Computer Engineering Senior Project Proposal

B. Eng. Computer Engineering

Academic Year 2559

Project title: Sugar Cane Grading from photo using machine learning

Group number: 2559:39

Group member:

1. Miss Pham Thi Mai Phuong 56070503447 [phuongmaipham@icloud.com](mailto:phuongmaipham@icloud.com)

Advisor:

Dr. Sally E. Goldin

**Introduction**

## **Keywords:** sugar cane grading, machine learning, image processing, convolutional neural network

## **Problem Statement, Motivation, and Potential Benefits**

Sugarcane is an important crop in Thailand. In order to produce high quality sugar, sugar companies need detailed information on the cane conditions in different fields. Features such as color and size of the leaves are some indicators of cane conditions (cane health).

A field’s condition, such as soil properties, seed quality or irrigation system, has some important consequences for those features. However, not every field is identical. Thus, the features and the resulting can quality vary from one field to another. This will cause a problem in collecting data: there are too many fields and it is too complicated for sugar companies to do exhaustive surveys to get the information that they need.

To address to this problem, this project will work in co-operation with staff from Mitrphol Sugar Company to create a software test bed for improving cane quality control over a large area. The test bed will be able to analyze the sugar cane health from mobile phone photos. The project will use machine learning to train the software to discriminate photos based on cane quality. The results obtained from this project will be useful for developing a real world system to allow individual farmers to send photos of their fields, which can be analyzed and classified to get more detailed information about cane health over a wide area. This project is thus important because it will help sugar companies gain better information with a lower surveying cost.

## **Project Types**

## This is a research - real world stakeholder project.

## 

## **Proposed Method**

The goal of this project is to develop a software test bed (not a final system) for experimenting with sugar cane images using supervised machine learning technique. The test bed will first extract the sugar cane crucial features from mobile phone photos. This test bed will then classify the cane photos into different health categories based on extracted features using supervised machine learning.

The project is expected to contain the following steps:

1. Research basic image processing concepts, machine learning concepts and algorithms with a focus on convolutional neural network
2. Collect, create and understand training/testing data
3. Write pre-processing software to standardize images
4. Extract desire features from input images (could be combined with ML frameworks, depending on which framework will be chosen)
5. Research and test/create prototypes for various ML frameworks (Potential libraries to look at: Caffe, Tensorflow, CUDA) on order to make a decision about which learning framework to use
6. Create experimental design - details of what parameters will vary; how to select training and test images (there are various strategies); how to analyze the results
7. Write scripts to control the experiments
8. Train the learning system
9. Test and analyze the result
10. Repeat training/testing (possibly) with 1) different framework; 2) different parameters.

## 

## **Original Engineering Content**

The main issue of this project is to classify mobile phone images into different classes using machine learning. There are already some existing works on this topic (i.e. recognizing images using machine learning). However, the problem of this project is more complicated because sugar cane photos are almost identical. Thus our test bed must be able to tell the difference between two slightly varied details such as two close color tones. Additionally, the choice of machine learning techniques varies depending on the problem. Therefore, we will need to choose and implement a proper machine learning technique that is able to classify the input images into the correct category. Furthermore, if we have time, we will produce a design for a system to receive photos sent from mobile phones, classify them and save the results.

## **Task Breakdown and Draft Schedule**

### **Task breakdown**

1. Analyze and determine the requirements of the project
2. Plan the project schedule
3. Work on introduction chapter of the report (chapter 1)
4. Research emphasizing on the following topics:

Work by other researchers on discriminating between similar images using machine learning

