An attempt of distinguishing human and chimpanzee in webcams and web scraper

By Wing Cheong Lo

Pasadena City College
Under supervision of Prof. Jamal Ashref

Introduction

Human verification is an important topic in the area of deep learning and many companies such as google, baidu spend lots of time on improving their model. Among all human verification artificial neural network models, faceNet from Google is the most successful since 2017. The model itself has a very high successful rate of 99.57% of detecting a human face. Detecting human face seems to be the best way of detecting human from a photo because human faces are unique and they are quite different from other animals. However, multi class classification is very hard to do in the area of deep learning because lots of photos are required to train the neural network. **This project aims at making a model which can distinguish human face and chimpanzee object from a photo.** Totally eleven models are implemented during the research. Among them, model "trainedModel-noCNN.h5" has the highest accuracy to diagnose a human object from a photo taken by the webcam. This may highly because **pooling and filter have reduced the amount of features in the photo. Therefore, our model with only one convolutional neural network (which is implemented in the input layer) has a high successful rate because it retains features in a selfie (such as background human selfies).**

Lo2

Tensorflow model

(loaded from `tf.keras.models.load_model(trainedModel-noCNN.h5).summary))`

```
                          Model: "trainedModel-noCNN.h5"
_________________________________________________________________
Layer (type)                 Output Shape                 Param #
=================================================================
conv2d (Conv2D)              (None, 298, 298, 16)         448
_________________________________________________________________
flatten (Flatten)            (None, 1420864)              0
_________________________________________________________________
dense (Dense)                (None, 25)                   35521625
_________________________________________________________________
dense_1 (Dense)              (None, 1)                    26
=================================================================
Total params: 35,522,099
Trainable params: 35,522,099
Non-trainable params: 0
#################################################################
```

Image input sizes : 300 pixels x 300 pixels x 3 color bytes
Loss : 0.363
Accuracy : 0.8671

Used formula

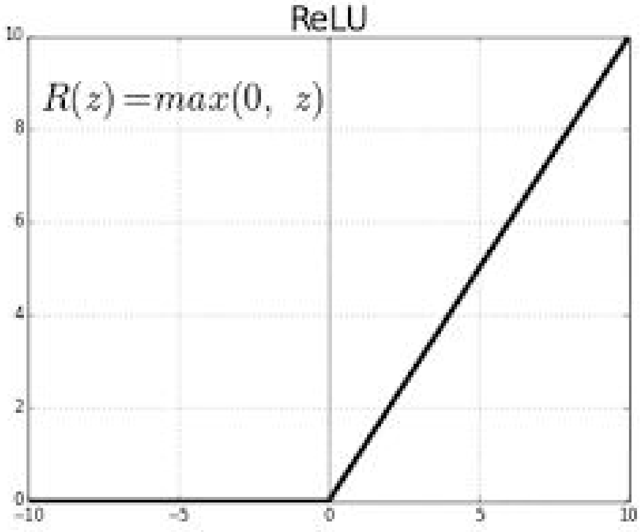
Input layer and hidden layer : relu
Output layer : sigmoid
Loss function : binary_crossentropy
Optimizer : SGD (Stochastic gradient descent

Model description

```
model = tf.keras.models.Sequential([

        # Note the input shape of all images are 300 x 300 x 3 bytes
        # Input layer
        # kernel size  : 16
        # fiter size   : 3x3
        # activation   : 'relu'
        # input_shpae  : 300 x 300 pixels with 3 bytes color
        # pooing layer : 2x2
        tf.keras.layers.Conv2D(16, (3,3), activation='relu',
input_shape=(300, 300, 3)),
        tf.keras.layers.MaxPooling2D(2, 2),

        # flatten the layers to be 2D matrix
        tf.keras.layers.Flatten(),

        #Sigmoid is used for two class while softmax is used for multipule
class
        # The main reason why to use sigmoid function is because it exists
between (0 to 1).
        # Therefore, it is especially used for models where we have to
predict the probability as an output.
        # Since probability of anything exists only between the range of 0
and 1, sigmoid is the right choice. Sigmund works better in
binary_crossentropy
        tf.keras.layers.Dense(1, activation='sigmoid')
    ])
    from tensorflow.keras.optimizers import SGD
    Optimizer = SGD(lr=1e-3, momentum=0.3, decay=0, nesterov=False)

    # comparison of two class
    model.compile(loss='binary_crossentropy',

                # after multiple attempts
                #   SGD is found to be most effective because
                # the loss significantly decreases after each epoch
                optimizer= Optimizer,
                metrics=['acc'])
    imageData_trainerGenerator = normalization.flow_from_directory(
        # path to store the training data set

"/home/kiroslo/pcc/deepLearningFinalProject/FINALPROJECT--CS003C/training_
data_set",
        # input size of the images
        target_size =(300, 300),
        # each batch : total number / 30
        batch_size = 30,
        # all images are stored as binary
        class_mode = 'binary')
```

