Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Deep Learning Methodologies in Assessing Logo Complexity**
Ονοματεπώνυμο Φοιτητή	**Κουβαράκης Γεώργιος**
Πατρώνυμο	**Μιχαήλ**
Αριθμός Μητρώου	**ΜΠΠΛ/ 14036**
Επιβλέπων	**Γεώργιος Τσιχριντζης**

Ημερομηνία Παράδοσης **Οκτώβριος 2018**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Γεώργιος Τσιχριντζής Καθηγητής	Ευθύμιος Αλέπης Επίκουρος	Διονύσιος Σωτηρόπουλος Δόκτωρ

Table of Contents

Abstract

In this document , we are going to write about Complexity measure an aesthetic measure and if we can extract features from companies logos that has been rated from some experts.
The experiment will use deep Convolutional Networks written in python programming language.

The dataset will be 208 images of company logos and we are going to use the original dataset and later we are going to create an augmented dataset from the original images.

There are going to be 4 evaluation stage with these 2 datasets in order to conclude about the experiment. In the first stage we are going to use a swallow net with Convolutional Layer, in the second we are going to use a 10 layers Convolutional network, in the third method we are going to use 10 layers but this time batch normalization between the layers. Finally, we are going to use a pre-trained deep network from imagenet and see if it made an improvement in relation to the previous techniques.

Furthermore, i am writing about the theory behind the aesthetic measures and the importance in information visualization.

(Ελληνική μετάφραση)
Σε αυτό το έγγραφο θα μελετήσουμε τον όρο πολυπλοκότητα πάνω στις αισθητικές μετρήσεις κάποιο λογότυπων εταιριών ,όπου έχουν βαθμονομηθεί από ειδικούς .Στο συγκεκριμένο πείραμα θα χρησιμοποιήσουμε<< Βαθειά μάθηση>> γραμμένη σε γλώσσα Python.

Τα δεδομένα μας είναι 208 λογότυπα ,όπου από αυτά θα «φτιάξουμε» καινούργια λογότυπα ,όπου θα μπουν και αυτά στα υπάρχουν και θα μεγαλώσουν τα δεδομένα μας αλλα δεν θα χαλάσουν την αυθεντικότητα των δεδομένων.

Το πείραμα θα έχει 4 στάδια ,όπου στο κάθε ένα θα δούμε πως συμπεριφέρονται ανάλογα άμα χρησιμοποιήσουμε τα αυθεντικά δεδομένα και αν χρησιμοποιήσουμε τα «φτιαχτά» .

Θα χρησιμοποιήσουμε «ρηχά δίκτυα» , δίκτυα 10 επιπέδων, δίκτυα 10 επιπέδων με batch normalization λειτουργιά ανάμεσα από το κάθε επίπεδο και τέλος θα χρησιμοποιήσω ένα προ εκπαιδευμένο δίκτυο με δικό μου ομαδοποιητή στο τέλος.

Επιπλέον θα ήθελα να προσθέσω ότι αναφέρεται στην εργασία η θεωρία πίσω από τα αισθητικά μέτρα.

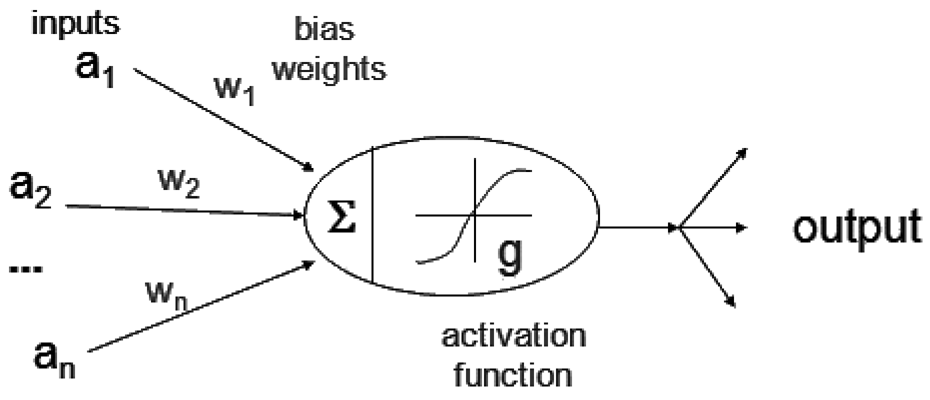
Introduction to Deep learning Approaches

What is really Deep Learning?

The term Deep Learning was introduced to the machine learning community by Rina Dechter in 1986 and to Artificial Neural Networks by Igor Aizenberg and colleagues in 2000,in the context of Boolean threshold neuron.

The first general, working learning algorithm for supervised, deep, feed-forward, multilayer perceptions was published by Alexey Ivakhnenko and Lapa in 1965. A 1971 paper described a deep network with 8 layers trained by the group method of data handling algorithm. After years of evolution in this specific scientific area we came to see some great techniques such as **Unsupervised Pre- trained Networks**, **Convolution Neural Networks (CNN)**, **Recurrent Neural Network** and **Recursive Neural Network**.

Deep Learning is in fact stacked neural networks with these networks are several layers. Each layer is made of node and this is the place that magic (computation) is happens. The inputs in these nodes are weights or coefficients that either amplify the input or dampen it, with that we are giving importance of the inputs that the algorithm is trying to learn. Every time these inputs-weights passing through the node and more specifically the activation function, is determining if the signal will affect or not the final output.



Auto-encoders

Unsupervised pre-trained network like **auto-encoders** are used to feed the layers with a training signal in order for the supervised learning not to start randomly.
The general idea is to stack some auto-encoders and in the last layers to use the supervised network for feature recognition.

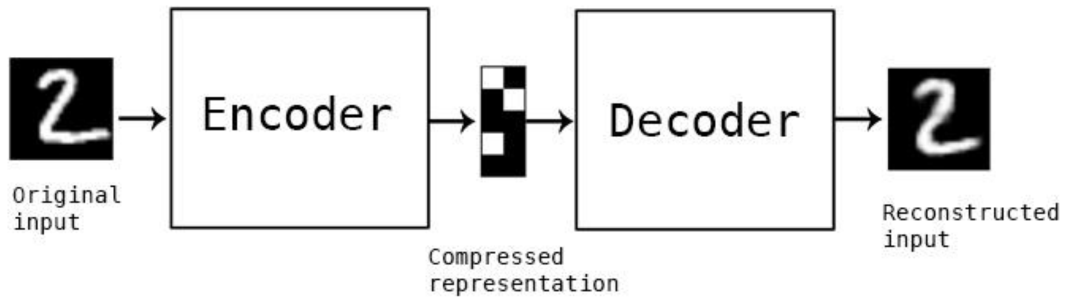


Figure 1

Convolution Neural Networks (CNN)

CNN or ConvNETS are used for analyzing visual imagery. The CNN has one layer of input and one layer of output but it includes many hidden layers. The number of hidden layers will be used, is based on how complex our problem is. A face recognition problem will demand more hidden layers (more parameters) than a logo recognition problem.

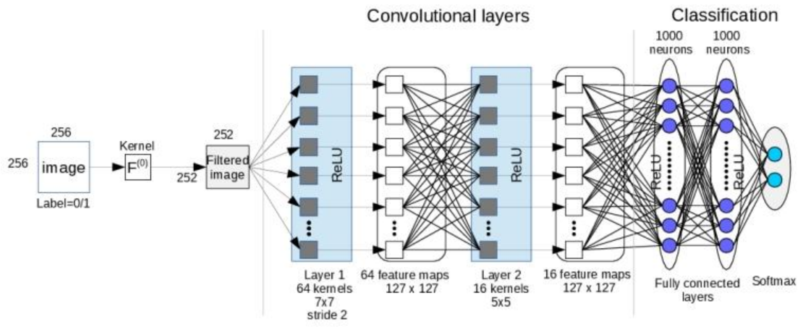


Figure 2

Recurrent Neural Networks

RNN or Recurrent Neural Networks are a sequential model. Sequential model is a model that transfer the information that has already learn in the previous layers (it is limited for only some steps behind).

Recurrent Neural networks are recurring over time. For example if you have a sequence
x = ['h', 'e', 'l', 'l']

This sequence is fed to a **single** neuron which has a single connection to itself.
At time step 0, the letter 'h' is given as input. At time step 1, 'e' is given as input. The network when unfolded over time will look like this. RNNs have shown great success in many NLP tasks.

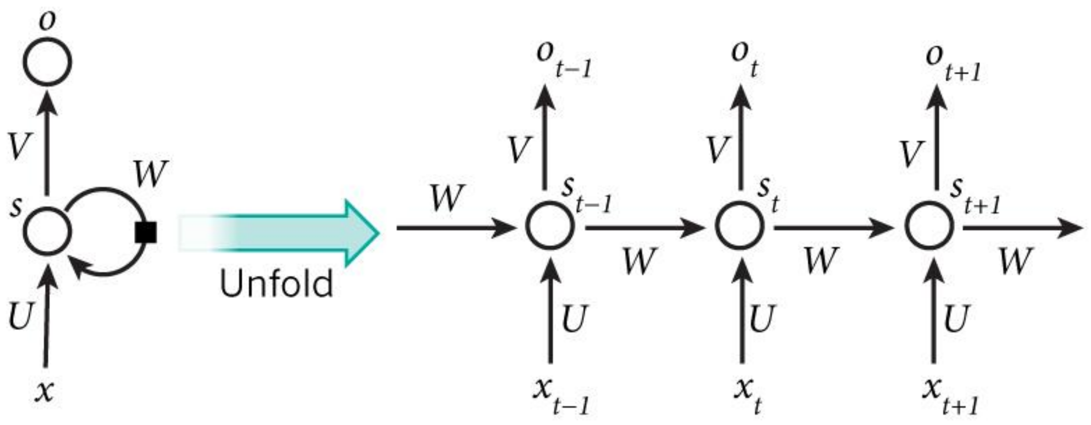


Figure 3

Recursive Neural Network

Recursive Neural Network is a model that will define its own weight with ultimate goal to produce a structured prediction over a variable size input structures.RNN have been successfully used in learning sequences and tree hieratical. Most use has been the NLP (natural languages processing).

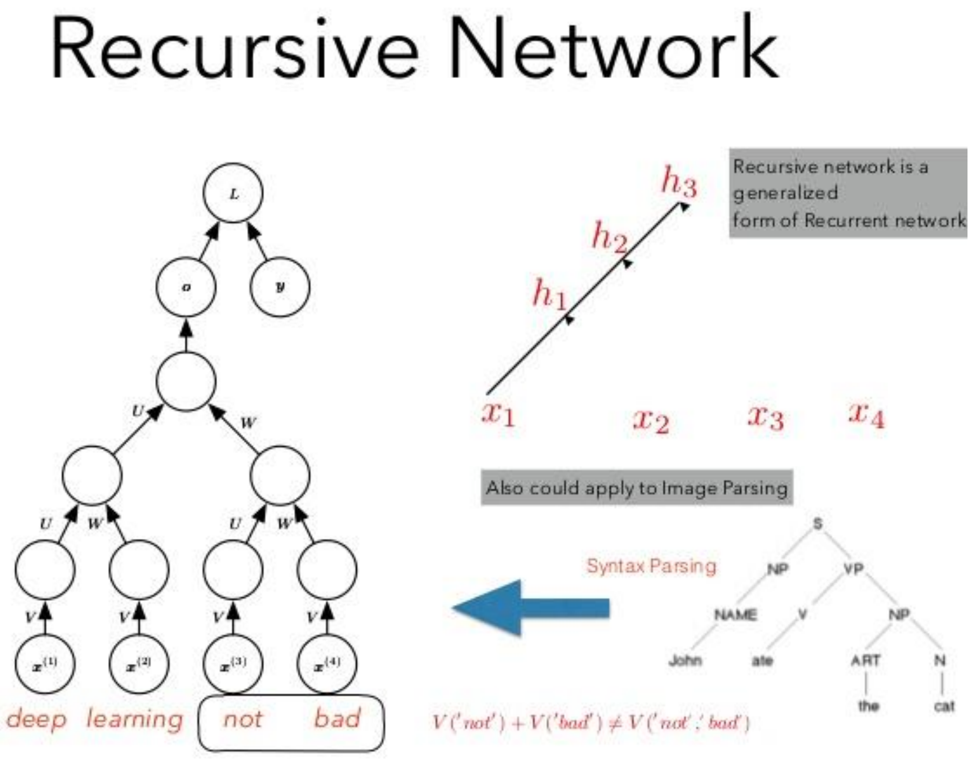


Figure 4

What can a deep learning network do?

