



Escola d'Enginyeria de Telecomunicació i
Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL FINAL DE GRAU

TÍTOL DEL TFG: Popularity Prediction on Instagram using Machine Learning

TITULACIÓ: Grau en Enginyeria de Sistemes de Telecomunicació

AUTOR: Eric Massip Cano

DIRECTOR: Kai-Lung Hua

DATA: 21 de juny del 2017

Títol: Popularity Prediction on Instagram using Machine Learning

Autor: Eric Massip Cano

Director: Kai-Lung Hua

Data: 21 de juny del 2017

Resum

Durant l'últim any, la recerca sobre noves maneres d'aprofitar Deep Learning pel profit de l'ésser humà ha crescut exponencialment. Al mateix temps, la nostra societat està evolucionant en direcció a noves formes de comunicació i interacció social. Instagram és una de les principals cares en aquest canvi en l'evolució humana.

Els sistemes de predicció i la Intel·ligència Artificial són temes que estan explotant ara mateix. Hi ha empreses que estan invertint en reptes i experts per predir aquell producte que l'usuari comprarà, o aquella música que li agradarà, o aquella pel·lícula que voldrà veure. Hi ha una nova branca per seguir que és Social Media. Predir la popularitat de les imatges, quina serà la més popular del dia, quin és el millor tall perquè el teu Selfie sigui més popular...

Aquest projecte tracta de posar aquestes dues realitats juntes. L'objectiu és predir quants likes rebrà un post abans de ser publicat a Instagram. Això ara només seria possible gràcies a Machine Learning. Machine Learning és aquest concepte que podem entrenar ordinadors per identificar patrons i dades, i aleshores utilitzar aquests patrons per predir noves dades. Nosaltres donarem mostres de posts a la nostra màquina perquè pugui trobar patrons i estimar i predir un resultat després d'haver estat donat algun altre input, en base als patrons que la màquina va aprendre.

El sistema constarà de tres seccions:

En primer lloc, la imatge d'entrada es classificarà en una categoria en funció del tema de la imatge utilitzant un model basat en Deep Learning. En aquest projecte tindrem en compte sis categories: Animals, Food, Friends, Landscape, Quote and Selfie.

En segon lloc, la imatge d'entrada es compara amb un conjunt de 200 imatges de la categoria seleccionada que ja tenen una puntuació entre 0 i 1 utilitzant un altre model de nou basat en Deep Learning. Un algoritme farà la comparació i es traduirà en un histograma. El punt màxim de l'histograma serà la puntuació computada de la imatge d'entrada.

En tercer lloc, utilitzarem aquesta puntuació calculada a partir de la segona secció com una variable en una regressió, per tal d'obtenir la predicció final de likes. Altres variables es tenen en compte en aquesta regressió també.

Com es pot veure a la descripció del sistema, sense Machine Learning seria impossible, fins i tot per a un ésser humà, d'identificar tots els patrons necessaris i predir amb precisió noves dades. Generalment, els humans poden crear un o dos bons models a la setmana; Machine Learning pot crear milers de models en una setmana.

Title: Popularity Prediction on Instagram using Machine Learning

Author: Eric Massip Cano

Director: Kai-Lung Hua

Date: June 21st 2017

Overview

In the last year, the research about new ways of using Machine Learning for the human profit has grown exponentially. At the same time, our society is evolving into new ways of communication and social interaction. Instagram is one of the faces of this change in the human evolution.

Prediction systems and Artificial Intelligence are topics that are exploding right now. Companies are investing in challenges and experts to predict that product that a user will want, or that music that he will like, or that film that he will want to watch. There is a new challenging branch to be followed and that is Social Media. Predict the popularity of pictures, which one will be the most popular of the day, which is the best cut for a selfie to be more popular...

This project tries to put these two realities together. The goal is to predict how many likes a post is gonna get before being posted on Instagram. This would only be possible right now thanks to Machine Learning. Machine Learning is this concept that we can train computers to identify patterns and data, and then use those patterns to predict off of new data. We will give samples of posts to our machine so it can find patterns and estimate and predict a result after being given some other input, based on the patterns that it learned.

The system will consist of three sections:

First, the input picture will be classified into one category depending on the theme of the picture using a retrained model based on Deep Learning. In this project we are going to take into account six categories: Animals, Food, Friends, Landscape, Quote and Selfie.

Second, the input picture will be compared with a set of 200 pictures from the selected category that already have a score between 0 and 1 using another retrained model based on Deep Learning. An algorithm will make the comparison and will result in a histogram. The maximum point of the histogram will be the computed score of the input picture.

Third, we will use that computed score from the second section as a variable in a regression, in order to get the final prediction of likes. Other variables are taken into account in this regression as well.

As you can see in the description of the system, without Machine Learning it would be impossible, even for a human being, to identify all the necessary patterns and predict off new data accurately. Humans can typically create one or two good models a week; machine learning can create thousands of models a week.

ACKNOWLEDGEMENTS

First of all, I would like to thank my professor in NTUST Kai-Lung Hua
for this opportunity and his guidance.
To my flatmates, Ougzhan and Tom,
for helping me understand statistics and probability using R.
To my lab mates, Daniel and Jonathan,
for being always there to discuss approaches or new branches of research.
To Diana,
for being the head of #ReplaceMeIfYouCan.
To Siraj Raval,
for his videos and for being my inspiration along this path.
And, as always, special thanks to my mum.

“Indeed we are the deep web. Two options for you, a red pill and a blue pill. If you take the red pill, you'll continue writing functional code with a smattering of if-then statements. You'll painstakingly write rule after rule so your machines can accomplish simple tasks and you won't hear from us again. But if you take the blue pill, you'll learn the superpower of our time, how to teach machines to learn for themselves. And we'll show you just how deep this rabbit hole goes.”, Siraj Raval, 2017

INDEX

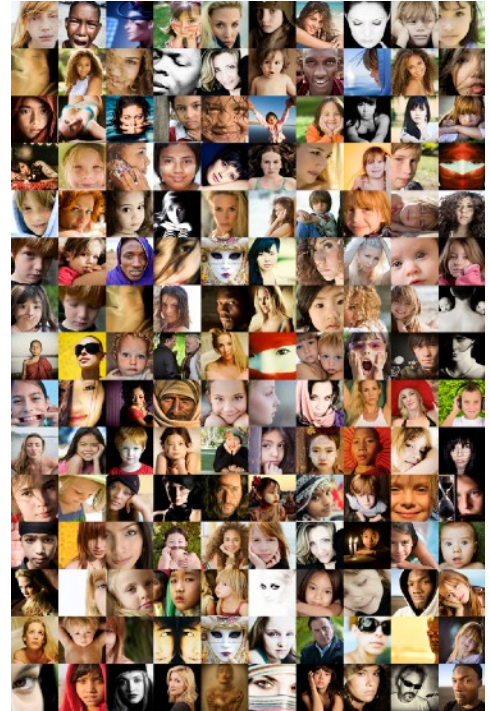
INDEX	1
CHAPTER 1. INTRODUCTION	2
CHAPTER 2. MACHINE LEARNING	3
2.1 Deep Learning	3
2.1.1 Convolutional Neural Networks	4
2.2 Support Vector Machine	5
How does it work?	5
2.2.1 Cross Validation	6
CHAPTER 3. DEVELOPMENT	7
3.1 Category Classification.....	7
3.1.1 Training Data	7
Using InstaBro to prepare the dataset	8
3.1.2 Training the model.....	9
3.1.3 Testing the model	10
3.2 Input Picture Score Computation.....	12
3.2.1 Why a normalized score?.....	12
Using Maximum and Minimum value	12
Computing the Normal Distribution	13
3.2.2 Comparison making pairs with OpenCV	13
3.3 Score into Likes Transformation	16
3.3.1 Support Vector Regression	16
Global User Data Regression	16
Specific User Data Regression	18
CHAPTER 4. RESULTS.....	20
4.1 Category Classification.....	20
4.2 Input Picture Score Computation.....	21
4.3 Score into Likes Transformation	22
Relative Error	29
CONCLUSIONS.....	31
REFERENCES.....	32

CHAPTER 1. INTRODUCTION

In the 21st century, social networks have increased in activity and users in our daily life exponentially. Especially due to the fact that right now, in some places of our planet, it is easier to have a smartphone than tap water at home.

The effect that these social networks have been able to make on our society is enormous. Changing not only lifestyles, but also ways to do business, marketing, journalism, even politics. One of the most significant life-changing issues due to the appearance of these networks is popularity, often hidden in the so-called word *Likes*. Instagram is one of the biggest reasons of this step in our society's evolution.

The main concern for these networks' users, apart from showing their lives to the world, is getting the biggest amount of likes and followers. There are even companies and business individuals the profits of whom depend on the amount of Likes that their posts and publications get.



Our goal is to be able to predict or estimate the quantity of likes that a picture is going to get on Instagram before it is actually posted. In order to solve this problem, we are going to develop a system that will use Machine Learning and Computer Vision to help us achieve this goal.

This document is structured in three parts. In the first part, we will talk about the theoretical background needed to understand the following parts, mainly focused on Machine Learning. The second part will be the development of the system, split into three major sections. And finally the third part will include the results of the testing of the experiment, followed by the conclusions.

The system will use the latest software technology to achieve our goal. We will use Python scripts to write our code. Tensorflow will be the library in charge of training and predicting using Deep Learning. OpenCV will be the library in charge of the computer vision part of the system, comparing, resizing images, etc. Bokeh will be the library in charge of plotting histograms and graphs. And finally Sci-Kit Learn, the library in charge of running the regressions.

CHAPTER 2. MACHINE LEARNING

Before talking about Deep Learning, it is important to present first the term Machine Learning. Machine Learning is a vast field that covers many specialties and is expanding extremely rapidly.

A good definition for this field was made up by Arthur Samuel way back in 1959: 'Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.' With Machine Learning, we can train a computer so that it can learn from some data and predict an output from the system after some input is send into the system, or predict future outputs after analyzing the patterns of the system.

2.1 Deep Learning

Deep Learning is a class of machine learning algorithms that use a cascade of many layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input. The algorithms may be supervised or unsupervised and applications include pattern analysis (unsupervised) and classification (supervised).

- **Supervised machine learning:** The program is “trained” on a pre-defined set of “training examples”, which then facilitate its ability to reach an accurate conclusion when given new data¹.
- **Unsupervised machine learning:** The program is given a bunch of data and must find patterns and relationships therein¹.



Fig. 1, Machine Learning tries to emulate the human brain with a computer

¹ Nick McCrea, 'An Introduction to Machine Learning Theory and Its Applications: A Visual Tutorial with Examples', 2014

2.1.1 Convolutional Neural Networks

In machine learning, a Convolutional Neural Network (CNN) is a type of neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex. Individual cortical neurons respond to stimuli in a restricted region of space known as the receptive field. The response of an individual neuron to stimuli within its receptive field can be approximated mathematically by a convolution operation².

