A PROJECT ON

"Fruit Classification and Detection using canvas"

SUBMITTED IN
PARTIAL FULFILLMENT OF THE REQUIREMENT
FOR THE COURSE OF
DIPLOMA IN ADVANCED COMPUTING FROM CDAC

SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY
'Plot no R/2', Market yard road,
Behind hotel Fulera,Gultekdi
Pune – 411037.
MH-INDIA

SUBMITTED BY:

Pratik Devnani

UNDER THE GUIDENCE OF:
Mr. Girish Gaikwad
Faculty Member
Sunbeam Institute of Information Technology, PUNE.

CERTIFICATE

This is to certify that the project work under the title ‘Fruit classification and detection’ is done by Pratik devnani in partial fulfillment of the requirement for award of Diploma in Advanced Computing Course.

Mr. Girish Gaikwad
Project Guide

Mrs . Pradnya Dindorkar
Course Co-Coordinator

Date:

ACKNOWLEDGEMENT

A project usually falls short of its expectation unless aided and guided by the right persons at the right time. We avail this opportunity to express our deep sense of gratitude towards Mr. Nitin Kudale (Center Coordinator, SIIT, Pune) and Mrs. Pradnya Dindorkar (Course Coordinator, SIIT ,Pune) and Project Guide Mr. Girish Gaikwad

We are deeply indebted and grateful to them for their guidance, encouragement and deep concern for our project. Without their critical evaluation and suggestions at every stage of the project, this project could never have reached its present form.

Last but not the least we thank the entire faculty and the staff members of Sunbeam Institute of Information Technology, Pune for their support.

Pratik Devnani

DBDA FEB 2019 Batch,

SIIT Pune

Table of Contents

1 Introduction

The aim of this paper is to propose a new dataset of images containing popular fruits. The dataset was named Fruits-dataset and can be downloaded from the addresses Google colab quickdraw. Currently the set contains 140000 images of 8 fruits and it is constantly updated with images of new fruits as soon as the authors have accesses to them. The reader is encouraged to access the latest version of the dataset from the above indicated addresses.

Having a high-quality dataset is essential for obtaining a good classifier. Most of the existing datasets with images contain both the object and the noisy background. This could lead to cases where changing the background will lead to the incorrect classification of the object.

Google announced that is working on an application named Google Lens which will tell the user many useful information about the object toward which the phone camera is pointing. First step in creating such ap- plication is to correctly identify the objects. The software has been released later in 2017 as a feature of Google Assistant and Google Photos apps. Currently the identification of objects is based on a deep neural network .

Such a network would have numerous applications across multiple do- mains like autonomous navigation, modeling objects, controlling processes or human-robot interactions. The area we are most interested in is creating an autonomous robot that can perform more complex tasks than a regular industrial robot. An example of this is a robot that can perform inspections on the aisles of stores in order to identify out of place items or understocked shelves. Furthermore, this robot could be enhanced to be able to interact with the products so that it can solve the problems on its own. Another area in which this research can provide benefits is autonomous fruit harvesting. While there are several papers on this topic already, from the best of our knowledge, they focus on few species of fruits or vegetables. In this paper we attempt to create a network that can classify a variety of species of fruit, thus making it useful in many more scenarios.

As the start of this project we chose the task of identifying fruits for several reasons. On one side, fruits have certain categories that are hard to differentiate, like the citrus genus, that contains oranges and grapefruits. Thus we want to see how well can an artificial intelligence complete the task of classifying them. Another reason is that fruits are very often found in
stores, so they serve as a good starting point for the previously mentioned project.

In November 2016, Google released an online game titled Quick, Draw! that challenges players to draw a given object in under 20 seconds. However, this is no ordinary game; while the user is drawing, an advanced neural net- work attempts to guess the category of the object, and its predictions evolve as the user adds more and more detail

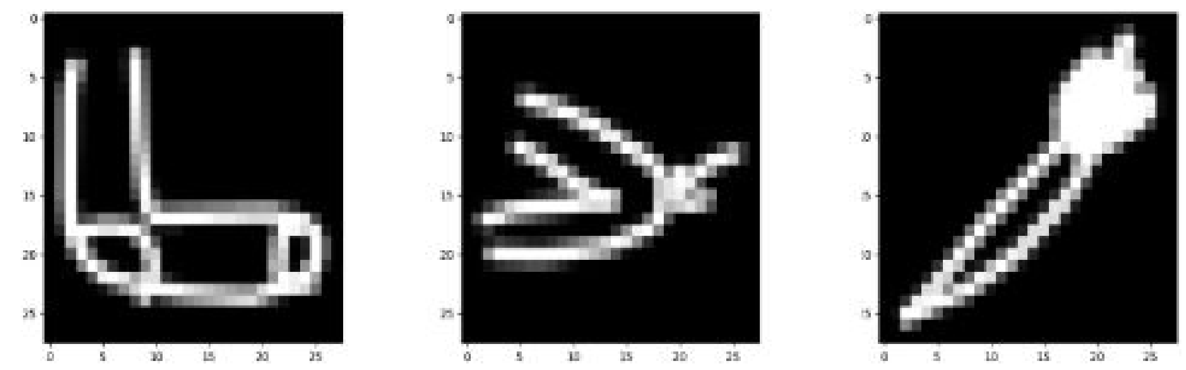
Beyond just the scope of Quick, Draw!, the ability to recognize and classify hand-drawn doodles has important implications for the development of artificial intelligence at large. For example, research in computer vision and pattern recognition, especially in subfields such as Optical Char- acter Recognition (OCR), would benefit greatly from the advent of a robust classifier on high noise datasets.

For the purposes of this project, we choose to focus on classification of the finished doodles in their entirety. While a simpler premise than that of the original game's, this task remains difficult due to the large number of categories (345), wide variation of doodles within even a single category, and confusing similarity between doodles across multiple categories.

Thus, we create a multi-class classifier whose input is a Quick, Draw! doodle and whose output is the predicted category for the depicted object.

