Dog Image Mood Classification Convolutional Neural Network

Deep Learning and Natural Languages Processing Project

Kfir Goldfarb, Shlomo Glick
Ariel University, Ariel

1. Abstract

Since convolutional neural networks models became the best way to learn on images in deep learning and artificial intelligence (for example: recognizing objects, apply filters etc.), one of the most common tasks known used by the convolutional neural networks models is image classification, this task is widely used in big applications and startup products.
In this paper we will show our experience with convolutional neural networks and how we trained a big convolutional neural network on complex data.
Firstly we can see related projects with the same task solving on the same data we used:

a. https://www.kaggle.com/code/sasakitetsuya/dog-emotion-classify-model-by-mobilenetv2-vgg16 - used softmax for all the 4 labels of the images and got accuracy of **50%**.
b. https://www.kaggle.com/code/kawonki/dog-s-emotion-predict-model - used softmax for all the 4 labels of the images and got accuracy of **41%**.

As we can see both related projects above has a low accuracies, this is because the data is very complex (well described below in the 5.a section), we wanted to get a simpler model that have only two labels for getting more accurate and strong model, also we was limited with machine performance power so we wanted to focus on less images data and more accurate model.

2. Introduction

Usually, images are represented in form of RGB matrix, each image matrix like this separated to 3 different table, each represent a colour channel scale of red, green or blue, each cell in table like this contains a colour value between 0 and 255, we want to train our network weights based on those values and this task can be very complex.
The main challenge of training our model is the large amount of calculations each train epoch and also how to build a fine convolutional neural network architecture that is able to learn a lot of images data.

3. Related Work

In the field of image classification in deep learning, we can split our work to two main branches:
How to build and train a convolutional neural network and how to use optimization and parameters manipulation to make the learning process better for a long term throw epochs (learning levels).
How to manipulate the images data using resizing, colour change and normalisation of colours values to make the neural network able to learn better from the images data.

4. Required Background

Basic knowledge in applied mathematics such as probability, calculus and linear algebra.
Knowledge in machine learning core topics such as optimization and convolutional neural networks.

5. Project Description

a. Data set

In this project we took the dataset from kaggle's dog emotions prediction and used the 'happy' and 'sad' folder, the motivation for taking those specific images is that the images are very complex and each image has its own 'distribution' values of colour and pixel structure, this is a big challenge for our network training.
We train our model over 9000 images (splitted equally for happy and sad image), originally each image in this dataset is in the size of 384x384 pixels and each has 3 tables for the RGB channels.
Before we trained our model on the data we normalise each image pixels values to be in range of 0 and 1 by using the normalisation equation: $V(p, c) = \frac{oldV(p,c)}{255}$, when V is the value of pixel p in channel c

Happy dog dataset sample:

Sad dog dataset sample:

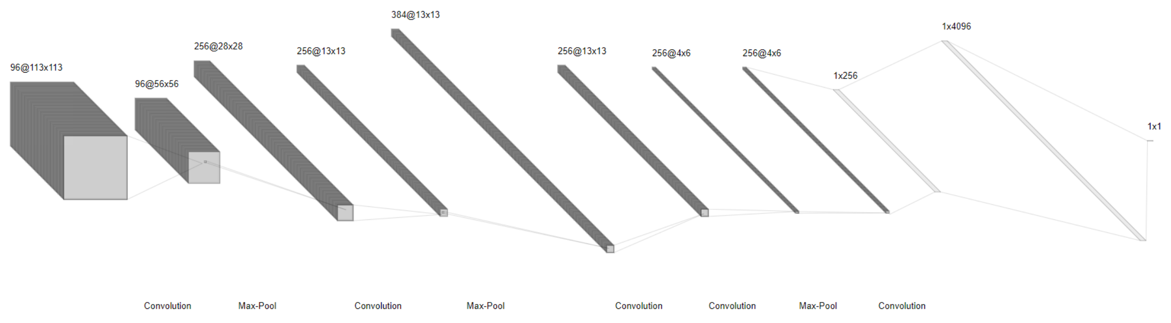
b. Problems with the data images

As we mentioned above, each Image in this dataset has its own aspect meaning different ‘distribution’ values of colour and pixel structure this is because each image was taken in a different view, angle, position, light etc. This was a big challenge for training our model and we felt this difficulty.