CNN's have a wide range of applications, including image and video recognition, recommender systems and natural language processing. We will use the already trained Inception v3 network for the first phase of our system as we will see later.

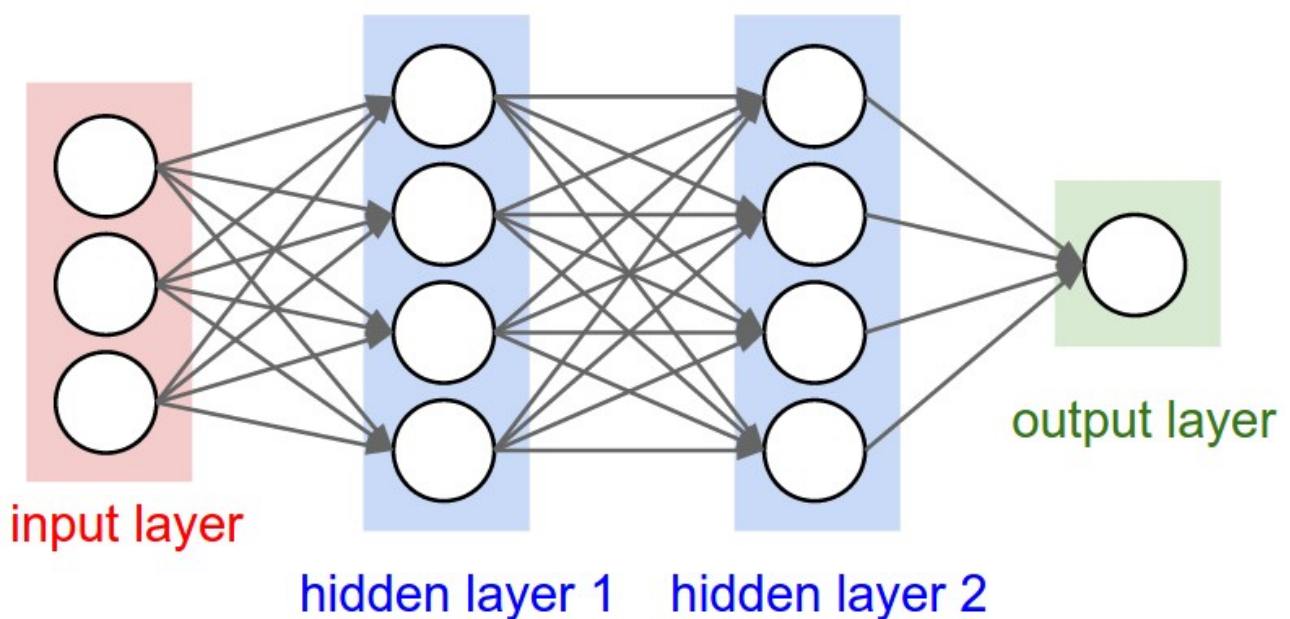


Fig. 2, Example of a regular 3-layer neural network.

Every blank circle we see in figure 2 represents one of the neurons we were talking about earlier. Thanks to these neurons, we have this sequence of layers that conform our neural network. Each of these layers will receive data from the neurons from their previous layer and will compute the output of neurons that will be sent to the next layer.

There are three main types of layers to build convolutional networks: Convolutional Layer, Pooling Layer and Fully-Connected Layer. We will stack these layers to form a full convolutional network architecture. The position of every layer in the stack, as well as the quantity of layers of each type, depends on the purpose of the model you want to train.

² "Convolutional Neural Networks (LeNet) – DeepLearning 0.1 documentation". *DeepLearning 0.1*. LISA Lab. Retrieved 31 August 2013.

2.2 Support Vector Machine

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well, as we can see in Fig. 3.³

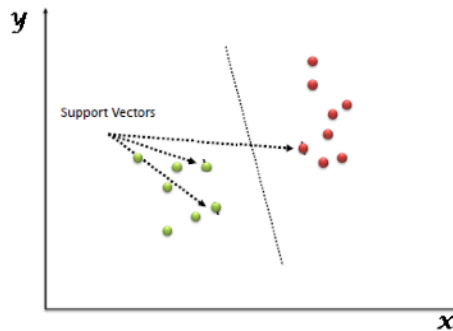


Fig. 3, Example of 2 classes divided by a hyper-plane

How does it work?

As described before, SVM consists of segregating into classification classes using hyper-planes. The question is which is the hyper-plane that separates better the classification classes? Let's see two examples:

In Fig. 4, we can see two hyper-planes that separate two classes in two different ways. We could think that B is better because it has higher margin, but SVM doesn't consider the margin, it just considers which one classifies the classes better, so it would choose A over B.

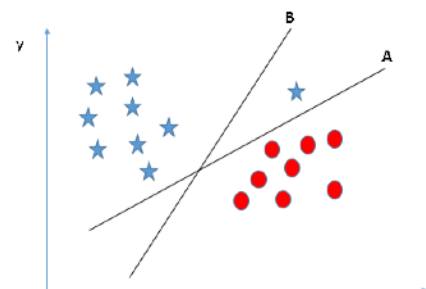


Fig. 4, Example of Linear Regression^x

In Fig. 5, we see a very difficult problem to solve with a linear hyper-plane. We would need another type of kernel to do this. We could use 'rbf' this time, as we will do with the regressions in section 3.

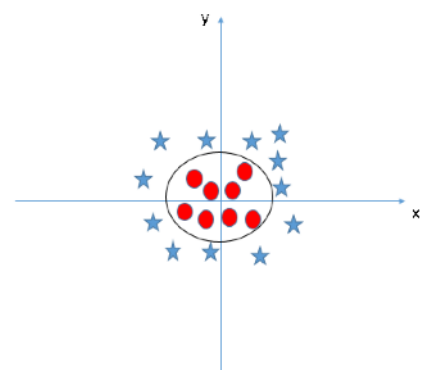


Fig. 5, Example of 'rbf' kernel

³ Sunil Ray, *Understanding Support Vector Machine algorithm from examples*, analyticsvidhya.com, 2015

2.2.1 Cross Validation

Cross validation is a model evaluation method that is better than residuals. The problem with residual evaluations is that they do not give an indication of how well the learner will do when it is asked to make new predictions for data it has not already seen. One way to overcome this problem is to not use the entire data set when training a learner. Some of the data is removed before training begins. Then when training is done, the data that was removed can be used to test the performance of the learned model on 'new' data. This is the basic idea for a whole class of model evaluation methods called *Cross Validation*.

K-fold cross validation is one way to improve over the holdout method. The data set is divided into k subsets, and the holdout method is repeated k times. Each time, one of the k subsets is used as the test set and the other $k-1$ subsets are put together to form a training set. Then the average error across all k trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set $k-1$ times. The variance of the resulting estimate is reduced as k is increased. The disadvantage of this method is that the training algorithm has to be rerun from scratch k times, which means it takes k times as much computation to make an evaluation. A variant of this method is to randomly divide the data into a test and training set k different times. The advantage of doing this is that you can independently choose how large each test set is and how many trials you average over.⁴

Fig. X is an example of a K-fold cross validation where $k = 7$. The green boxes would be the training data for every k . The grey boxes would be the testing data for every k . In every split or k , the model is trained with the 'new' training data (the green boxes) and the new testing data (the grey box). After that, we compute the accuracy score of the model and save that score. As soon as we have the 7 scores, we sum all of them and divide between 7, that gives the cross validation accuracy score.

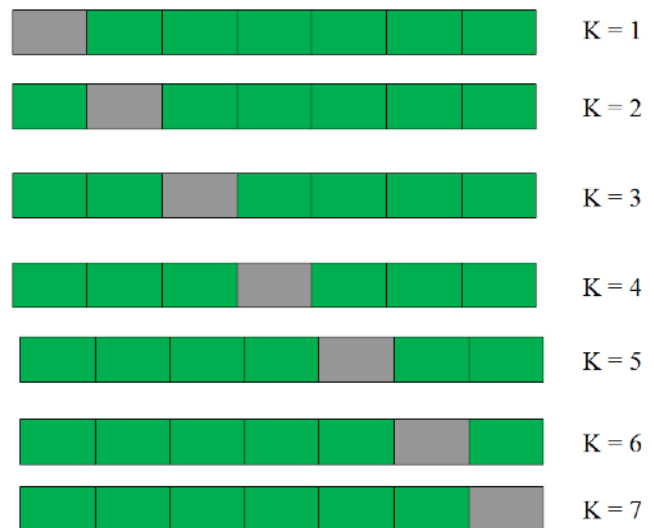


Fig. 6, Example of 7-fold cross validation

In the third section of our system, we will have to do a regression with not many points of data. Cross validation helps us know the accuracy of our model much more accurately than just computing the score. As I said, it is very useful when you do not have many points of data because you can train your model with all the training data you have, you do not have to split into training and testing data.

⁴ Schneider J., "Cross Validation", 1997

CHAPTER 3. DEVELOPMENT

The development of the system to predict the amount of likes that a picture is going to get before being posted on Instagram will be split in three sections:

1. **Category Classification**
2. **Input Picture Score Computation**
3. **Score into Likes Transformation**

3.1 Category Classification

3.1.1 Training Data

After analyzing the behavior of a great variety of users, I realized that there was a correlation between the style of every post and the resulted likes that the post would get. So we decided to classify the pictures into different categories. Furthermore, this would help us out in the second section (Input Picture Score Computation) as we will be able to compare pictures of the same category and, as a result same style, to compute a score.

