Atributes Classification using a Convolutional Neural Network

Alejandro Trujillo
Universidad de los Andes
Bogotá, Colombia
af.trujilloa@uniandes.edu.co

Santiago Martínez
Universidad de los Andes
Bogotá, Colombia
s.martinez1@uniandes.edu.co

Abstract

The present paper has the intent to show the processes of designing a convolutional neural network (CNN) and applying the design architecture to a multi-label classification problem.

1. Introduction

Attribute classification is an important task when it comes to computer vision and artificial intelligence, as it helps a machine understand semantically an image and possibly respond in certain ways depending on the intent of the algorithm. In this paper we'll use the celebA dataset to train a neural network in order to classify attributes in images of celebrities.

2. Methodology

2.1. Dataset Description

The Dataset used for this problem was celebA. This dataset is composed by 202599 annotated images with 162770 images in the train set, 19867 in the test set and 19962 in the validation set. Some example can be seen in figure 1.

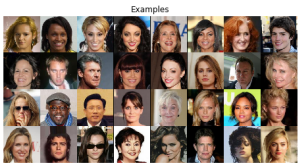

Figure 1. Example of images in the dataset

Apart from that, each image is annotated with the presence or lack of multiple attributes, which include mustache, blonde hair, curly hair, attractive among others.

Our intent is to focus on 10 attributes and come up with a method to classify if an image has them or not.

2.2. Non-Neural Network Approach

One way to tackle this problem would be to use a conventional classifier like, for example, an SVM. It'd be necessary to train multiple SVM (one per attribute) and feed it a descriptor of the training images which include said attribute as positives and those that don't as negatives. The descriptor is more complicated, as this attributes don't follow a very clear visual pattern, however, including as much information as possible would be a way to go, the descriptor then would include a histogram of the texton map of the image, color difference, Pyramidal HOG, between others, all of them concatenated. The problem of including too much information is that the more descriptors one uses, the slower the algorithm becomes. While using the algorithm on the wild, this is very inconvenient. However, state of the art methods don't use this kind of approach anymore, as CNNs yield better results and don't require as much time while testing.

2.3. Simple Network design

Firstly, we tried a small Neural Network based on "LeNet" composed by 3 convolutional layers and 1 linear layer. For all convolutional layers, we used a kernel size of 3. And between each of them, we used a ReLu non-linearity and a max pooling of size 2. After the first 2 convolutional layers we added a dropout with a probability of 0.25, the same was done after the third one. The classifier consisted of a linear layer.

2.4. Big Network Design

The proposed Network consists of 5 convolutional layers. the first layer kernel's size is 5 pixels, the rest of them have a kernel size of 3. Between each of these layers, we added a Relu non-linearity and a maxpool of size 2

and stride 1. After this layers comes the classifier, which consists of one linear layer. This design yields a total of 3946826 parameters in the net. This network was heavily inspired by AlexNet, however subtle changes were made, as we don't use three linear layers as a classifier but one, which could lead to less accuracy but also less parameters to optimize.
The selected optimizer was a Stochastic Gradient Descent (SGD) with a learning rate of 0.01. The loss function used was a Binary cross-Entropy loss, which unlike softmax, is independent for each vector component (class in our case) which means that the loss calculated for each component is unaffected by other component values. Being our problem a multi-label one, this function results convenient. [1]

2.5. Difficulties during Design

The first problem we had to approach was to create the dataset class in such way that pytorch would be able to understand it, and at the same time dividing it in its three categories. This task also included limiting the labels to the 10 selected for analysis. Another problem arose when we tried to run the train script, as the linear layers required an specific input in order to work properly. This led to adding a new step on the transformation step. This step consisted of resizing the input image to the appropriate size. for the small model, the input images were resized to 28x28 (as it was based on LeNet) and for the big Net, the input images were resized to 96x96.

3. Results

4. Conclusions

The correct design of neural networks requires time and a clear understanding of the parameters used, not enough time means the Net won't be correctly trained and would be inaccurate, and designing a Net that's simple isn't appropriate for most computer vision problems, for which reason our simple conbolutional neural network didn't perform good enough and our large one couldn't be completely trained.

References

[1] *Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names", Gombru.github.io, 2019. [Online]. Available: https://gombru.github.io/2018/05/23/cross* $_{e}ntropy_{l}oss/.[Accessed: 26 - Apr - 2019]$.