Model choice explanation

Input layer and hidden layer -- relu

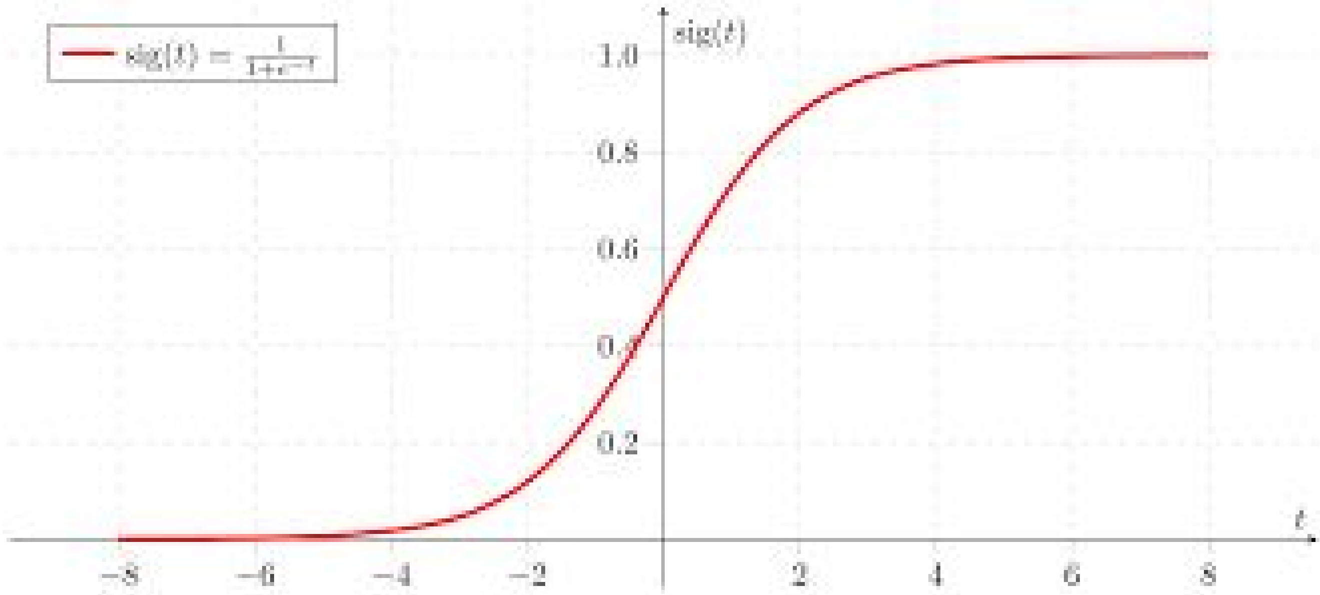


$$f(x) = x^+ = \max(0, x),$$

The formula , relu (**rectified linear unit)** is widely used in the area of Image processing. This is because relu has a formula of max (0,x), which means that it will ignore all negative data and treats all positive data in 1 to 1 ratio.

In this project, **formula relu is used as the activation of input layer and hidden layer because the value of Image pixels ranges from 0 to 255** (even range from 0 to 1 after normalization (Normalization means dividing the value of RGB by 255.0)). So when the neural network is processing the images data, all features will be retained.

Output layer -- sigmoid



In this project, Sigmoid instead of Softmax is used as the Output layer function because of relu and binary classification. Typically, Softmax function seems to be a more steady choice in most cases. But Sigmoid is used over here because of normalization (Image pixels RGB value / 255.0).