Our decision was to split into 6 categories:

- **Animals**
- **Food**
- **Friends**
- **Landscape**
- **Quote**
- **Selfie**

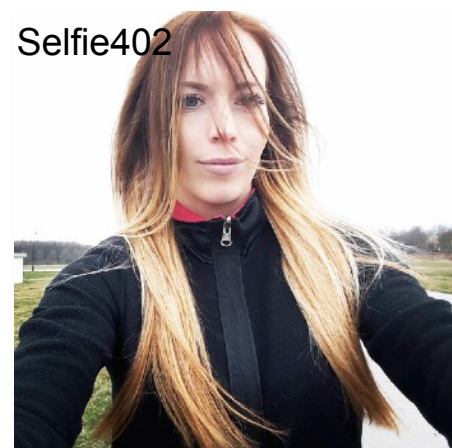
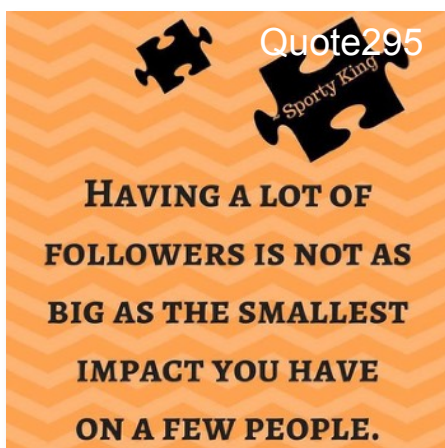
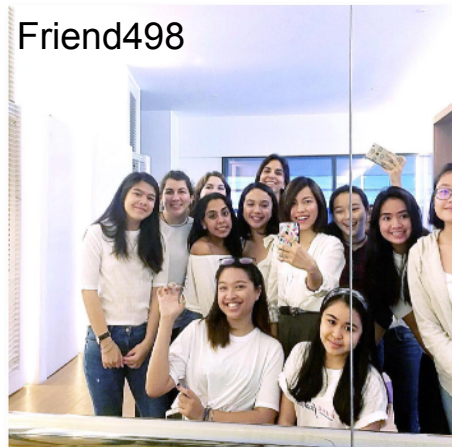
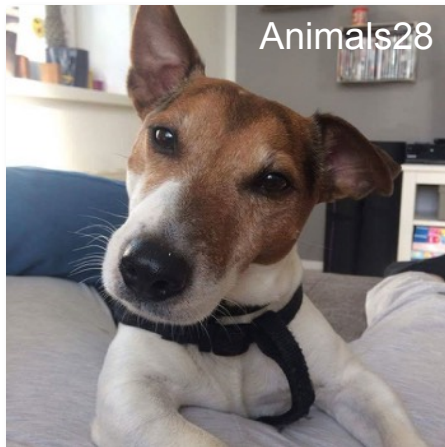
Obviously, this was our choice but we could include more and more categories in order to increase the accuracy of the system, being careful not to overfit it.

This is the first step to develop the system and, as we are dealing with Big Data Science, we need to have a dataset to train our model. Apart from the image file, more data about the post will be needed, amount of Likes, Posts, Followers and Following, of the owner of the picture.

Using InstaBro to prepare the dataset

We agreed to have at least 1.000 pictures of each category, that would give us a 6.000 photos dataset. I used a tool called InstaBro⁵ to complete this task.

With this tool, I was able to filter posts using a hashtag, for example, I would ask InstaBro to show me the latest 10.000 posts with the #animals. After that, I would choose the ones that I want and, with the most important feature of this tool, I can export to a CSV file all the data of every post as well as download all the selected posts into the desired folder.



⁵ InstaBro, all rights reserved to Boris Karulin

3.1.2 Training the model

At this point, we have our dataset with 6.000 photos ready. It is time to train the model, but training a model from scratch would mean using a lot of computing effort, optimizing hyper parameters, etc. Instead, we are going to do transfer learning here. Transfer Learning means we are starting with a model that has already been trained on another problem and then, we will be retraining it on our problem.

We are going to use the Inception v3 network. Inception v3 is trained for the ImageNet Large Visual Recognition Challenge using the data from 2012, and it can differentiate images between 1.000 different classes, like basketball, dog or car. We will use this same network, but retrain it to tell apart our six categories.

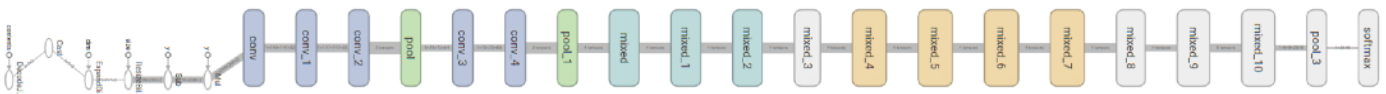


Fig. 7, Simplified picture of the layers' stack of the inception network

As we can see in Fig. 7, the architecture of the network is very complex, therefore, these layers are not only pre-trained, but also very valuable at finding and summarizing information that will help classify most images.

We will use TensorFlow to retrain the Inception v3 network and classify pictures depending on our six categories. What we are really doing is retraining only the last layer of the stack, called 'fully-connected layer' or 'Bottleneck'. While the retraining script is running, a series of step outputs appear:

- **Training accuracy** shows the percentage of the images used in the current training batch that were labeled with the correct class.
- **Validation accuracy** is the precision (percentage of correctly-labelled images) on a randomly-selected group of images from a different set.
- **Cross entropy** is a loss function that gives a glimpse into how well the learning process is progressing (lower numbers are better here).

As you might be wondering, we have not separated our dataset into training data and testing data. That is because, when retraining, TensorFlow gathers some random data from our dataset and trains the model over that, getting a training accuracy result. Then, it tests the model on the rest of the data it did not gather for the training, getting a validation accuracy result. This is done for every step so it is a trustful accuracy result.

If the train accuracy is high but the validation accuracy remains low, that means the network is overfitting and memorizing particular features in the training images that aren't helpful more generally.


```

2017-05-16 04:19:10.096765: Step 3960: Train accuracy = 98.0%
2017-05-16 04:19:10.096852: Step 3960: Cross entropy = 0.137409
2017-05-16 04:19:10.411349: Step 3960: Validation accuracy = 93.0% (N=100)
2017-05-16 04:19:13.488840: Step 3970: Train accuracy = 98.0%
2017-05-16 04:19:13.488934: Step 3970: Cross entropy = 0.118002
2017-05-16 04:19:13.751980: Step 3970: Validation accuracy = 94.0% (N=100)
2017-05-16 04:19:16.350452: Step 3980: Train accuracy = 96.0%
2017-05-16 04:19:16.350526: Step 3980: Cross entropy = 0.134668
2017-05-16 04:19:16.609771: Step 3980: Validation accuracy = 93.0% (N=100)
2017-05-16 04:19:19.370856: Step 3990: Train accuracy = 97.0%
2017-05-16 04:19:19.370934: Step 3990: Cross entropy = 0.127454
2017-05-16 04:19:19.660670: Step 3990: Validation accuracy = 97.0% (N=100)
2017-05-16 04:19:22.061232: Step 3999: Train accuracy = 98.0%
2017-05-16 04:19:22.061310: Step 3999: Cross entropy = 0.088377
2017-05-16 04:19:22.342595: Step 3999: Validation accuracy = 96.0% (N=100)
Final test accuracy = 94.3% (N=579)

```

Fig. 8, step 3960 to 3999 retraining the Inception v3 network for our category classification



As we can see at the bottom of Fig. 8 and zoomed in in Fig. 9, the final test accuracy combining the 4000 steps is 94.3%, which is a very good result.

Final test accuracy = 94.3% (N=579)

Fig. 9, Final test accuracy of our retrained model

3.1.3 Testing the model

Now that our model is ready, let's test it with some random data from the web. None of the pictures in the next table are from the training data and they were selected randomly.

URL	Picture	Result
https://cdn.playbuzz.com/cdn/3231fb60-6394-4b01-999b-1eca59833ce5/043f003e-f31a-4e55-9637-942e61fc6b7e.jpg		<p>animalscategory (score = 0.68803)</p> <p>friendscategory (score = 0.13242)</p> <p>foodcategory (score = 0.06125)</p> <p>landscapecategory (score = 0.05454)</p> <p>quotecategory (score = 0.04675)</p> <p>selfiecategory (score = 0.01701)</p>
https://bocao.com.do/uploads/img21-03-2015-1931595193.jpg		<p>foodcategory (score = 0.99432)</p> <p>animalscategory (score = 0.00430)</p> <p>landscapecategory (score = 0.00079)</p> <p>selfiecategory (score = 0.00027)</p> <p>quotecategory (score = 0.00020)</p> <p>friendscategory (score = 0.00011)</p>



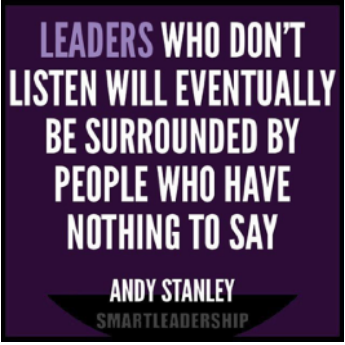

URL	Picture	Result
http://cdn.playbuzz.com/cdn/f5d11eff-a65e-43c2-92f6-70f6277bc753/78ee1799-5bac-45fc-8ba0-8c14a74c08d9.jpg		<pre> friendscategory (score = 0.53461) selfiecategory (score = 0.30900) landscapecategory (score = 0.11550) animalscategory (score = 0.02283) quotecategory (score = 0.01704) foodcategory (score = 0.00103) </pre>
https://lh3.googleusercontent.com/oPgTLhxBCyxNuaeLi7evXS Aeh-_X46oerOoFBSC4Gs1qI-_NDiwAcR1HDcdN-1WGm_klrf_ShR_AVnqZuHbHUrY=s1024		<pre> landscapecategory (score = 0.97403) animalscategory (score = 0.01904) selfiecategory (score = 0.00440) foodcategory (score = 0.00096) quotecategory (score = 0.00079) friendscategory (score = 0.00078) </pre>
https://s-media-cache-ak0.pinimg.com/736x/89/f4/5b/89f45bb8db60d9727f915fcb6bb70658.jpg		<pre> quotecategory (score = 0.99987) selfiecategory (score = 0.00005) landscapecategory (score = 0.00004) friendscategory (score = 0.00003) foodcategory (score = 0.00001) animalscategory (score = 0.00000) </pre>
https://s-media-cache-ak0.pinimg.com/736x/6e/c9/48/6ec9488ecd46d37fda7a7e898b7f6393.jpg		<pre> selfiecategory (score = 0.99335) friendscategory (score = 0.00639) quotecategory (score = 0.00010) landscapecategory (score = 0.00009) animalscategory (score = 0.00006) foodcategory (score = 0.00002) </pre>

Fig. 10, Table testing the results for each category with the retrained model

As we can see in the third column of Fig. 10, the friends category and the selfie category sometimes have some difficulties because they are similar, but almost all the time the model gets it right. Something similar happens when we are dealing with a picture of a person and an animal, it is difficult or not so clear even for us humans to classify those kind of pictures.

3.2 Input Picture Score Computation

The idea of this second part of the system is, after we have classified the input picture into one of our categories, we are going to compute a normalized score between 0 and 1 for this picture.

3.2.1 Why a normalized score?

Well, one of the biggest problems that we had to face is that the amount of Likes that a picture will get is extremely affected by the user variables factor, such as Posts, Following and especially Followers. This means that we had to think of a way to extract that popularity factor out of the picture, so we had to get a normalized score totally non-dependent on the user variables. For example:

We have a user A with 100 followers who posts a picture which gets 10 likes.

Another user B with 100.000 followers posts a picture which gets 10.000 likes.

Clearly, the user variables factor is affecting the resulted likes that both these users get. It does not mean that the post of user B is much better than the one of user A. In fact, the estimated score for both pictures should be the same, as we can see if we divide the resulted likes between their followers:

$$\text{User A} \rightarrow \frac{10}{100} = 0,1 \qquad \text{User B} \rightarrow \frac{10000}{100000} = 0,1$$

The relation is the same, although we cannot use this formula for all of our pictures because every comparison of users have different magnitudes. So we had to think about other ways to normalize the scores of each picture of our training data.