Machine learning methods for image classification and the available libraries

Basic image processing concepts

1. Create the project proposal and get feedbacks
2. Test prototypes for various learning frameworks and make a decision on which learning framework to use
3. Collect and create dataset
4. Study and understand the dataset
5. Create experimental design
6. Complete progress report for the first semester
7. Work on theory and background chapter of the report (chapter 2)
8. Work on methodology chapter of the report (chapter 3)
9. Prepare for presentation for the first semester
10. Write pre-processing software to standardize images
11. Write scripts to control the experiments
12. Test the system and fix bugs
13. Train and test the system with different parameters
14. Analyze the results
15. Complete final report for the second semester (Result + conclusions, chapter 4 and 5)
16. Create poster and prepare for presentation for the second semester

### **Draft Schedule**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Operation** | **Project Duration** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **2016** | | | | | | | | | | | | | | | | | | | | **2017** | | | | | | | | | | | | | | | |
| **August** | | | | **September** | | | | **October** | | | | **November** | | | | **December** | | | | **January** | | | | **February** | | | | **March** | | | | **April** | | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1 | Analyze and determine the requirements of the project |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | Plan the project schedule |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | Work on Introduction chapter of the report |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | Research emphasizing on the following topics: |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 1. Work by other researchers on discriminating between similar images using machine learning |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 2) ML method for image classification especially CNN and the available libraries |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 3) Basic image processing concept |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 | Create the project proposal and get feedbacks |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 | Test prototypes for various learning frameworks and make a decision on which learning framework to use |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 | Collect and create dataset |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 | Study and understand the dataset |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 | Create experimental design |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 | Complete progress report for the first semester |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 11 | Prepare for presentation for the first semester |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 12 | Work on Background and Theory chapter of the report |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 13 | Work on Methodology chapter of the report |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 14 | Write pre-processing software to standardize images |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | H | H | H |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 15 | Write scripts to control the experiments |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 16 | Test the system and fix bugs |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 17 | Train and test the system with different parameters |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 18 | Analyze the results |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 19 | Complete final report for the second semester |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 20 | Create poster and prepare for presentation for the second semester |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

\*\* H:holiday

## **Deliverables for Term 1**

* Experimental data set
* Experimental design
* Some prototype using the selected framework
* Decision on what learning framework(s) to use, with justification

## **Deliverables for Term 2**

* Complete experimental design of the test bed
* Software test bed with desirable results
* Results and data analysis

**Background and related theories**

1. **Data Augmentation**

Overfitting describes the problem when the model is tailored to fit the random noise in one specific sample rather than reflecting the overall population. The easiest and most common method to reduce overfitting on image data is to artificially enlarge the dataset using label-preserving transformations [1,2]. To each of the sampled batch, we will apply n ∈ [1, 100] random transformations. Each of these random transformations is a combination of several elementary forms transformation, which we will describe shortly [1,7]. The benefit of using little computation is that we will not have to store the pre-processed image on the disk.

Image transformation contains two major approaches: point operators (sometimes called 1-to-1 pixel transforms) and neighborhood (or area-based) operators. In point operators, each output pixel value is strictly a function of the corresponding input pixel value. Brightness and contrast adjustments are two examples of such transformation. Techniques like convolution, which we will discuss later on in part 2.1, are not 1-to-1 transform [6]. Here we will discuss brightness and contrast adjustments as the first and the second form of data augmentations.

The first form of data augmentations is brightness adjustment. We add or subtract a constant amount of light to all input pixel in order to change an image brightness, as suggested in the following function:

g(i,j)= f(i,j)+β (1)

Where f(i,j) is the pixel located in the *i-th* row and *j-th* column of the input image, g(i,j) is the pixel located in the *i-th* row and *j-th* column of the output image and β is a bias parameter which is used to control the image brightness. Brightness adjustment is equivalent to shifting the contents of the histogram left (subtraction) or right (addition).



Figure 1.2.1. brightness

The second form of data augmentations is contrast adjustment. Contrast is the different between the maximum and the minimum pixel intensity of an image. To change the contrast of an image, we change the range of the luminance value presented in the input pixels. It is mathematically suggested by the following function:

g(i,j)= α f(i,j) (2)

Where f(i,j) is the pixel located in the *i-th* row and *j-th* column of the input image, g(i,j) is the pixel located in the *i-th* row and *j-th* column of the output image and α is a weight parameter which is used to control the image contrast.



