification

Thesis supervisor:

Prof.univ.dr.ing. Liliana Dobrică

Student,
Cristinel-Adrian Popescu

Table of Contents

Introduction

A central theme in machine learning is that of sequential decision-making. This is the mission to decide, from experience, the sequence of actions to be taken in an uncertain environment in order to achieve certain objectives. A series of sequential steps in decision-making tasks cover a wide range of possible applications with the potential to have an impact in areas such as robotics, healthcare, smart grids, finance or self-driving cars are just a few of them. Machine learning (abbreviated as ML) provides automated methods that can detect patterns in data and use them to perform certain tasks. In addition to learning by consolidation, two essential types of machine learning tasks can be considered: supervised learning, the task of deducing a classification, or regression from labeled training data, and the opposite: unsupervised learning being the task of drawing inferences from datasets consisting of unlabeled input data.

During the last few years, computer software applications have gone a dramatic transformation from simple data processing to machine learning and in some cases artificial intelligence, thanks to the availability and accessibility of a huge volume of data collected through sensors and the internet. The idea of machine learning demonstrates and grows the fact that the computational system has the ability to improve itself with the advancement of time. This study paper targets machine learning and computer vision domains, explores and analytically evaluates an application in the computer vision domain using some deep learning / machine learning techniques, and predicts future candidates.

After batch processing of the dataset in the SaaS platform, Google

Colab (similar to Anaconda Jupyter Notebook but with more facilities like allowing running TensorFlow 2.0 on any hardware) using their free access to an Nvidia GPU, the commonly used algorithm are neural networks. The most recent ML applications in computer vision are object classification, object detection, extraction of relevant information from images, graphical or text documents, and video images. Machine learning and computer vision hope to bring into the computers / robots the human capabilities for data sensing, data understanding, and action taking based on the past and present outcomes in the future. Computer vision is an essential part of the (Industrial) Internet of Things, and brain human interfaces. Complex human activities are recognized and monitored in multimedia streams using machine learning, artificial intelligence, and computer vision. There are a few well-established methods for prediction and analysis such as supervised learning, unsupervised learning, and semi-supervised learning. These methods use various machine learning algorithms such as Support Vector Machines (SVM), K-Nearest Neighbours (KNN), Naive Bayes, Decision Trees, Logistic Regression, etc.

State of the art

In these two critical areas of recent research, computer vision uses image and pattern mappings in order to find solutions based on requirements /human needs and the connections between them. Computer vision can help automate the monitoring, inspection, and surveillance duties by seeing an image as an array of pixels. Machine learning like deep learning is a subset of artificial intelligence, the automatic analysis or annotation even in real-time in videos is the outcome of computer vision and machine learning. Each of the components that are involved in delivering an image to the computer plays an essential role in the machine vision process. Thus, illumination is usually crucial to bring out the objects we are interested in. Triggered frame grabbers and cameras with full-frame sensors or higher are essential equipment for the image to be captured at the right time with the right exposure. Lenses are especially important for acquiring a sharp, accurate, and aberration-free image. Still, none of these components can "see" or extract the information we are interested in from the image. This is similar to human vision. Without our eyes, we cannot see, yet, even with eyes, we can't see anything without our brain [1]. The eye is only a sensor that delivers data to the brain for interpretation / analysis. Extend this analogy a little further, even if we are myopic, we can still see but a little bad. It is clear that the processing of the images delivered to the computer system by the image sensors is truly the core of machine vision. Before we can dive into this study, we need to investigate the fundamental data structures that are involved in ML-computer-vision applications. We will look at the data structures for 2D images, regions, and subpixel-precise contours. An image is the rudimentary data structure in computer vision since this is the data that an image acquisition device like a camera or a frame-grabber typically delivers to the computer’s memory, via a higher speed bus.

A pixel may be thought of as a collection of the light radiation/energy that falls on the sensor element throughout the exposure time, integrated over the light's spectral distribution and the sensor's spectral response. Depending on the camera, the sensor's spectral response will often include the full visible spectrum as well as a portion of the short IR spectrum. In this situation, the camera will return one energy sample per pixel, resulting in a single-channel grey value picture. RGB cameras, on the other hand, return three samples per pixel, resulting in a three-channel picture. These will be the two most common (also the most important) types of sensors used in applications that require computer vision. As a result, in order to cover all potential applications, a picture may be thought of as a collection of an indefinite number of channels. An image channel may be thought of as a two-dimensional array of floats/integers. In the computer science / data science domain, this is also the data structure used to represent images.

One of the challenges in machine vision is to find areas in a picture that have specific qualities, such as by using a threshold operation. As a result, at the very least, we require a representation for a random subset of the pixels in an image. Also, for the segmentation algorithm, areas must be able to extend beyond the picture borders in order to prevent artifacts. A binary picture has a grey value of 0 for points that are not in the region and 1 for points that are in the region. Like an addiction, we may represent several items in the image as label images, in which the grey value encodes the region to which the point belongs. A label of 0 is often used to indicate locations which are not included in any area and values greater than 0 are used to represent the various regions. It is therefore straightforward to talk about areas in the image from an abstract standpoint. However, it is not immediately evident how to effectively portray regions. The usage of binary pictures to represent areas is instantly suggested by this concept. A binary picture has a grey value of 0 for points that are not in the region and 1 (or any other number other than 0) for points that are in the region. As an addition, we may represent several items in the image as label images, in which the grey value encodes the region to which the point belongs. A label of 0 is often used to indicate locations that are not included in any area, and values greater than 0 are used to represent the various regions. The representation of areas as binary pictures has one clear disadvantage: it must hold (often a large number of) points not included in the region. Unfortunately, the representation is inefficient: we must store at least one bit for each point in the image. Because bytes are significantly quicker to access than bits, the representation frequently employs one byte per point. This representation is similarly inefficient in terms of run time: determining which points will be included in the area. The usage of binary pictures to represent areas is instantly suggested by this concept. A binary picture has a grey value of 0 for points that aren't in the region and 1 (or any other number excluding 0) for points that are in the region. As an addition, we may represent several items in the image as label images, in which the grey value encodes the region whereby the point belongs. A label of 0 is often used to indicate locations which are not included in any area, and a value greater than 0 is being used to represent the various regions. The representation of areas as binary pictures has one clear disadvantage: it must hold (often a large number of) points which is not included in the location. The representation is inefficient: we must store at least one bit for each point in the image [2]. Because bytes are significantly quicker to access than bits, the representation frequently employs one byte per point. This approach is also inefficient in terms of execution time: to decide which points are included in the specific location, we must conduct a test for each point in the binary picture. Furthermore, storing areas that extend to negatives coordinates as binary pictures is hard, which leads to complicated methods. Because numerous areas are represented as label pictures, overlapping regions can't be represented, which causes complications when morphological operations are done on the regions. As a result, an efficient representation that simply saves the points contained in an area would be quite valuable.