Using Maximum and Minimum value

The first way I used was getting the info of the most recent 50 photos posted by that user, exporting the CSV file from InstaBro. Following, I computed the Minimum and Maximum value of likes inside this 50 photo sack and using the next formula to determine the normalized score between 0 to 1 for the selected picture:

$$\text{Score} = \frac{\text{Likes} - \text{Minimum}}{\text{Maximum} - \text{Minimum}}$$

Computing the Normal Distribution

The second way I used was also getting the info of the most recent 50 photos posted by that user, exporting the CSV file from InstaBro. After that, I calculated the Average and Standard Deviation values for this 50 photo sack. Finally, I computed the Normal Distribution using the accumulated distribution form to get a normalized score between 0 to 1 for the selected picture.

3.2.2 Comparison making pairs with OpenCV

Now that we had two ways of normalizing these scores, I used both of them on approximately 200 pictures from each of our categories as training data. So afterwards, we will be able to test two different approaches to see which one works better.

Once the input picture is classified into one category in the first section of the system, the input picture is going to be compared with each and every photo that we gathered from that category and computed a normalized score.

In order to this, we paired the 200 pictures of each category between every one of them. If the score of the left picture was higher than the right picture, we would classify it in the class 1 folder; if the score of the left picture was lower than the left picture, we would classify it in the class 0 folder.

After that, we had two folders for every category, one for the MaxMin values and another one for the NormDist values. Inside each one of those, there were two more folders which would be the classes, 1 and 0. Both folders contained the total quantity of pairs that would sum up to 200 photos x 199 photos = 39600 pairs (199 pictures to compare because obviously, we don't compare a picture with itself). We used the same technique as we did in the first section, we retrained the last layer of the Inception v3 network with this training data. We finally had 12 models, 2 per category.

	Animals	Food	Friends	Landscape	Quote	Selfie
MaxMin Accuracy	76,3 %	75,7 %	75,8 %	73,1 %	68 %	74,2 %
NormDist Accuracy	76,6 %	77,4 %	75,8 %	72,3 %	67,6 %	75,9 %

Fig. 11, Accuracy test results after retraining the 12 models, two per category, one for MaxMin score values and one for NormDist values

Now considering that we have an input picture to predict, the idea consists of making pairs with every picture from the comparison set of the category. For every pair, our already retrained model will classify it into class 1 or class 0. Class 1 means input picture is better, class 0 means compared picture is better. So if we get class 1, we will add one point to every 2-decimal value over the score of the compared picture. If it classifies class 0, we will add one point to every 2-decimal value below the score of the compared picture.

After making pairs and classifying the input picture with the 200 pictures of the comparison set of the category, this will build a histogram and the maximum point of that histogram will be considered the score for the input picture.

Let's see this with an example:



Input picture



Food1
Score = 0,5



Food2
Score = 0,75



1

Class 1, the score of Food1 is 0,5 so we will add one point to every 2-decimal value, meaning adding one point to 0,5 over 0,5, 0,51, 0,52, 0,53, etc.



0

Class 0, the score of Food2 is 0,75 so we will add one point to every 2-decimal value below 0,75, meaning adding one point to 0,74, 0,73, 0,72, 0,71, etc.

After going over the 200 comparison set pictures of the selected category, we will have a histogram with a maximum point that we will use as the computed score for the input picture.

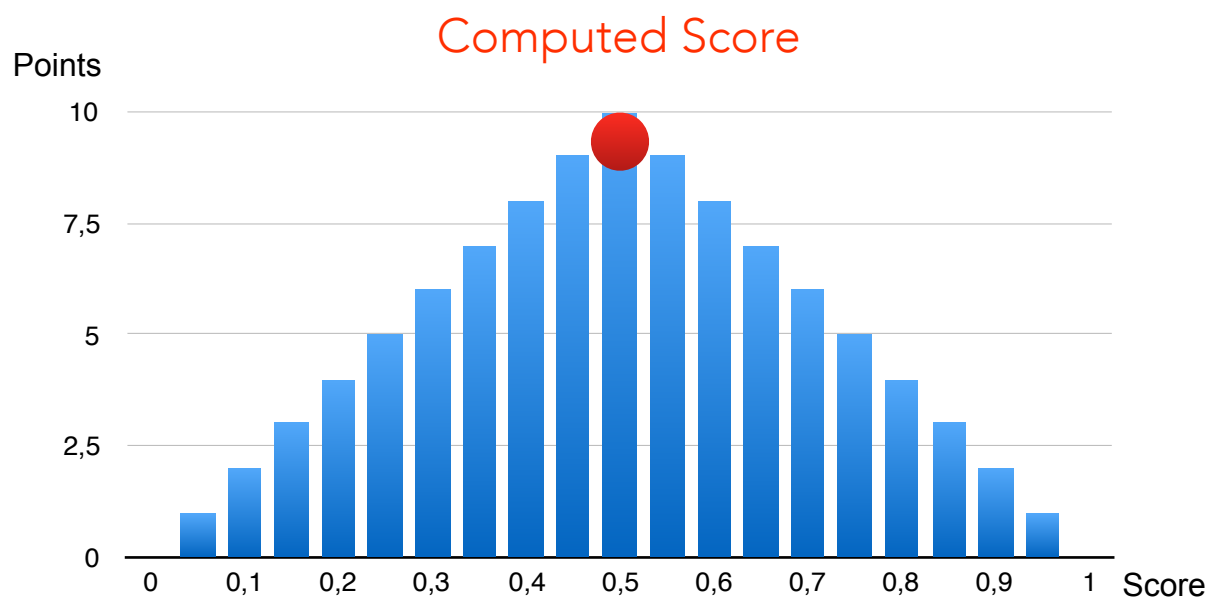


Fig. 12, Histogram after getting the computed score

3.3 Score into Likes Transformation

This is the last section of our system, we have a normalized score for the input picture and we just have to convert it into an amount of Likes, depending on the variables of the user who is trying to post this picture.

3.3.1 Support Vector Regression

We are going to do a regression using the SVR function in the Scikit-Learn library for Python. This way we can give some data to the model and it will predict a resulted quantity of Likes.

Again, we have two approaches for doing so, we will go over both of them.

Global User Data Regression

If we add all the pictures' data that have a normalized score from every category into a CSV file, we will end up having more than 1.200 rows with the name of the picture, quantity of Likes, Posts, Followers and Following.

We can use all that data to train a model using the SVR function. If the model is accurate, it will be able to predict the amount of Likes just introducing the normalized score of the input picture that we got from the previous section, the quantity of Posts, Followers and Following. I rounded the Likes column so that for 33 Likes, it would say 30; and for 158, it would say 160. That way it would be easier for the model to give good results because there would be less labels.

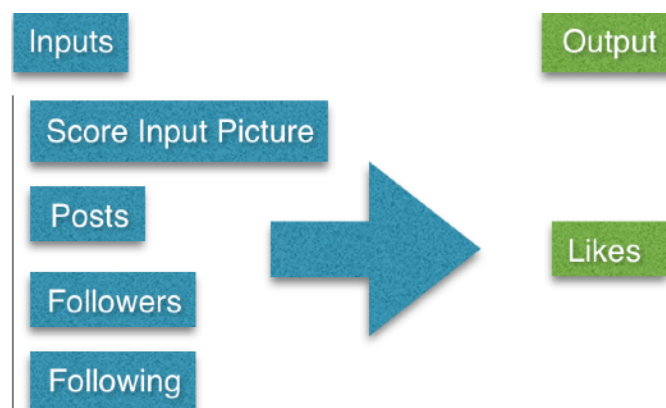


Fig. 13, Schema about our support vector machine

I tried a few kernels for training the model, but clearly the best one was 'rbf'. I optimized the model playing with the hyper parameters but the results were not very good. The best results I got were using the parameter $C = 8.000$ and $\gamma = 1^{-8}$.

As we can see in Fig. 14, the results are not good, 0.38 of R^2 is very low. We can keep this model and test it as well but it is really likely not going to get good results.

I realized though that the biggest amount of pictures of the 1.200 in the CSV file had less than 260 Likes. Consequently, I wanted to try to train the model removing those high values of Likes to see if that improved the accuracy of the model, even though we were taking some of the training data out.

```

Predicted in the 10 range: 0.19183673469387755 % ( 47 / 245 )
Predicted in the 20 range: 0.11836734693877551 % ( 29 / 245 )
Predicted in the 40 range: 0.23673469387755103 % ( 58 / 245 )
Predicted in the 60 range: 0.1510204081632653 % ( 37 / 245 )
Predicted in the 80 range: 0.044897959183673466 % ( 11 / 245 )
Predicted in the 100 range: 0.04897959183673469 % ( 12 / 245 )
Predicted in the over 100 range: 0.20816326530612245 % ( 51 / 245 )
Accuracy = 0.437643932162
Predicted in the 10 range: 0.1557377049180328 % ( 38 / 244 )
Predicted in the 20 range: 0.12704918032786885 % ( 31 / 244 )
Predicted in the 40 range: 0.25 % ( 61 / 244 )
Predicted in the 60 range: 0.16393442622950818 % ( 40 / 244 )
Predicted in the 80 range: 0.0860655737704918 % ( 21 / 244 )
Predicted in the 100 range: 0.036885245901639344 % ( 9 / 244 )
Predicted in the over 100 range: 0.18032786885245902 % ( 44 / 244 )
Accuracy = 0.452677713667
Predicted in the 10 range: 0.11475409836065574 % ( 28 / 244 )
Predicted in the 20 range: 0.0942622950819672 % ( 23 / 244 )
Predicted in the 40 range: 0.1721311475409836 % ( 42 / 244 )
Predicted in the 60 range: 0.16393442622950818 % ( 40 / 244 )
Predicted in the 80 range: 0.12295081967213115 % ( 30 / 244 )
Predicted in the 100 range: 0.06147540983606557 % ( 15 / 244 )
Predicted in the over 100 range: 0.27049180327868855 % ( 66 / 244 )
Accuracy = 0.175052540428
Predicted in the 10 range: 0.15163934426229508 % ( 37 / 244 )
Predicted in the 20 range: 0.13524590163934427 % ( 33 / 244 )
Predicted in the 40 range: 0.21311475409836064 % ( 52 / 244 )
Predicted in the 60 range: 0.12295081967213115 % ( 30 / 244 )
Predicted in the 80 range: 0.08196721311475409 % ( 20 / 244 )
Predicted in the 100 range: 0.08196721311475409 % ( 20 / 244 )
Predicted in the over 100 range: 0.21311475409836064 % ( 52 / 244 )
Accuracy = 0.373806487999
Predicted in the 10 range: 0.11885245901639344 % ( 29 / 244 )
Predicted in the 20 range: 0.14754098360655737 % ( 36 / 244 )
Predicted in the 40 range: 0.25 % ( 61 / 244 )
Predicted in the 60 range: 0.18442622950819673 % ( 45 / 244 )
Predicted in the 80 range: 0.11065573770491803 % ( 27 / 244 )
Predicted in the 100 range: 0.03278688524590164 % ( 8 / 244 )
Predicted in the over 100 range: 0.1557377049180328 % ( 38 / 244 )
Accuracy = 0.485404078055
Cross Validation Score = 0.384916950462

