Human Emotion Classification using Deep Convolution Neural Networks

Krishna Pranay Angara

Department of Computer Science, 2022.

University of Central Florida.

Abstract

The project discussed is based on the Image classification problem using deep convolution neural networks. The convolution networks use multiple hidden layers by taking in the data extracted using beautiful soup framework to classify the emotions of the human object present in the images. The multiclass model generated learns through the data of images to recognize the emotions. The images are classified into happy and sad while feeding the model. Multiple activations are applied to understand the tendencies of accuracy, based on activation functions and the number of layers used in the network. The best model based on the ROC curve is used for the prediction.

1.Introduction

Computer Vision is a sub-branch of Artificial Intelligence. Computer Vision deals with Image and object recognition through real time or on existing data sets to understand the object and its surroundings. In the year 1959, popular work by Hubel and Wiesel [1] discussed how the visual cortex in cats' brain handles understanding spatial data field. The visual image is not taken at once rather, the receptive fields would be covering smaller regions of this spatial data and multiple receptive fields with overlapping spatial regions fire neurons to the visual cortex. Together they form the complete spatial data field. Hubel and Wiesel proposed a technique for using a similar method to identify patterns in a dataset. Followed by this work a basic convolution neural network has been created in 1980 by Knihiko Fukushima [3]. The convoluted layers take multiple smaller regions of the earlier layer, and the feature extraction is done through spatial averaging. The final layer correctly identifies the object. Later in the year 1994, the first ever Convolution neural network for computing high resolution images has been successfully implemented by Yann LeCun, Leon Bottou, Yosuha Bengio and Patrick Haffner. Named after Yann LeCun, the LeNet-5 architecture discusses the convolution layer modules including the input layer, Sampling layer or feature extraction layer, fully connected layer and the final output layer [2]. Feature extraction stands out as being a crucial step as these values would decide the learning points of all the neurons for n hidden layers.

Max Pooling has been a popular feature extraction method. The maximum elements of a region from previous layers are considered and resized to send through the next layer of neurons. This

preserves the key features and reduces the layer size efficiently. The concept of max pooling was introduced by Yamaguchi in 1990 [4].

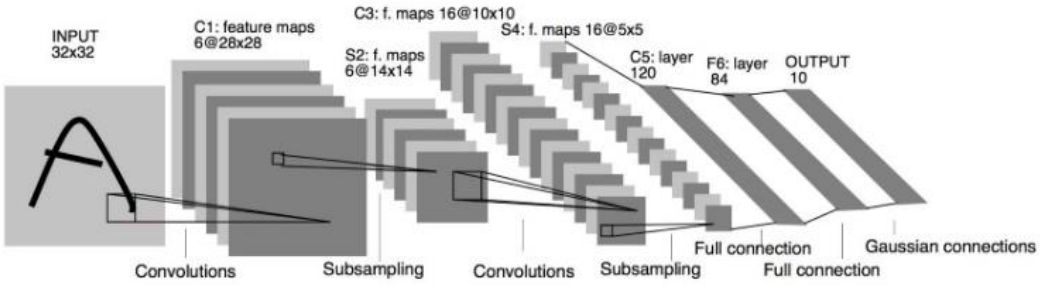

Figure 1: The Conv Net Architecture (LeNet5) created by Yann LeCun, Leon Bottou, Yosuha Bengio and Patrick Haffner.

Computer Vision has come a long way from trivial Matrix operations for image manipulations to using real time video in aiding major health and defense applications. Mathematics at its core, we will try to understand how we could use convolution neural networks in object detection and how labeled data could help the model predict emotions of a person. We use supervised learning for training our model with train data and then evaluate it with random images.

Neural networks, much like our brain, use neurons to learn parts of data from the training set and give us a prediction based on previously learnt values. Emotions give meaning and purpose to human life. They are a particularly important phenomenon that affects human ways of life. We will try to build a model that can predict 2 very important emotions out of many, i.e., happy and sad. The neurons in our brain communicate via electric signals, The neurons in the neural network communicate through numbers and mathematical operations.

