# Abstract

This document supports the interactive object recognition artificial-intelligence technology demonstrator that recognises apples, bananas, and oranges.

This document teaches you what the AI system does. What all the technical jargon is. How to use it and how to learn how to make it.

# What it is and How it works

The object recognition system uses a CNN (convolutional neural network) to classify what object the camera is seeing.

*A Convolutional neural network is a class of deep neural network. It’s particularly suited to the analysis of image-based data.*

*A deep neural network is one of the many different types of architecture of deep learning.*

*Deep learning is a machine learning method that uses (artificial) neural networks. Artificial neural networks are inspired by the neural networks that comprise the brains of animals.*

*Machine learning is about algorithms that learn to carry out tasks without needing absolutely explicit/detailed instructions. The algorithms learn from recurring aspects/features or patterns of features across the data on which they’re trained. It is a subset of artificial intelligence.*

*Artificial intelligence is a system/machine/device that uses feedback from its environment (closed-loop feedback) in order to achieve a target output.*

*The model has to be trained first. What we have here is a type of learning called supervised learning, which means that the model learns through trial and error and refers to the label of the image (apple, banana, etc) to correlate features it sees in the input image with the objects in question.*

An image is fed into the input layer of the model. Inside every layer lie neurons. Between every layer lie the weights. The second layer is the convolution layer. Convolution layers are where feature matching happens. All inter-layer neurons have weighted links.

As the image goes from a convolution layer to the next layer (usually an activation layer), the input is multiplied by the weights between neurons, goes through a threshold filter called an activation function, and determines the output values of the neurons. In the next layer, which is the pooling layer, the operation being carried out to determine the neuron values is called pooling.

What pooling does is condense the size of the layer, essentially narrowing down the features and eventually condensing an assortment of features into a classified object. As we go further down the model, more convolutional, activation, and pooling, layers are added. In the end, the output layer’s neuron that corresponds to the object in question will have the highest activation intensity.

The hope is that the recognition/classification accuracy of the object in the image relative to what the image label is as high as possible. The image is sent through the model again and the weights between the layers are changed. If it’s more accurate this time the computer knows its heading in the right direction if not, it assigns a “cost” function and adjusts the weights differently.

It knows it’s going the right way because of gradient descent. Gradient descent is an optimisation tool, and there are many methods of doing it. They observe the rate-of-change (gradient) of how accurate or inaccurate the model is becoming with each pass. The more accurate, the lower the “cost”. The job of gradient descent is to arrive at the lowest cost/inaccuracy or smallest gradient possible.

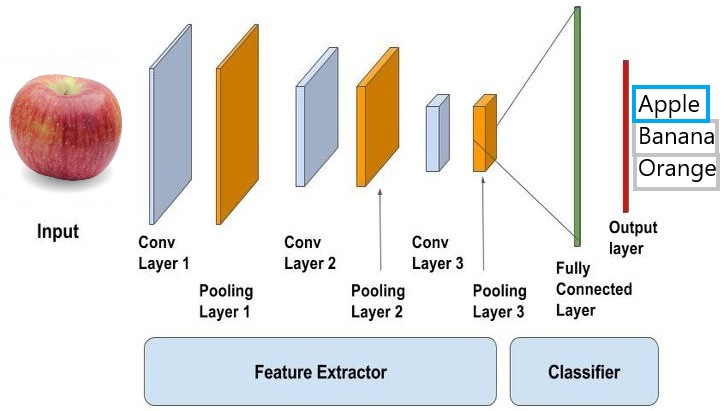
This is deep learning. The computer realises that the classification accuracy is not 100% so, through trial and error, it adjusts the values of the weights until the output for classification is as accurate as possible.

## Difficulties and hurdles

The accuracy of your model depends mainly on the quality of the input image data you feed it. It’s difficult to know what the neural network is learning and how it’s learning it. If your training images have recurring patterns/other objects (like tables and chairs in photos of apples), it might think that a table is part of an apple and if it’s not present in the input image, then an “apple” is not present.