Classification

So deep learning can solve classification problems like detect faces, identify expression (joy, angry) identify sign, classify text as a spam.

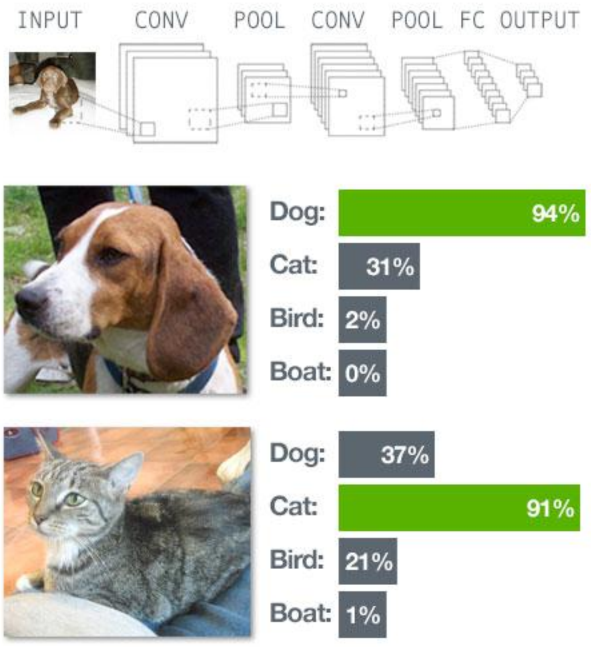


Figure 5

A classification task depends on labeled dataset and this is what we know as supervised training.

Clustering task

Searching for pattern, comparing documents, images, sounds, detection on unusual behavior.This is clustering tasks and they do not need labeled data, it is called unsupervised learning. Unsupervised learning has the potential to produce high accurate models.

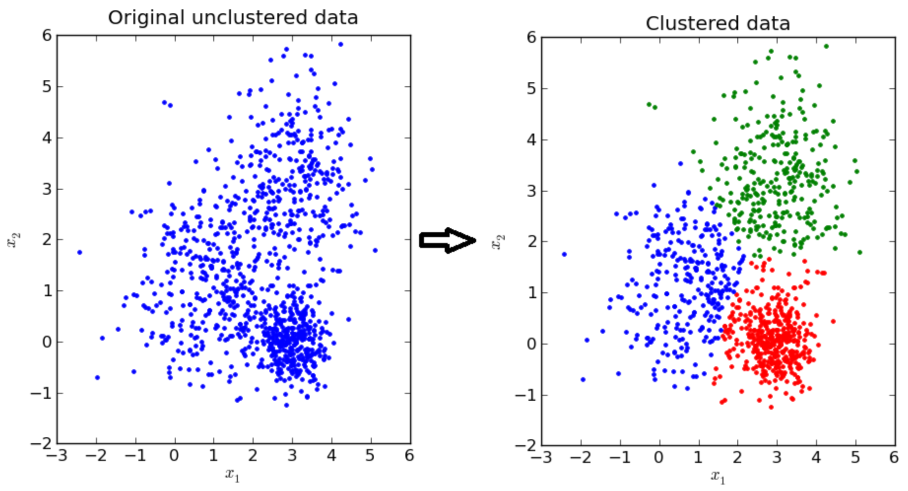


Figure 6

Predictive analytics

Regressions, using classification and correlation between an image and a value from the label, this can call a static prediction. But with enough data we can use past events and future events to establish a connection between them.

Using a regression on these events we may predict event of the future, with the meaning of future we don't necessary mean time relevant future. Let's see an example bellow, Let's use a row with numbers separated by comma, like this [1, 1, 2, 3, 1, 1, 2, 4, 5, 5, 3, 1, 1, …..]

We will find with a naked eye a pattern in this row of numbers, that after the two 1 in a row comes the 2. This was an example of what a past and future connection is all about.

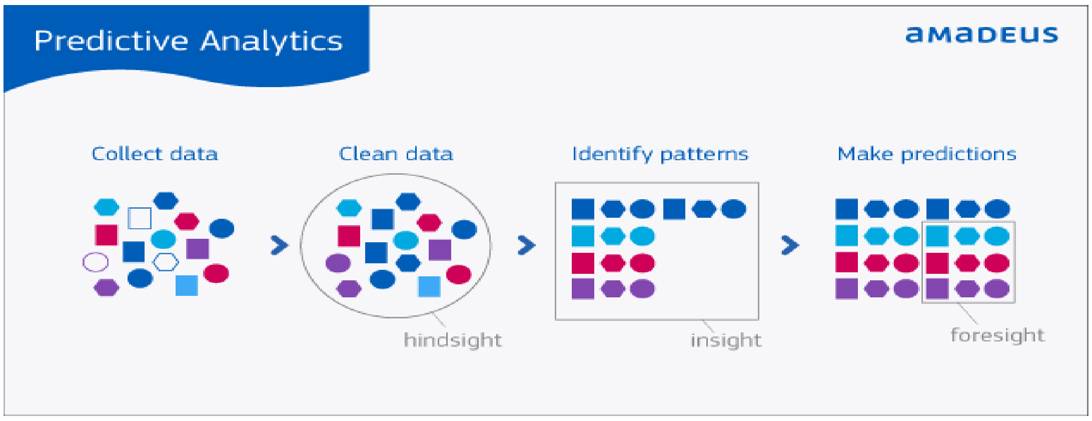


Figure 7

Feature extraction versus feature generator

Feature selection is also called variable selection or attributes selection (see figure below). Is the process of selecting a subset of relevant feature for use in model construction.

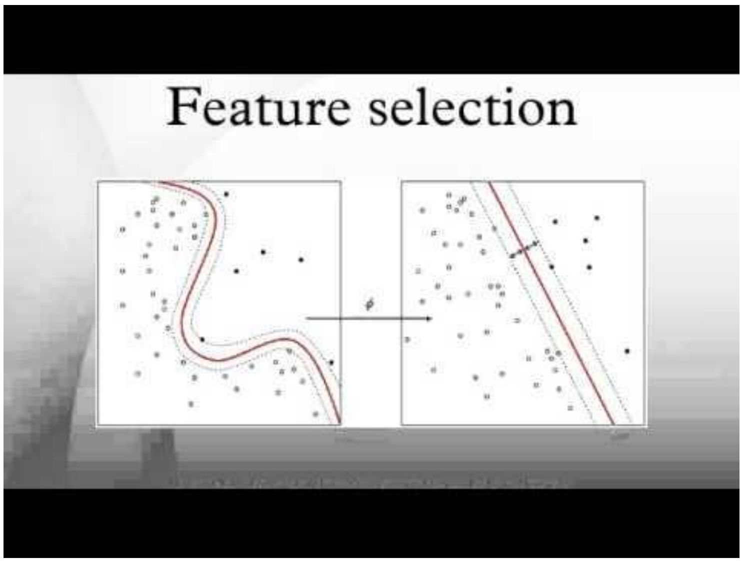


Figure 8

Feature extraction on the other hand is something like dimension reduction that represents the interesting parts of the actual image as a compact feature vector.
Deep learning models are used (among other techniques) for automatic feature extraction.

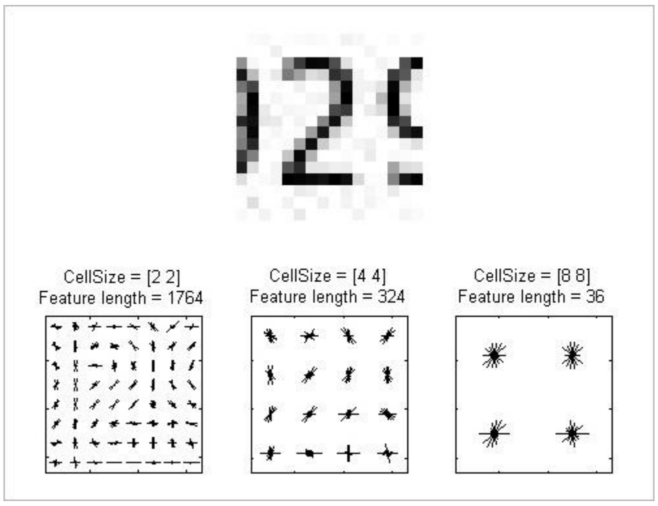


Figure 9

Importance of Data

The real question is “Is Data more important than Algorithms in Deep learning?” The answer is very complicated, it is true that these techniques need a large dataset in order to create a generalize model, so it is an essential part for our models. But it isn’t only that, you need to tune a lot of parameters in order to succeed.

However, there are some techniques with pre-trained models like imagenet. All you have to do is use your own data, probably few thousands, and you have trained you own data but only with the latest layers, you see as we explained earlier in the first layers we generate the shapes and the shades (see section) and in the last layers put our classifier.

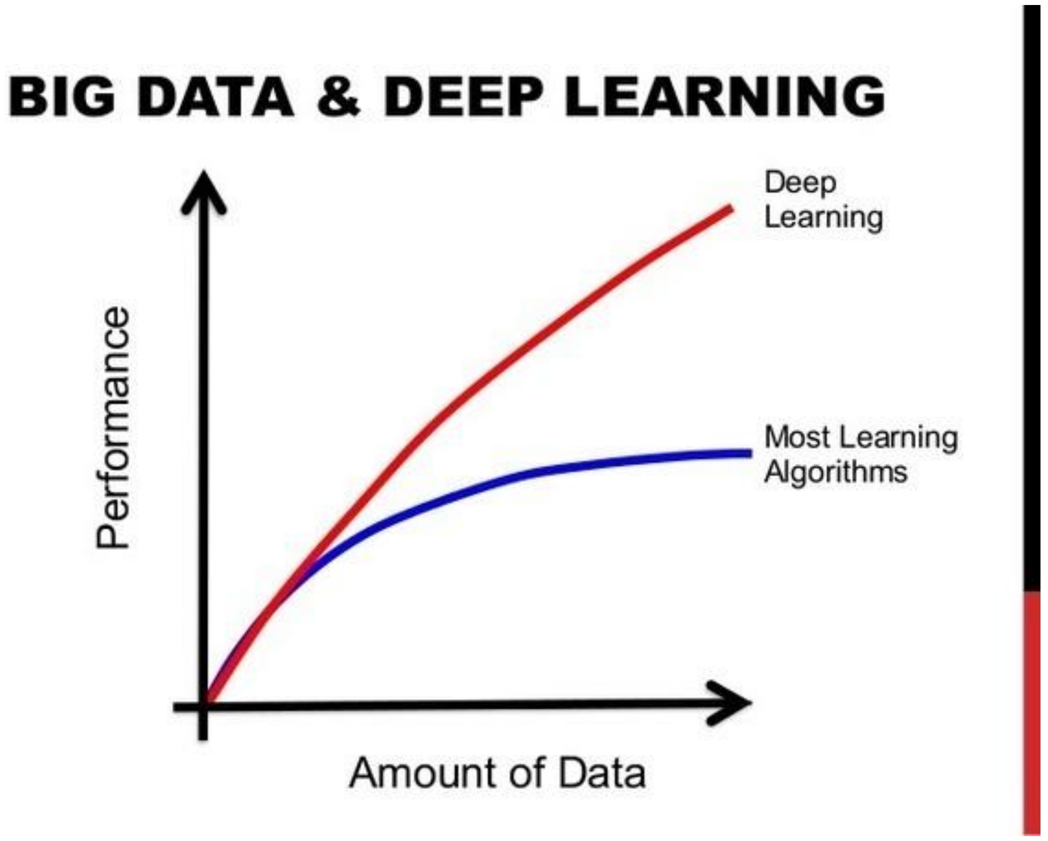


Figure 10

It’s ok with the motto “more data will produce more accurate models ,but what can we do if we don’t have the luxury having a big dataset even with a pre-trained model.
First of all, **get more data** by that I mean go to research facilities, schools and use open databases.