```

Fig. 14, Cross Validation Score for our model running with 5 splits

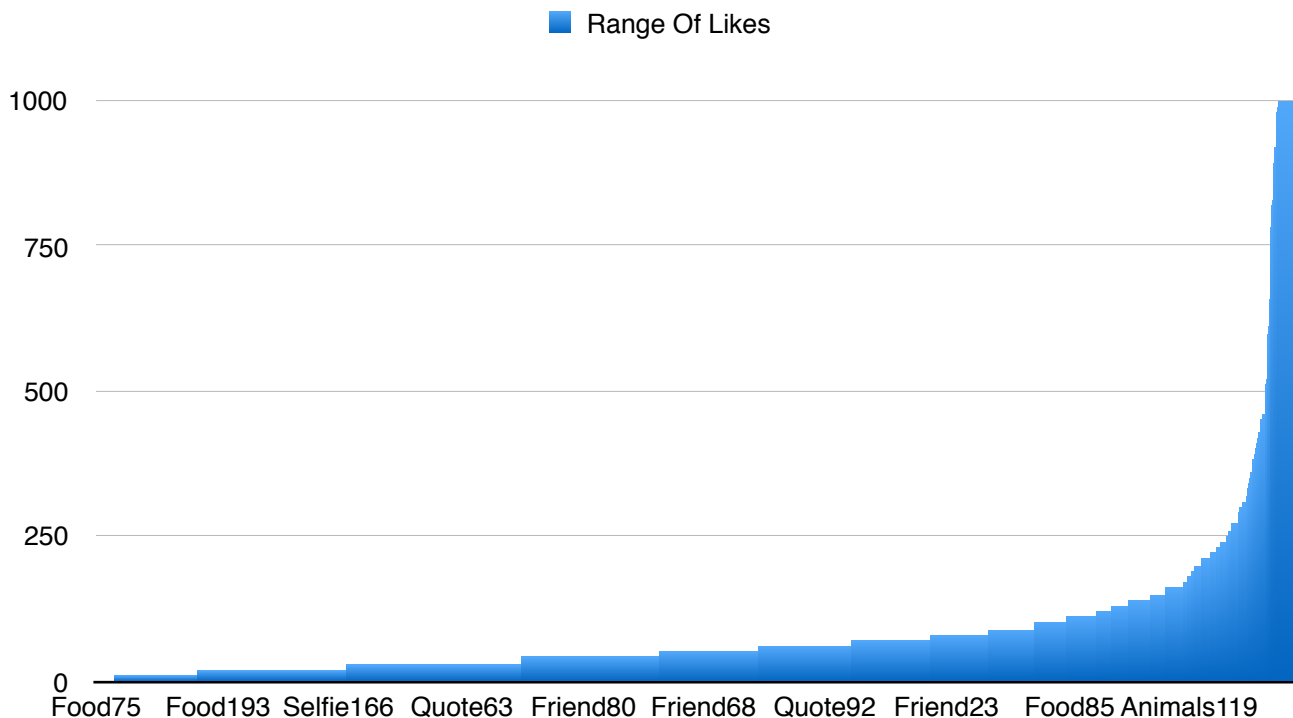


Fig. 15, Graph showing all the pictures of the training data and their correspondent range of likes

The results using less data were still not acceptable, they barely improve 1 and 2 points , respectively. The positive feedback is that it improved a little bit, so we were right when we said that those high points were disturbing our regression.

These bad results are clearly due to the fact that we have 4 independent variables and 1 fixed variable, and the linearity between those 4 variables and the fixed variable is not random, but close. That means that a post can have 50 likes with its user 100 posts, 100 followers and 100 following; and another post can have 50 likes with its user 5.000 posts, 5.000 followers and 5.000 following. These examples happen and that is the reason why our accuracy results are so bad. But anyways, we must look for another solution to guarantee the success of the system.

Specific User Data Regression

The idea of this second path is basically treating the data the same way as with the Global Data Regression, but instead of using random data from many different users, using the data from the specific user that wants to post the input picture that we received. For experiments, I used InstaBro again, which has a good filter for username as well and we can export it into a CSV file.

Having the data from every post of that specific user in a CSV file lets us use the same approach as we did in the second section. We have to get a normalized score for every post that we want to use in order to train the model. As we already know, we have two ways of doing that, using the Maximum and Minimum value or the Normal Distribution. We did both because that way we can test both of them and see which one is better.

One of the important issues to take care of using Specific User Data Regression though, is the quantity of posts that are going to be used for the regression. Some users have less than 50 posts, others have 1.000. At least we know for a fact that 50 posts is enough to do an accurate regression, so that would be the minimum limit to compute a Specific User Data regression.

Posts	Followers	Following	Quantity of posts used in the regression	R^2
538	779	688	151	0,982
299	159	151	101	0,865
189	217	554	76	0,85
54	274	146	51	0,828

Fig. 16, Table showing the test results for some users using Specific User Data Regression

These results are much more optimistic. As I said before, a minimum amount of 50 posts would be necessary to achieve accurate results. Also, with this table we can say that more posts means better results too.

Specific User Data Regression looks more accurate than Global Data Regression, but it would be more difficult to implement because with the global regression we just have to train our model with our training data and we can use it as many times as we want. On the other hand, with the specific user regression we have to get the data from every post of that user in real time first, and also compute the Minimum and Maximum values and/or the Normal Distribution.

The easier way to implement that approach would be using the Instagram API User Endpoints⁶ via HTTP GET calls. In the same Python script we will use to do the regression, we could ask for the information of a user to the Instagram API. The Instagram API would give us all the data we need about the posts of that user. After that, depending on the quantity of posts that user has, we could train our model with more or less points but remember, never less than 50.

⁶ <https://www.instagram.com/developer/endpoints/users/>

CHAPTER 4. RESULTS

With the objective of testing the accuracy of the system with different users and posts, I used Instabro once again to gather 100 random posts of every category. I collected the photos as well as the information of the posts and the users. In total, I had 600 pictures as testing data, 100 per category.

4.1 Category Classification

The accuracy of the first section of the system was nearly excellent so, as you might think, we got quite good results. They are described better in the confusion matrix of Fig. 17.

	Animals	Food	Friends	Landscape	Quote	Selfie
Animals	100	0	0	0	0	0
Food	0	99	0	0	0	1
Friends	2	0	85	0	0	13
Landscape	1	0	0	99	0	0
Quote	0	0	0	0	98	2
Selfie	0	0	7	2	1	90

Fig. 17, Confusion matrix resulted after the classification of 600 pictures, 100 per category

The lowest accuracy resulted from the 6 categories was 85 %, from the Friends category followed by 90 % from the Selfie category, which is still a very good result. Animals, Food, Friends and Quote are over 98 %, even difficult to improve.

After all, the first section of the system behaves properly so we can move forward to the second section.

4.2 Input Picture Score Computation

In the following table we are going to see the histograms of three testing pictures from the Selfie category after going through the second section of the system, running with the NormDist score values. Remember that the column Computed Score represents the maximum point of the histogram.

The X-axis goes from 0 to 1 in steps of 2-decimal frames, 0, 0,01, 0,02, 0,03, etc.
The Y-axis represents a normalized value from the points that every frame of the X-axis got.

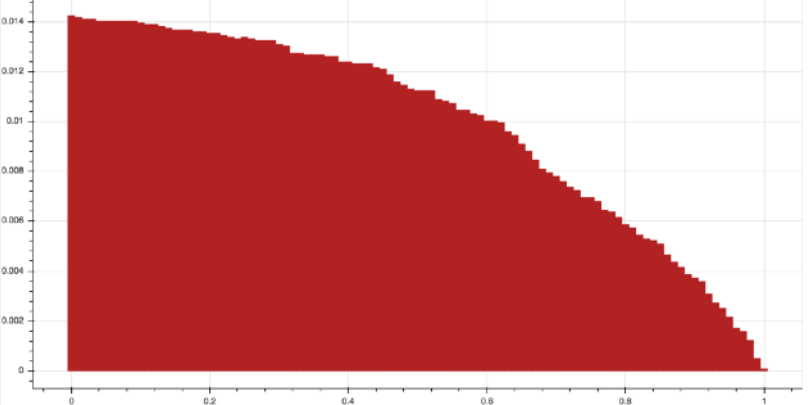
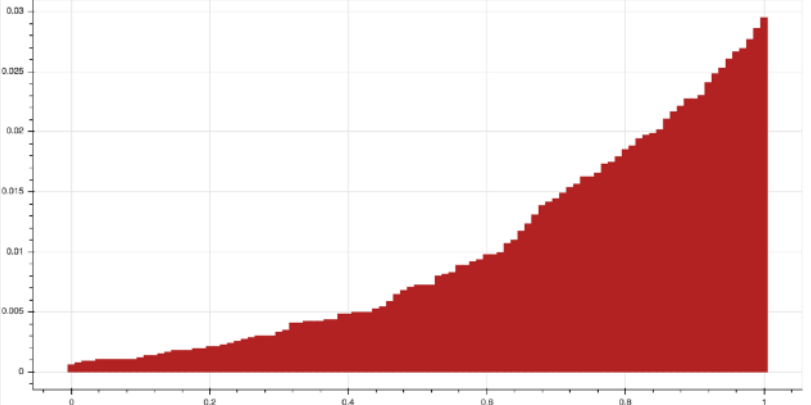
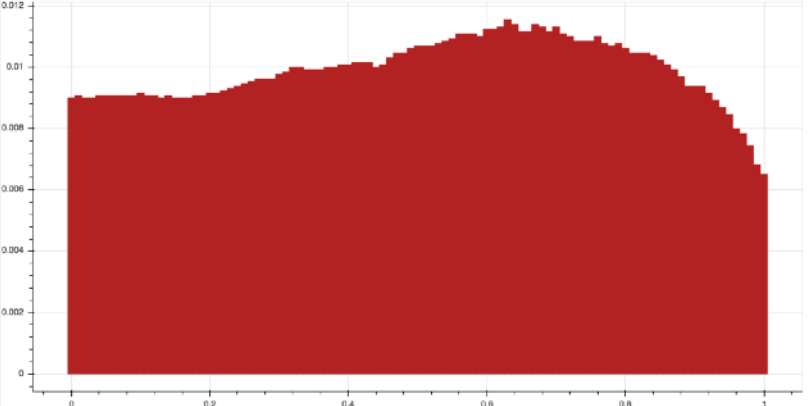
Testing picture	Computed Score	Histogram
SelfieTest7	0	
SelfieTest27	1	
SelfieTest12	0,63	

Fig. 18, Table showing three examples of histograms and computed scores from the Selfie category

4.3 Score into Likes Transformation

At this point of the experiment, we have each one of the 600 pictures with a computed score from 0 to 1. Now it is time to undo that conversion between score and likes using the regression. As you already know, we have two types of regressions, two types of computing scores and 6 categories, that means we will have to analyze 24 different graph results.