2. DATA

Google publicly released a Quick, Draw! dataset con- taining over 50 million images across 345 categories. There are multiple different representations for the images. One dataset represents each drawing as a series of line vectors, and another contains each image in a 28x28 grayscale ma- trix. Because we focus on classification of the entire doodle in this project, we use the latter version of the dataset. We treat each 28x28 pixel image as a 784 dimensional vector.



To test our models, we split the data into three different folds: 70% for training, 15% for validation, and 15% for testing. To reduce computation time and storage of the data, we decided to create a smaller subset of the original dataset by randomly sampling 1% of the drawings from each cate- gory.

As a result, we obtain approximately 350,000 examples for the training set and 75,000 examples each for the valida- tion and testing set. Furthermore, the number of drawings in each category is balanced, so this leaves approximately 1000 examples per category in the training dataset.

3. Deep learning

In the area of image recognition and classification, the most successful results were obtained using artificial neural networks. These networks form the basis for most deep learning models.

Deep learning is a class of machine learning algorithms that use multi-ple layers that contain nonlinear processing units . Each level learns to transform its input data into a slightly more abstract and composite rep-resentation . Deep neural networks have managed to outperform other machine learning algorithms. They also achieved the first superhuman pat- tern recognition in certain domains . This is further reinforced by the fact that deep learning is considered as an important step towards obtaining Strong AI. Secondly, deep neural networks - specifically convolutional neural networks have been proved to obtain great results in the field of image recognition.

In the rest of this section we will briefly describe some models of deep artificial neural networks along with some results for some related problems.

Most modern deep learning models are based on artificial neural networks, specifically, Convolutional Neural Networks (CNN)s, although they can also include propositional formulas or latent variables organized layer-wise in deep generative models such as the nodes in deep belief networks and deep Boltzmann machines

In deep learning, each level learns to transform its input data into a slightly more abstract and composite representation. In an image recognition application, the raw input may be a matrix of pixels; the first representational layer may abstract the pixels and encode edges; the second layer may compose and encode arrangements of edges; the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face. Importantly, a deep learning process can learn which features to optimally place in which level *on its own*. (Of course, this does not completely obviate the need for hand-tuning; for example, varying numbers of layers and layer sizes can provide different degrees of abstraction.)

The "deep" in "deep learning" refers to the number of layers through which the data is transformed. More precisely, deep learning systems have a substantial *credit assignment path* (CAP) depth. The CAP is the chain of transformations from input to output. CAPs describe potentially causal connections between input and output. For a feedforward neural network, the depth of the CAPs is that of the network and is the number of hidden layers plus one (as the output layer is also parameterized). For recurrent neural networks, in which a signal may propagate through a layer more than once, the CAP depth is potentially unlimited.No universally agreed upon threshold of depth divides shallow learning from deep learning, but most researchers agree that deep learning involves CAP depth > 2. CAP of depth 2 has been shown to be a universal approximator in the sense that it can emulate any function.Beyond that more layers do not add to the function approximator ability of the network. Deep models (CAP > 2) are able to extract better features than shallow models and hence, extra layers help in learning features.

Deep learning architectures are often constructed with a greedy layer-by-layer method. Deep learning helps to disentangle these abstractions and pick out which features improve performance.

For supervised learning tasks, deep learning methods obviate feature engineering, by translating the data into compact intermediate representations akin to principal components, and derive layered structures that remove redundancy in representation.

Deep learning algorithms can be applied to unsupervised learning tasks. This is an important benefit because unlabeled data are more abundant than labeled data. Examples of deep structures that can be trained in an unsupervised manner are neural history compressors and deep belief networks

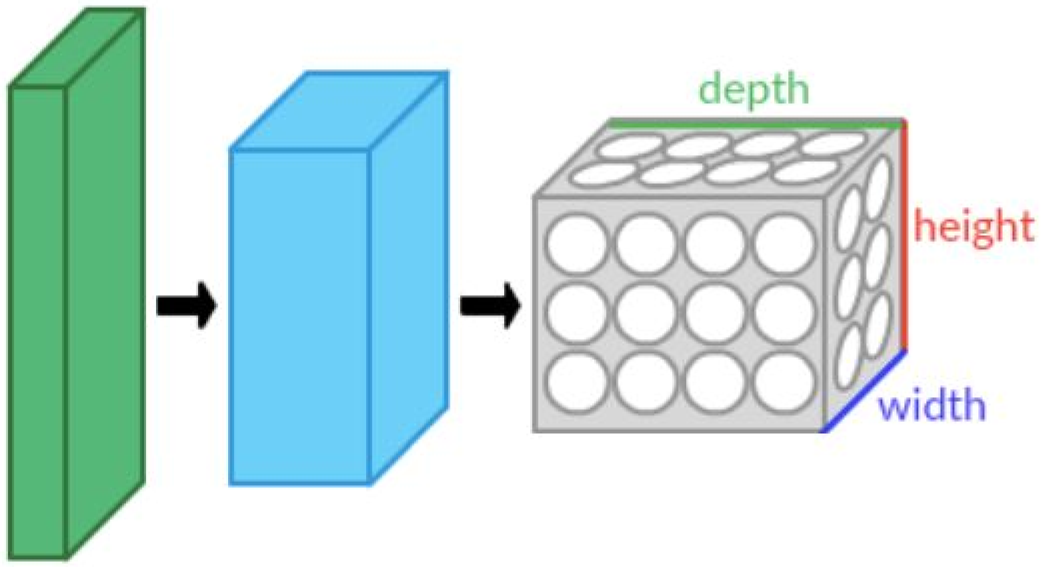
4. Convolutional neural networks

Convolutional neural networks (CNN) are part of the deep learning models. Such a network can be composed of convolutional layers, pooling layers, ReLU layers, fully connected layers and loss layers . In a typical CNN architecture, each convolutional layer is followed by a Rectified Linear Unit (ReLU) layer, then a Pooling layer then one or more convolutional layer and finally one or more fully connected layer. A characteristic that sets apart the CNN from a regular neural network is taking into account the structure of the images while processing them. Note that a regular neural network converts the input in a one dimensional array which makes the trained classifier less sensitive to positional changes.