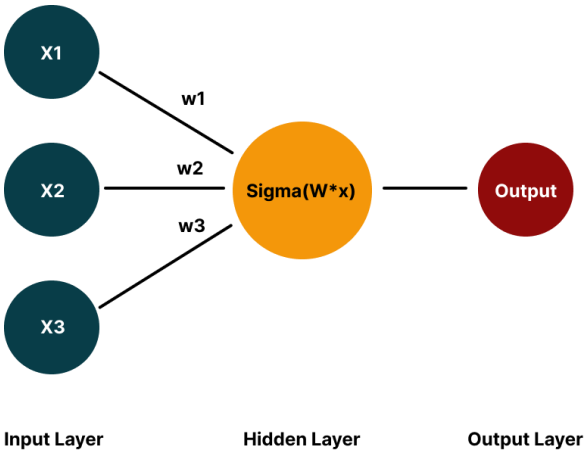

Figure 2: The Neural Networks would be taking in the inputs of x1, x2 and x3 in the input layers and are passed through hidden layer where we could use an activation function for predicting the output.

Several proposed frameworks help us in building neural networks seamlessly. The precision and accuracy of these models could be managed via hyper parameter tuning. Regardless of the model efficiency, data integrity and correctness are huge factors in Image detection. Normalizing the data for removing noise and data being independent of geometric distortion is crucial.

2.Method

Through web scraping using beautiful soup framework, happy and sad data is collected. The Source would be taking in the request URL and based on the source link would collect images in that webpage. We could modify this URL to include webpage 2,3 and 4 to collect data. The images would then be labeled based on the image download number and stored in the image path specified by us. A total of 480 images have been considered regardless of the image resolution. Some of the images with their labels could be shown in figure 3.

Import all the necessary packages and libraries for operating on image data and model generation. TensorFlow using keras is utilized for neural networks. Various functions such as sequential, conv2D, MaxPooling2D and Dense layers are imported for tuning the neural network. A roc curve is imported to estimate the best network for prediction. AUC and precision enable us to measure the accuracy of models.