Secondly, you may need to create data, by that I do not mean to use random data. You can use filters, resizes, reshapes, noise cancellation or addiction, etc. This is known as augmented data.

Lastly, I recommend rescaling your data to fit your activation function are using sigmoid (figure 7) is using binary (0, 1), tanh(figure 6) is using (-1, 1), normalize your y values to fit in all the area of your activation function

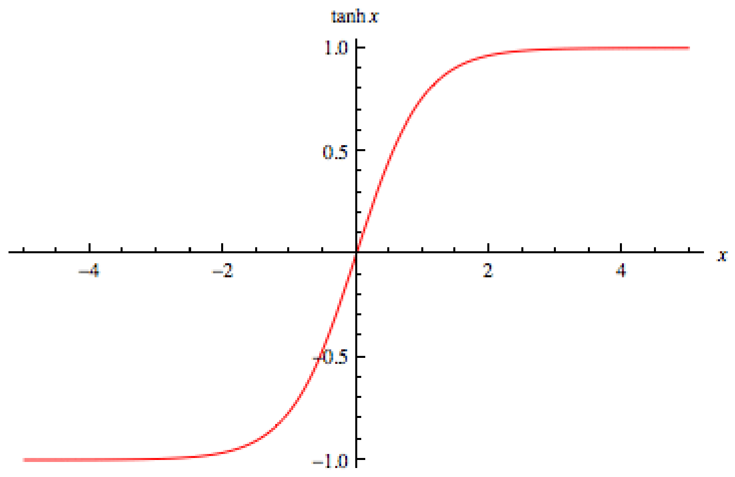


Figure 11

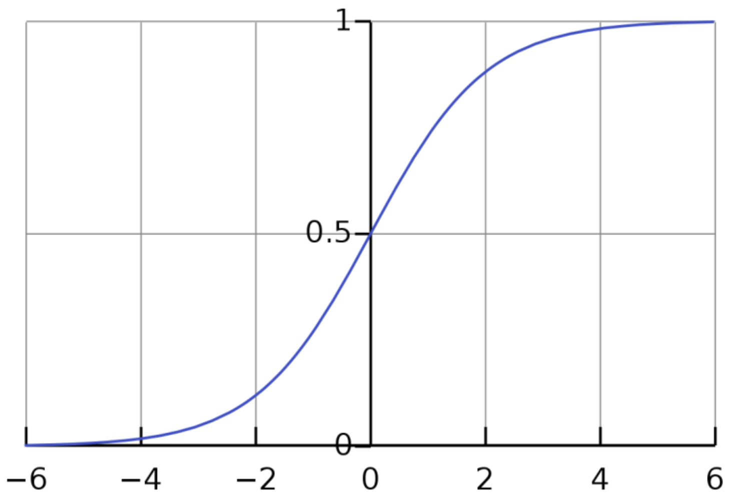


Figure 12

<u>Measuring Aesthetics</u>

Aesthetics is a science that characterizes the good, bad, pretty, ugly in the nature and most importantly in the arts such as images, painting, music, landscapes, etc. It is part of our philosophy in relation to the divinity, our value about the truth, morality and harmony. It comes from the Greek word "aisthitos" («αισθητός»), which means the vision through our senses.

A lot of philosophers and great professors had tried to measure it or had used it as a factor regarding their work and some of them were Pythagoreans, Platon, and Aristotelis.

What is complexity in aesthetics? And most importantly how can we measure it?

Complexity can be measured mainly with human ratings but we may be able to compute it using a machine.

As we may know when a human rates something, that can easily be mistaken from an objective rate to a subjective rate and that is the human factor and is called personal taste. In the following images we are going to see the differences in black and white images (not even colored).

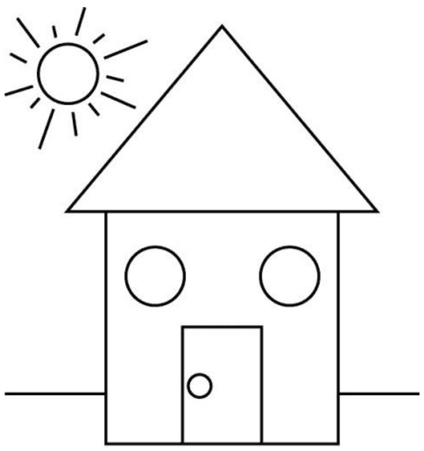


Figure 13

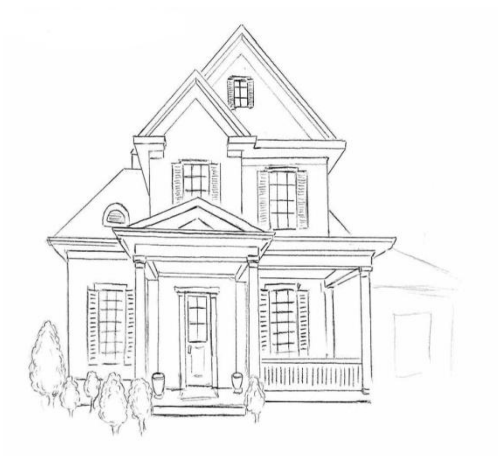


Figure 14

As you can see in Figure 1 we can easily describe it(even with words) and in Figure 2 is more difficultto described it,that is a simple example of how to describe complexity. Complexity is a term that is more theoretical than practical.
Scientists have tried to compute complexity but it was not achievable at that time, so instead they have tried to define it and then use some other methods to "compute" it.
So now we will see the following ways of expressing the **complexity:**

1. **Compression based on image complexity**

2. **Entropy as a measure to express complexity**

COMPRESSION-BASED IMAGE COMPLEXITY

Compression Ratio

The most common complexity defined method is introduced by Kolmogorov, which says *the complexity of an object is the length of the shortest binary computer to describe it*. However Kolmogorov method is not computational. So for that matter, we will define a CR (compression Ratio)

$$CR = \frac{Imag\_file\_size}{file\_size\_after\_compresion}$$

So now, we can define the Image Complexity (IC):

$$IC_{ls} = \frac{1}{CR}$$

Ls= lossless

In computation aesthetics, loss compression and distortion are factors that may describe by definition the image complexity. So, for that matter we will see 2 more algorithms (describing IC by definition):

$$IC_{RMSE\,(q)} = \frac{RMSE(q)}{CR(q)}$$

Factor **q** stands for the grade of the compressor, the higher the grade, the less compression has been done, so that means better quality of the image. RMSE is the root mean square error between the original image and the lossy compressed image
And the third method is:

$$IC_{LY} = \frac{1}{CR_{(q)}}$$

LY is lossy compression

These three algorithms are close in defining the complexity of an image based on compression. In the figure and table below we will see the results of an experiment made by to demonstrate the correlation between compression level, complexity(figure 3) and the coefficient between the different algorithms(table 1)

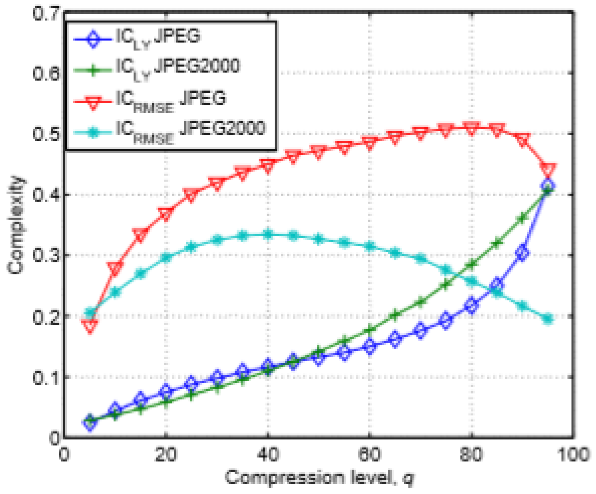
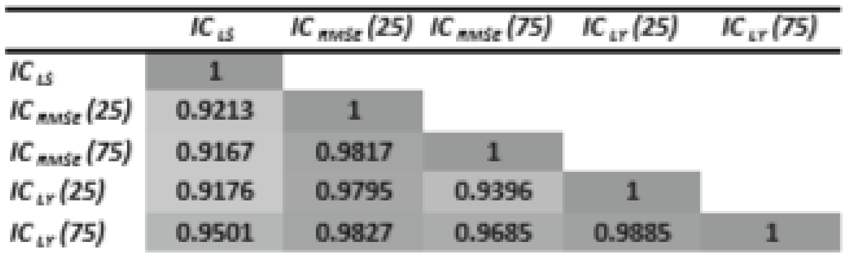


Figure 15



	IC_{LS}	$IC_{RMSE}(25)$	$IC_{RMSE}(75)$	$IC_{LY}(25)$	$IC_{LY}(75)$
IC_{LS}	1				
$IC_{RMSE}(25)$	0.9213	1			
$IC_{RMSE}(75)$	0.9167	0.9817	1		
$IC_{LY}(25)$	0.9176	0.9795	0.9396	1	
$IC_{LY}(75)$	0.9501	0.9827	0.9685	0.9885	1

As we can see the relation between them is very close, this result came out of 500 uncompressed image of 512x512 resolution.

Run Length encoding as a measure of Processing Complexity

RLE(Run Length Encoding) is one of the simplest encodings we can find, therefore I will give an example about this:
Let’s assume we have this string ‘GGGREECCCCEEDDDDDDDRR’ if one digit of this takes 2 bits then the output will have:

21 digit X 2 bits = 42 bits,

With RLE we can easily encode it to this 3G2E4C2E7D2R (13 digit), now let’s do the math again:

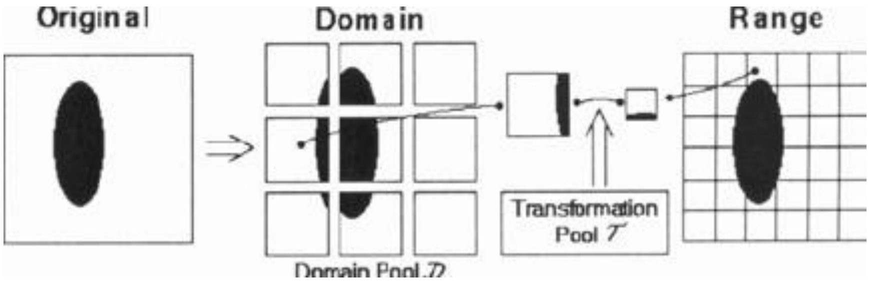
13 digits X 2 bits= 26bits

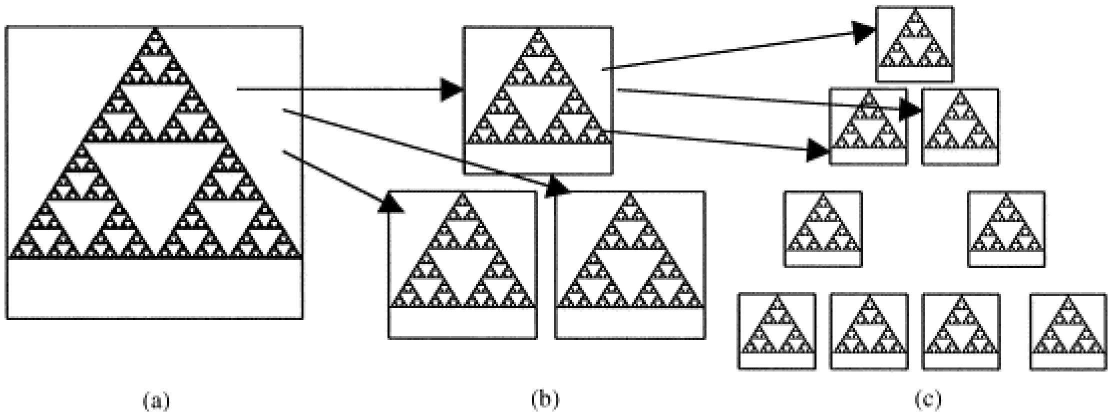
As we understand RLE is a lossless encoding and to estimate the complexity will be based on the compression rate. In other words, the higher the compression rate, the better the encoding works, the smaller the complexity is.