Figure 1.2.2. contrast

We can combine the brightness and contrast control in a single operation. It is mathematically represented as:

g(i,j)= α f(i,j) + β (2)

Where f(i,j) is the pixel located in the *i-th* row and *j-th* column of the input image, g(i,j) is the pixel located in the *i-th* row and *j-th* column of the output image, α and β are weight and bias parameter respectively which are used to control the image contrast and brightness.

The two other forms of data augmentation that we will apply are scaling and horizontal flipping. Scaling can be done by multiply the scale of the patch by a factor between 0.7 and 1.4 (as suggested by Alexey Dosovitskiy, Jost Tobias Springenberg and Thomas Brox) [7].

Artificially enlarge the dataset using the combination of these three forms of image translation in image pre-processing has two extra benefits. Firstly, it allows the network to “look at” the seed image at different perspectives. Secondly, it allows the CNN model to work properly, because CNN often require to be fed with a considerably large dataset [1,4]. Notice that data augmentation is different from image pre-processing.

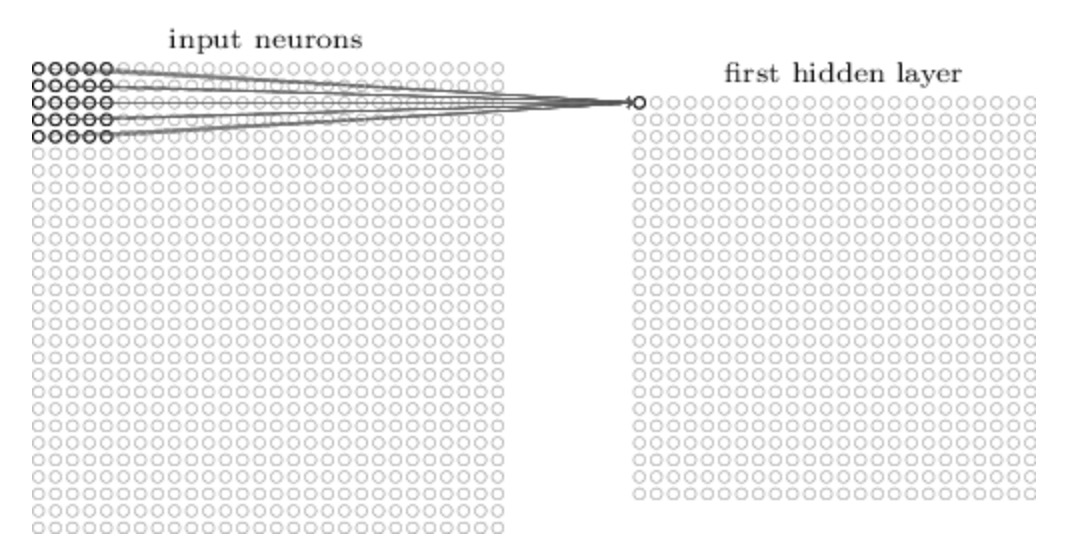
1. **Data preprocessing**
2. **Data processing** 
   1. **Convolutional Neural Networks (CNN)** 
      1. **The architecture**CNN is a state-of-the-art technique for image recognition [1,2,3,4,5]. It is a multilayer neural networks, consists of one or more convolutional layers, followed by subsampling layers (sometimes called pooling layers), and one or more fully connected layers.

Figure 3.1.1. a convolutional neural networks

* + 1. **Convolutional layer**

Four parameters are required in convolutional layer: the number of layer K, receptive field size F, the stride S and the amount of zero padding P. This layer accepts input of volume size and produces an output of volume size [(.

