International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2019 IJSRCSEIT | Volume 5 | Issue 1 | ISSN : 2456-3307
DOI : https://doi.org/10.32628/CSEIT195150

Emotion Detection using Facial expressions with Transfer Learning

Sairama Geetanand, Shaik Nazeer Basha, Sajja Nageswara Rao, Rayudu Srinivas
CSE, Vasireddy Venkatadri Institute of Technology, Guntur, Andhra Pradesh, India

ABSTRACT

The use of machines to perform different tasks is constantly increasing in society. Providing machines with perception can lead them to perform a great variety of tasks. Machine perception requires that machines understand about their environment. Recognizing facial emotions will thus help in this regard. We use the TensorFlow library and the Inception model and apply transfer-learning for our dataset to retrain the model. We then identify the facial emotions: happiness, sadness, anger, surprise, disgust and neutral.
Keywords: Deep learning, TensorFlow, Inception, Transfer Learning, Convolutional neural networks.

I. INTRODUCTION

Facial expressions play a very important role in human communication. The human face is the richest source of emotions. As society continues to make more use of human-machine interactions, it is important for machines to be able to interpret facial expressions in order to improve their authenticity or to make them less machine and more human. Our brains make vision seem easy. It doesn't take any effort for humans to tell apart a lion and a tiger, read an article, or recognizing a human's face. But these are actually hard problems to solve with a computer: they only seem easy because our brains are incredibly good at understanding images.
Emotion can be recognized through a variety of means such as voice intonation, body language, and more complex methods such electroencephalography (EEG). However, the easier, more practical method is to examine facial expressions. There are seven types of human emotions shown to be universally recognizable across different cultures: anger, disgust, fear, happiness, sadness, surprise, contempt. Interestingly, even for complex expressions where a mixture of emotions could be used as descriptors, cross-cultural agreement is still observed. Therefore, a utility that detects emotion from facial expressions would be widely applicable. Such advancement could bring applications in medicine, marketing and entertainment. The task of emotion recognition is particularly difficult for two reasons: (1) There does not exist a large database of training images and (2) classifying emotions can be difficult depending on whether the input image is static or in a transition frame into a facial expression. The latter issue is particularly difficult for real-time detection where facial expressions vary dynamically.

In the last few years the field of machine learning has made tremendous progress on addressing these difficult problems. In particular, the discovery of the model called a deep convolutional neural network can achieve reasonable performance on hard visual recognition tasks matching or exceeding human performance in some domains. It is now being used by popular tech-giants, Facebook being one of the leading users. Researchers have demonstrated steady progress in computer vision by validating their work against ImageNet an academic benchmark for computer vision.

II. METHODS AND MATERIAL

A. Dataset

We scraped image urls from google images, using java script, downloaded the images from the generated urls using python. The images are then filtered manually

and the ones that don‟t suit the emotion are removed. The current dataset of images we used contain 2192 images for the six emotions : Happy, Sad, Angry, Surprise, Disgust, Neutral.

B. TensorFlow

Tensor Flow (TF) [2] is an open source software library for machine learning written in Python and C++. The main reason behind it is that TF was developed by Google Brain Team. Google has already been using TF to improve some tasks on several products. These tasks include speech recognition in Google Now, search features in Google Photos, and the smart reply feature in Inbox by Gmail. Some design decision in TF have lead to this framework to be early adopted by a big community. One of them is the ease of going from prototype to production. There is no need to compile or to modify the code to use it on a product. Then, the framework is not only thought as a research tool, but as a production one. Another main design aspect is that there is no need to use different API when working on CPU or GPU. Moreover, the computations can be deployed over desktops, servers and mobile devices. A key component of the library is the data flow graph. The sense of expressing mathematical computations with nodes and edges is a TF trademark. Nodes are usually the mathematical operations, while edges define the input / output association between nodes. The information travels around the graph as a tensor, a multidimensional array. Finally, the nodes are allocated on devices where they are executed asynchronously or in parallel when all the resources are ready.

C. Inception

Inception is a pre-trained deep neural network, developed to classify images. It took around 2 weeks to train the inception model using 8 Tesla K40 GPUs and it costed around $30,000. It considers around 25 million parameters and does 5 billion add-multiply operations to classify a single image. Given today‟s computation power, it can be done in fraction of a second. Inception takes images as its input. It can process only JPEG format images. The recommended resolution is 299*299. If the image is of higher resolution, it will be compressed automatically. It produces a private class of array of the image as its output. It was developed as a part of ImageNet Large Sale Visual Recognition challenge 2014. It can classify almost every day-to-day objects. The architecture of the Inception is briefly described in . Inception consists of 22 layers. The penultimate layer is called as “Bottlenecks”. The final layer is called as softmax layer. This is the layer that can be retrained to classify the required image group. We retrained it to classify 6 emotions: neutral, happy, sad, surprise, disgust and anger. There are 3 metrics. Training accuracy describes the accuracy of the inception model in classifying the images on which it was trained. Validation accuracy describes the accuracy of the model in classifying the images that are not part of the training dataset. Cross entropy is the loss function. It describes the error rate.