Fractal compression as a measure of processing complexity

Fractal compression is a lossy compression method that uses iterated function systems (IFS) to reduce an image into fractal-based IFS, this algorithm works better in images that have similarities or patters.IFS fractals are more related to set theory than fractal geometry.

With fractal compression, encoding is extremely computationally expensive because of the search used to find the self-similarities. Decoding, however, is quite fast. While this asymmetry has so far made it impractical for real time applications, when a video is archived for distribution from disk storage or file downloads, fractal compression becomes more competitive



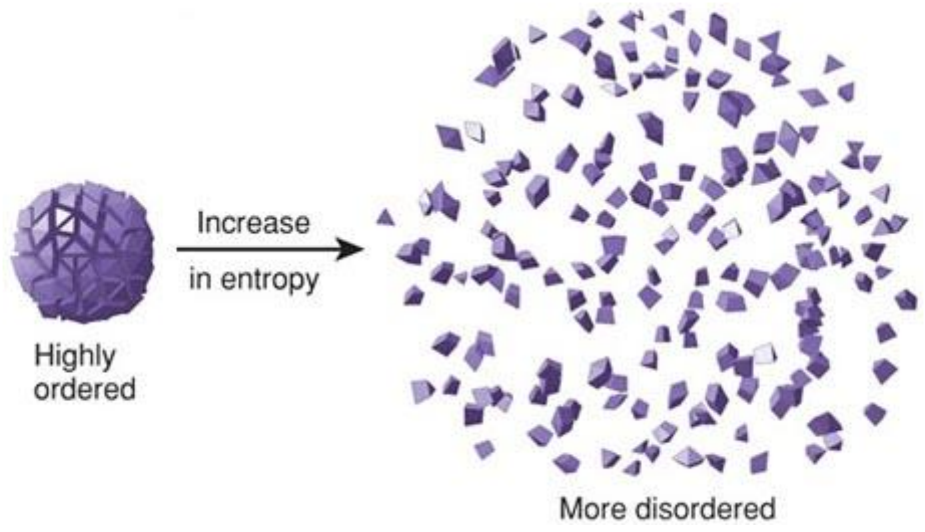


Entropy Measure

If Shannon Entropy is applied to the color values of pixels across an image, it can be used to calculate the unpredictability of an image, thus obtaining a reasonable estimate of the image's visual complexity. Shannon entropy and Compression Ratio (explained before) are expressing fitness values to images , this has been calculated by scanning across the pixels of the image and counting the occurrence of each color. From this, a probability distribution is created, and the below equation is applied

$$H(X) = \sum_{i=1}^{n} p(x_i) log_2 p(x_i)$$

This approach of Shannon entropy as a complexity measure only considers the spread of colors rather than their distribution of the images. So this way of computing may miss important qualities of the image.



Deep Learning Classification with Respect to Complexity

We are going to experiment how complexity can be “extracted” from images using deep learning algorithm and see if we can relate the models with human rated complexity measure.

Deep learning technique

Using Convolution Neural Networks is a must, because those are recommended for image recognition. Furthermore we are going to see swallow and deep nets and pre-trained networks .Every model will be evaluated with datasets

Dataset Description

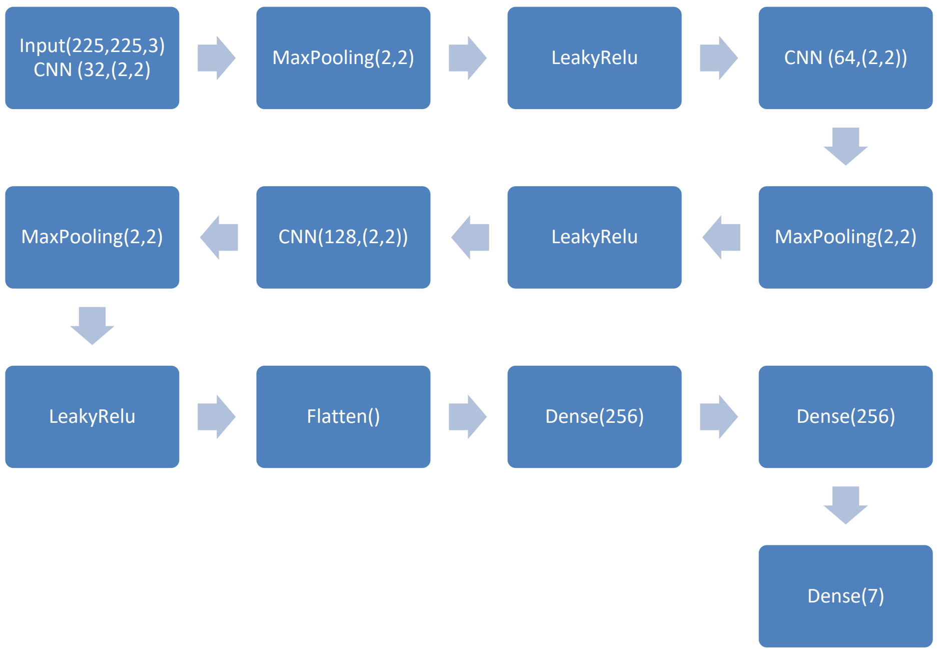
Our **original sampling** is 208 company logos and the augmented dataset will be the same 208 sampling but this time we are going to rotate the images in order to balance our dataset. This means that we are going to rotate only the classes that are not appear that much in our dataset

How are we going to evaluate our results?

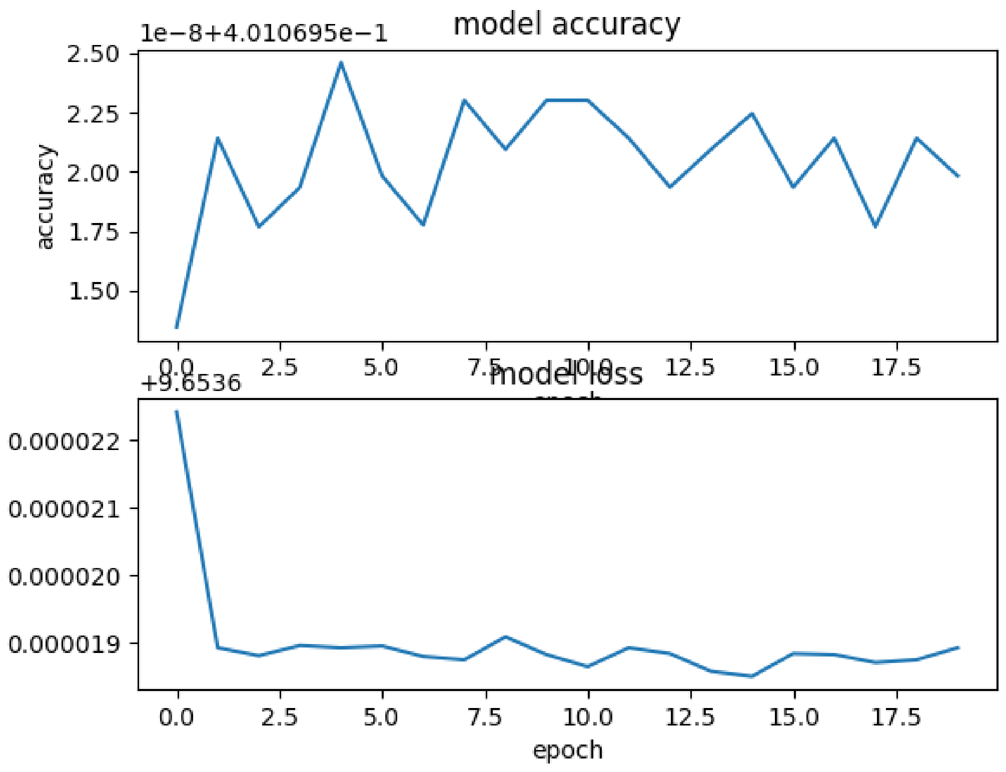
And finally my conclusion is going to be based on confusion matrix and MAE (mean absolute error) to understand if the output is approved or denied.

Experiment and Results

Swallow nets (1-2 Hidden Layer)



Part 1



Accuracy Percentage is:47,61 %

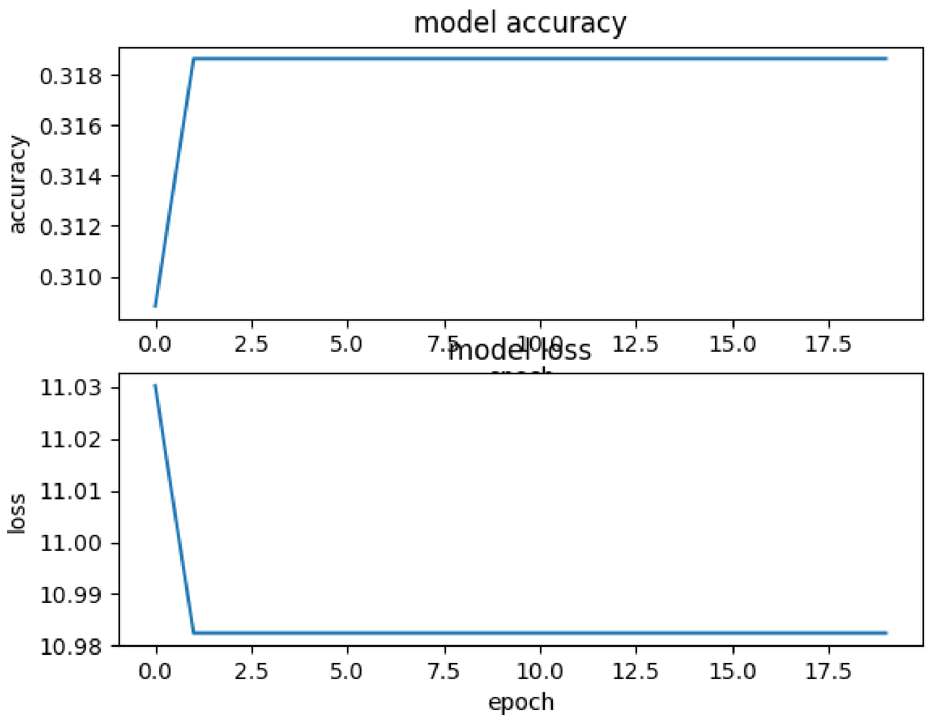
Below is the confusion matrix for the validation of our model

0	**3**	**0**	**0**	**0**	**0**	**0**
0	**10**	**0**	**0**	**0**	**0**	**0**
0	**6**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**
0	**2**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**

MAE is: 0.7142857142857143

In this particular swallow deep net we can be sure that it can classify more than one class, so that 0.47% is not representative of the overall accuracy of the model