Think of the input image as a square of n x n neurons. The value of each neuron is the corresponding pixel intensity of the input image. We will map a localized region of the input neurons to a neuron in the hidden layer. These localized regions of input neuron are called local receptive fields.



n x n input neurons

Figure 4.1.2.1. local receptive fields

Each mapping from input layer to a hidden neuron at the hidden layer next to it learns a weight , and each corresponding hidden neuron leans a bias b. This weight and bias together make up a kernel (sometimes called a filter). The kernel value is shared among all neurons in the same hidden layer. The output generated by the , hidden neuron is a convolutional function between an input neuron , ’s value and its weight:

(3)

Where

is neural activation function

b is the shared value for the bias

the weight of the connection at the row and the column of the receptive field

is the input pixel value at the ( row and the column of the receptive field where j is row position of the hidden layer and k is the of the hidden layer

x is the total number of row in the receptive field

y is the total number of column in the receptive field

The combination of outputs generated by a hidden layer is called a feature map. To calculate the value of the next pixel in the feature map, will move our kernel k pixel along both width and height consequently (in which case we would say a stride length of k is being used) and re-apply the convolution function (3), until we run out of input neuron.



Figure 4.1.1.2.1 convolutional process - An input image of size 7x7x3 is filtered by 2 3×3x3 convolutional kernels with stride 2 which create 2 feature maps.

Each feature map can detect one feature at different locations of the same input image. To detect many features from one input image, we need several features maps. To generate several feature maps, we will need several kernels. A complete convolutional layer consists of several different features maps.

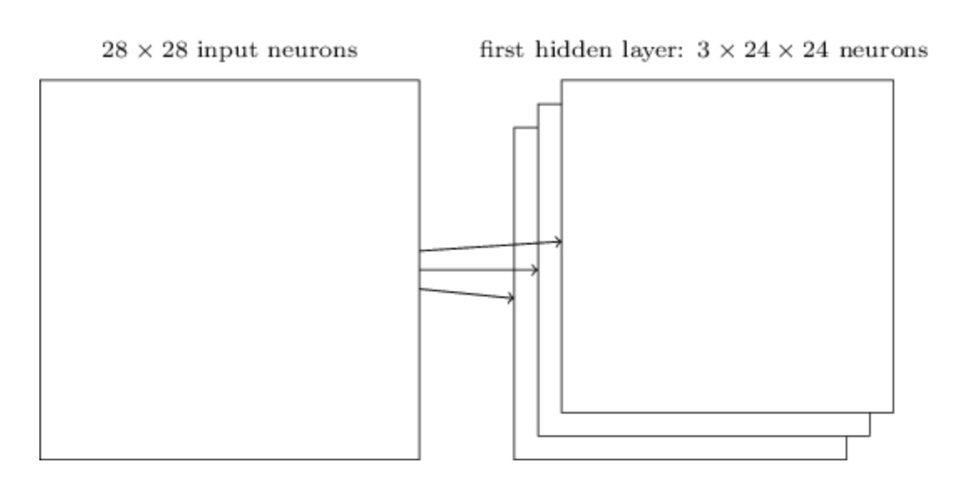


Figure 4.1.2.3. the first hidden layer consists of several features maps

* + 1. **The use of zero-padding**
    2. **Pooling layer**

Pooling layer requires two parameters: filter size F and stride S. It accepts input with volume size and produces an output of volume size [(.

A convolutional layer is usually followed by a pooling layer. Its function is to simplify every feature map in the previous layer at every depth slide spatially. The benefit of performing this step is to reduce the amount of parameters, computation in the network and therefore overfitting. A successful pooling layer should be able to preserve critical information while being invariant to troublesome deformations [9].