Dataset

The playground prediction competition's dog breed identification dataset available from Kaggle, the most used and known website by the data scientists community, was utilized for this study. The organizers of the competition provided a training set and a test set of images of dogs. Each image has a filename that is its unique id, all labels data comes into a comma-separated values (CSV) file. The collection comprises around 120 distinct dog breeds, each with at least 150 photos [3]. The beneficiaries of the competition have a dataset with a total of 20 thousand photos. The main goal is to create a good enough classifier capable of determining a dog's breed from an image in order to improve services that use machine learning and deep learning algorithms like Google Photos mobile application, for example, in scanning something into an image and displaying pieces of information regarding that something.

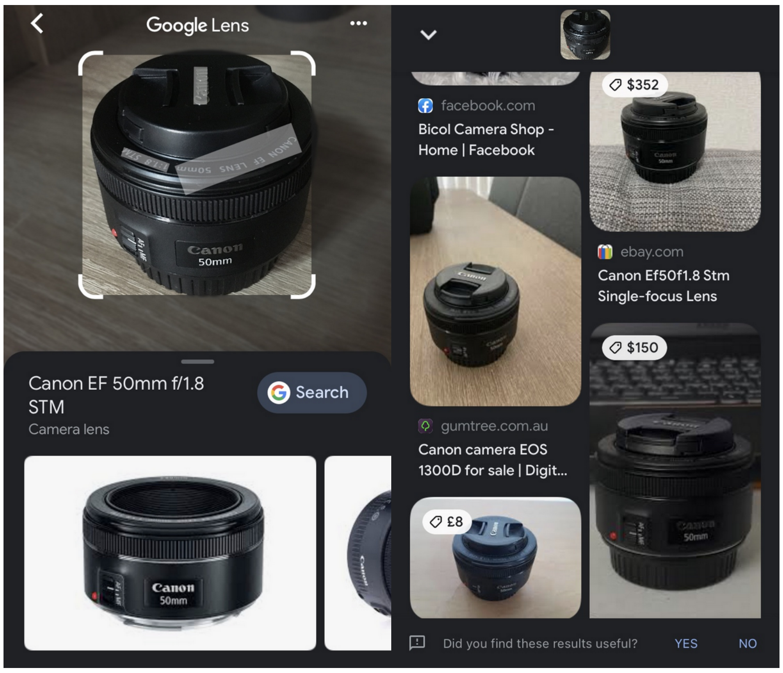

Google Lens
Canon EF 50mm f/1.8 STM
Camera lens
Search
facebook.com
Bicol Camera Shop - Home | Facebook
$352
ebay.com
Canon Ef50f1.8 Stm Single-focus Lens
$150
gumtree.com.au
Canon camera EOS 1300D for sale | Digit...
£8
Did you find these results useful?
YES
NO

Figure 1: Camera lens scanned in Google Photos app for desire to purchase one

The tool used in this project to work with “almost everything” and the data specifically for pre-processing was Python. Python is an open-source programming platform that is mainly used for statistical computing, machine learning, AI, and in almost every domain which was first designed by Guido van Rossum in 1991. Pypi or "Python Package Index" is the official third-party software repository for Python, is the place where users can find tools and libraries matching all they need, from microcontrollers to data science/analytics, web, and so on. For example, SciKit Learn library has a huge arsenal of statistical techniques such as linear and nonlinear modeling, classification, clustering, and many more, deep and machine learning plus artificial intelligence solutions along with highly extensible graphical techniques such as seaborn or matplotlib. Once the dataset was obtained, being an image dataset the most important or rather the most important step before applying any sort of algorithms, was to perform pre-processing on that data, so as to convert all the information from a human readable format into a machine readable format.

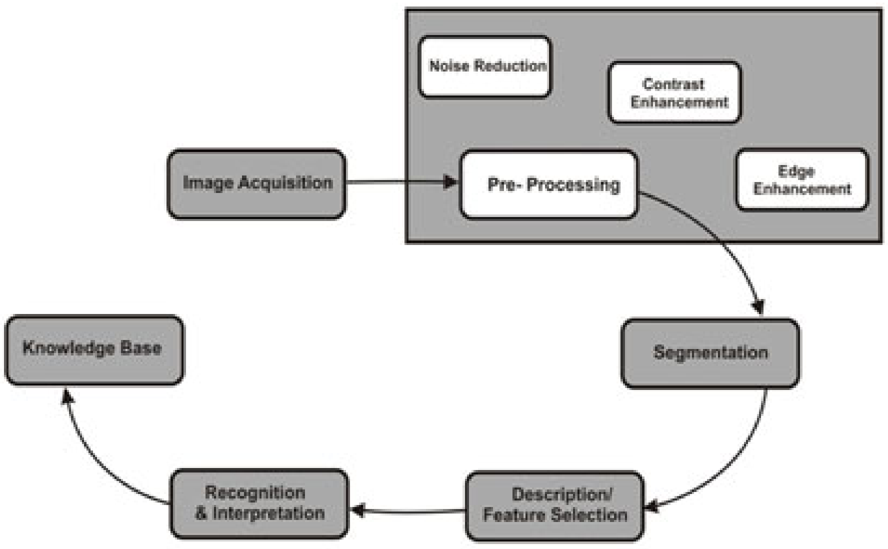


Figure 2: Digital image processing steps in ML/DL [4]