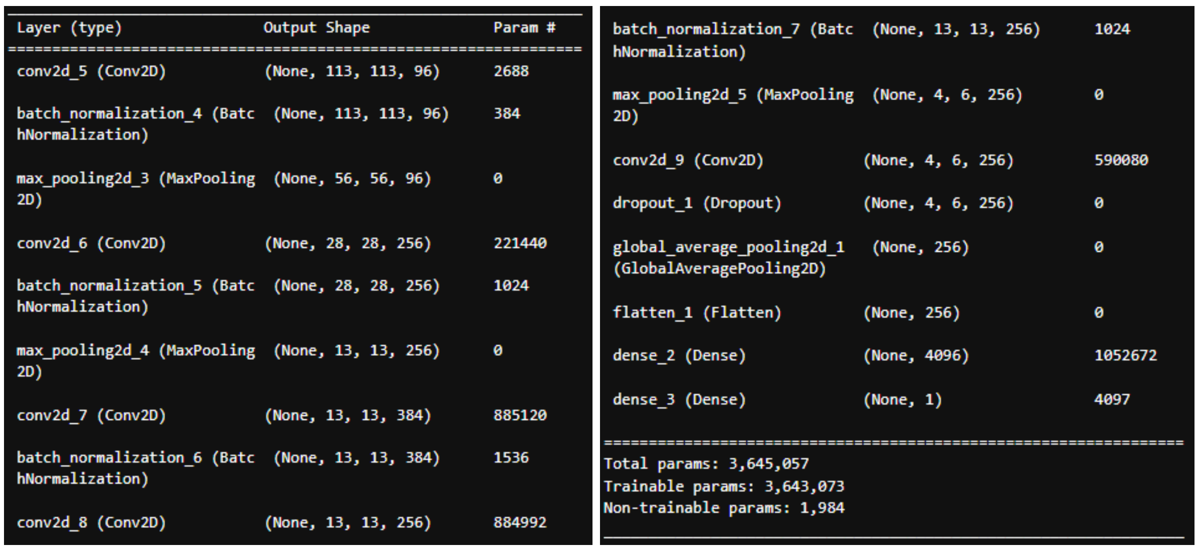
c. Model

The main idea of our model architecture is to used common activation function each layer, using max pooling and convolutional layers, the goal of the model is to perform a binary classification between ‘happy’ and ‘sad’ dog images, this is because our task to to classify only two classes, instead of using the softmax method we used the sigmoid function and binary cross-entropy method.

Model architecture illustration:



Model summary and parameters:



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 113, 113, 96)      2688

batch_normalization_4 (Batc  (None, 113, 113, 96)      384
hNormalization)

max_pooling2d_3 (MaxPooling  (None, 56, 56, 96)        0
2D)

conv2d_6 (Conv2D)            (None, 28, 28, 256)       221440

batch_normalization_5 (Batc  (None, 28, 28, 256)       1024
hNormalization)

max_pooling2d_4 (MaxPooling  (None, 13, 13, 256)       0
2D)

conv2d_7 (Conv2D)            (None, 13, 13, 384)       885120

batch_normalization_6 (Batc  (None, 13, 13, 384)       1536
hNormalization)

conv2d_8 (Conv2D)            (None, 13, 13, 256)       884992
```

```
batch_normalization_7 (Batc  (None, 13, 13, 256)       1024
hNormalization)

max_pooling2d_5 (MaxPooling  (None, 4, 6, 256)         0
2D)

conv2d_9 (Conv2D)            (None, 4, 6, 256)         590080

dropout_1 (Dropout)          (None, 4, 6, 256)         0

global_average_pooling2d_1   (None, 256)               0
(GlobalAveragePooling2D)