Pooling is done via a summary statistic over a region of neurons in the preceding layer. The summary statistic in is defined by the norm of inputs in the pools. If node are in the pool, the output of the pooling process can be mathematically represented as:

(4)

Two widely used pooling techniques can be derived from function (4). The first one is max pooling. This is when p → ∞

(5)

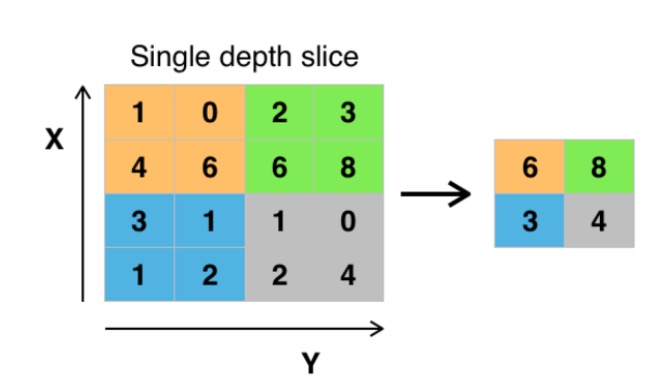


Figure 4.1.4.1. pooling layer – An input single depth slice of size 4 x 4 is filtered by a 2x2 kernel with stride 2 which results in a pooling feature map of size 2 x 2

When p = 2, we have pooling:

(6)

A filter size of 2x2 and a stride of length 2 are often applied with these two pooling techniques. Average pooling (when P = 1) was often used historically but has recently fallen out of favor because it gives less competitive results than the other two.

An another known pooling technique is called subsampling [10]. The output of subsampling function is given by:

β + b) (7)

Where

β is a trainable scalar

b is a bias

n x n is the size of the receptive field

According to some research [4,9], there is no best pooling technique. We will need to try applying several pooling techniques to our problem to see which one yields the best result.

* + - 1. **Overlap pooling**

A. Krizhevsky et al. has taken a step further to the traditional pooling [1]. Suppose a pooling layer consists of a grid of pooling units, each summarizes a neighborhood of size z x z and stride s. Traditionally, we set z = s, where we obtained non overlap pooling. Now we will set s < z to obtain overlap pooling. A. Krizhevsky et al. has pointed out this is a more effective pooling technique as it reduces the top-1 error rate by 0.4% and top-5 error rate by 0.3%.

* + - 1. **Exploiting viewpoints in pooling**

This approach was introduced by Sander et al. in 2014. After preprocessing and data augmentation, viewpoints are extracted by the combination of flipping, rotating, cropping to the input image. Notice that the combination of those operations is necessary because it reduces redundancy. All viewpoints are presented to the same convolutional architecture. The resulting feature maps from each viewpoint are first concatenated, then processed by a set of fully connected layer to obtain predictions. The benefit of exploiting viewpoints in pooling is that it allows the network to “look at” the image at different angles.