Among the best results obtained on the MNIST dataset is done by using multi-column deep neural networks. As described in paper, they use multiple maps per layer with many layers of non-linear neurons.

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that *convolve* with a multiplication or other dot product. The activation function is commonly a RELU layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution. The final convolution, in turn, often involves backpropagation in order to more accurately weight the end product.

Though the layers are colloquially referred to as convolutions, this is only by convention. Mathematically, it is technically a *sliding dot product* or cross-correlation. This has significance for the indices in the matrix, in that it affects how weight is determined at a specific index point.



4.1 CONVOLUTION LAYER

Convolutional layers are named after the convolution operation. In mathe- matics convolution is an operation on two functions that produces a third function that is the modified (convoluted) version of one of the original functions. The resulting function gives in integral of the pointwise multi- plication of the two functions as a function of the amount that one of the original functions is translated .

A convolutional layer consists of groups of neurons that make up kernels. The kernels have a small size but they always have the same depth as the input. The neurons from a kernel are connected to a small region of the input, called the receptive field, because it is highly inefficient to link all neurons to all previous outputs in the case of inputs of high dimensions
such as images. For example, a 100 x 100 image has 10000 pixels and if the first layer has 100 neurons, it would result in 1000000 parameters. Instead of each neuron having weights for the full dimension of the input, a neuron holds weights for the dimension of the kernel input.

The kernels slide across the width and height of the input, extract high level features and produce a 2 dimensional activation map. The stride at which a kernel slides is given as a parameter. The output of a convolutional layer is made by stacking the resulted activation maps which in turned is used to define the input of the next layer.

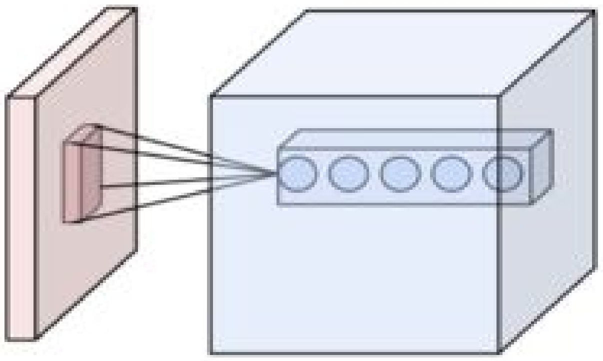
Applying a convolutional layer over an image of size 32 X 32 results in an activation map of size 28 X 28. If we apply more convolutional layers, the size will be further reduced, and, as a result the image size is drastically reduced which produces loss of information and the vanishing gradient problem.

The strides causes a kernel to skip over pixels in an image and not include them in the output. The strides determines how a convolution operation works with a kernel when a larger image and more complex kernel are used. As a kernel is sliding the input, it is using the strides parameter to determine how many positions to skip.

ReLU layer, or Rectified Linear Units layer, applies the activation function max(0, x). It does not reduce the size of the network, but it increases its nonlinear properties.

When dealing with high-dimensional inputs such as images, it is impractical to connect neurons to all neurons in the previous volume because such a network architecture does not take the spatial structure of the data into account. Convolutional networks exploit spatially local correlation by enforcing a sparse local connectivity pattern between neurons of adjacent layers: each neuron is connected to only a small region of the input volume.

The extent of this connectivity is a hyperparameter called the receptive field of the neuron. The connections are local in space (along width and height), but always extend along the entire depth of the input volume. Such an architecture ensures that the learnt filters produce the strongest response to a spatially local input pattern.



4.2 Pooling layers

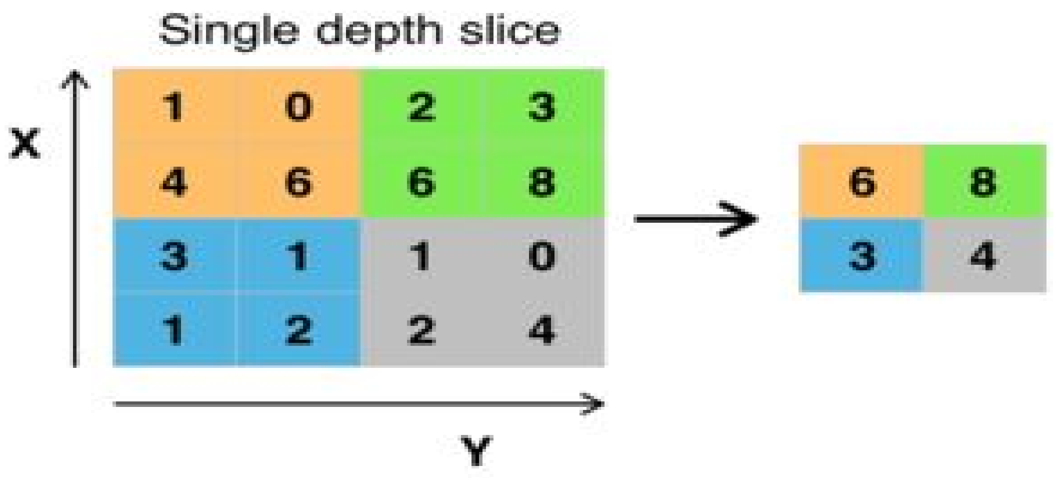
Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling among which *max pooling* is the most common. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum.

Intuitively, the exact location of a feature is less important than its rough location relative to other features. This is the idea behind the use of pooling in convolutional neural networks. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters, memory footprint and amount of computation in the network, and hence to also control overfitting. It is common to periodically insert a pooling layer between successive convolutional layers in a CNN architecture.The pooling operation provides another form of translation invariance.