Before discussing the resulted graphs, it is important to say that, in order to run this experiment, we used a computer with an NVIDIA GTX1080Ti GPU. It would take approximately 3 minutes to run every picture of the testing data.

The graphs are formed by 8 bars. The X-axis represents the range in which the predicted value of likes fits in comparison to the original value. The Y-axis represents the sum of predictions fitted in every range. The possible ranges are:

- **10 Range:** between -5 and 5. Ex.: original value = 100, predicted value = 105 or 95.
- **20 Range:** between -10 and +10. Ex.: original value = 100, predicted value = 110 or 90.
- **40 Range:** between -20 and +20. Ex.: original value = 100, predicted value = 120 or 80.
- **60 Range:** between -30 and +30. Ex.: original value = 100, predicted value = 130 or 70.
- **80 Range:** between -40 and +40. Ex.: original value = 100, predicted value = 140 or 60.
- **100 Range:** between -50 and +50. Ex.: original value = 100, predicted value = 150 or 50.
- **1000 Range:** between -500 and +500. Ex.: original value = 1000, predicted value = 500 or 1500.
- **Over 1000 Range:** below -500 or higher than +500. Ex.: original value = 1000, predicted value = 499 or 1501.

On one hand, the best graphs will be the ones with higher bars in the lowest ranges, so higher bars on the left side is better. Because the lowest ranges represent more accuracy when predicting, the predicted value is closer to the original value. On the other hand, the worse ones will be the ones with higher bars in the highest ranges, so higher bars on the right side is worse.

In the following pages we will be able to see the graphs for every method out of the 24 tested in the experiment.

Category	Method	Graph																				
Animals	MaxMin Global Regression	<table border="1"> <caption>Data for MaxMin Global Regression</caption> <thead> <tr> <th>Predicted Range</th> <th>Sum(Predictions)</th> </tr> </thead> <tbody> <tr><td>0</td><td>11</td></tr> <tr><td>10</td><td>12</td></tr> <tr><td>20</td><td>13</td></tr> <tr><td>30</td><td>11</td></tr> <tr><td>40</td><td>8</td></tr> <tr><td>50</td><td>4</td></tr> <tr><td>100</td><td>4</td></tr> <tr><td>1000</td><td>37</td></tr> <tr><td>Over 1000</td><td>4</td></tr> </tbody> </table>	Predicted Range	Sum(Predictions)	0	11	10	12	20	13	30	11	40	8	50	4	100	4	1000	37	Over 1000	4
	Predicted Range	Sum(Predictions)																				
	0	11																				
	10	12																				
20	13																					
30	11																					
40	8																					
50	4																					
100	4																					
1000	37																					
Over 1000	4																					
Norm Dist Global Regression	<table border="1"> <caption>Data for Norm Dist Global Regression</caption> <thead> <tr> <th>Predicted Range</th> <th>Sum(Predictions)</th> </tr> </thead> <tbody> <tr><td>0</td><td>7</td></tr> <tr><td>10</td><td>11</td></tr> <tr><td>20</td><td>19</td></tr> <tr><td>30</td><td>13</td></tr> <tr><td>40</td><td>8</td></tr> <tr><td>50</td><td>4</td></tr> <tr><td>100</td><td>4</td></tr> <tr><td>1000</td><td>33</td></tr> <tr><td>Over 1000</td><td>5</td></tr> </tbody> </table>	Predicted Range	Sum(Predictions)	0	7	10	11	20	19	30	13	40	8	50	4	100	4	1000	33	Over 1000	5	
Predicted Range	Sum(Predictions)																					
0	7																					
10	11																					
20	19																					
30	13																					
40	8																					
50	4																					
100	4																					
1000	33																					
Over 1000	5																					
MaxMin Specific User Regression	<table border="1"> <caption>Data for MaxMin Specific User Regression</caption> <thead> <tr> <th>Predicted Range</th> <th>Sum(Predictions)</th> </tr> </thead> <tbody> <tr><td>0</td><td>11</td></tr> <tr><td>10</td><td>16</td></tr> <tr><td>20</td><td>17</td></tr> <tr><td>30</td><td>14</td></tr> <tr><td>40</td><td>12</td></tr> <tr><td>50</td><td>2</td></tr> <tr><td>100</td><td>2</td></tr> <tr><td>1000</td><td>25</td></tr> <tr><td>Over 1000</td><td>3</td></tr> </tbody> </table>	Predicted Range	Sum(Predictions)	0	11	10	16	20	17	30	14	40	12	50	2	100	2	1000	25	Over 1000	3	
Predicted Range	Sum(Predictions)																					
0	11																					
10	16																					
20	17																					
30	14																					
40	12																					
50	2																					
100	2																					
1000	25																					
Over 1000	3																					
NormDist Specific User Regression	<table border="1"> <caption>Data for NormDist Specific User Regression</caption> <thead> <tr> <th>Predicted Range</th> <th>Sum(Predictions)</th> </tr> </thead> <tbody> <tr><td>0</td><td>11</td></tr> <tr><td>10</td><td>16</td></tr> <tr><td>20</td><td>17</td></tr> <tr><td>30</td><td>14</td></tr> <tr><td>40</td><td>12</td></tr> <tr><td>50</td><td>2</td></tr> <tr><td>100</td><td>2</td></tr> <tr><td>1000</td><td>25</td></tr> <tr><td>Over 1000</td><td>3</td></tr> </tbody> </table>	Predicted Range	Sum(Predictions)	0	11	10	16	20	17	30	14	40	12	50	2	100	2	1000	25	Over 1000	3	
Predicted Range	Sum(Predictions)																					
0	11																					
10	16																					
20	17																					
30	14																					
40	12																					
50	2																					
100	2																					
1000	25																					
Over 1000	3																					

Category	Method	Graph																		
Food	MaxMin Global Regression	<table border="1"> <caption>Data for MaxMin Global Regression</caption> <thead> <tr> <th>Predicted Range</th> <th>Sum of Predictions</th> </tr> </thead> <tbody> <tr><td>0</td><td>8</td></tr> <tr><td>10</td><td>13</td></tr> <tr><td>20</td><td>10</td></tr> <tr><td>30</td><td>7</td></tr> <tr><td>40</td><td>9</td></tr> <tr><td>50</td><td>5</td></tr> <tr><td>1000</td><td>45</td></tr> <tr><td>Over 1000</td><td>2</td></tr> </tbody> </table>	Predicted Range	Sum of Predictions	0	8	10	13	20	10	30	7	40	9	50	5	1000	45	Over 1000	2
	Predicted Range	Sum of Predictions																		
	0	8																		
	10	13																		
20	10																			
30	7																			
40	9																			
50	5																			
1000	45																			
Over 1000	2																			
Norm Dist Global Regression	<table border="1"> <caption>Data for Norm Dist Global Regression</caption> <thead> <tr> <th>Predicted Range</th> <th>Sum of Predictions</th> </tr> </thead> <tbody> <tr><td>0</td><td>7</td></tr> <tr><td>10</td><td>9</td></tr> <tr><td>20</td><td>14</td></tr> <tr><td>30</td><td>13</td></tr> <tr><td>40</td><td>7</td></tr> <tr><td>50</td><td>9</td></tr> <tr><td>1000</td><td>37</td></tr> <tr><td>Over 1000</td><td>4</td></tr> </tbody> </table>	Predicted Range	Sum of Predictions	0	7	10	9	20	14	30	13	40	7	50	9	1000	37	Over 1000	4	
Predicted Range	Sum of Predictions																			
0	7																			
10	9																			
20	14																			
30	13																			
40	7																			
50	9																			
1000	37																			
Over 1000	4																			
MaxMin Specific User Regression	<table border="1"> <caption>Data for MaxMin Specific User Regression</caption> <thead> <tr> <th>Predicted Range</th> <th>Sum of Predictions</th> </tr> </thead> <tbody> <tr><td>0</td><td>9</td></tr> <tr><td>10</td><td>14</td></tr> <tr><td>20</td><td>21</td></tr> <tr><td>30</td><td>15</td></tr> <tr><td>40</td><td>5</td></tr> <tr><td>50</td><td>8</td></tr> <tr><td>1000</td><td>27</td></tr> <tr><td>Over 1000</td><td>1</td></tr> </tbody> </table>	Predicted Range	Sum of Predictions	0	9	10	14	20	21	30	15	40	5	50	8	1000	27	Over 1000	1	
Predicted Range	Sum of Predictions																			
0	9																			
10	14																			
20	21																			
30	15																			
40	5																			
50	8																			
1000	27																			
Over 1000	1																			
NormDist Specific User Regression																				

Category	Method	Graph
Friends	MaxMin Global Regression	 <p>A bar chart showing the sum of predictions for different predicted ranges. The y-axis is labeled 'Sum(Predictions)' and ranges from 0 to 25. The x-axis is labeled 'Predicted Range' and has categories: 10, 20, 30, 40, 50, 100, 1000, and Over 1000. The bars are red. The values are approximately: 10: 12, 20: 12, 30: 19, 40: 12, 50: 8, 100: 9, 1000: 27, Over 1000: 1.</p>
	Norm Dist Global Regression	 <p>A bar chart showing the sum of predictions for different predicted ranges. The y-axis is labeled 'Sum(Predictions)' and ranges from 0 to 20. The x-axis is labeled 'Predicted Range' and has categories: 10, 20, 30, 40, 50, 100, 1000, and Over 1000. The bars are red. The values are approximately: 10: 18, 20: 20, 30: 17, 40: 11, 50: 10, 100: 2, 1000: 21, Over 1000: 1.</p>
	MaxMin Specific User Regression	 <p>A bar chart showing the sum of predictions for different predicted ranges. The y-axis is labeled 'Sum(Predictions)' and ranges from 0 to 30. The x-axis is labeled 'Predicted Range' and has categories: 10, 20, 30, 40, 50, 100, 1000, and Over 1000. The bars are red. The values are approximately: 10: 22, 20: 12, 30: 31, 40: 10, 50: 8, 100: 2, 1000: 15, Over 1000: 0.</p>
	NormDist Specific User Regression	 <p>A bar chart showing the sum of predictions for different predicted ranges. The y-axis is labeled 'Sum(Predictions)' and ranges from 0 to 30. The x-axis is labeled 'Predicted Range' and has categories: 10, 20, 30, 40, 50, 100, 1000, and Over 1000. The bars are red. The values are approximately: 10: 22, 20: 12, 30: 31, 40: 10, 50: 8, 100: 2, 1000: 15, Over 1000: 0.</p>