After finishing the pre-processing on the data, the images required to be saved in a format acceptable for the researcher to train a model on. The greyscale pictures, which we're now in a two-dimensional matrix or tensors / numerical representations, were then saved into a single dimension array or vector, essentially a flat-file with 784 characteristics per image, the result of transforming the 28*28 grid into a single vector. Once the images have been properly transformed into vectors with characteristics, the researcher puts the data into a CSV file, which can now be immediately accessed by the system and is in a machine-readable format. We repeated the process for all 120 dog breeds, which was time-consuming but resulted in the creation of a folder containing CSV files for all 120 dog breeds. Instead of dialing all 120 files every time the analyst wanted to apply an algorithm, I decided to merge all 120 CSV files into one single CSV file, adding id and labels to each breed to make it easier to classify, which saves the researcher a lot of time instead of having to call 120 separate files at executable. An intriguing study translates depth pictures to RGB images using a color map and then employs a deep convolutional neural network pre-trained on color images. This method's expansion increases recognition performance even further by integrating a multi-modal technique to learn associated features from the pre-trained model. 3D convolutional neural networks are becoming more popular as faster GPUs and specialised Nvidia CUDA frameworks for deep learning become available [5].

Google Colab was the SaaS platform used for building this application for the simple reason that they offer for processing Nvidia GPUs and Google “in-house” TPUs facilitating very fast processing of numerical calculations, ideal in image processing and machine learning. Having an GUI similar to Anaconda Jupyter Notebook, the media storage used for dataset and saving and loading models was Google Drive.