Part 2



Percentage Accuracy of the model is: 30,4%

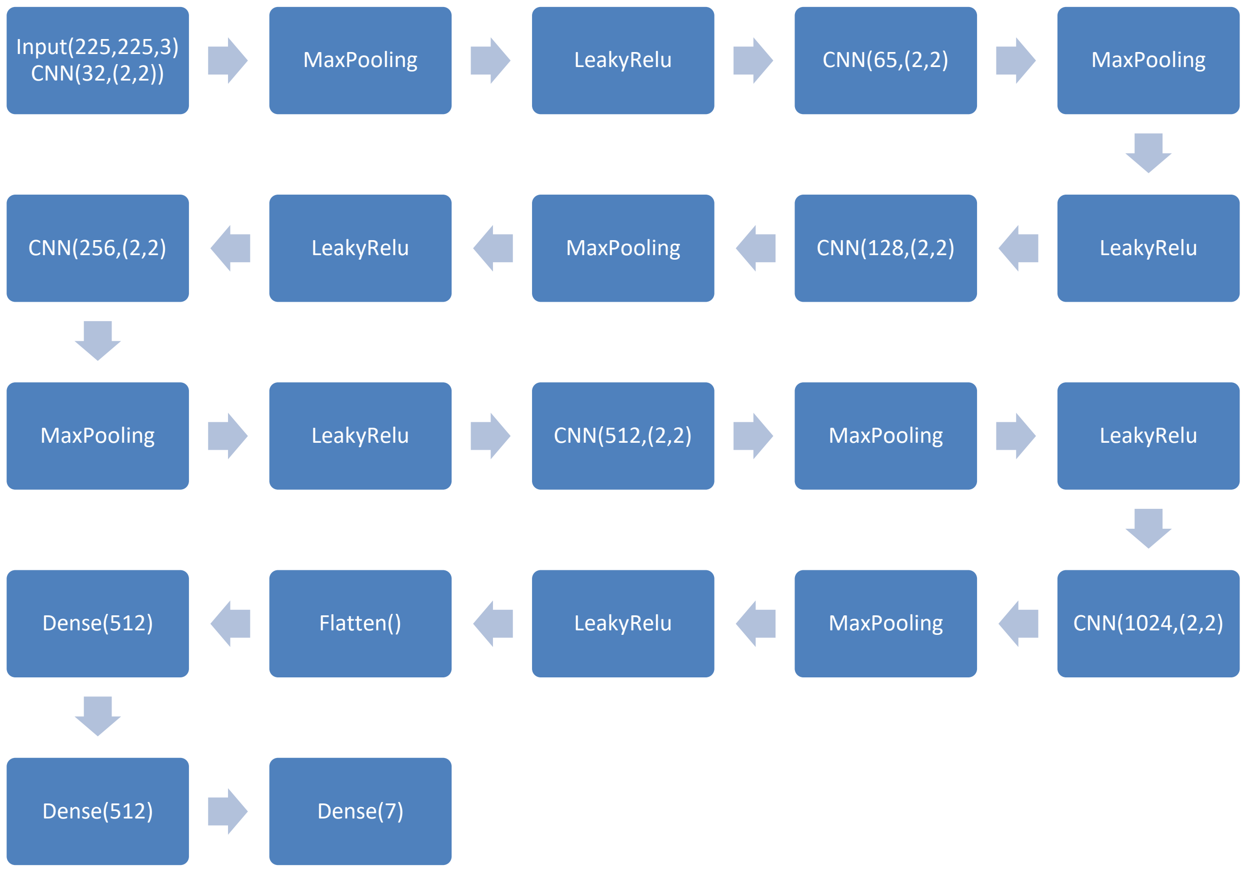
Below is the confusion matrix for the validation of our model

0	**0**	**10**	**0**	**0**	**0**	**0**
0	**0**	**11**	**0**	**0**	**0**	**0**
0	**0**	**14**	**0**	**0**	**0**	**0**
0	**0**	**8**	**0**	**0**	**0**	**0**
0	**0**	**3**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**

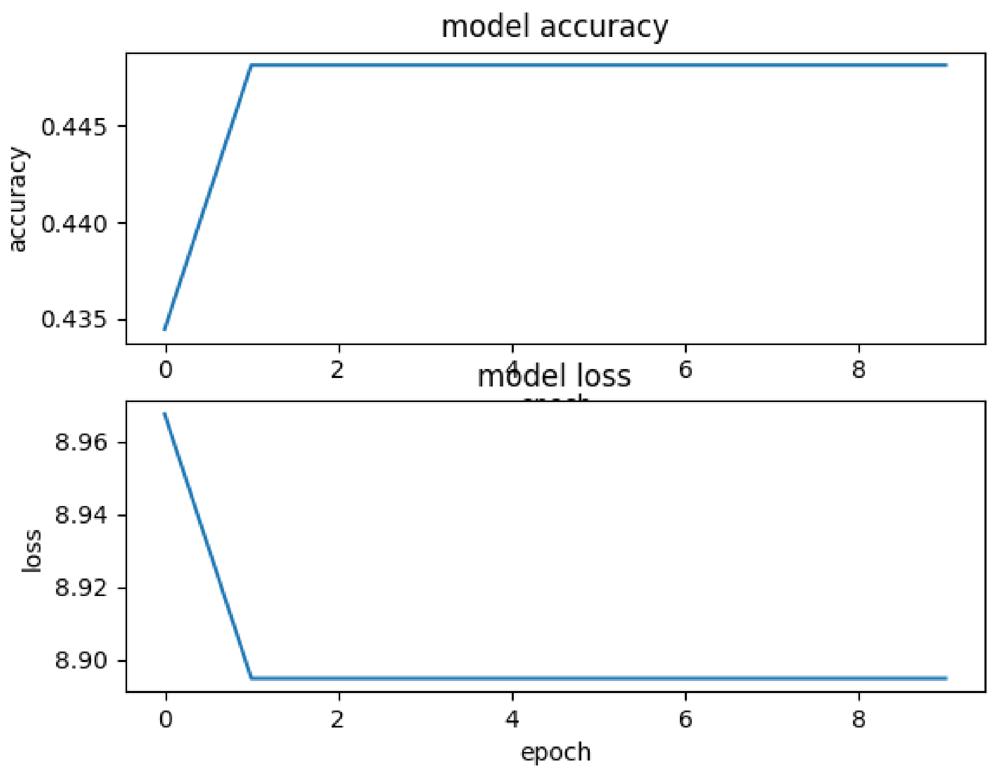
MAE is : 0.9782608695652174

This time the augmented dataset we used made things worse than before, with 30 %

MODEL 2(5 Hidden Layers)



Part 1



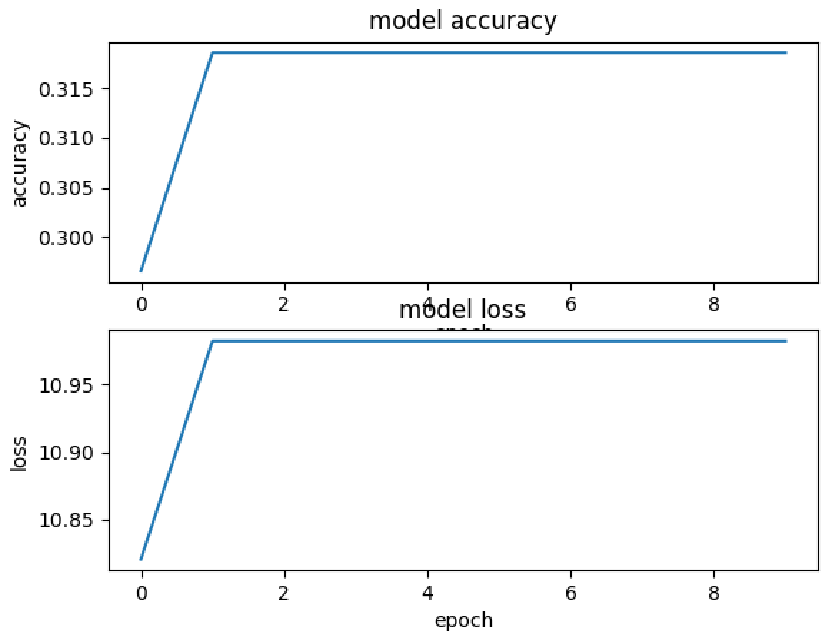
Accuracy percentage is 47,62%

MAE is: 0.7142857142857143

Another model that focuses in one class instead of trying to define other as well, even if they are false

0	**3**	**0**	**0**	**0**	**0**	**0**
0	**10**	**0**	**0**	**0**	**0**	**0**
0	**6**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**
0	**2**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**

Part 2



Accuracy percentage is 30,43%

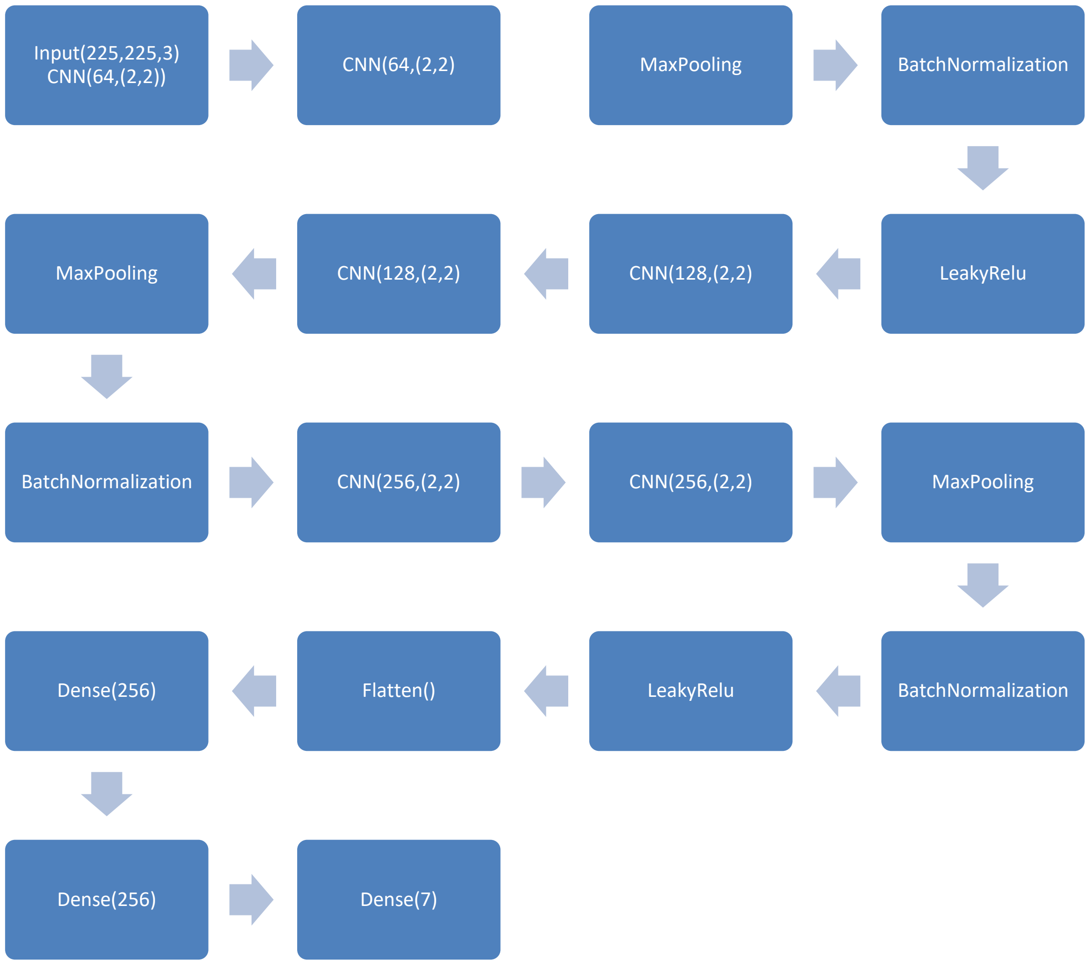
Below is the confusion matrix for the validation of our model