Category	Method	Graph																		
Landscape	MaxMin Global Regression	<table border="1"> <caption>Data for MaxMin Global Regression</caption> <thead> <tr> <th>Predicted Range</th> <th>Sum (Predictions)</th> </tr> </thead> <tbody> <tr><td>10</td><td>8</td></tr> <tr><td>20</td><td>4</td></tr> <tr><td>40</td><td>19</td></tr> <tr><td>60</td><td>11</td></tr> <tr><td>80</td><td>14</td></tr> <tr><td>100</td><td>6</td></tr> <tr><td>1000</td><td>37</td></tr> <tr><td>Over 1000</td><td>1</td></tr> </tbody> </table>	Predicted Range	Sum (Predictions)	10	8	20	4	40	19	60	11	80	14	100	6	1000	37	Over 1000	1
	Predicted Range	Sum (Predictions)																		
	10	8																		
	20	4																		
40	19																			
60	11																			
80	14																			
100	6																			
1000	37																			
Over 1000	1																			
Norm Dist Global Regression	<table border="1"> <caption>Data for Norm Dist Global Regression</caption> <thead> <tr> <th>Predicted Range</th> <th>Sum (Predictions)</th> </tr> </thead> <tbody> <tr><td>10</td><td>8</td></tr> <tr><td>20</td><td>4</td></tr> <tr><td>40</td><td>19</td></tr> <tr><td>60</td><td>11</td></tr> <tr><td>80</td><td>14</td></tr> <tr><td>100</td><td>6</td></tr> <tr><td>1000</td><td>37</td></tr> <tr><td>Over 1000</td><td>1</td></tr> </tbody> </table>	Predicted Range	Sum (Predictions)	10	8	20	4	40	19	60	11	80	14	100	6	1000	37	Over 1000	1	
Predicted Range	Sum (Predictions)																			
10	8																			
20	4																			
40	19																			
60	11																			
80	14																			
100	6																			
1000	37																			
Over 1000	1																			
MaxMin Specific User Regression	<table border="1"> <caption>Data for MaxMin Specific User Regression</caption> <thead> <tr> <th>Predicted Range</th> <th>Sum (Predictions)</th> </tr> </thead> <tbody> <tr><td>10</td><td>8</td></tr> <tr><td>20</td><td>8</td></tr> <tr><td>40</td><td>17</td></tr> <tr><td>60</td><td>9</td></tr> <tr><td>80</td><td>8</td></tr> <tr><td>100</td><td>13</td></tr> <tr><td>1000</td><td>37</td></tr> <tr><td>Over 1000</td><td>1</td></tr> </tbody> </table>	Predicted Range	Sum (Predictions)	10	8	20	8	40	17	60	9	80	8	100	13	1000	37	Over 1000	1	
Predicted Range	Sum (Predictions)																			
10	8																			
20	8																			
40	17																			
60	9																			
80	8																			
100	13																			
1000	37																			
Over 1000	1																			
NormDist Specific User Regression	<table border="1"> <caption>Data for NormDist Specific User Regression</caption> <thead> <tr> <th>Predicted Range</th> <th>Sum (Predictions)</th> </tr> </thead> <tbody> <tr><td>10</td><td>12</td></tr> <tr><td>20</td><td>13</td></tr> <tr><td>40</td><td>19</td></tr> <tr><td>60</td><td>10</td></tr> <tr><td>80</td><td>12</td></tr> <tr><td>100</td><td>13</td></tr> <tr><td>1000</td><td>22</td></tr> <tr><td>Over 1000</td><td>1</td></tr> </tbody> </table>	Predicted Range	Sum (Predictions)	10	12	20	13	40	19	60	10	80	12	100	13	1000	22	Over 1000	1	
Predicted Range	Sum (Predictions)																			
10	12																			
20	13																			
40	19																			
60	10																			
80	12																			
100	13																			
1000	22																			
Over 1000	1																			

Category	Method	Graph												
Quote	MaxMin Global Regression	 <p>A bar chart showing the sum of predictions for different predicted ranges using MaxMin Global Regression. The y-axis is labeled 'Sum(Predictions)' and ranges from 0 to 20. The x-axis is labeled 'Predicted Range' and has categories: 10, 100, 1000, 10000, and Over10000. The bars are red. The values are approximately: 10: 16, 100: 11, 1000: 9, 10000: 19, Over10000: 15.</p> <table border="1"> <thead> <tr> <th>Predicted Range</th> <th>Sum(Predictions)</th> </tr> </thead> <tbody> <tr> <td>10</td> <td>16</td> </tr> <tr> <td>100</td> <td>11</td> </tr> <tr> <td>1000</td> <td>9</td> </tr> <tr> <td>10000</td> <td>19</td> </tr> <tr> <td>Over10000</td> <td>15</td> </tr> </tbody> </table>	Predicted Range	Sum(Predictions)	10	16	100	11	1000	9	10000	19	Over10000	15
	Predicted Range	Sum(Predictions)												
	10	16												
	100	11												
1000	9													
10000	19													
Over10000	15													
Norm Dist Global Regression	 <p>A bar chart showing the sum of predictions for different predicted ranges using Norm Dist Global Regression. The y-axis is labeled 'Sum(Predictions)' and ranges from 0 to 25. The x-axis is labeled 'Predicted Range' and has categories: 10, 100, 1000, 10000, and Over10000. The bars are red. The values are approximately: 10: 19, 100: 20, 1000: 24, 10000: 12, Over10000: 7.</p> <table border="1"> <thead> <tr> <th>Predicted Range</th> <th>Sum(Predictions)</th> </tr> </thead> <tbody> <tr> <td>10</td> <td>19</td> </tr> <tr> <td>100</td> <td>20</td> </tr> <tr> <td>1000</td> <td>24</td> </tr> <tr> <td>10000</td> <td>12</td> </tr> <tr> <td>Over10000</td> <td>7</td> </tr> </tbody> </table>	Predicted Range	Sum(Predictions)	10	19	100	20	1000	24	10000	12	Over10000	7	
Predicted Range	Sum(Predictions)													
10	19													
100	20													
1000	24													
10000	12													
Over10000	7													
MaxMin Specific User Regression	 <p>A bar chart showing the sum of predictions for different predicted ranges using MaxMin Specific User Regression. The y-axis is labeled 'Sum(Predictions)' and ranges from 0 to 25. The x-axis is labeled 'Predicted Range' and has categories: 10, 100, 1000, 10000, and Over10000. The bars are red. The values are approximately: 10: 19, 100: 20, 1000: 24, 10000: 12, Over10000: 7.</p> <table border="1"> <thead> <tr> <th>Predicted Range</th> <th>Sum(Predictions)</th> </tr> </thead> <tbody> <tr> <td>10</td> <td>19</td> </tr> <tr> <td>100</td> <td>20</td> </tr> <tr> <td>1000</td> <td>24</td> </tr> <tr> <td>10000</td> <td>12</td> </tr> <tr> <td>Over10000</td> <td>7</td> </tr> </tbody> </table>	Predicted Range	Sum(Predictions)	10	19	100	20	1000	24	10000	12	Over10000	7	
Predicted Range	Sum(Predictions)													
10	19													
100	20													
1000	24													
10000	12													
Over10000	7													
NormDist Specific User Regression	 <p>A bar chart showing the sum of predictions for different predicted ranges using NormDist Specific User Regression. The y-axis is labeled 'Sum(Predictions)' and ranges from 0 to 25. The x-axis is labeled 'Predicted Range' and has categories: 10, 100, 1000, 10000, and Over10000. The bars are red. The values are approximately: 10: 28, 100: 19, 1000: 25, 10000: 11, Over10000: 7.</p> <table border="1"> <thead> <tr> <th>Predicted Range</th> <th>Sum(Predictions)</th> </tr> </thead> <tbody> <tr> <td>10</td> <td>28</td> </tr> <tr> <td>100</td> <td>19</td> </tr> <tr> <td>1000</td> <td>25</td> </tr> <tr> <td>10000</td> <td>11</td> </tr> <tr> <td>Over10000</td> <td>7</td> </tr> </tbody> </table>	Predicted Range	Sum(Predictions)	10	28	100	19	1000	25	10000	11	Over10000	7	
Predicted Range	Sum(Predictions)													
10	28													
100	19													
1000	25													
10000	11													
Over10000	7													

Category	Method	Graph
Selfie	MaxMin Global Regression	 <p>A bar chart showing the sum of predictions for different predicted ranges. The y-axis is labeled 'Sum(Predictions)' and ranges from 0 to 30. The x-axis is labeled 'Predicted Range' and has categories: 10, 100, 1000, 10000, and Over 10000. The bars are red. The values are approximately: 10: 9, 100: 2, 1000: 24, 10000: 16, 100000: 11, 1000000: 6, 10000000: 31, Over 10000000: 1.</p>
	Norm Dist Global Regression	 <p>A bar chart showing the sum of predictions for different predicted ranges. The y-axis is labeled 'Sum(Predictions)' and ranges from 0 to 30. The x-axis is labeled 'Predicted Range' and has categories: 10, 100, 1000, 10000, and Over 10000. The bars are red. The values are approximately: 10: 14, 100: 12, 1000: 14, 10000: 11, 100000: 20, 1000000: 24, Over 1000000: 1.</p>
	MaxMin Specific User Regression	 <p>A bar chart showing the sum of predictions for different predicted ranges. The y-axis is labeled 'Sum(Predictions)' and ranges from 0 to 30. The x-axis is labeled 'Predicted Range' and has categories: 10, 100, 1000, 10000, and Over 10000. The bars are red. The values are approximately: 10: 16, 100: 13, 1000: 28, 10000: 17, 100000: 9, 1000000: 3, 10000000: 13, Over 10000000: 1.</p>
	NormDist Specific User Regression	 <p>A bar chart showing the sum of predictions for different predicted ranges. The y-axis is labeled 'Sum(Predictions)' and ranges from 0 to 30. The x-axis is labeled 'Predicted Range' and has categories: 10, 100, 1000, 10000, and Over 10000. The bars are red. The values are approximately: 10: 16, 100: 13, 1000: 28, 10000: 17, 100000: 9, 1000000: 3, 10000000: 13, Over 10000000: 1.</p>

Relative Error

We would like to have a better approach at saying the accuracy of the system so we thought about computing the relative error for every tested post. Some posts had an original value of 10 likes while the system predicted 50; others had 1000 and the system predicted 1050. Those examples have the same absolute error (50) but the first one has a relative error prediction of 400 %, whereas the second one has a relative error of just 5 %.