The pooling layer operates independently on every depth slice of the input and resizes it spatially. The most common form is a pooling layer with filters of size 2×2 applied with a stride of 2 downsamples at every depth slice in the input by 2 along both width and height, discarding 75% of the activations. In this case, every max operation is over 4 numbers. The depth dimension remains unchanged**.**

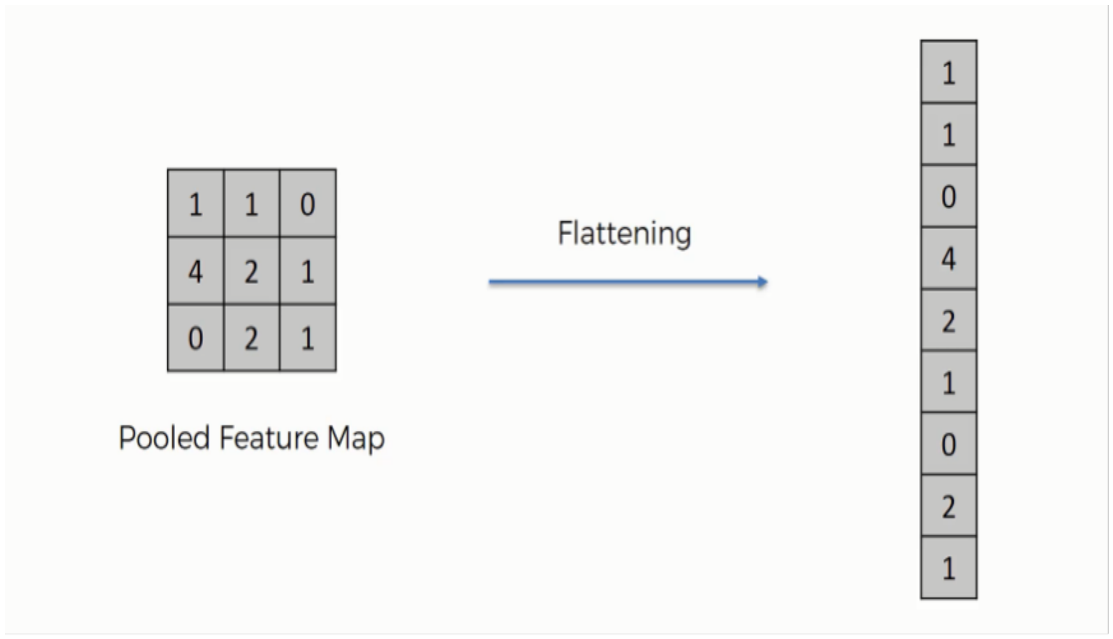
In addition to max pooling, pooling units can use other functions, such as average pooling or ℓ_2-norm pooling. Average pooling was often used historically but has recently fallen out of favor compared to max pooling, which performs better in practice.

Due to the aggressive reduction in the size of the representation,there is a recent trend towards using smaller filters or discarding pooling layers altogether.



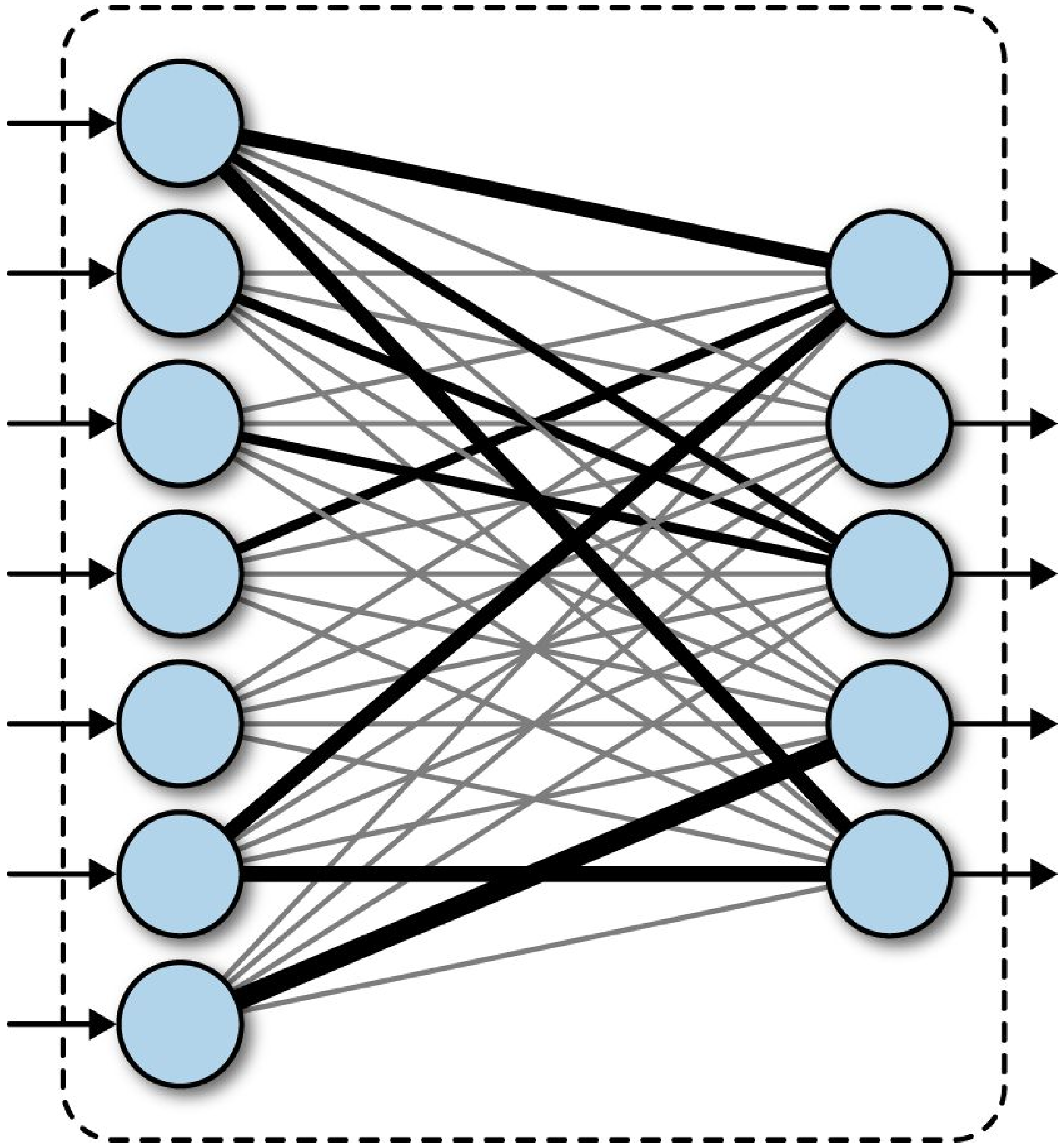
4.3 Flatten

Flattening is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a *fully-connected* layer. In other words, we put all the pixel data in one line and make connections with the final layer. And once again. What is the final layer for? The classification of 'Fruits.'