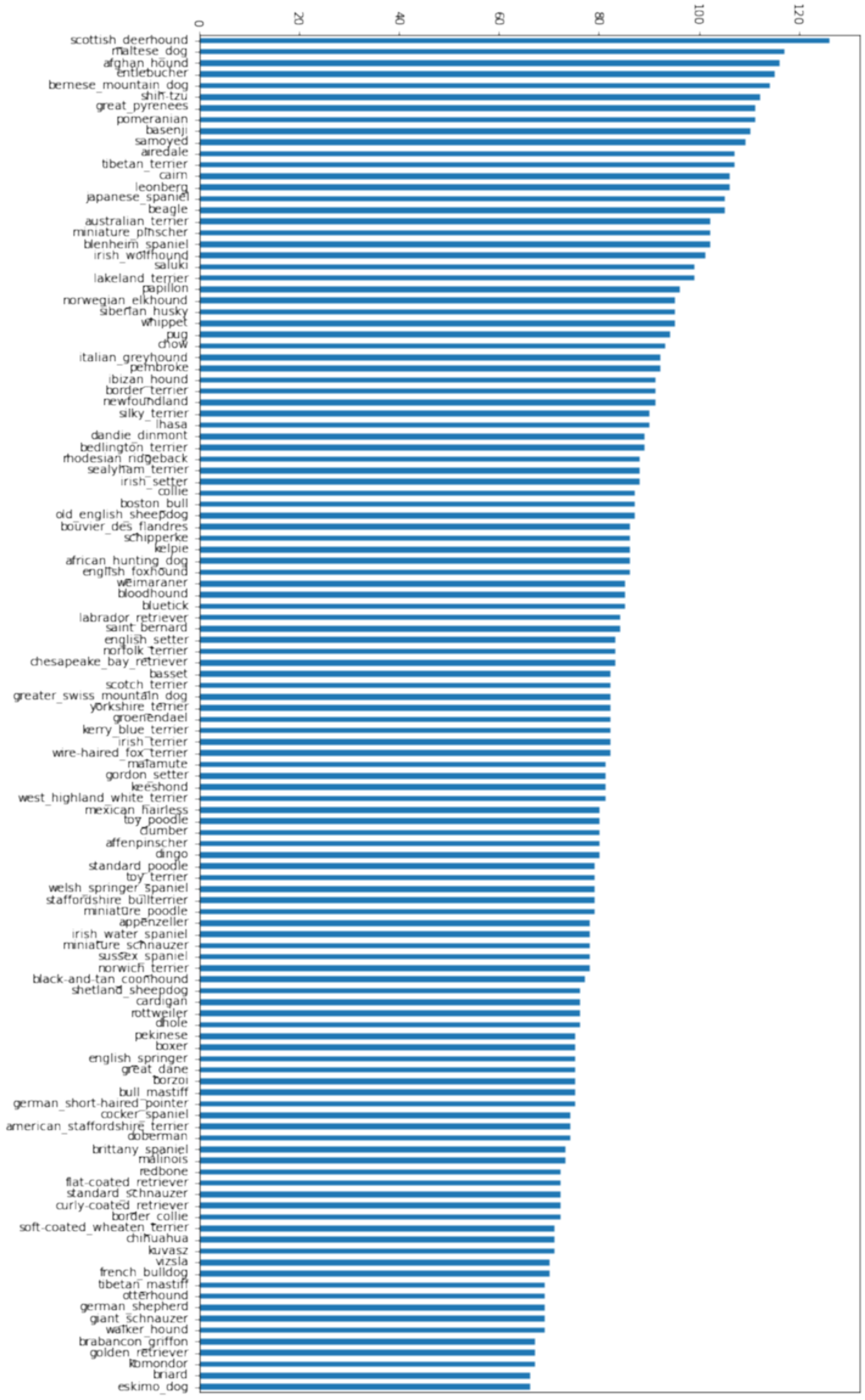


Figure 3: Count of distinct breeds presents in the dataset

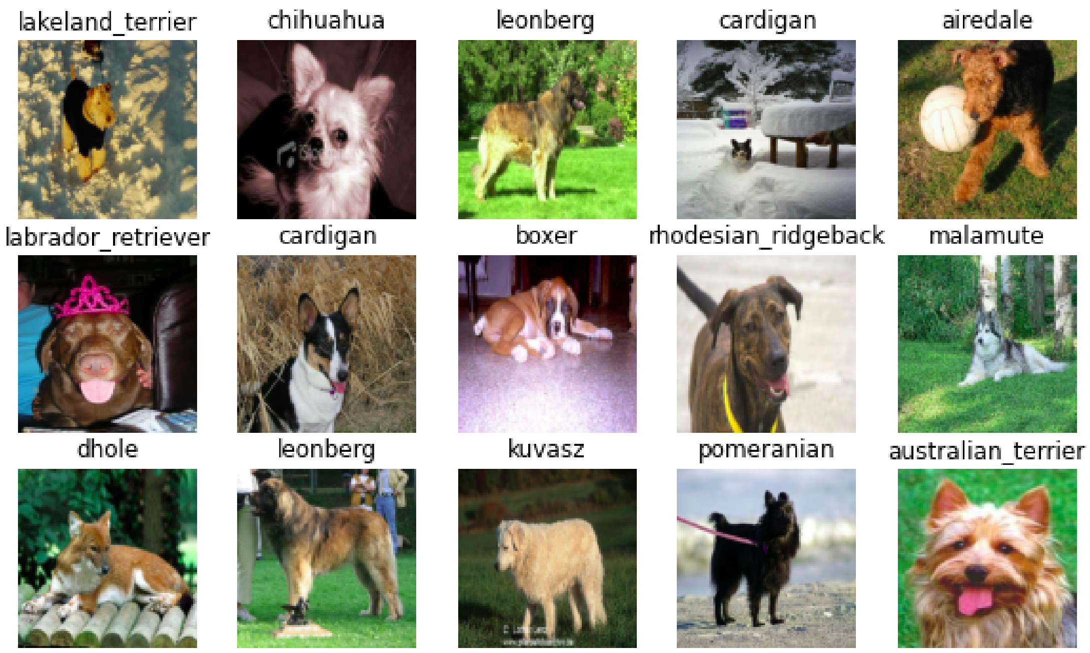
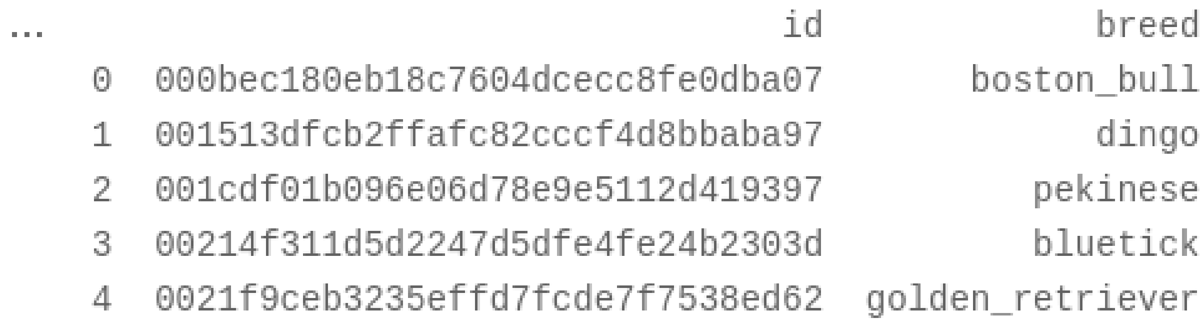


Figure 4: Viewing 15 samples of the data from the training batch



```
...                                id            breed
    0  000bec180eb18c7604dcecc8fe0dba07      boston_bull
    1  001513dfcb2ffafc82cccf4d8bbaba97            dingo
    2  001cdf01b096e06d78e9e5112d419397         pekinese
    3  00214f311d5d2247d5dfe4fe24b2303d         bluetick
    4  0021f9ceb3235effd7fcde7f7538ed62  golden_retriever
```

Figure 5: Image label and corresponding breed

Feature Extraction

Finding structures in computer images data is the focus of high-level feature extraction. One way to automatically recognize faces, for example, is to extract the component characteristics. This necessitates the evacuation of significant face features such as the eyes, ears, and nose for a human example, the procedure is not very different from recognizing an animal. To locate them, we can utilize their shape, the white area of the eyes is ellipsoidal, the mouth, as well as the eyebrows, might seem like two lines. Shape extraction entails determining their location, orientation, and size. This feature extraction method is analogous to how humans see the world, discuss fundamental geometric forms such as triangles, circles, and squares. More complicated images can be broken down into a system of simple forms. The arrangement of forms may help analysis in various applications. In the case of human face analysis, expected to detect the eyes above and on each side of the nose, and the mouth below the nose. In feature extraction, we often aim for invariance qualities so that the extraction procedure does not vary depending on the circumstances that are chosen or defined. That is, whatever the value of any parameter that might impact the appearance of a shape, approaches should find shapes reliably and robustly. We want resilience to variations in illumination level as a basic invariant, we strive to locate a form whether it will be light or dark. In general, as far as there is a contrast between a shape and its surrounding, the shape may be considered to exist and identified. Obviously, any computer vision system will fail in low-light settings you can't see anything when it's absolutely black. The second most critical characteristic after lighting is position: we want to find a form wherever it appears. Then, we frequently want to identify a form that is not rotated, assuming that the object or camera has an unknown orientation, this is known as rotation or orientation invariance. Then, we may try to determine the item at whatever size it seems to be, which could be due to physical change or how close the object is to the camera. This necessitates size or scale independence. These are the primary invariance features that we will look for in our form extraction algorithms. Nature, on the other hand, has a habit of rolling things underneath our feet because there is always noise in images because of low illumination. Also because we're talking about shapes, keep in mind that there may be more than one in the image. When one is placed on top of the other, it occludes, or hides, the other, so not all of the shape of one object is seen.