In faceNet, google uses Softmax instead of Sigmoid because Softmax is a better choice than Sigmoid in terms of multi class classification. However, Sigmoid function works better for binary classification. **The main reason to use sigmoid function is because RGB value exists between (0 to 1).** Therefore, it is especially used for models where we have to predict the probability as the output. Since the prob of anything exists only between the range of 0 and 1, Sigmoid is the right choice.

Loss -- binary_crossentropy

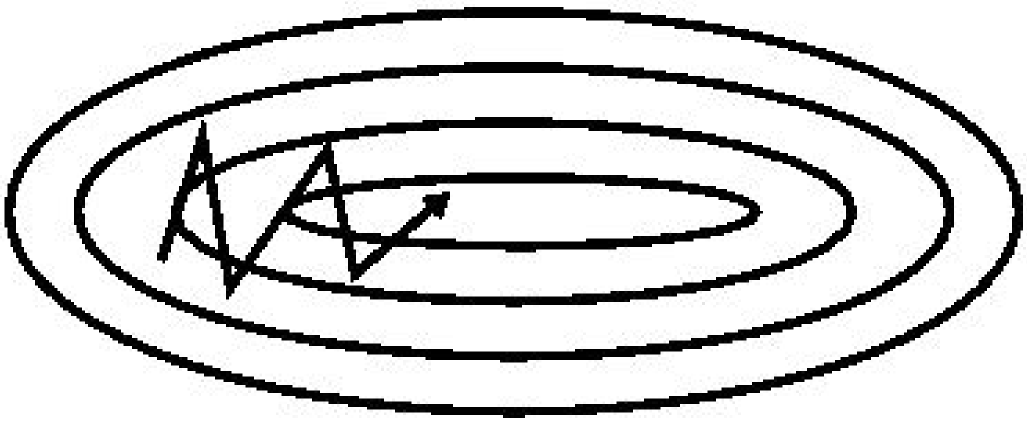
Binary_crossentropy is a loss function used for two class classification. It works excellently for comparison between two classes. In this project, there are few reasons for the implement use of Binary_corssentropy. Firstly, chimpanzee and human (including body as well) are basically two classes. Secondly, Binary_crossentropy is a cheap loss function because it requires not much datasets to train a neural network AI to be able to distinguish a human or a chimpanzee.

However, there are many limitations for this loss function. Firstly, it only enables an AI to distinguish two classes. Objects other than Chimpanzee and human cannot be effectively distinguish. Therefore, when we put a photo containing horse object to test by implementing the model, it may not get a result that what we want **because the neural network has never been trained with this type of objects**. **The object is rare to the neural network.** It will randomly predict the object to be chimpanzee and human.

Google faceNet uses **triplet loss function** for their model. *Triplet loss* is a *loss function* for artificial neural networks where a baseline (anchor) input is compared to a positive (truthy) input and a negative (falsy) input. This means that triplet loss function needs at least two photos for the same object. Based on this requirement, the developer of faceNet believes that their best model, [VGGFace2](https://www.robots.ox.ac.uk/~vgg/data/vgg_face2/), has a dataset consists of ~3.3M faces and ~9000 classes, which means that it is a very expensive loss function.

This project uses 6700 photos, 3400 photos for human faces selfie and 3300 Chimpanzee photos. **Such a small dataset will not be enough for triplet loss function.** Therefore, binary_crossentropy as the loss function is used for the project, even if it is not the best loss function formula in human face recognition.