0	**3**	**10**	**0**	**0**	**0**	**0**
0	**10**	**11**	**0**	**0**	**0**	**0**
0	**6**	**14**	**0**	**0**	**0**	**0**
0	**0**	**8**	**0**	**0**	**0**	**0**
0	**2**	**3**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**

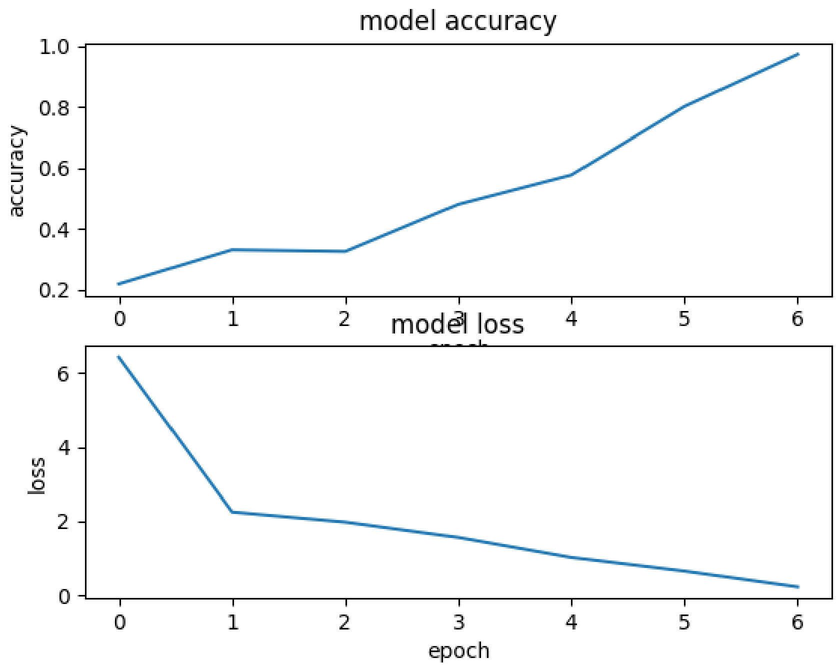
MAE is: 0.9782608695652174

In this model we can see that our model is predicting 2 classes out of 7 with a high value of MAE . The other classes are not having even a false prediction.

Model 3(5 Hidden Layers with Batch Normalization)



Part one



Accuracy Percentage is 23,81%

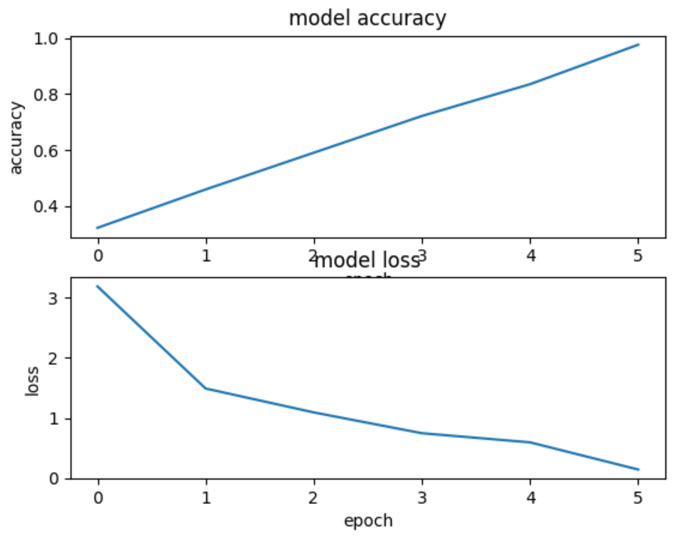
Below is the confusion matrix for the validation of our model

2	**1**	**0**	**0**	**0**	**0**	**0**
5	**13**	**1**	**1**	**0**	**0**	**0**
3	**2**	**0**	**1**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**
1	**1**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**

MAE is: 1.1904761904761905
We can see that it tried to predict 4 classes, but only 2 with success.

In this particular model we can see that the model is learning something but it may needs more data to extract a result.

Part two



Accuracy Percentage is 52,631%

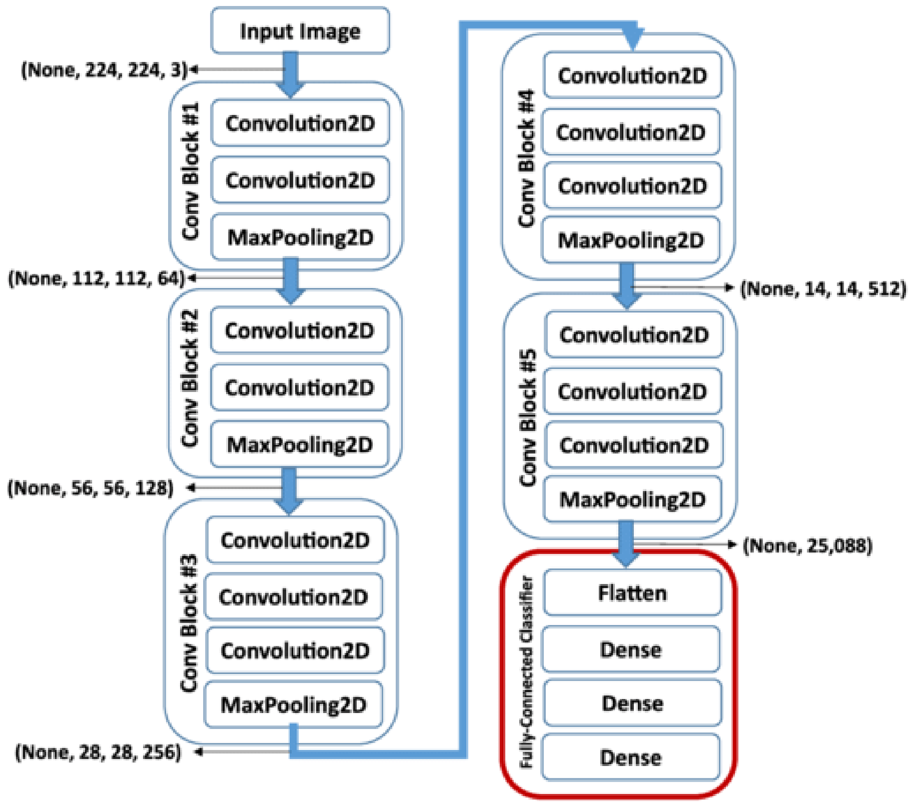
Below is the confusion matrix for the validation of our model

0	**1**	**0**	**0**	**0**	**0**	**0**
0	**17**	**8**	**1**	**0**	**0**	**0**
0	**3**	**10**	**2**	**0**	**0**	**0**
0	**4**	**1**	**3**	**1**	**0**	**0**
0	**2**	**1**	**1**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**	**0**

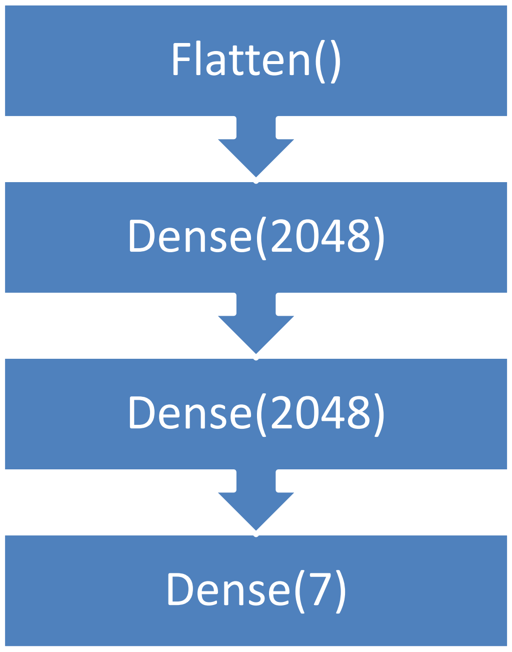
MAE is: 0.7719298245614035

This tried to predict 4 classes,3 classes were predicted with success and the other without. As we can see the Accuracy percentage increased even thought it isn't statistical significant and the Mean Absolute Error dropped.

Pre-Trained



The Fully-Connected Classifier is below



Part one

Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Actual Class	Predicted
0.12	0.07	**0.75**	0.01	0.00	0.01	0.05	4	2
0.00	**0.83**	0.08	0.00	0.00	0.00	0.00	1	1
0.01	0.01	**0.98**	0.00	0.00	0.00	0.00	2	2
0.00	0.01	0.00	**0.98**	0.00	0.00	0.00	4	3
0.00	**0.99**	0.00	0.00	0.00	0.00	0.00	1	1
0.00	0.00	**0.47**	**0.45**	0.07	0.00	0.00	2	2
0.02	0.00	**0.50**	0.00	**0.42**	0.06	0.00	1	2
0.02	**0.88**	0.00	0.09	0.00	0.00	0.01	0	1
0.05	0.28	0.13	**0.52**	0.00	0.01	0.02	1	3
0.11	**0.46**	0.32	0.01	0.09	0.01	0.01	1	1
0.20	**0.60**	0.05	0.14	0.00	0.00	0.00	2	1
0.02	0.00	0.00	**0.97**	0.00	0.00	0.00	2	3
0.14	**0.78**	0.02	0.01	0.01	0.05	0.00	1	1
0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0	1
0.1	0.09	0.20	**0.42**	0.17	0.01	0.01	0	3
0.00	**1**	0.00	0.00	0.00	0.00	0.00	1	1
0.02	**0.64**	0.32	0.00	0.00	0.01	0.01	1	1
0.01	**0.98**	0.00	0.02	0.00	0.00	0.00	2	1
0.32	0.02	**0.39**	**0.22**	0.00	0.05	0.00	1	2
0.00	**0.94**	0.00	0.05	0.01	0.00	0.00	1	1
0.14	**0.69**	0.04	0.14	0.00	0.00	0.00	2	1

The accuracy on this model is: 42, 87%

Confusion matrix cannot be constructed with imagenet because it produces percentage of each the classes so I will try to get the higher value as the activated neuron.
This model achieved 9/21(Accuracy: 42, 87%) predictions.
The strong classes where class 1, with total 7/9 predictions 77, 7% of the right predicted classes. More statistical data is that:

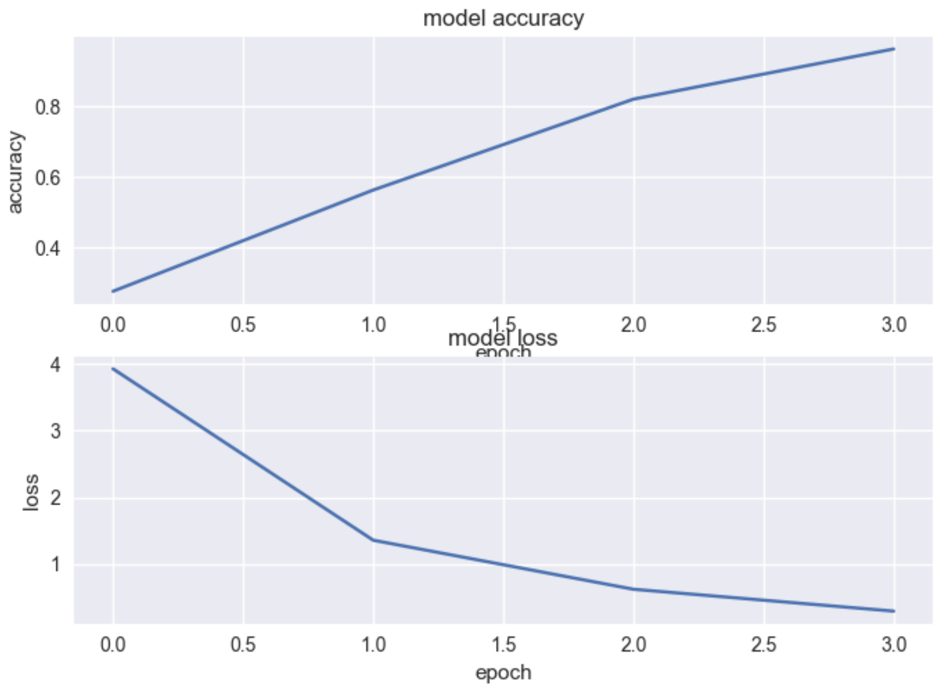
Class 0	0/0
Class 1	7/12
Class 2	2/5
Class 3	0/4
Class 4	0/0
Class 5	0/0
Class 6	0/0