However, before we can create picture analysis algorithms, we must first extract the forms. Extraction is more difficult than detection because extraction requires us to obtain a representation of a form, such as its location and size, just as detection only requires us to know that a shape exists inside an image. This can be successfully completed by considering the intensity values or by correlating the pixels to a predefined pattern. If the brightness of the shape is understood, the pixels that compose the shape may be retrieved by categorizing the pixels based on a specified intensity threshold in a first way [6]. Alternately, if the background picture is known, the pixels that form the shape of an item placed on the backdrop can be removed. Template matching is a model-based method that finds the best correlation between one known model and the pixels in an image to extract the form. There are several methods for calculating the correlation between the template and the picture. Correlation can be done by taking into account either the image or frequency domains. In addition, the template may be defined by considering luminance or a binary form. The Hough transform offers an efficient template matching mechanism for binary templates. This method can extract basic shapes like linear and polynomial forms but also arbitrary shapes [7]. In almost any case, the difficulty of such a method can be decreased by taking into account the forms' invariant properties.

Thresholding is a resulted extraction technique in which the pictures may be seen as a consequence of attempting to isolate the eyes from the surroundings. If the brightness of the shape to be retrieved is understood to be what defines it, then thresholding a picture at that intensity level should discover the shape. The observed brightness of the target form is obviously sensitive to variations in lighting, if the image illumination changes, so will the observed brightness of the target shape [8]. Any thresholding strategy will fail except if the threshold level is configured to respond to changes in brightness level. Its appeal stems from its simplicity, thresholding does not need a lot of computing work. If the light intensity varies linearly, histogram balancing will provide a picture that does not change. Unfortunately, the outcome of histogram balancing is subject to noise, shadows, and different lighting, noise can have a significant impact on the final image, rendering a thresholding strategy. Any thresholding strategy will fail except if the threshold level is configured to respond to changes in brightness level. Its appeal stems from its simplicity, thresholding does not need a lot of computing work. If the light intensity varies linearly, histogram balancing will provide a picture that does not change. Unfortunately, the outcome of histogram balancing is subject to noise, shadows, and different lighting; noise can have a significant impact on the resultant image, rendering a thresholding strategy ineffective. Thresholding following intensity normalization was far less susceptible to noise but since noise is expanded with the original image and has a significant effect on the stretching process. However, it is still vulnerable to shadows and changing lighting. Similarly, it can only be used in settings where the lighting can be precisely regulated. This criterion applies to every application that employs basic thresholding. If the overall illumination level cannot be regulated, edge magnitude data can be thresholded since it is indifferent to the overall brightness level due to the discretization process [8]. But, edge data is rarely continuous, and gaps in the recognized perimeter of a form might occur. Another significant challenge, which also relates to thresholding raw brightness data, is that there is frequently more than one form. When the patterns are stacked on top of each other, one disrupts the other, and the shapes must be separated. Before thresholding, another method is to remove a picture from a known backdrop. This implies that the backdrop is accurately understood, otherwise, much more information than just the target feature would plainly emerge in the final image, the subtraction will be impossible if there is noise on either image, particularly both. There is no implicit form description in this technique, but if the thresholding procedure is adequate, it is straightforward to estimate fundamental shape characteristics such as location.

The intensity histogram depicts how distinct brightness levels are taken in an image; the range of brightness levels is used to calculate picture contrast. The histogram graphs the number of pixels with a specific brightness level vs the brightness level. The brightness of 8-bit pixels spans from 0-black to 255-white. The histogram demonstrates that not all of the grey levels are utilized and that the lowest and greatest intensity levels are near together, indicating moderate contrast. The histogram comprises the dark areas of the picture, such as the hair and the iris of the eye, in a zone between 100 and 120 brightness values [10]. The brighter elements are mostly about the skin. If the image was black or darker in general, the histogram would be skewed toward black. If the image was brighter but less contrasty, the histogram would be narrower and more concentrated towards the whiter brightness values. This histogram demonstrates that we did not use all of the possible grey levels. As a result, we may expand the image to employ all of them, and the image will become clearer. This is a mainly aesthetic enhancement to improve the image's look. Many fundamental image processing processes, as detailed in this chapter, are focused on improving the look, particularly in light of future processing. If the perfect histogram is known, the histogram can also tell whether or not there is noise in the image. We may wish to eliminate this noise not just to improve the image's aesthetic, but also to make the work of (and present the subject better for) following feature extraction approaches easier.