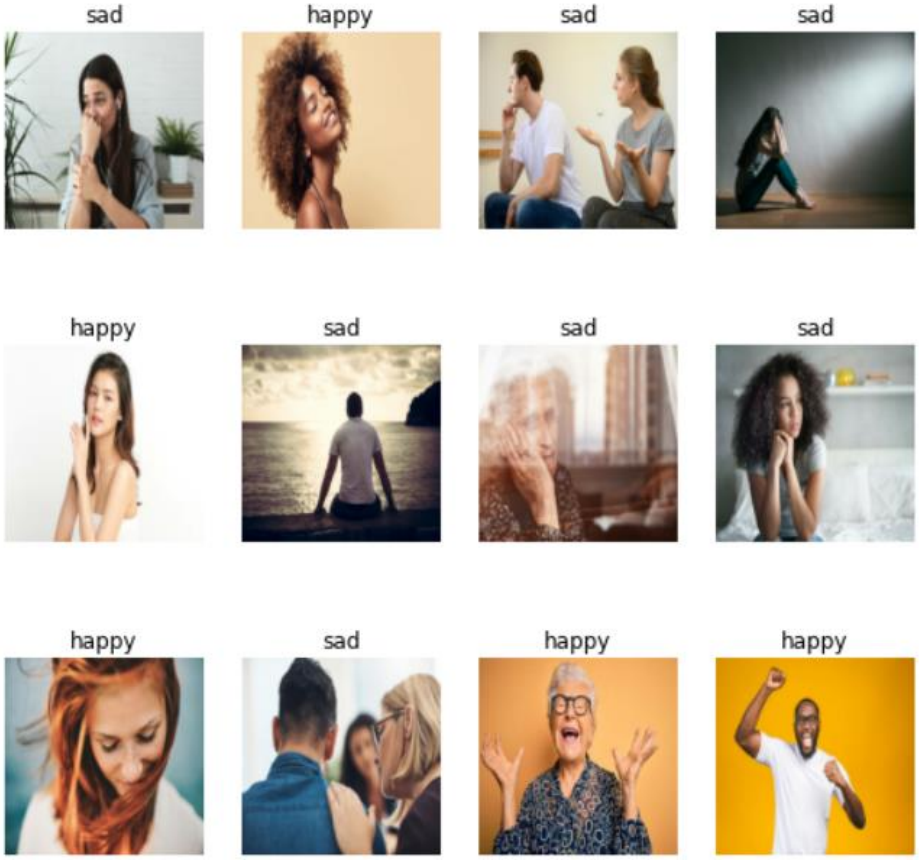

Figure 3: Some of the images with their labels could be shown above.

The images would be considered as NumPy matrices. The values would range from 1 to 255. Working on this data would be computationally intensive. For more efficiency we will normalize the data and reduce the pixel values. Split the data to train, validation and test values.

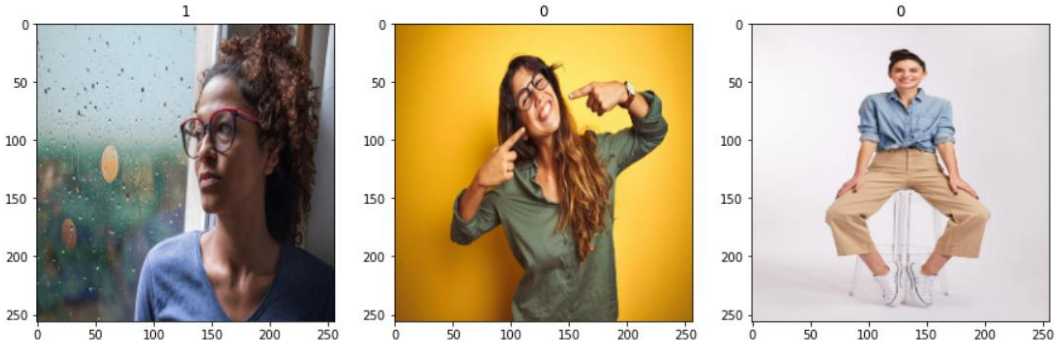

Figure 4: The 1 value is taken as sad images and 0 as happy images.

Understanding various activation function combinations with varying number of layers is the primary aim of the paper. There are several nonlinear activation functions that could be applied on complex convolution networks such as sigmoid, Relu and Tanh. Throughout the discussion we will be considering the different number of layers and mainly 3 combinations of activation functions mentioned above.

According to Tomasz Szandala, Relu has much better execution and training time when compared with other activation Functions. We would analyze if this true using our convolution neural networks. The tendencies of activation efficiency if calculated based on training 10,000 images of CIFAR10 dataset with different activation functions [5]. The results and findings can be seen in figure 5.