As we see only 3 classes are predicted through our model and only Class 1 has enough prediction to take it statistically significant

Part two

Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Actual Class	Predicted
0.00	0.00	**0.94**	0.01	0.05	0.00	0.00	2	2
0.01	**0.71**	0.24	0.03	0.00	0.00	0.00	1	1
0.02	0.23	0.15	0.03	**0.54**	0.01	0.02	2	4
0.11	0.12	**0.73**	0.03	0.01	0.01	0.00	0	2
0.01	0.00	**0.43**	**0.53**	0.01	0.00	0.01	3	3
0.01	0.05	**0.44**	**0.44**	0.01	0.01	0.01	0	3
0.45	0.08	**0.40**	0.07	0.00	0.00	0.00	0	2
0.57	0.22	0.16	0.01	0.04	0.00	0.00	4	0
0.00	**0.91**	0.04	0.05	0.00	0.00	0.00	0	1
0.03	0.00	0.08	0.01	**0.87**	0.00	0.00	2	4
0.69	0.28	0.00	0.02	0.01	0.00	0.00	1	0
0.26	**0.29**	**0.23**	**0.20**	0.01	0.00	0.01	1	1
0.01	**0.52**	0.02	**0.42**	0.03	0.00	0.00	0	1
0.49	0.07	0.26	0.13	0.02	0.01	0.01	0	0
0.05	**0.63**	0.31	0.00	0.01	0.00	0.00	2	1
0.00	0.01	0.19	**0.42**	0.12	0.26	0.00	1	3
0.05	0.00	**0.78**	0.16	0.00	0.00	0.00	2	2
0.16	**0.24**	0.03	0.05	0.19	**0.30**	0.04	3	1
0.00	0..01	**0.99**	0.00	0.00	0.00	0.00	0	2
0.02	**0.93**	0.01	0.00	0.04	0.00	0.00	1	1
0.02	**0.62**	0.01	0.00	0.21	0.00	0.00	2	1
0.00	**0.95**	0.00	0.05	0.00	0.00	0.00	1	1
0.64	0.00	0.34	0.00	0.01	0.00	0.00	0	0
0.04	0.23	0.00	**0.33**	0.01	0.01	**0.38**	3	3
0.07	0.16	0.21	**0.46**	0.10	0.00	0.00	2	3
0.01	**0.53**	0.35	0.11	0.00	0.00	0.00	1	1
0.00	**0.94**	0.00	0.05	0.01	0.00	0.00	1	1
0.02	**0.81**	0.06	0.09	0.01	0.00	0.00	3	1
0.00	0.01	0.21	0.06	**0.62**	0.09	0.00	4	4
0.17	**0.63**	0.09	0.10	0.01	0.00	0.00	3	1
0.10	0.25	**0.64**	0.00	0.00	0.00	0.00	2	2
0.74	0.03	0.10	0.12	0.01	0.00	0.00	2	0
0.00	**0.88**	0.01	0.11	0.00	0.00	0.00	2	1
0.01	**0.54**	0.08	0.11	0.03	0.22	0.02	3	1
0.00	0.00	**1**	0.00	0.00	0.00	0.00	0	2
0.00	0.01	0.07	0.00	**0.87**	0.00	0.04	4	4
0.07	0.00	**0.92**	0.00	0.00	0.00	0.00	0	2
0.20	**0.40**	0.32	0.08	0.00	0.00	0.00	3	1
0.02	0.02	**0.71**	0.23	0.00	0.02	0.00	1	2
0.09	0.02	**0.40**	**0.41**	0.03	0.03	0.02	1	3
0.00	0.19	0.00	**0.79**	0.01	0.00	0.00	1	3
0.07	0.03	**0.83**	0.05	0.00	0.00	0.01	2	2

0.03	0.05	0.03	**0.80**	0.04	0.01	0.04	3	3
0.01	0.13	0.04	0.24	**0.54**	0.00	0.04	2	4
0.00	0.00	**0.99**	0.00	0.00	0.00	0.00	2	2
0.00	0.01	**0.99**	0.00	0.00	0.00	0.00	2	2



This model achieved 20/46 of the predicted classes. With Accuracy : 43,47%

Class 0	2/5
Class 1	6/16
Class 2	6/12
Class 3	3/8
Class 4	2/5
Class 5	0/0
Class 6	0/0

In the table above, we can see that with the augmented dataset we manage to predict more classes and better distributed, it predicted 5 classes out of 7.

Conclusion

We used a bunch of networks in order to find a conclusion, swallow networks(1-2 hidden layers) as we can see is not that effective in image classification, we need more depth in order to learn features, shapes ,lines, colors. Even though we used 5 hidden layer model we are not able to make it work.

We were able to make it work with a 5 hidden layer model, but this time after every layer to re-establish the new limits of our weights. In this model we saw an accepted result even it is far beyond the ideal. We saw the difference between the original dataset and the augmented.

Later we used a pre-trained network to create the weights and we added our own classifier to predict our classes. We can see that the augmented dataset it predicted 5 classes we **43% percent accuracy**

Future work

Our future is take the weights from aesthetics measures and we will try to classify medical images.

Code Snippets

```
import keras
from keras.models import Sequential,Input,Model
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.layers.advanced_activations import LeakyReLU
from keras import optimizers
from sklearn.metrics import mean_absolute_error
import six.moves.cPickle as pickle
import numpy
from keras.utils import to_categorical

from sklearn.model_selection import train_test_split
import sys
from functions_general import train_valid_augmented_data
import matplotlib.pyplot as plt
import tensorflow as tf
from collections import Counter
from sklearn.utils import class_weight
from keras.callbacks import Callback

from sklearn.metrics import confusion_matrix
```

Keras library has a lot of functions that help you build your layers,set your inputs,help you Normalize your data, implementing activation functions.

Tensorflow library is the core of my model,it's the place that magic happens. In more technical term is the backend system that is doing the computation and runs our model.

Sklearn library is build for the machine learning give a lot of tools and plug and play function that helps with your dataset before implementing them on the model or after the model to show your output in more scientific and understandable way.

Pickle library is producing files that help you store your objects,lists,arrays,etc . With few words we can say that stores data that can be used later.

Matplotlib library is creating plots to represent your output in a graphic way.

Functions_general is a custom library i created to manipulate the data structure or create weights for my labels.

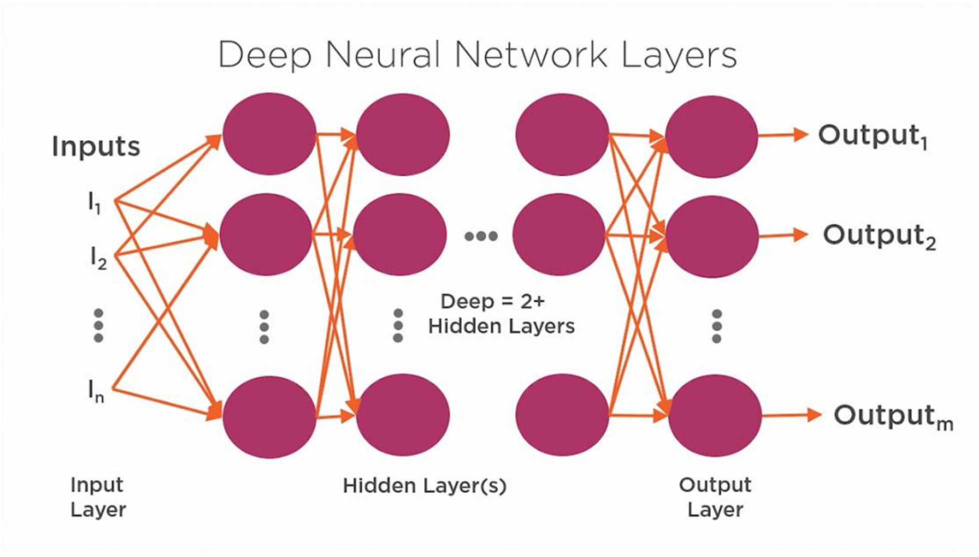
Keras library its very easy to use if you have the right structure on the data.

```
fashion_model = Sequential()
fashion_model.add(Conv2D(64,kernel_size=(2,2),activation='relu',
input_shape=(225,225,3),padding='same'))
```

In the first line we are declare our model what structure it will have,**sequential** is for stacked layers and after declaring our **active neurons** =64 ,**kernel_size**=(2,2)), our **activation** function="relu" and finally the **Input_shape**=(225,225,3) while **padding**='same' means that the output has the same length as the input.

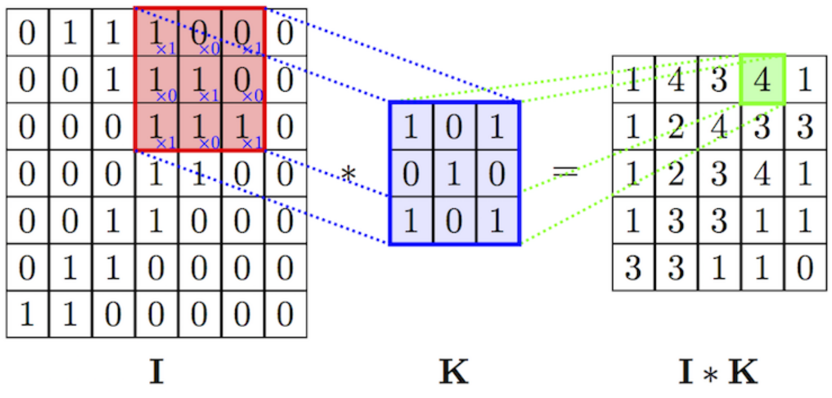
Using the function **add()** we can start stacking layers in our model.Simple right?

Active neurons:Is the number of neurons that is used in the specific layers

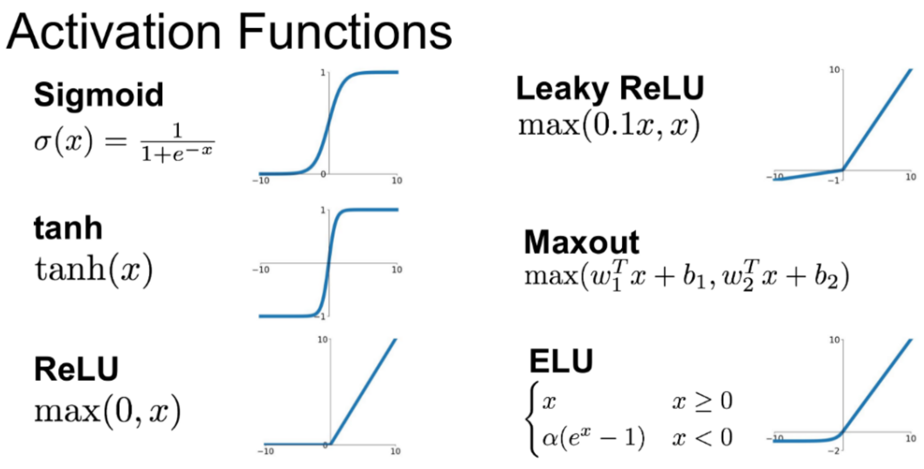


in this example above,we having the first layer with 3 neurons,and all the hidden with 3 neurons.