Detecting contrast, defined as a difference in intensity, helps highlight the borders of features within an image because this is where image contrast occurs. Because the intensity of an item varies with its surroundings, this is how eyesight perceives its perimeter. Essentially, an object's border is a step-change in intensity levels. The edge is at the step-change location. We may use first-order differentiation to identify the edge position because it highlights change; first-order differentiation offers no response when applied to signals which do not modify. The initial edge detection operators investigated here are group operators whose output approximates the outcome of first-order differentiation. Differentiating neighboring dots might reflect a difference in intensity. Differentiating horizontally adjacent points detects vertical variations in intensity and is commonly referred to as a horizontal edge detector due to its operation. Because the difference is zero, a horizontal operator will not detect horizontal variations in intensity [11].

A region often describes components or interior points that are bounded by a border or perimeter, which is sometimes referred to as the area's contour. The shape of a contour is often referred to as its form. A point is said to be on the border / contour if it is part of the area and has at minimum one pixel in its vicinity that is not. The border is commonly identified via contour following starting at one place on the contour and work our way around it, either clockwise or anti-clockwise, looking for the next contour point. To identify the interior and border points of an area, we must analyze the nearby connections between pixels. Connectivity rules are used to explain these relationships. There are two commonly used definitions of connectivity: 4-way or 4-neighborhood, in which only near neighbors are considered for connection, or 8-way or 8-neighborhood, in which all eight pixels around a specified pixel are considered for connection. Both forms of connection can be used to define a border or an area, but they're always complementary. That example, if the border pixels are connected in four directions, the region pixels will be connected in eight directions, and vice versa. We can, imaginary, see that given a diagonal boundary, 4-way connection produces a staircase boundary, but 8-way connectivity produces a diagonal line created from either the points at the neighborhood's corners. This is in contrast to the pixels on the perimeter.

Fourier-type descriptors enable us to shape descriptions using the power of Fourier theory. The basic concept is to define a contour with a collection of integers that indicate the frequency components of the entire form. We may use frequency analysis to choose a limited group of values, the Fourier coefficients, to characterize a form instead of any noise (for example, the noise affecting the spatial position of the boundary pixels). The standard procedure for obtaining a Fourier representation of a curve consists of two major parts. First, we must generate a representation of a curve; next, we must extend it using the Fourier theory. We can get varied tastes by mixing multiple curve representations and Fourier expansions. In this section, we will look into Fourier operators of angular and complicated contour representations. Fourier expansions, on the other hand, can be created for different curve representations. Aside from the specification of the curve, the decision of Fourier expansion determines the growth and features of the description. If we think that a curve's trace forms a periodic function, we may utilize a Fourier series expression. As a result, we could create a representation based on the Fourier transform. We might utilize different Fourier integral formulations in this scenario. In pattern recognition, this is the most popular approach to describe forms. It is vital to note that, whereas an image's curve is made up of discrete dots / pixels, Fourier descriptors are designed for continuous curves. This is useful since it results in a discrete collection of Fourier descriptors [12]. Furthermore, keep in mind that the pixels in the image are really the sampled points of a continuous curve in the image. The approach, on the other hand, corresponds to the definition of the integral of a continuous curve. In actuality, we have a sampled version rather than a continuous curve, numerical integration is used to estimate the expansion.

Convolutional Neural Networks CPU vs GPU Performance

For the implementation of the proposed CNN model used for my application was Google Colab is a free cloud service offered by Google for Machine Learning and AI that includes free GPU processors mostly from Nvidia, pre-installed libraries, is built on Jupyter Notebook, supports bash commands, and stores notebooks in Google Drive. TensorFlow 2.0 and Keras, which are focused on deep machine learning, are the primary libraries utilized. Keras is the most popular framework in the domain due to its simplicity. Keras has been "embedded" into TensorFlow 2.0 through the module tf.keras. CNN additionally makes use of the following tools: OpenCV, Scikit-Learn, NumPy Array, Pandas, and Matplotlib. The major goal is to analyze and compare CNN times on CPU and GPU architectures. Aside from observing the performance of CNN MobileNetV2 in various architectures. Pooling has the function of simplifying the preceding layer's information, in this example, the map of attributes. As with convolution, a region unit is established to pass over the full output of the preceding layer, often using a 2x2 matrix. The unit is in charge of condensing the information in that region into a single value. To select this aspect, you must first decide how you want the summary to be done. The most often used approach is MaxPooling, which retrieves the unit's greatest number and transmits this value to the output [9]. This data summarization reduces the number of weights to be learned by the Neural Network while also avoiding overfitting. Flattening takes the matrix formed in the Pooling phase as input and performs a transformation in the image matrix, altering its format to an array, that is, it turns the matrix to an attributes vector [9]. At this point, the Neural Network gets the data from the Flattening layer and sends the characteristics vector as input for Neural Network training, which is a computer model based on the human central nervous system. In this method, you will be able to spot patterns in a big amount of data and categorize them. Tensors (n-dimensional arrays) are used to model all data in TensorFlow, with the elements having one of a small number of basic types, such as int32, float32, or string (where string may represent any binary data). Tensors naturally represent the inputs and outputs of basic mathematical operations in many machine learning algorithms: for example, a matrix multiplication takes two 2-D tensors and creates a 2-D tensor; and a batch 2-D convolution takes two 2-D tensors and produces a 2-D tensor. At the most basic level, all TensorFlow tensors are dense. To represent sparse data, TensorFlow provides two options: encode the data into variable length of string elements of a dense tensor, or use a tuple of dense tensors, e.g., an n-D sparse tensor with m non-zero elements can be represented in coordinate-list format as a m n matrix of coordinates and tensor's form can change in one or more dimensions, allowing sparse tensors with varying amounts of elements to be represented [13].