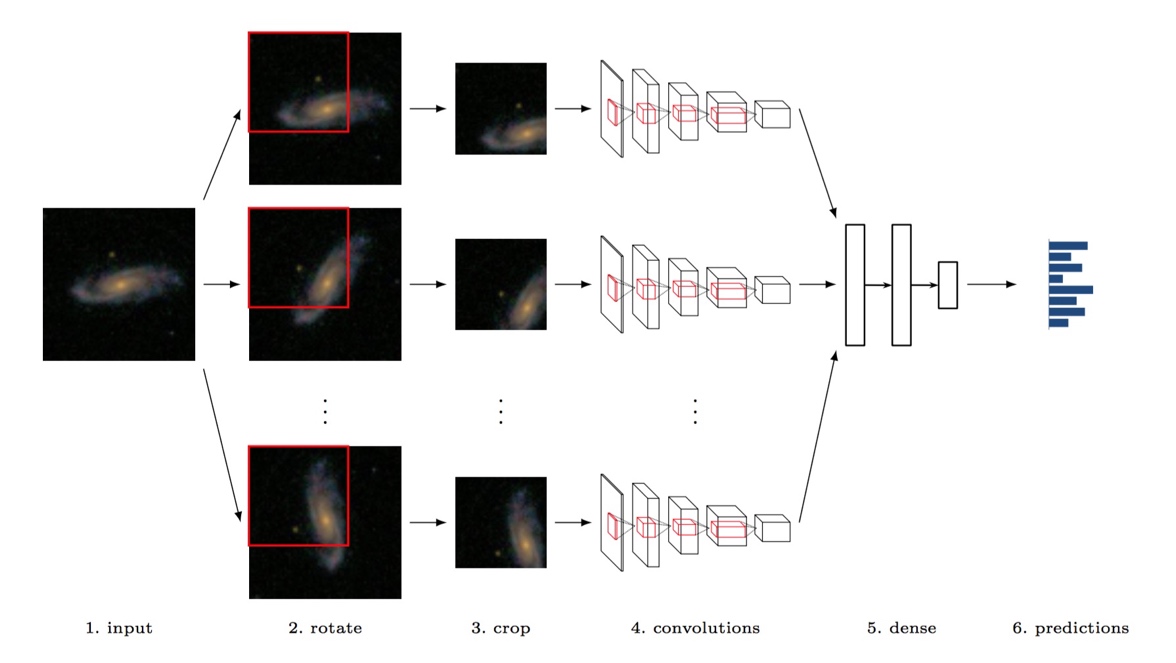


Figure 4.1.4.1. overview viewpoints exploitation in pooling. Different viewpoints are extracted from the original image (2,3). Each viewpoint is fed to a separated convolutional architecture (4). Results generated by each convolutional architecture are concatenated and feed into dense layers (5) to obtain prediction (6).

* + 1. **ReLU non-linearity**

A standard way to model a neuron output is with tanh function:

f(x) = (8)

Rectified Linear Unit is a modern way to model a neuron output f as a function of input x. It computes the function:

f(x) = max(0,x) (9)

Where x = Wx + b (10)

which means the output is 0 when the input is less than 0 and the output is a linear with slope 1 when x the input is greater than 0.

ReLU has two advantages over tanh function. Deep learning neural networks with ReLU was found to train significantly faster than networks with tanh unit. This is because in training time with gradient descent, saturating nonlinearities are much slower than non- saturating nonlinearities. Secondly, while tanh function involves expensive operations, ReLU can be implemented easily by thresholding a matrix of activations at zero. A convolutional layer is often followed by ReLU.

* + 1. **Fully connected layers**

Flatten output from the last pooling layer is proceeded by a set of fully connected layers. Fully connected layers are originated from multi-layers feed forward neural networks. They consist of an input layer, one more hidden layers and an output layer that connected in acyclic graph. Each layer is made up of neurons. Neurons between two adjacent layers are pairwise connected, but neurons in one layer share no connection. Cycles are not allowed since it will create an infinite loop. No direct connection exits between input and output layer.

The inputs are fed into the neurons making up the input layer. The outputs produced by this layer are then weighted and passed simultaneously to the next layer known as hidden layer. Hidden layer’s outputs are again weighted and used to input to an another hidden layer and so on. It is arbitrary how many hidden layers there should be, but normally we only use one. The weighted output of the last hidden layer are input to an output layer, where the prediction for the given tuples will be produced.

Neurons in the input layer are called input units. Neurons in the hidden layers and the output layers are called neurodes or sometimes referred to as output units. The number of input units are not necessarily equal number of input units. There can be more or less number of hidden units than number of input or output units. Each output unit applies a nonlinear (activation) function to its input. This function is suggested as followed:

(11)

Where   
 is output from each hidden node j, where node j precedes node k  
 is the weight of the connection between node j and k  
 is the bias of node k  
 is the output computed by node k