The following table show the relative error of every category, hence the accuracy of every category, excluding the posts with a relative error higher than 100%. The bar charts show the quantity of posts in every range between 0 and 100% of relative error in 10% frames. Exactly as before, the higher bars on the left side of the graph, the less relative error so the highest accuracy we will have.

Category	Predictions with relative error over 100%	Relative Error	Accuracy	Graph																						
Animals	11	38,52 %	61,48 %	<table border="1"> <caption>Animals Relative Error Distribution</caption> <thead> <tr> <th>Relative Error Range</th> <th>Number of Posts</th> </tr> </thead> <tbody> <tr><td><10%</td><td>13</td></tr> <tr><td><20%</td><td>12</td></tr> <tr><td><30%</td><td>16</td></tr> <tr><td><40%</td><td>15</td></tr> <tr><td><50%</td><td>9</td></tr> <tr><td><60%</td><td>3</td></tr> <tr><td><70%</td><td>6</td></tr> <tr><td><80%</td><td>4</td></tr> <tr><td><90%</td><td>7</td></tr> <tr><td><100%</td><td>4</td></tr> </tbody> </table>	Relative Error Range	Number of Posts	<10%	13	<20%	12	<30%	16	<40%	15	<50%	9	<60%	3	<70%	6	<80%	4	<90%	7	<100%	4
Relative Error Range	Number of Posts																									
<10%	13																									
<20%	12																									
<30%	16																									
<40%	15																									
<50%	9																									
<60%	3																									
<70%	6																									
<80%	4																									
<90%	7																									
<100%	4																									
Food	6	38,92 %	61,08 %	<table border="1"> <caption>Food Relative Error Distribution</caption> <thead> <tr> <th>Relative Error Range</th> <th>Number of Posts</th> </tr> </thead> <tbody> <tr><td><10%</td><td>14</td></tr> <tr><td><20%</td><td>13</td></tr> <tr><td><30%</td><td>17</td></tr> <tr><td><40%</td><td>15</td></tr> <tr><td><50%</td><td>7</td></tr> <tr><td><60%</td><td>12</td></tr> <tr><td><70%</td><td>10</td></tr> <tr><td><80%</td><td>5</td></tr> <tr><td><90%</td><td>5</td></tr> <tr><td><100%</td><td>5</td></tr> </tbody> </table>	Relative Error Range	Number of Posts	<10%	14	<20%	13	<30%	17	<40%	15	<50%	7	<60%	12	<70%	10	<80%	5	<90%	5	<100%	5
Relative Error Range	Number of Posts																									
<10%	14																									
<20%	13																									
<30%	17																									
<40%	15																									
<50%	7																									
<60%	12																									
<70%	10																									
<80%	5																									
<90%	5																									
<100%	5																									

Category	Predictions with relative error over 100%	Relative Error	Accuracy	Graph																						
Friends	16	38,1 %	61,9 %	<table border="1"> <caption>Relative Error Distribution for Friends</caption> <thead> <tr> <th>Relative Error Range</th> <th>Frequency</th> </tr> </thead> <tbody> <tr><td><10%</td><td>13.5</td></tr> <tr><td><20%</td><td>6.0</td></tr> <tr><td><30%</td><td>17.0</td></tr> <tr><td><40%</td><td>14.0</td></tr> <tr><td><50%</td><td>6.0</td></tr> <tr><td><60%</td><td>8.0</td></tr> <tr><td><70%</td><td>5.0</td></tr> <tr><td><80%</td><td>8.0</td></tr> <tr><td><90%</td><td>5.0</td></tr> <tr><td><100%</td><td>1.0</td></tr> </tbody> </table>	Relative Error Range	Frequency	<10%	13.5	<20%	6.0	<30%	17.0	<40%	14.0	<50%	6.0	<60%	8.0	<70%	5.0	<80%	8.0	<90%	5.0	<100%	1.0
Relative Error Range	Frequency																									
<10%	13.5																									
<20%	6.0																									
<30%	17.0																									
<40%	14.0																									
<50%	6.0																									
<60%	8.0																									
<70%	5.0																									
<80%	8.0																									
<90%	5.0																									
<100%	1.0																									
Landscape	3	38,86 %	61,14 %	<table border="1"> <caption>Relative Error Distribution for Landscape</caption> <thead> <tr> <th>Relative Error Range</th> <th>Frequency</th> </tr> </thead> <tbody> <tr><td><10%</td><td>13.5</td></tr> <tr><td><20%</td><td>12.0</td></tr> <tr><td><30%</td><td>11.0</td></tr> <tr><td><40%</td><td>9.0</td></tr> <tr><td><50%</td><td>14.0</td></tr> <tr><td><60%</td><td>17.0</td></tr> <tr><td><70%</td><td>6.0</td></tr> <tr><td><80%</td><td>6.0</td></tr> <tr><td><90%</td><td>2.0</td></tr> <tr><td><100%</td><td>3.0</td></tr> </tbody> </table>	Relative Error Range	Frequency	<10%	13.5	<20%	12.0	<30%	11.0	<40%	9.0	<50%	14.0	<60%	17.0	<70%	6.0	<80%	6.0	<90%	2.0	<100%	3.0
Relative Error Range	Frequency																									
<10%	13.5																									
<20%	12.0																									
<30%	11.0																									
<40%	9.0																									
<50%	14.0																									
<60%	17.0																									
<70%	6.0																									
<80%	6.0																									
<90%	2.0																									
<100%	3.0																									
Quote	13	43,19 %	56,81 %	<table border="1"> <caption>Relative Error Distribution for Quote</caption> <thead> <tr> <th>Relative Error Range</th> <th>Frequency</th> </tr> </thead> <tbody> <tr><td><10%</td><td>12.0</td></tr> <tr><td><20%</td><td>12.0</td></tr> <tr><td><30%</td><td>9.0</td></tr> <tr><td><40%</td><td>12.0</td></tr> <tr><td><50%</td><td>8.0</td></tr> <tr><td><60%</td><td>7.0</td></tr> <tr><td><70%</td><td>6.0</td></tr> <tr><td><80%</td><td>10.0</td></tr> <tr><td><90%</td><td>7.0</td></tr> <tr><td><100%</td><td>4.0</td></tr> </tbody> </table>	Relative Error Range	Frequency	<10%	12.0	<20%	12.0	<30%	9.0	<40%	12.0	<50%	8.0	<60%	7.0	<70%	6.0	<80%	10.0	<90%	7.0	<100%	4.0
Relative Error Range	Frequency																									
<10%	12.0																									
<20%	12.0																									
<30%	9.0																									
<40%	12.0																									
<50%	8.0																									
<60%	7.0																									
<70%	6.0																									
<80%	10.0																									
<90%	7.0																									
<100%	4.0																									
Selfie	8	37,89 %	62,11 %	<table border="1"> <caption>Relative Error Distribution for Selfie</caption> <thead> <tr> <th>Relative Error Range</th> <th>Frequency</th> </tr> </thead> <tbody> <tr><td><10%</td><td>19.0</td></tr> <tr><td><20%</td><td>12.0</td></tr> <tr><td><30%</td><td>11.0</td></tr> <tr><td><40%</td><td>13.0</td></tr> <tr><td><50%</td><td>5.0</td></tr> <tr><td><60%</td><td>8.0</td></tr> <tr><td><70%</td><td>6.0</td></tr> <tr><td><80%</td><td>10.0</td></tr> <tr><td><90%</td><td>6.0</td></tr> <tr><td><100%</td><td>2.0</td></tr> </tbody> </table>	Relative Error Range	Frequency	<10%	19.0	<20%	12.0	<30%	11.0	<40%	13.0	<50%	5.0	<60%	8.0	<70%	6.0	<80%	10.0	<90%	6.0	<100%	2.0
Relative Error Range	Frequency																									
<10%	19.0																									
<20%	12.0																									
<30%	11.0																									
<40%	13.0																									
<50%	5.0																									
<60%	8.0																									
<70%	6.0																									
<80%	10.0																									
<90%	6.0																									
<100%	2.0																									

Fig. 19, Relative error results of every category described in percentage and graphs

CONCLUSIONS

- The small uncertainty between Friends and Selfie category. Summing up the misclassifications from both classes give us 20 misclassifications out of 200. It's a 90 % accuracy so, even though we know that there is some uncertainty, it is not a big step to overcome.
- As you might have noticed, MaxMin Global Regression and NormDist Global Regression are sharing the same graph. In the global regressions, there are four independent variables (score, posts, followers, following) and one fixed variable (likes). If we have a look at the independent variables magnitudes, we can see that score will always be between 0 and 1; posts can go from 50 to 4000 or 5000; followers can be any result; and following, same as followers. The score variable is so small compared to the other three variables that it doesn't really affect to the predicted value, maybe just one or two decimals, so the histogram will always be the same. One way to fix this would be normalizing the values of posts, followers and following, so they have the same importance as the rest of variables.
- Second thing we can point out is that the specific user regressions give better results in all the categories. As we had in mind before testing, the accuracy of the specific user regression is higher than the one of the global regression.
- Comparing the MaxMin value graphs and the NormDist score value graphs in every graph, we can affirm that NormDist score values give better results than MaxMin values.
- The Quote category is the one with the worst accuracy and that makes sense, because the photos classified as quotes are very similar to each other hence difficult to predict accurately.
- After analyzing these results, we can clearly say that the accuracy of our system is approximately 61 %.

61 % is an even better result than expected because, as soon as the project started, we knew that there was another variable that was not in the data. People are different and they behave differently on Instagram, but moreover, their followers behave totally different too. So it is still a big challenge to try to predict that behavior apart from all the other variables that are taken into account.

REFERENCES

1. "Convolutional Neural Networks (LeNet) – DeepLearning 0.1 documentation". *DeepLearning 0.1*. LISA Lab. Retrieved 31 August 2013.
2. C.-W. Hsu, C.-C. Chang, C.-J. Lin. LIBSVM: a library for support vector machines, "A Practical Guide to Support Vector Classification", Department of Computer Science National Taiwan University, last updated May 19, 2016
3. McCrea N., "An Introduction to Machine Learning Theory and Its Applications: A Visual Tutorial with Examples", TOPTAL, 2014. Retrieved from <https://www.toptal.com/machine-learning/machine-learning-theory-an-introductory-primer>
4. Ray S., "Understanding Support Vector Machine algorithm from examples", Analytics Vidhya, 2015. Retrieved from <https://www.analyticsvidhya.com/blog/2015/10/understaing-support-vector-machine-example-code/>
5. Schneider J., "Cross Validation", 1997. Retrieved from <https://www.cs.cmu.edu/~schneide/tut5/node42.html>
6. Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., Rabinovich A., "Going Deeper with Convolutions", IEEE Explore, 2015
7. Tensorflow Documentation, https://www.tensorflow.org/api_docs/python/tf/
8. Tensorflow for Poets, https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/?utm_campaign=chrome_series_machinelearning_063016&utm_source=gdev&utm_medium=yt-desc#0