To perform the tests, before predicting what dog breed is the present in the image of a single dog or multiple dogs, it is necessary to load an image into RAM memory (in case of a CPU speed test) and respectively, into GPU memory (in case of speed test using graphical processor). The graphical processor used is Nvidia Tesla P100.n terms of computational speed, the version designed for the Tesla P100 GPU demonstrated benefits, cutting execution time by up to 10.7 times when compared to CPU execution. Also, change the Artificial Neural Network's algorithm for the possibility using the Google Cloud TPU execution environment, which contains TPUs with greater archiving than the free TPU offered by Google Colab, providing higher performance than a GPU in terms of processing / execution speed, and examine the impact of other hyperparameters, such as learning rate, because running scripts on a TPU architecture needs some scripts modifications. In both cases our images are resized to a standard size of 224 pixels, grouped in a batch size of 32, facilitating the most efficient learning due to batch processing for the training and validation scenarios until the 10th epoch, further, the model no longer has an efficient learning rate.

Final results and conclusions

I expected to perform a performance measurements in order to examine the model performance I developed. Confusion matrix, accuracy with log loss were among the performance metrics I obtained. The confusion matrix demonstrate how effectively the model was trained and how well it anticipated the test data set. Regarding individual photos, it compared the genuine dog breed to the anticipated dog breed. I acquired the following confusion matrix for the training and testing data sets for my optimal model.

As can be seen in Figure 6 (LEFT), the model is training successfully since there is a distinct diagonal line representing the genuine breed that matches the projected breed. Then, looking at the testing confusion matrix in Figure 6 (RIGHT), I saw that the diagonal line, which indicates the correct predictions, is not as common. Instead, I saw that there are a few dots dispersed across the remainder of the confusion matrix that reflect inaccurate predictions. The testing log loss and accuracy were also provided with the help of verbose. These measures are a good representation of the true rates of log loss and accuracy because they are being passed through cross-validation, also provide a good point of view of how the system is performing. The optimal model produced for log loss a minimum of "0.0306" and a maximum value of "1.3556". Our optimal / ideal model has a minimum accuracy of "0.6706" - 67.06% and a maximum accuracy of "0.9973" - 99.73%. A future improvement of this model is to increase the number of splits utilized. These results were obtained using only five cross-validation splits. An increase to ten splits will allow the model to train more which will potentially increase the accuracy rate of predictions, and certainly, time performance will decrease significantly.

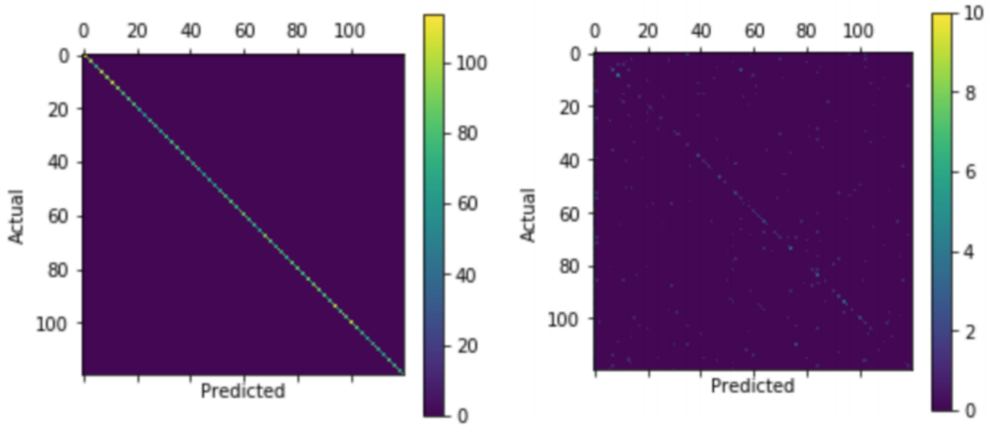


Figure 6: (LEFT) Confusion matrix on the training data-set. The primary diagonal is the evidence of good performance during training scenario.

(RIGHT) Confusion matrix on the test data-set. This shows a clear diagonal pattern, additional classes are also mistakenly predicted. While this appears to go beyond the random noise level.

Attempts to forecast dog breeds were performed using the performance indicators gathered. One method for making predictions is to use the test set, which was divided into at the start of training. I used the test set to pass in an image from the set and see what the algorithm predicted. To figure this out, I looked at the prediction probability and found the largest value in the array. The maximum value index indicates the expected label of the dog breed. I was able to extract the real value of the dog breed in the image since this input image is an image from the original data. By comparing the value returned by the algorithm to the real value, I was able to ascertain whether or not the system properly anticipated that specific image. I was able to retrieve the name of the breed suggested by the machine using the anticipated breed index. Which allowed me to evaluate the dog's input image to what the algorithm predicted it to be.

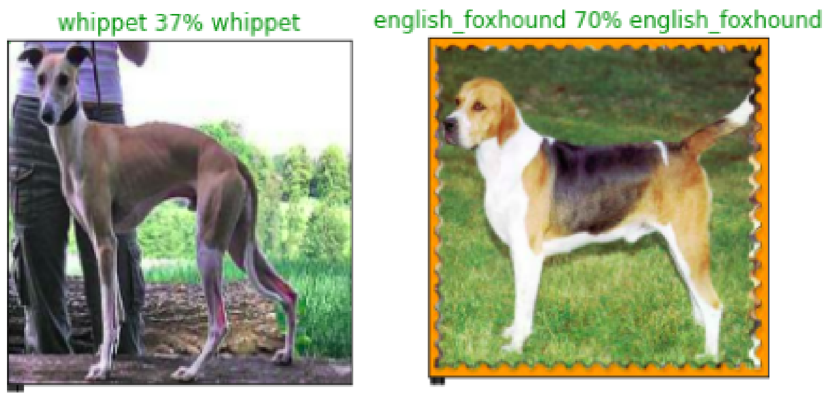


Figure 7: Examples of correct breed prediction for whippet and english foxhound dogs