flatten_1 (Flatten)          (None, 256)               0

dense_2 (Dense)              (None, 4096)              1052672

dense_3 (Dense)              (None, 1)                 4097

=================================================================
Total params: 3,645,057
Trainable params: 3,643,073
Non-trainable params: 1,984
_________________________________________________________________
```

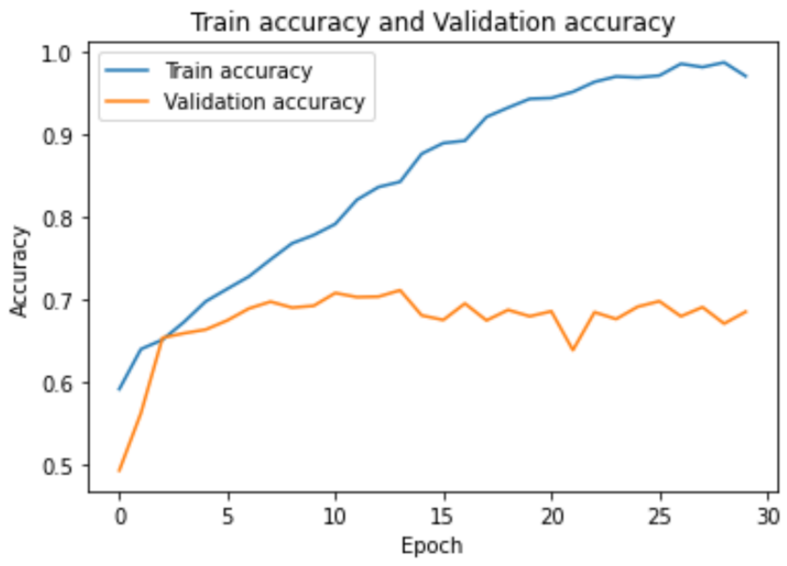
d. Training

Each image we used to train our network was in size of 227 x 227 with 3 channels of RGB so each image dimension was (3, 227, 227), we used 9000 images in the training process when we took 20% of the data for validation set.
Before training our model we shuffle the data that the training will train in random images.
We trained our network for 30 epochs using 0.000001 learning rate each, and batch size was 32, the optimizer we have used is the Adam optimizer.
Each epoch we tested the model to show us the train and validation loss value and accuracy.

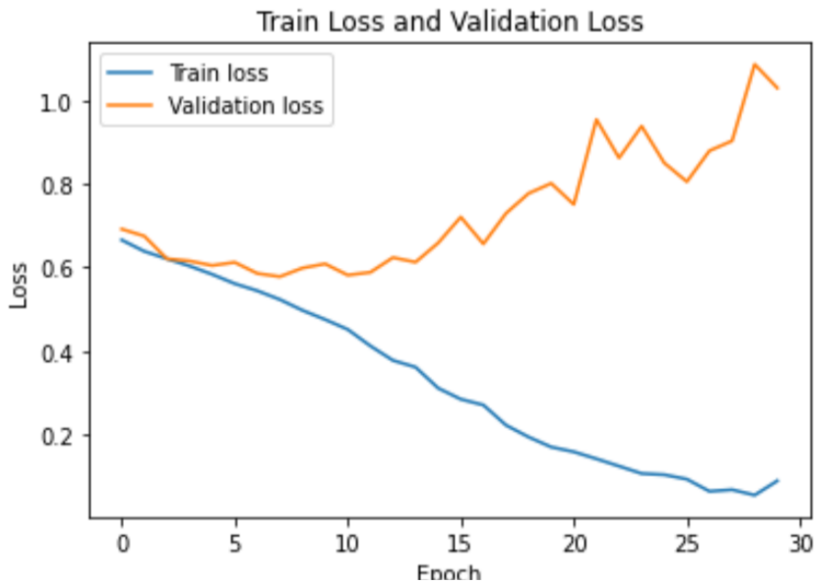
e. Results

The final results of training the model was:
Validation set accuracy value: **76.40%** and loss value: **0.5434**.
When the train set accuracy value: **80.78%** and loss value: **0.4165**.
We can see the graph those values we approached each epoch while we trained the model:

Accuracy:



Loss:



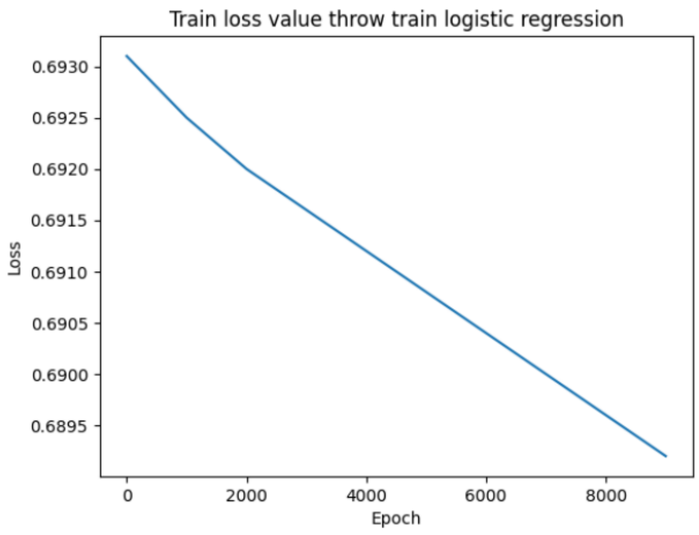
f. Conclusions & Discussion

Our main conclusion on this project was that it is very difficult to train a convolutional neural network on large scales and big calculations especially with our complex images dataset when each image has its own aspect, in addition because our model was big network and we trained it on a large amount of data the training