Fig.1: The 3 metrics: Train accuracy, Class entropy, Validation accuracy

D. Implementation

Once we are sure that inception is correctly installed and is working correctly we retrain the model for our dataset like so. Modern object recognition models have millions of parameters and can take weeks to fully train. Transfer learning is a technique that shortcuts a lot of this work by taking a fully-trained model for a set of categories like ImageNet, and retrains from the existing weights for new classes. We retrained the final layer from scratch, while leaving all the others untouched.
How is it done?

We evaluate whether features extracted from the activation of a deep convolutional network trained in a fully supervised fashion on a large, fixed set of object recognition tasks can be re-purposed to novel generic tasks. Our generic tasks may differ significantly from the originally trained tasks and there may be insufficient labelled or unlabeled data to conventionally train or adapt a deep architecture to the new tasks. We investigate and visualize the semantic clustering of deep convolutional features with respect to a variety of such tasks, including scene recognition, domain adaptation, and fine-grained recognition challenges. Before we start any training, we need a set of images to teach the network about the new classes you want to

recognize. Here we use a dataset of emotions gathered from a variety of sources. Once you have the images, from the root of your TensorFlow source directory you can begin the training process.

We load the pre-trained Inception v3 model, remove the old top layer, and train a new one on the emotion photos we've downloaded. None of the emotions were in the original ImageNet classes the full network was trained on. The magic of transfer learning is that lower layers that have been trained to distinguish between some objects can be reused for many recognition tasks without any alteration. The script can take thirty minutes or more to complete, depending on the speed of the machine. The first phase analyzes all the images on disk and calculates the bottleneck values for each of them. 'Bottleneck' is an informal term we often use for the layer just before the final output layer that actually does the classification. This penultimate layer has been trained to output a set of values that's good enough for the classifier to use to distinguish between all the classes it's been asked to recognize. That means it has to be a meaningful and compact summary of the images, since it has to contain enough information for the classifier to make a good choice in a very small set of values. The reason our final layer retraining can work on new classes is that it turns out the kind of information needed to distinguish between all the 1,000 classes in ImageNet is often also useful to distinguish between new kinds of objects. Because every image is reused multiple times during training and calculating each bottleneck takes a significant amount of time, it speeds things up to cache these Bottleneck values on disk so they don't have to be repeatedly recalculated and if you rerun the script they'll be reused so you don't have to wait for this part again.

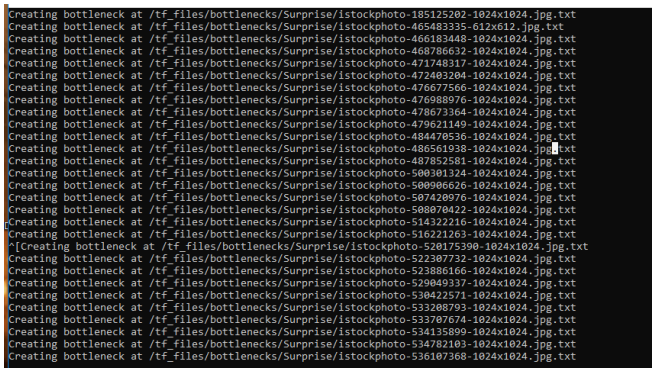

Fig.2 : TensorFlow creating bottleneck values for each image

Fig.3: Bottleneck values generated for an image

Once the bottlenecks are complete, the actual training of the top layer of the network begins. We'll see a series of step outputs, each one showing training accuracy, validation accuracy, and the cross entropy. The training accuracy shows what percent of the images used in the current training batch were labelled with the correct class. The validation accuracy is the precision on a randomly-selected group of images from a different set. The key difference is that the training accuracy is based on images that the network has been able to learn from so the network can overfit to the noise in the training data. A true measure of the performance of the network is to measure its performance on a data set not contained in the training data -- this is measured by the validation accuracy. 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. Cross entropy is a loss function which gives a glimpse into how well the learning process is progressing. The training's objective is to make the loss as small as possible, so you can tell if the learning is working by keeping an eye on whether the loss keeps trending downwards, ignoring the short-term noise.