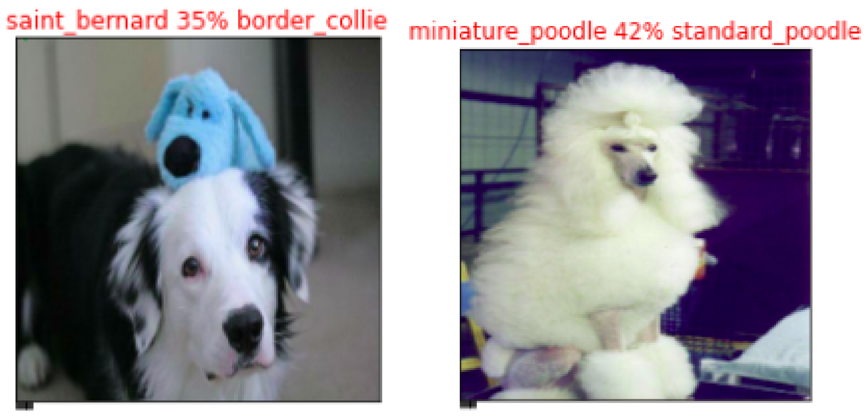

saint_bernard 35% border_collie
miniature_poodle 42% standard_poodle

Figure 8: Examples of wrong predictions

In Figure 8 we can see two samples of wrong predictions by the model in left image we have a border collie dog as true label and the model is confussed by the face similarity with a saint bernard dog having a confidence level of 35%. Actually for the right image the prediction is more interesting due to the striking similarities between the two breeds, with a confidence level of 42% and a predicted label of miniature poodle instead of a standard poodle, our model predicted the wrong value being confused by the very close similarities between the two breeds which shows that no model of machine learning or artificial intelligence can be perfect.

Some other way to predict images is by inputting your own images into the software. For example, I was able to input five images of dogs unknown to the model, images that are not part of the dataset, images captured at a puppy beauty park event. And the question is let's see what the system predicted it to be, because in four out of five images are two dogs from the same breed instead of one for increasing the difficulty of a correct prediction. The system will again output the probabilities for the images provided being each breed in the data set. I was able to evaluate whether or not it was accurately predicted. Unfortunately, it did not properly predict one canine out of the five photographs I entered. The algorithm predicted a vizsla breed (Figure 9), yet both dogs in the photography were labrador breeds, much like the first image on the list. Other races that were successfully predicted include a doberman, pomeranian, appenzeller, and labrador retriever.

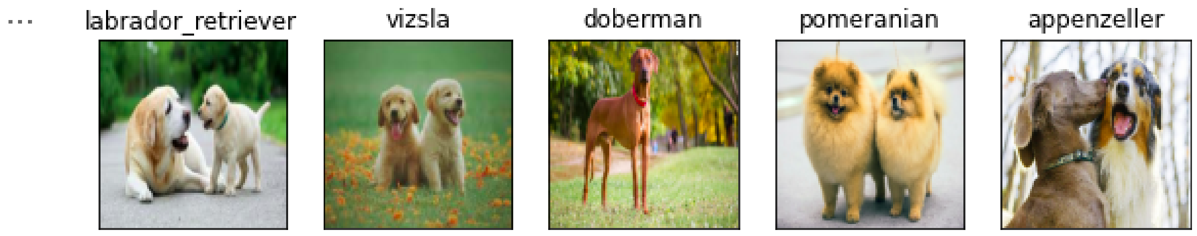


Figure 9: Predictions made on data unknown by model

In conclusion, the Convolutional Neural Network appears to be an effective algorithm for dog breed detection. Newer technology and computing power are produced, and more study work may be completed over the course period for better categorization. By organizing comparable techniques and improved computing methods, the procedure may be made more efficient with deep learning models, and breed classification can be conducted further using deep neural networks.

References

[1] In Steger, C., In Ulrich, M., & In Wiedemann, C. Machine vision algorithms and applications, Pg 20-30, 2018.

[2] Lapray, P.J., Wang, X., Thomas, J.B., and Gouton, P. Multispectral filter arrays: Recent advances and practical Implementation. Sensors, Pg 626–659, 2014.

[3] https://www.kaggle.com/c/dog-breed-identification/data

[4] https://madhavuniversity.edu.in/digital-image-processing.html

[5] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard. Multimodal deep learning for robust RGB-D object recognition. IEEE/RSJ International Conference on, pages 681–687, 2015.

[6] Altman, J. and Reitbock, H. J. P., A Fast Correlation Method for Scale and Translation Invariant Pattern Recognition, IEEE, pp. 46 –58, 1984.

[7] Ballard, D. H., Generalising the Hough Transform to Find Arbitrary Shapes, CVGIP, 13, pp. 112–122, 1981.

[8] M. H. Chowdhury and W. D. Little, "Image thresholding techniques," IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing. Proceedings, pp. 585-589, 1995.

[9] Abiwinanda N., Hanif M., Hesaputra S.T., Handayani A., Mengko T.R. Brain Tumor Classification Using Convolutional Neural Network, Pp 183-189, 2019.

[10] Zebin Sun, Wenquan Feng, Qi Zhao, Lidong Huang, "Brightness preserving image enhancement based on a gradient and intensity histogram," J. Electron. Imag. 24(5) 053006, 10 September 2015.

[11] Mark Nixon and Alberto S. Aguado. Feature Extraction &amp; Image Processing for Computer Vision, Third Edition (3rd. ed.). Academic Press, Inc., USA, 2012.

[12] Van Otterloo, P. J., A Contour-Oriented Approach to Shape Analysis, Prentice Hall International (UK) Ltd, Hemel Hempstead, 1991.

[13] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Zheng, X., (Google Brain), TensorFlow: A system for large-scale machine learning, 2016.