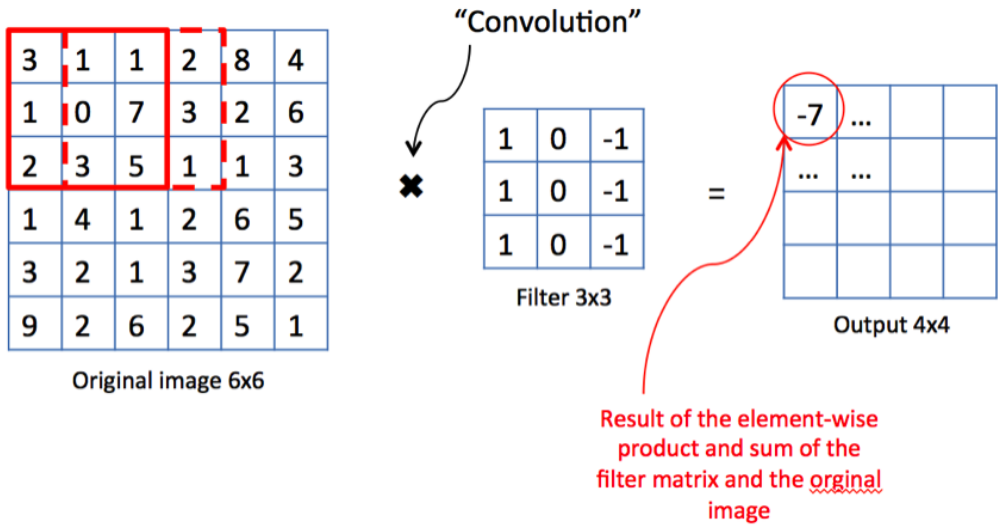
Optimizer -- SGD (Stochastic gradient descent)

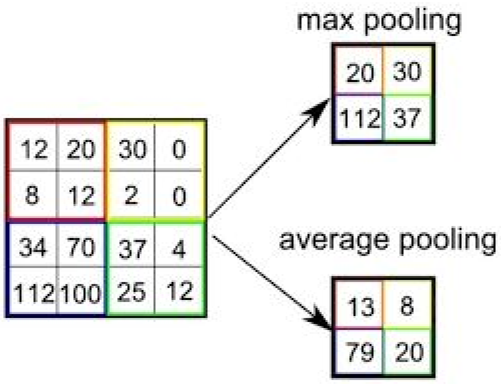


Eleven neural network AI models are trained in this project, and SGD has a higher accuracy rate than RMSprop and adam. This is perhaps SGD doesn’t automatically adjust the loss but RMPprop and adam does that. **When it comes to training the model, the loss sometimes increases after each epoches when using RMPprop and adma.** However, loss should be as low as possible when it comes to Binary_crossentropy. Low loss rating is a good indicator which may mean the AI has a very high tendency to distinguish two classes accurately.

Because of the property of SGD, loss when gradually decreases after each epochs. Loss decrease approximately 0.05 after each epoch.

Drawback of using more convolutional neural network





CNN, Convolutional Neural Network is widely used in most deep learning project. The reason for this is because CNN removes most useless pixel data and retains most important feature for the image. When it comes to human face selfie, this skill are highly used to detect important part of human face such as nose shape, fur color.

However, in this project, model has no convolutional neural network is found to be the best when it comes to predict a human by using a webcam. This may be because when no many pooling and filter is used, most features of a photo is retained, including the background to take a photo. **According to the dataset,**

most background of 3300 chimpanzee photos are forest, jungle. And background of 3400 human selfies has very few green color. This leads to the result that when using a webcam to take a selfie, it has a very high probability to be predicted to be a human object because of the color of the background.

However, this may also mean that it would be highly possible that a human who takes a selfie inside a forest would be distinguished as a chimpanzee because the model is trained with all RGB values of a photo.

Limitation

The result of not using convolutional neural network could be serious. This is because most features of a photo are retained in the training data set. **The trained model may treat background color as an important factor to distinguish between a human and a chimpanzee.** Therefore, when a chimpanzee takes a webcam inside a room, or a human takes a webcam in a forest, would be wrongly classificated.

When using the program to download photos with web scraper, some chimpanzees photos are wrong diagnosed as human. This could be because when a photo is downloaded from a web scraper, the size of the photo is changed. **The content inside the photo is rare to the model and therefore it makes a wrong prediction.** This may mean that the way to design the model should be changed and triplet may be a better loss function than binary_crossentropy when it comes to human verification. Because triplet enables comparison between multiple classes.

Citation

https://www.kaggle.com/slothkong/10-monkey-species

Alexander Freytag and Erik Rodner and Marcel Simon and Alexander Loos and Hjalmar Kühl
 "Chimpanzee Faces in the Wild: Log-Euclidean CNNs for Predicting Identities and Attributes of Primates"**,
German Conference on Pattern Recognition (GCPR), 2016 .

humanSelfieDataSet -- University of Central Florida {
@inproceedings{kalayeh2015selfie,
title={How to Take a Good Selfie?}