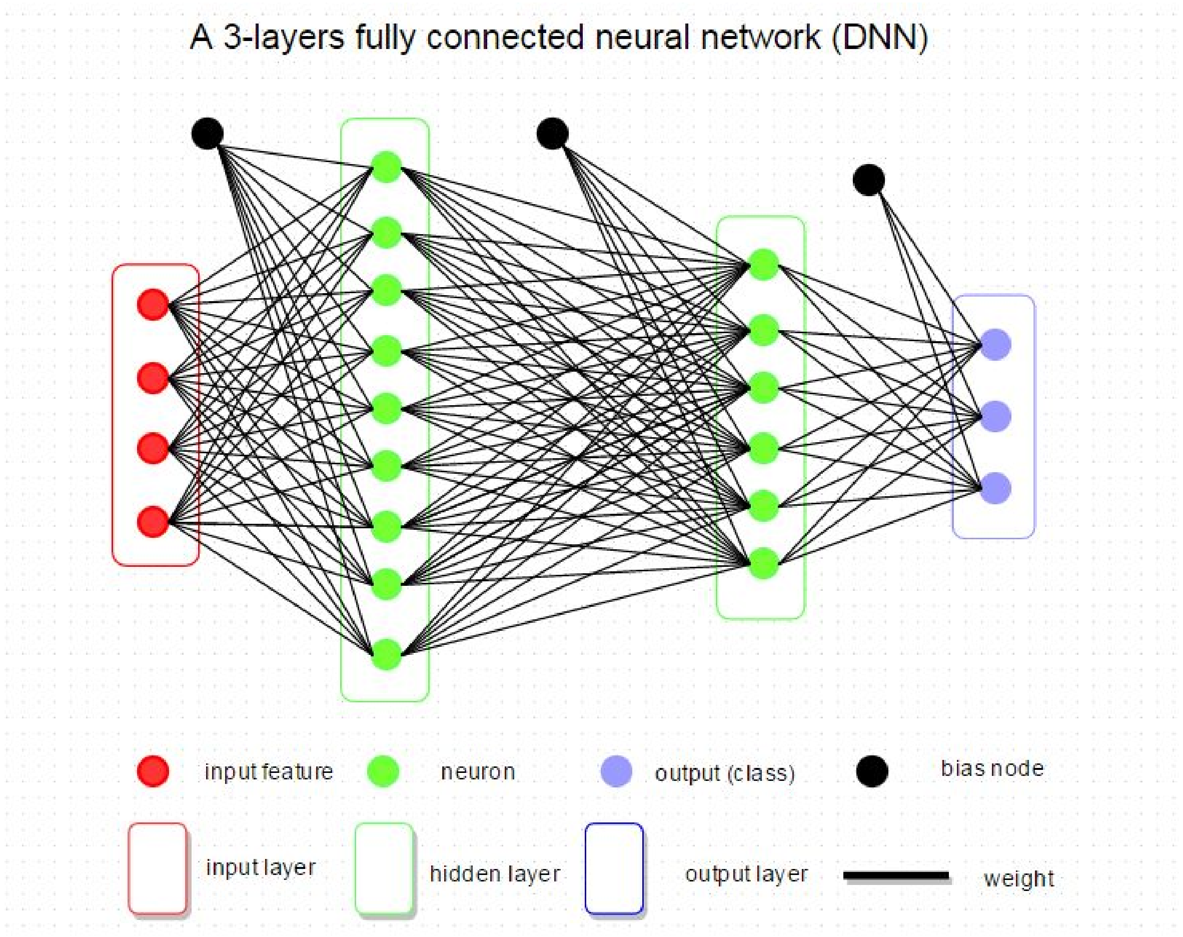


4.4 Fully Connected Layer

Fully connected layers are layers from a regular neural network. Each neuron from a fully connected layer is linked to each output of the previous layer. The operations behind a convolutional layer are the same as in a fully connected layer. Thus, it is possible to convert between the two.



The nodes in fully connected networks are commonly referred to as "neurons." Consequently, elsewhere in the literature, fully connected networks will commonly be referred to as "neural networks." This nomenclature is largely a historical accident.



4.5 Loss Layer

Loss layers are used to penalize the network for deviating from the expected output. This is normally the last layer of the network. Various loss function exist: softmax is used for predicting a class from multiple disjunct classes, sigmoid cross-entropy is used for predicting multiple independent probabilities (from the [0, 1] interval)