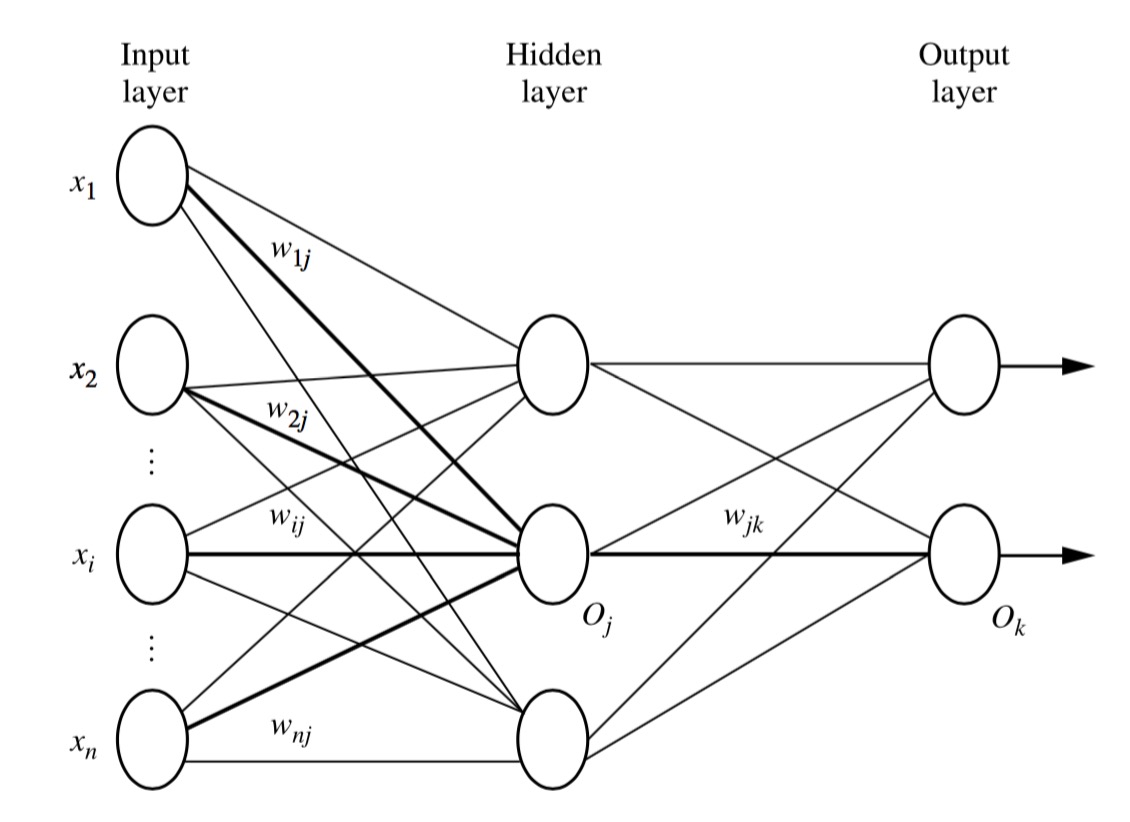
****

Figure 4.1.6.1. fully connected layers. Photo from Data Mining Concepts and Techniques, 3rd edition, Jiawei Han & Micheline Kamber

* + 1. **Back propagation**

Back propagation is a process to adjust the learnt weight and bias by comparing the network’s prediction for each tuple with the known target value. The aim is to minimize the error between the network’s prediction and the known target value. The target value can either be a known class label or a continuous value. The weight and bias modification process is done in a backward direction, from softmax layer through fully connected, pooling and ReLU layers to the convolutional layer. The forward and backward process is repeated until all the weights in the network converges. The steps are described next.

Initialize all weights and bias in the network with some small random values and choose a learning rate (l)

Propagate the inputs forward the networks using our initialized weights and bias. That is, for each input unit k, the net output can be computed by equation (11). This net input is then applied an activation function to produce an output for unit k, as describe in the function below:

(12)

This function can map an input to an output in a range between 0 and 1. We will repeat this process until we reach the output layer, which gives the prediction for the networks.