Try having a different background, lighting, and other non-recurring objects in the image. The more you have the better. Usually, more than a thousand images are required but if your training images and working environment (what the camera looks at when the system is deployed) look similar enough to the computer, you can make a reasonably accurate classifier/model with less than a hundred training images.

This trial and error, rewriting the values of the weights, deep learning operation requires a lot of computing power or a lot of training time. GPUs tend to be faster at neural network tasks than CPUs, so if you have the option of using a GPU for training, by all means, utilise it.



# How to Make/use it

If you wish to make use of the following information, you are expected to have above-average computer use competency since you will be doing computer programming.

To follow this section, you will need to:

1. Have access to a personal computer
2. Familiarise yourself with the Python programming language, what the website is, where you can find tutorials, and how to install it.
3. Install the latest version of Python (3.X.X)
4. Learn to write programmes in Python
5. Learn how to install libraries in Python

## Options

There are many programming languages to choose from, but I’ve recommended Python because all the libraries you need are available, and it’s an easy language to learn.

There are many ways of doing object recognition using CNNs, I recommend YOLO because it’s fast and tends to be able to recognise object better than other systems because it looks at segments of the input image instead of the image as a whole for doing object recognition and object localisation.

Another popular method is using a CNN framework from Google called TensorFlow. If you choose frameworks like TensorFlow or Theano, I recommend using a high-level interface called Keras.

## Using a pre-trained model in YOLO

The best performing CNN object recognition system today is YOLO. It’s a special object recognition system that looks for features throughout an image differently to other types of convolutional neural network.

It does both recognition and object localisation within the image frame that gets fed to the input layer.

Download pre-trained weights from <https://pjreddie.com/darknet/yolo/> for a generic recognition model.

**Install the libraries OpenCV and NumPy**. Read the comments to figure out what the line is doing.

Use this code: <https://github.com/Mr-645/CNN_samples/blob/master/YOLO_fruit_camera.py>

This example of code was written for labelling, and drawing boxes around apples, bananas, and oranges.

## Train own model in YOLO

If you want to train your own YOLO model, refer to this link. <https://github.com/thtrieu/darkflow#training-on-your-own-dataset>. The training regime will be different for newer or older versions of YOLO, what platform you’re running on, and what framework you’re using YOLO with. This particular link is for ‘darkflow’ (darknet ported to TensorFlow) on a Windows machine.

## train and USe a model in TensorFlow

This example is for the classification of a cat or a dog. The reason why I’m giving you this example is because, this code and model actually work with decent accuracy, unlike the fruit recognition model I was trying to train. The other thing is that this type of CNN only tells you what’s in the image instead of drawing a box around the object in the photo. You will need to read the programme text output to read the classification.

Assuming you have an elementary knowledge of programming in Python, remember to install the libraries the code needs, and refer to the comments for help. You will need to learn how to use TensorFlow and Tensorboard, fortunately, there are tutorials available online.

This particular training code can write a Tensorboard log during training. Tensorboard is a tool that lets you see logged data in the browser. You want to see the validation loss decrease over time and the validation accuracy increase over time. For a full tutorial, go to this link: <https://pythonprogramming.net/tensorboard-analysis-deep-learning-python-tensorflow-keras/>

Use my code for training and running the model: <https://github.com/Mr-645/CNN_samples/blob/master/TF_cat_dog.py>

### Explanation of details

Training samples: the total number of images in ‘training’ folder to learn from

Validation samples: the total number of images in ‘validation’ folder to test against

Epochs: number of times to go through CNN updating weights

Batch: images per epoch

The optimiser fields are for assigning a type of optimiser to the CNN. The recommended option is “Adam”. Refer to the gradient descent paragraph in the “How it works” section above for what optimisers do.

The loss field refers to the loss function assigned to the CNN, and the function you choose depends on what types of objects you want to classify. If you’re classifying between two objects (or just recognising one), select a binary loss function, otherwise choose one that’s categorical.

The same goes for the “class mode” field.

For more information on training a neural network in TensorFlow with Keras, go to this link: <https://pythonprogramming.net/introduction-deep-learning-python-tensorflow-keras/>