5. Fruit Data set

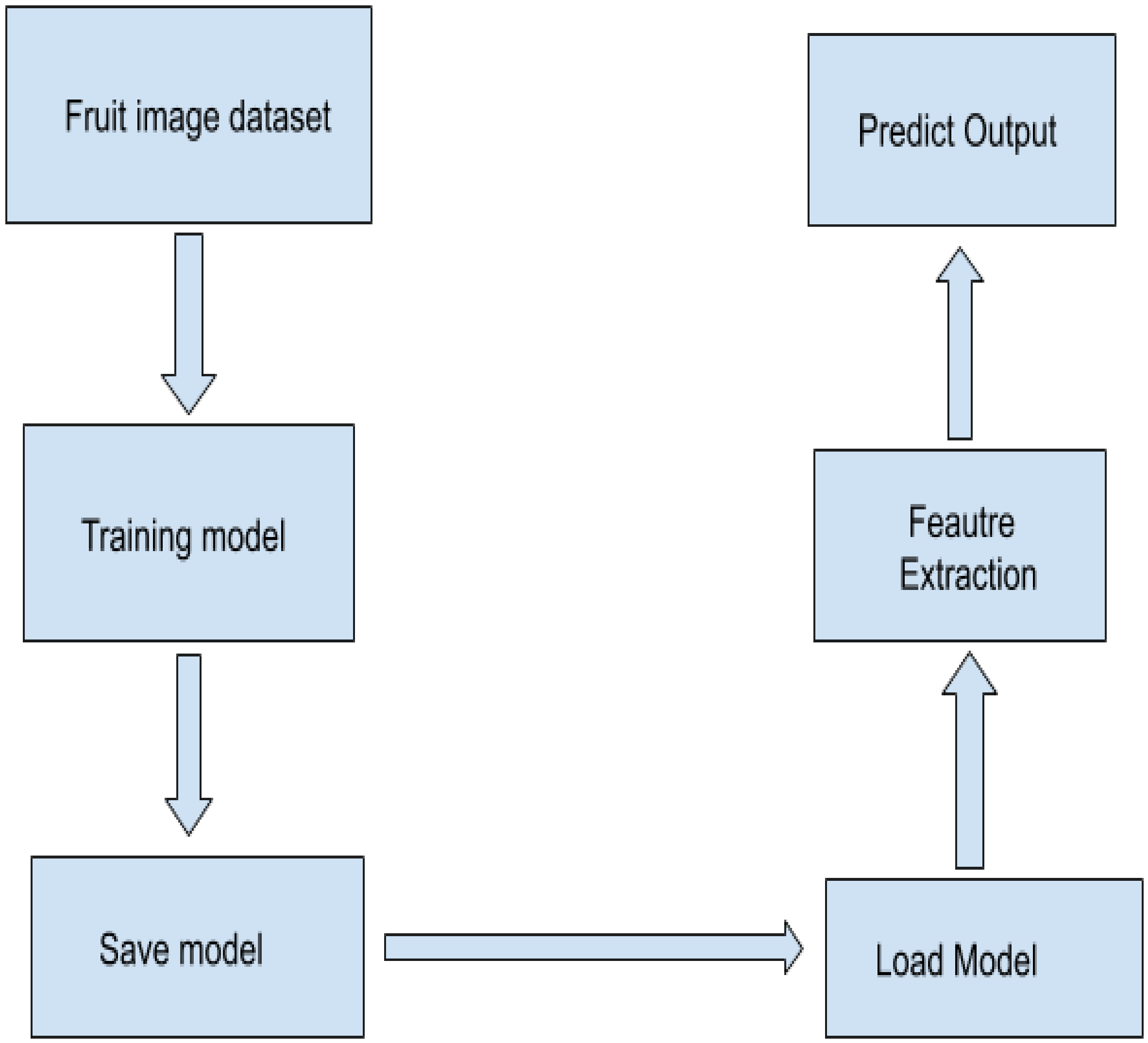
Jonas Jongejan, a creative technologist and member of the Google Creative Lab, was jetlagged and heading to an internal hackathon. He was wracking his brain for something fun to make that was related to artificial intelligence. Then it hit him: What if he took a machine learning algorithm built to identify objects in photos, and trained it with images of line drawings and doodles instead? He discovered that it worked pretty well, and even when it failed, the results were entertaining–while also offering a small peek into how computers see. "The fact that it was not perfect, that you could play with it," Jongejan says. "That was where *Quick, Draw!* was born."

Within six months, *Quick, Draw!* had gone viral. People from all over the world were playing the experimental game, which asks you to doodle an object or animal while the neural network tries to guess what it is. It was on the front page of Reddit. Dozens of publications, *Co.Design*included, covered the quirky little site.

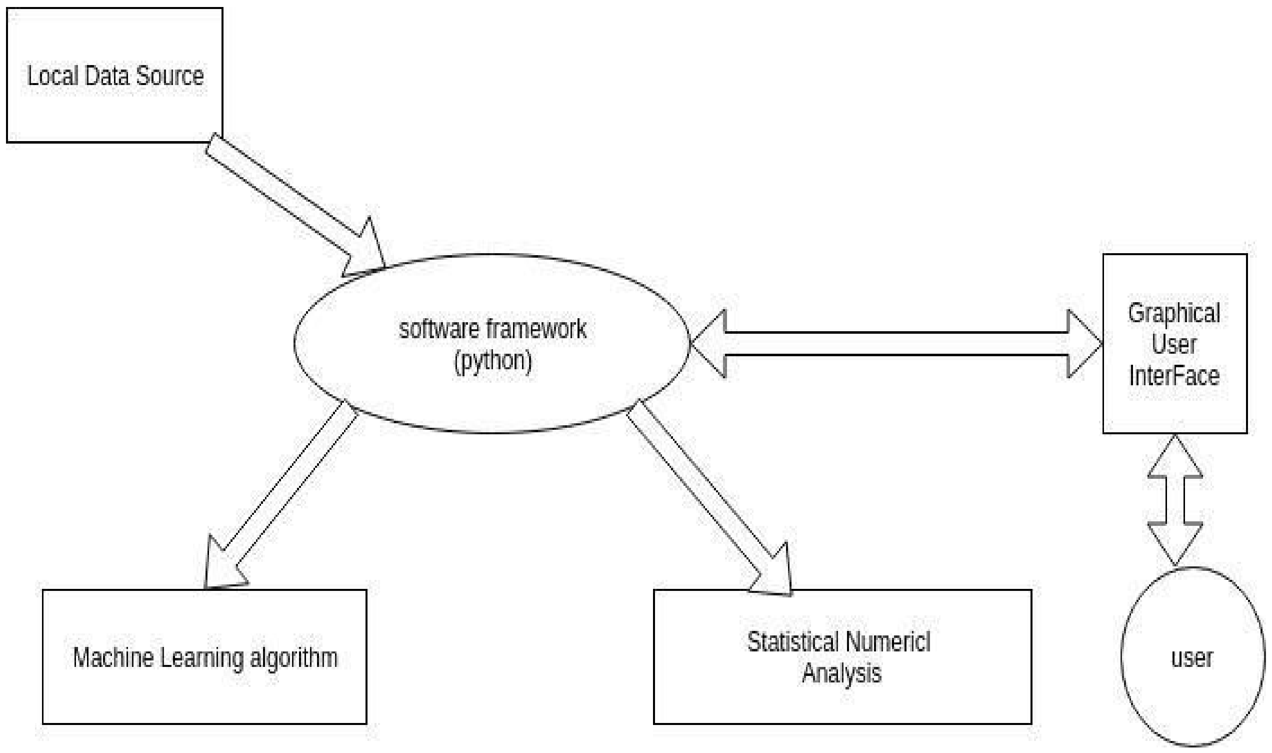
Quick, Draw! also had an unintended consequence. Anyone who played the game–now millions of people–was contributing their drawings to a new data set full of doodles. Together, they were creating a whole new set of drawings on which to train a neural network more advanced than the intentionally imperfect *Quick, Draw!* algorithm. The group's next game, AutoDraw, was built from the data collected from the first. It went further than Jongejan's original; while players still doodled objects as the neural net tried to guess, it also provided them with a clip art version of their drawing, whether it was a car or a tree–clearly useful functionality.