Kernel: is he convolutional window between our 2d image I and our Kernel, that will produce interesting feature about our image.



activation function: In simple words its the main factor if the neurons will be activated or not. Some of the most known activation functions are shown below.

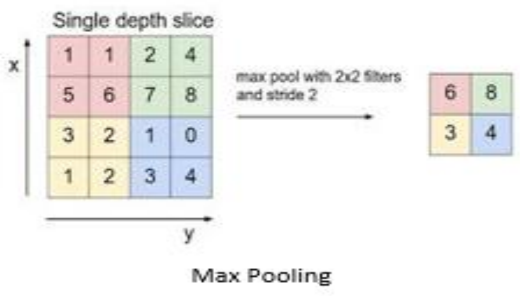


Input_shape: As the name of the variable give away ,it is the shape of our input data. An example is,in my model i use 208 images with 225,225,3 shape. So i am declaring only the shape of one image.

Padding: Means that the output will be the shame as the input.

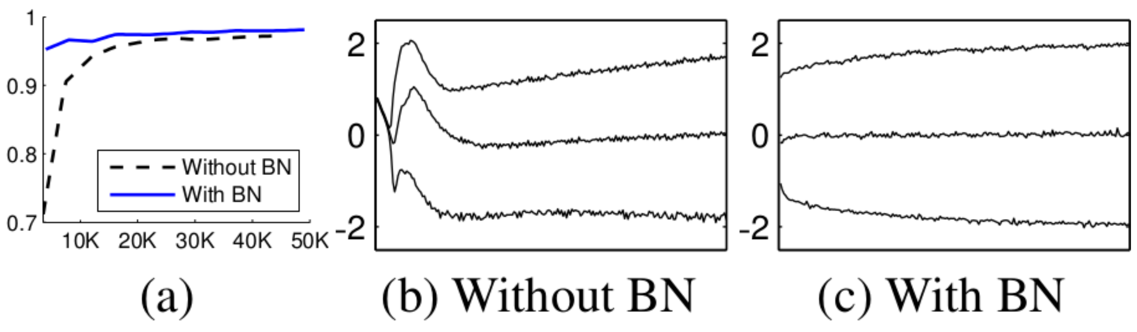
```
fashion_model.add(Conv2D(64, (2,2), activation='relu',padding='same'))
fashion_model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
fashion_model.add(BatchNormalization())
fashion_model.add(LeakyReLU(alpha=0.1))
```

In the first line we are stacking our next layer of our model,after the first layer we are using MaxPooling2D(filter) in this phase ,we are trying to downsampling our data in order to make assumption about our data the (2,2) will run over our array and replace the the starting array with a single value of the max number of that filter.



Later we are adding **Batchnormalization()** in our hidden data this time. But what are we actually doing in this function.

We normalize the input layer by adjusting and scaling the activations. For example, when we have features from 0 to 1 and some from 1 to 1000, we should normalize them to speed up learning. If the input layer is benefiting from it, why not do the same thing also for the values in the hidden layers, that are changing all the time, and get 10 times or more improvement in the training speed.



And finally the **LeakyRelu=Activation** is the same as the activation inside the convolution layer,but now we can change some values for better performance.

```
fashion_model.add(Flatten())
fashion_model.add(Dense(256, activation='relu'))
fashion_model.add(Dense(256, activation='relu'))
fashion_model.add(Dense(7, activation='softmax'))
```

Flatten() is a function that reshapes the data to fit in the vectorized layers later. Below is an example :

```
model = Sequential()
model.add(Conv2D(64, 3, 3,
            border_mode='same',
            input_shape=(3, 32, 32)))
# now: model.output_shape == (None, 64, 32, 32)

model.add(Flatten())
# now: model.output_shape == (None, 65536)
```

Dense() is a simply a layer where each unit or neuron is connected to each neuron in the next layer.

In our last layer we see a dense of 7 neurons and an activation different from the other,this is our classifier where softmax activates or don’t the neurons based on the classes we have.If we got 3 classes we are going to use 3 neurons,if we got 7 classes we are going to use 7 neurons and so on.

```
sgd = optimizers.SGD(lr=0.01)

fashion_model.compile(loss='categorical_crossentropy',optimizer=sgd,
    metrics=['accuracy'])

fashion_model.summary()

with tf.device('/gpu:0'):
     history = fashion_model.fit(train_X, train_label,
     class_weight=class_weight1,batch_size=20,epochs=10,
     verbose=2,callbacks=[EarlyStoppingByAcc()])
```

Optimizer is the like the hero of our model, using the output of the loss function tries to correct the model by playing with the weights that has been created.

In the second line we compile our model the last step before the takeoff(training),we are setting the loss function to use categorical data(7 classes).

Summary is the overview of our model.how many layers it has,how many parameters are trainable and so on.

tf.device('/gpu:0') is the selected device that will do the calculations.

Fit is the function that do the training which takes the train data with their labels(train_X,train_label),

class_weight is used for unbalanced data such as mine and setting the weight of each class based on their numbers

Batch size is the number of data that will take each time for computation

Epochs is how many times will run the the training in order to achieve higher accuracy percentage

Callbacks is a function that is doing something every time an epoch is completed.

Verbose is about showing or not the training process

Configuration

To begin with we need python above 3.5 and i recommend anaconda framework because it has pre-installed a lot of libraries. Let's see now more detailed about the everything you need to install on your machine before running deep learning algorithms in python.

My machine specifications and recommended :

	Own	Recommended
CPU	Intel i5 7th gen 7400-6MB cache	Intel® Xeon® Processor E7-4809 v4 - 20 MB cache
GPU	Nvidia 1050 Ti 4GB	Nvidia Titan X - 12 MB(supports multiple GPU)
RAM	DDR4 - 8GB - 2100MHz	DDR4 - 32 GB - 2100MHz

1.Installing Python 3.5 or Anaconda

using **pip** or **conda** then you can download all the libraries that may missing

2.After the Installation you need to set the environmental variables on your machine

here is a guide of how to do it.

3.Now we are going to see which libraries we need and some extras

Open a command prompt and we start type:

1. pip install keras
2. conda install numpy
3. Download and install graphiz
4. pip install pydot
5. Download and install Cuda
6. pip3 install tensorflow-gpu

These are the libraries you need to run Keras using GPU,after each installation like graphiz or Cuda you need to add these into the environmental variables.

Furthermore, you need to be careful on not using multiple versions of the libraries(tensorflow is case sensitive) because they conflict each other

In tensorflow-gpu there is a documentation on which version you need to match in order to run everything smoothly

References

1. Alexandra Forsythe , Visual Complexity: Is That All There Is?

2. Jurgen Schmidhuber ,The Swiss AI Lab IDSIA ,Deep Learning in Neural Networks: An Overview

3. Hrushikesh Mhaskar and Qianli Liao and Tomaso Poggio ,Learning Functions: When Is Deep Better Than Shallow

4. Li Deng and Dong Yu , Deep Learning: Methods and Applications

5. Yann LeCun and Marc'Aurelio Ranzato ,Atlanta, 2013-06-16,Deep Learning Tutorial

6. Ayon Dey, Machine Learning Algorithms: A Review

7. Dumitru Erhan, Pierre-Antoine Manzagol,Yoshua Bengio,Samy Bengio and Pascal Vincent ,The Difficulty of Training Deep Architectures and the Effect of Unsupervised Pre-Training

8. Yoshua Bengio, Aaron Courville, and Pascal Vincent ,Unsupervised Feature Learning and Deep Learning: A Review and New Perspectives

9. Farzin Yaghmaee, Mansour Jamzad, Introducing A New Method for Estimation Image Complexity According To Calculate Watermark Capacity

10. Penousal Machado, Amílcar Cardoso , Computing Aesthetics

11. Jaume Rigau, Miquel Feixas, and Mateu Sbert ,Conceptualizing Birkhoff's Aesthetic Measure Using Shannon Entropy and Kolmogorov Complexity

12. Jaume Rigau, Miquel Feixas, and Mateu Sbert , Informational Aesthetics Measures

13. Jaume Rigau, Miquel Feixas, Mateu Sbert , Informational Dialogue with Van Gogh's Paintings

14. Daniel L Atkins, Roman Klapaukh, Will N Browne and Mengjie Zhang ,Evolution of Aesthetically Pleasing Images Without Human-In-The-Loop

15. Michael Batty , Robin Morphet , Paolo Masucci ,Kiril Stanilov , Entropy, complexity, and spatial information

16. Mario Ignacio Chacón Murguía, Alma Delia Corral Sáenz and Rafael Sandoval Rodríguez , A Fuzzy Approach on Image Complexity Measure

17. Sandhya Singh and Prof. Dolley Shukla , A Review on Various Measures for Finding Image Complexity

18. S. Khalighy, G. Green , C. Scheepers and C.Whittet , MEASURING AESTHETIC IN DESIGN

19. Daniel Filonik, Dominikus Baur ,Measuring Aesthetics for Information Visualization

20. Honghai Yu and Stefan Winkler IMAGE COMPLEXITY AND SPATIAL INFORMATION

21. Matthieu Perreira da Silva, Vincent Courboulay, Pascal Estraillier, IMAGE COMPLEXITY MEASURE BASED ON VISUAL ATTENTION

22. Dhiraj Joshi, Ritendra Datta, Elena Fedorovskaya, Xin Lu, Quang-Tuan Luong, On Aesthetics and Emotions in Images: A Computational Perspective

23. Jukka Perkio and Aapo Hyv¨arinen ,Modelling image complexity by independent component analysis, with application to content-based image retrieval

24. Renjie Zhou, Chen Yang, Jian Wan , Wei Zhang, Bo Guan , and Naixue Xiong, Measuring Complexity and Predictability of Time Series with Flexible Multiscale Entropy for Sensor Networks
25. ANSI T1.801.03, “Digital transport of one-way video signals – parameters for objective performance assessment,” American National Standards Institute, 1996

26. E. C. Larson and D. M. Chandler, “Most apparent distortion: Full-reference image quality assessment and the role of strategy,” J. Electronic Imaging, vol. 19, no. 1, pp. 011006, 2010

27. R. Cilibrasi and P. M. B. Vit´anyi, “Clustering by compression,” IEEE Trans. Information Theory, vol. 51, pp. 1523–1545, 2005

28. T. Guha and R. K. Ward, “On image similarity, sparse representation and Kolmogorov complexity,” arXiv Computing Research Repository, vol. abs/1206.2627, 2012.

29. J. Perki¨o and A. Hyv¨arinen, “Modelling image complexity by independent component analysis, with application to content-based image retrieval,” in Proc. 19th International Conference on Artificial Neural Networks (ICANN), Limassol, Cyprus, 2009, vol. 2, pp. 704–714.

30. T. M. Cover and J. A. Thomas, Elements of Information Theory, Wiley-Interscience, 2nd edition, July 2006.

31. M. Li, X. Chen, X. Li, B. Ma, and Paul M. B. Vit´anyi, “The similarity metric,” IEEE Trans. Information Theory, pp. 863–872, 2003

32. Matthieur Perreira Da Silva, Vincent Courboulay, Pascal Estraillier-IMAGE COMPLEXITY MEASURE BASED ON VISUAL ATTENTION ,L3i, University of La Rochelle Avenue M. Crépeau 17042 La Rochelle cedex 01 France

33. https://keras.io/