By default, it will run 4,000 training steps. Each step chooses ten images at random from the training set, finds their bottlenecks from the cache, and feeds them into the final layer to get predictions. Those predictions are then compared against the actual labels to update the final layer's weights through the back-propagation process. As the process continues you should see the reported accuracy improve, and after all the steps are done, a final test accuracy evaluation is run on a set of images kept separate from the training and validation pictures. This test evaluation is the best estimate of how the trained model will perform on the classification task. You should see an accuracy value of between 90% and 95%, though the exact value will vary from run to run

since there is randomness in the training process. This number is based on the percent of the images in the test set that are given the correct label after the model is fully trained.

Tensor Board summaries make it easier to understand, debug, and optimize the retraining. We can visualize the graph and statistics, such as how the weights or accuracy varied during training. The Inception v3 network with a final layer retrained to our categories to get the output graph and a text file containing the labels –output labels.

III. RESULTS AND DISCUSSION

After the training process, we provide the retrained model with the image we wish to classify. Note that the retrained model can classify images only with the classes we provided initially in the retraining inception part, so if the image is for example, an image of a tall building the result would be the best possible match among the classes. The system can identify only the images it is trained for just like humans, seeing something we have never seen before we shall not be able to identify it. It will classify the images in the weights of percentage, for example if an image is happy it will identify say 0.9025 to say it

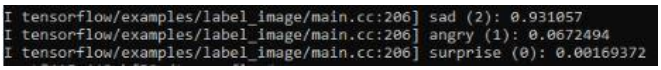

Fig.4: Test image classification values.

The above image shows the classification value of an image, which depicts the emotion „sad", and its value generated is 0.931057 which approximates to 93%.

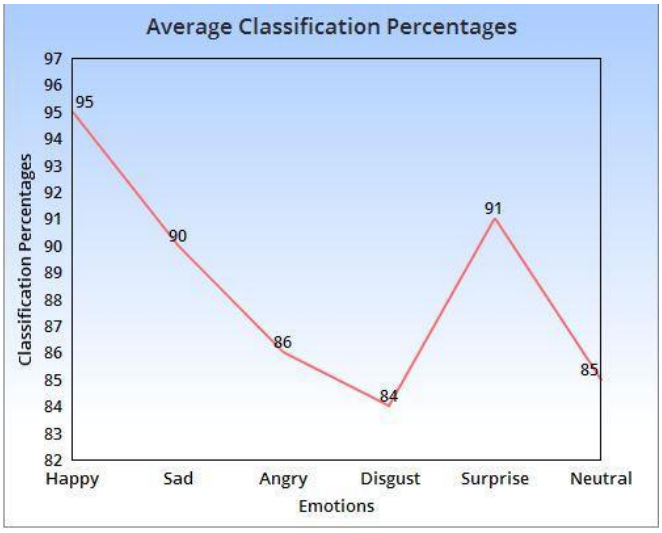

Fig.5: A line graph depicting the average percentage values, obtained by testing various images of the six emotions.

The graph values are taken by testing the output label with various images, some taken from the dataset and some outside of it. Each emotion is tested using 15 images and the average of all these values is then converted to percentage, which are given as inputs to the graph.

IV.CONCLUSION

In this work, we detect and recognize emotions in static human images using TensorFlow and Inception model. The future enhancement can be an action that is done upon recognition of the emotions. If we get a sad emotion, we can have the system play a song or tell a joke or send his/her best friend a message. This can be the next step of AI where the system can understand, comprehend the user's feelings and emotions and react accordingly. This bridges the gap between machines and humans.

We can also have an interactive keyboard where the users can just use the app and the app will then identify the emotion and convert that emotion to the emoji of choice.

V. REFERENCES

[1]. Raghav Puri, Archit Gupta, Manas Sikri (2018). Emotion Detection using Image Processing in Python. 5th International Conference on "Computing for Sustainable Global Development (IndiaCom)" IEEE conference ID : 42835.

[2]. https://codelabs.developers.google.com/codelabs/tensorflow-for-poets

[3]. Hong-Wei Ng, Viet Dung Nguyen, Vassilios Vonikakis, Stefan Winkler(2015). Deep Learning for Emotion Recognition on Small Datasets Using Transfer Learning.

[4]. Alexandr Rassadin, Alexey Gruzdev, Andrey Savchenko(2017). Group-level Emotion Recognition Using Transfer Learning from Face Identification.

[5]. https://www.tensorflow.org/hub/tutorials/image_retraining

[6]. https://www.tensorflow.org/install/source

[7]. https://docs.docker.com/toolbox

Cite this article as :

Sairama Geetanand, Shaik Nazeer Basha, Sajja Nageswara Rao, Rayudu Srinivas , "Emotion Detection using Facial expressions with Transfer Learning ", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 5 Issue 1, pp. 229-234, January-February 2019.

Available at doi :
https://doi.org/10.32628/CSEIT195150

Journal URL : http://ijsrcseit.com/CSEIT195150