Jongejan never imagined that *Quick, Draw!* would end up with more than 50 million doodles, or that it would spawn research papers, coding tools, and even an analysis of how people are really bad at drawing flamingos. His game is the biggest runaway success of Google's AI Experiments project, and perhaps the best illustration of its value to the company.

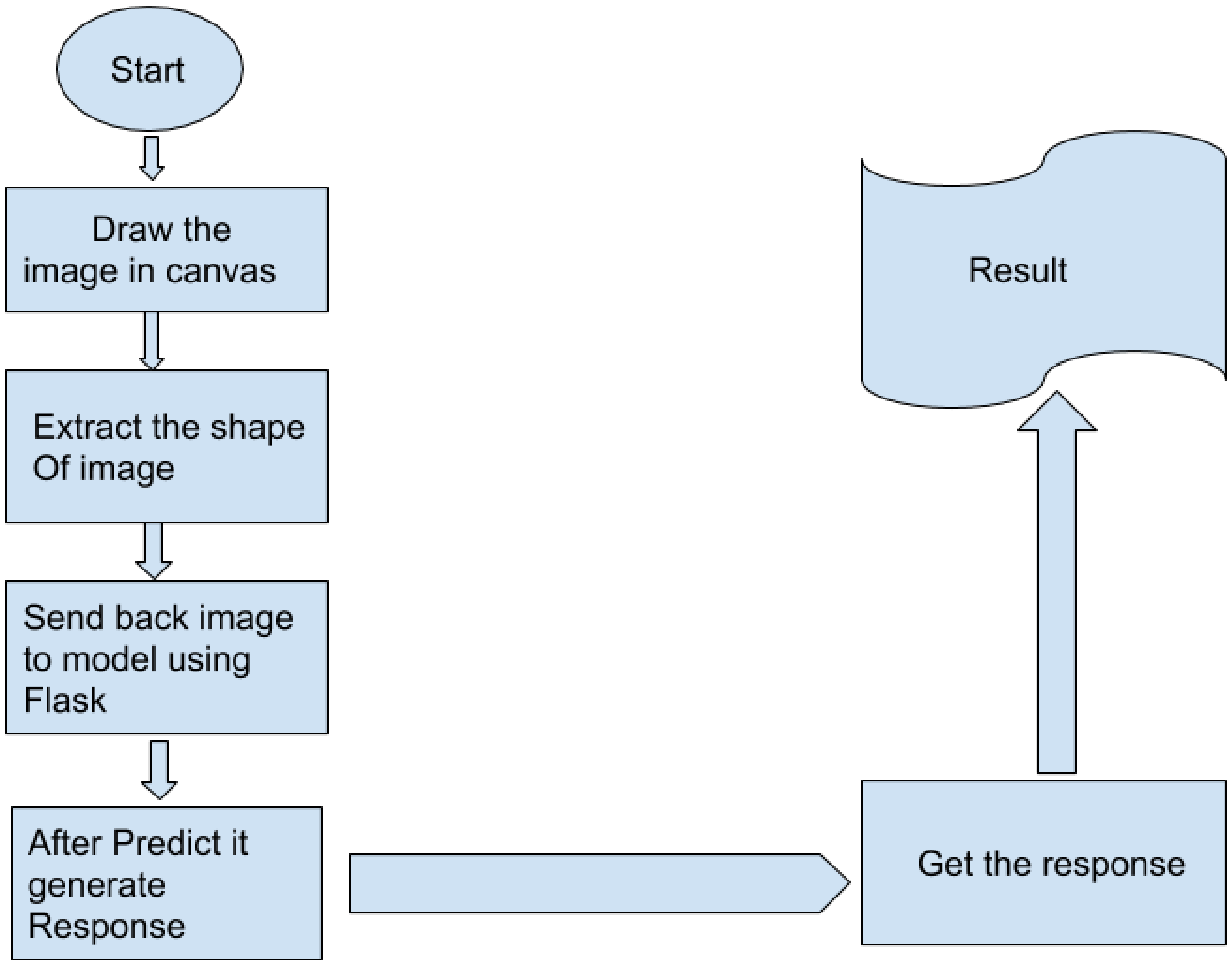
6. Block Diagram



7. System Flow Diagram


Local Data Source
software framework (python)
Graphical User InterFace
Machine Learning algorithm
Statistical Numericl Analysis
user

8. Data Flow diagram


Start
Draw the image in canvas
Extract the shape Of image
Send back image to model using Flask
After Predict it generate Response
Get the response
Result

9. Proposed Algorithm

1.K-means Clustering

k-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as barycenter of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new center. A loop has been generated. As a result of this loop we may notice that the k centers change their location step by step until no more changes are done or in other words centers do not move any more.

Algorithmic steps for k-means clustering

Let X = $\{x_1,x_2,x_3,\ldots\ldots..,x_n\}$ be the set of data points and V = $\{v_1,v_2,\ldots\ldots.,v_c\}$ be the set of centers.

1) Randomly select *'c'* cluster centers.

2) Calculate the distance between each data point and cluster centers.

3) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers..

4) Recalculate the new cluster center using:

$$v_i = (1/c_i) \sum_{j=1}^{c_i} x_i$$

where, *'c_i'* represents the number of data points in i^{th} cluster.