process required a high performance machine with good CPU, GPU and RAM, the total time we spend only on training our model for all the attempted we did is ~80 hours total, to make the model more accurate and better we needed to train our model on a bigger dataset and to make our network big but we couldn’t do it because of the need of a very high performance machine that we cannot afford.
But with the resources we have we think we done our best and approche good results model for this type of images dataset.

6. Previous Attempts

a. Logistic Regression

In previous attempts we have implemented a logistic regression model using TensorFlow 1.x library, we train this model on 4800 images each image was in dimension of (3, 150, 150), the model was trained for 10000 epochs with learning rate of 0.00000001 and using Gradient Descent Optimizer to minimise the loss function of the logistic regression.
We can see the loss value of train dataset each epoch we trained the model:



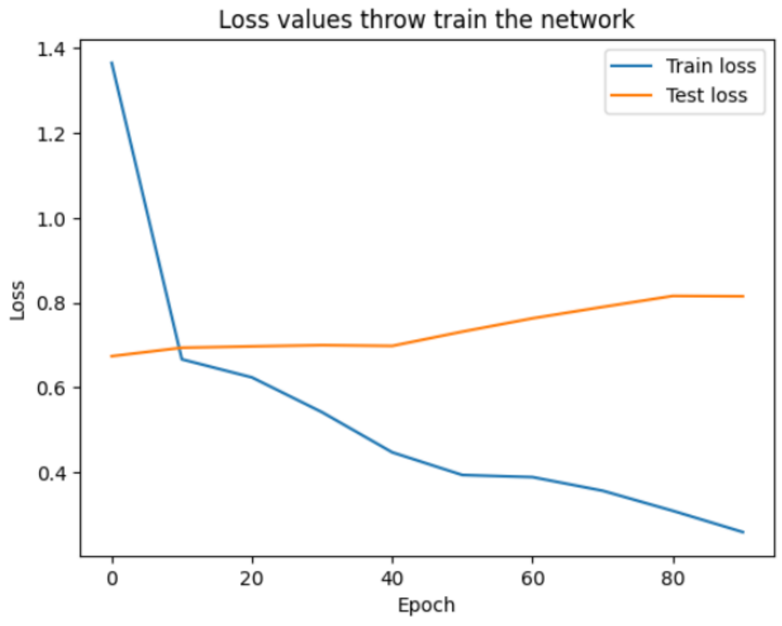
We used 1200 images for the test set for this attempt and got testset accuracy of: **54%**.

b. Basic Neural Network Classifier

In this attempt we implemented a basic neural network classifier using TensorFLow 1.x library, we used 4000 images to train our model and more 1000 image for testset, each image dimension was (1, 224, 224) using grey scale, the network architecture is:

1. Input: 224 x 224
2. First Dense: 2000 neurons
3. Second Dense: 700 neurons
4. Third Dense 200 neurons
5. Fourth Dense: 70 neurons
6. Output: 2 neurons

We train the network for 100 epochs and each 10 epoch calculate the training loss and test loss value:



We tested the model on the 1000 images test set and got accuracy of **60%**.