Dataset Description

The dataset we plan to use is a collection of images of fruits taken from different rotated angles. The total size of the dataset is 65429 images. It is divided into a training and test set of 48905 images and 16421 images respectively. There are 95 different fruits which means the classification algorithm we implement will have 95 classes to choose from. There is also a smaller dataset of 103 images with multiple fruits in the same image. This can be useful to test our algorithm on more complex training sets to see how they would perform in situations more similar to the real world. The dataset also does not combine different varieties of fruit so apples are separated into five different classes corresponding to five varieties.

The images are sized as 100 x 100 pixels. While it is possible to use individual pixel values converted from RGB to HSV, this will cause the problem to be extremely complex due to the high number of parameters and hence makes it infeasible to use without some preprocessing. We take advantage of computer vision libraries and existing methods to extract features from the image and then use those as inputs for our ML algorithms. Some options available to us are the Histogram of Oriented Gradients (HOG) or a more localized feature detection algorithm such as SURF or SIFT. SURF has support for it built into MATLAB so it is easier to use but the processing time for these algorithms also need to be taken into account. This makes HOG the best option available for us to use.

Neural Network

We plan to train a neural network on a dataset. A neural network is a collection of perceptrons that work together similar to neurons firing in our brain. While, they can be physical too, they are mostly implemented as algorithms in Machine Learning examples. Neural Networks and especially their subclass of convolutional neural networks (CNN) are commonly used in image analysis and object detection.

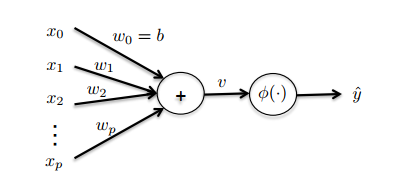


Fig. Structure of a perceptron

The perceptron works by adding a series of weighted sum of features in addition to a commonly added bias term. If the sum satisfies a condition, it activates a specific output. Perceptrons may be connected in a wide configuration of series or parallel specifications depending on the requirements of the model.

In mathematical terms, this translates to:

Where:

There is an activation function that is used to decide whether the inputs will cause a positive or negative output. This will depend on our problem. We will try to use simple activation functions such as the ReLu to simplify the complexity of the problem. This is necessary to make it computationally feasible to solve depending on the machine capabilities but there are a few challenges that we will need to account for with this approach. Due to the unbounded nature of this activation function, it can blow the activation which might force us to use a sigmoid function such as tanh instead.

The ReLu function can be described by the statement:

Stacking perceptrons leads to a neural network. A typical topography of a neural network is shown below.

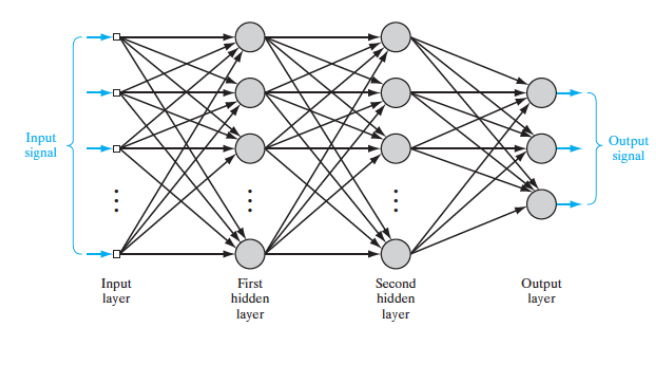


Fig. Typical neural network structure

There are a couple of challenges that are come with the territory of designing a neural network. These are choosing the number of layers and the type of layer such as a convolution or a pooling layer. This will be particularly necessary to reduce the complexity of the problem. Max pooling layers decrease the complexity and hence the computational resource necessary by decreasing the amount of parameters. This has the added advantage of tackling another challenge inherent to neural networks which is of overfitting. When implementing a Neural Networks have a tendency of overfitting data which give really high training accuracy but a bad test accuracy. By reducing the parameters, we can decrease this tendency to overfit and hence give more accurate results.