5) Recalculate the distance between each data point and new obtained cluster centers.

6) If no data point was reassigned then stop, otherwise repeat from step 3

2. CNN

It is assumed that reader knows the concept of Neural Network.

When it comes to Machine Learning, Artificial Neural Networks perform really well. Artificial Neural Networks are used in various classification task like image, audio, words. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural Network. In this blog, we are going to build basic building block for CNN.

Before diving into the Convolution Neural Network, let us first revisit some concepts of Neural Network. In a regular Neural Network there are three types of layers:

1. Input Layers: It's the layer in which we give input to our model. The number of neurons in this layer is equal to total number of features in our data (number of pixels incase of an image).
2. Hidden Layer: The input from Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layers can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by addition of learnable biases followed by activation function which makes the network nonlinear.
3. Output Layer: The output from the hidden layer is then fed into a

logistic function like sigmoid or softmax which converts the output of each class into probability score of each class.

The data is then fed into the model and output from each layer is obtained this step is called feedforward, we then calculate the error using an error function, some common error functions are cross entropy, square loss error etc. After that, we backpropagate into the model by calculating the derivatives. This step is called Backpropagation which basically is used to minimize the loss.

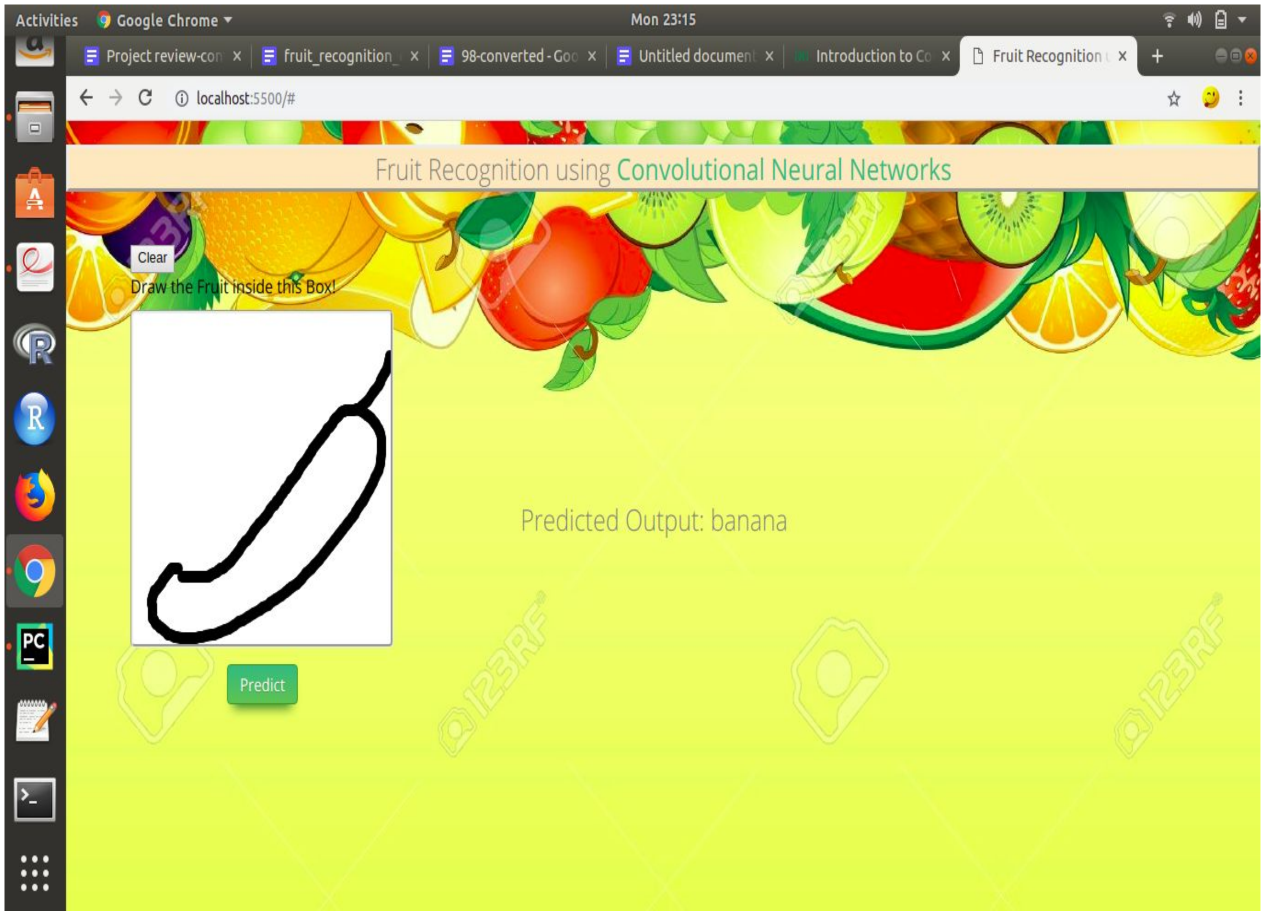
10. Conclusion and Further Work

We described a new and complex database of images with fruits. Also we made some numerical experiments by using TensorFlow library in order to classify the images according to their content.

From our point of view one of the main objectives for the future is to improve the accuracy of the neural network. This involves further experi- menting with the structure of the network. Various tweaks and changes to any layers as well as the introduction of new layers can provide completely different results. Another option is to replace all layers with convolutional layers. This has been shown to provide some improvement over the net- works that have fully connected layers in their structure. A consequence of replacing all layers with convolutional ones is that there will be an increase in the number of parameters for the network . Another possibility is to replace the rectified linear units with exponential linear units. According to paper , this reduces computational complexity and add significantly better generalization performance than rectified linear units on networks with more that 5 layers. We would like to try out these practices and also to try to find new configurations that provide interesting results.

In the near future we plan to create a mobile application which takes pictures of fruits and labels them accordingly.

Another objective is to expand the data set to include more fruits. This is a more time consuming process since we want to include items that were not used in most others related papers.



Fruit Recognition using Convolutional Neural Networks

Predicted Output: banana