After the networks give a prediction, we will calculate the error for each neuron in the output layer by computing

= (13)

Where

ok is the actual output of unit k  
 Tk is the known target value of the given training tuple

This error is then propagated backward in the networks. For each

hidden layer unit, the error is calculated by

= (14)

Where

is the actual output of unit j  
 is the error of the hidden unit j  
 is the error of the hidden unit k, where the hidden unit j precedes the hidden unit k  
 is the weight of the connection between unit k and unit j

The calculated error is then used to update the weight and bias at each unit. For each unit j, the change in weight is calculated by:

(15)

and consequently the weight of unit j is updated by:

(16)

Accordingly, for each unit j, the change in bias is calculated by:

(17)

and consequently the bias of unit j is updated by:

(16)

The strategy that we presented is often referred to as case updating. In CNN, the weights and bias increments are built up in variables and only updated after the networks have gone through all the tuples in the training set. Each iteration through the training set is an epoch. This technique is called epoch updating.

Training stops when one of the following conditions is reached:

* All in the previous epoch are bellow some pre-specified threshold
* A pre-specified number of epochs has reached
* The percentage of tuples misclassified in the previous epoch is below some thresh- old
  + 1. **Model evaluation**
  1. **CNN vs. Fine Grained recognition**
  2. **Training on multiple GPUs**

1. **Data post-processing** 
   1. **Dropout**

**Reference**

[1] A. Krizhevsky, I. Sutskever, G.E. Hinton. ImageNet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25 (NIPS’2012), 2012.

[2] Evgeny A. Smirnov\*, Denis M. Timoshenko, Serge N. Andrianov. Comparison of Regularization Methods for ImageNet Classification with Deep Convolutional Neural Networks. 2013 2nd AASRI Conference on Computational Intelligence and Bioinformatics

[3] Jonathan Krause, Timnit Gebru, Jia Deng, Li-Jia Li, Li Fei-Fei. Learning Features and Parts for Fine-Grained Recognition. Invited paper. Supported by an ONR MURI grant and the Yahoo! FREP program.

[4] [Aäron van den Oord](http://reslab.elis.ugent.be/aaron), [Ira Korshunova](http://irakorshunova.github.io/), Jeroen Burms, [Jonas Degrave](http://317070.github.io/), [Lionel Pigou](http://lpigou.github.io/), [Pieter Buteneers](https://twitter.com/pieterbuteneers). Classifying plankton with deep neural networks. First prize of The [National Data Science Bowl](https://www.kaggle.com/c/datasciencebowl) competition

[5] Comparison of Regularization Methods for ImageNet Classification with Deep Convolutional Neural Networks. 2013 2nd AASRI Conference on Computational Intelligence and Bioinformatics. Evgeny A. Smirnov\*, Denis M. Timoshenko, Serge N. Andrianov

[6] [Computer Vision: Algorithms and Applications](http://szeliski.org/Book/). Richard Szeliski  
[7] Unsupervised feature learning by augmenting single images. Alexey Dosovitskiy, Jost Tobias Springenberg and Thomas Brox.   
[8] [Deep Learning](http://www.iro.umontreal.ca/~bengioy/dlbook/), draft book in preparation, by Yoshua Bengio, Ian Goodfellow, and Aaron Courville  
[9] Signal recovery from Pooling Representations. Joan Bruna, Arthur Szlam and Yann LeCun   
[10] Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. 20th International Conference on Artificial Neural Networks (ICANN), Thessaloniki, Greece, September 2010. Dominik Scherer, Andreas Mu ̈ller , and Sven Behnke  
 [11] Sander Dieleman, Kyle W. Willett and Joni Dambre.Rotation-invariant convolutional neural networks for galaxy morphology prediction. Mon. Not. R. Astron. Soc. 000, 1–20 (2014).   
[12] Data Mining Concepts and Techniques, 3rd edition, Jiawei Han & Micheline Kamber