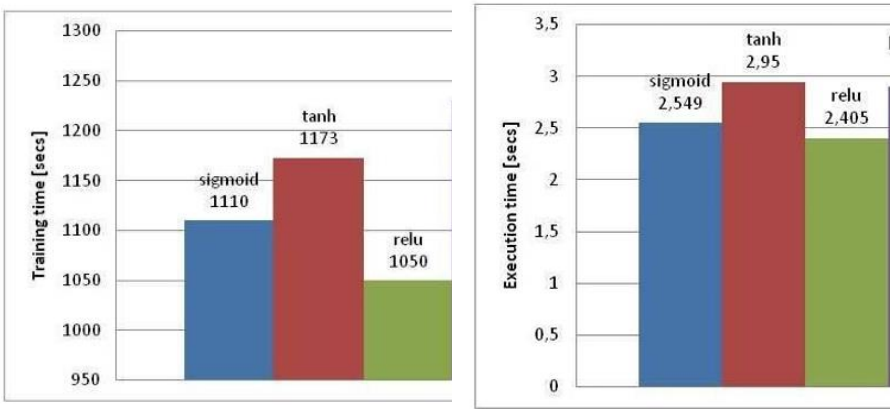

Figure 5: Training and execution analysis

2.1 Model 1

The Model 1 takes in an input size of 256*256 with 3 channels. The convolution layers take in the images and are sent through 16 layers of neurons where the kernel size of 5*5 would be applied to those 16 kernels and the results would be sent to 32 layers of neurons and then again

through 16 layers. The Kernel size and padding are consistent across all the layers. The hidden layers would be using RElu while the final dense layer uses sigmoid activation function. RElu or a Rectified Linear Unit function outputs the input if the value is positive else zero (max{input,0}) as shown in formula 1.

The activation layer acts after the summation step and then defines output accordingly. The simple form of computation based on the formula below allows the model to run more smoothly and efficiently.

f(x)= {x, for x>0 0, for x<=0, {0, x}}

Formula 1: Relu Activation Function

The Rectilinear Activation Function is more commonly used in deep learning models because of this reason. The sparsity of a model is an important feature i.e., The number of unwanted neurons in the model needs to be ignored and the number of neurons that could provide necessary information needs to be considered. The number of unwanted neurons is significantly less in Relu as it nullifies the 0 values and will only consider the neurons with values greater than 0. The graph when considering inputs x and outputs f(x) or y can be seen in figure 6.

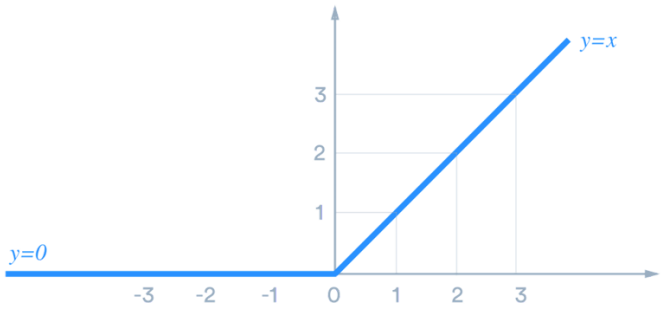

Figure 6: Rectified linear function graphical representation

2.2 Model 2

The emphasis is more on sigmoid activation function instead of Rectified Linear Unit. All the hidden layers would be using Sigmoid except the final layer which would be using RElu (Rectified Linear Unit). The sigmoid function has a range of (0,1). This helps us learn nonlinearly separable values. This synonymous logistic function helps us classify our outputs to either true or false. The mathematical equation for applying logistic function would be shown in the figure below.

$$f(x) = \frac{1}{1+e^{-(x)}}$$

Figure 7: Sigmoid function that return 0 or 1 based on input x and Euler's value e

Regardless of the usefulness of sigmoid activation function, when used along with RElu produced reduced accuracies when compared to model1. The increase in the number of hidden layers and epochs rendered no major improvement in improved accuracy.

2.3 Model 3

In model 3 we have considered the use of tanh (tangent hyperbolic function). An important concept in deep learning is understanding gradients. To extract a good amount of information in the images we need a good understanding of gradients. Gradients are the building blocks for image processing. When compared with sigmoid, tanh gives us greater gradients [6]. The mathematical equation for performing tanh can be seen in the following figure8 [7].

$$g(x) = \frac{e^x}{1 + e^x}$$

$$tanh(x) = 2g(2x) - 1$$

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

Figure 8: tanh function as an improvement of sigmoid function.

The higher the gradient information the more learnings would be. Thus, tanh has been considered instead of sigmoid to understand its influence in improving accuracy. The gradients when plotted could be as represented in figure 9.

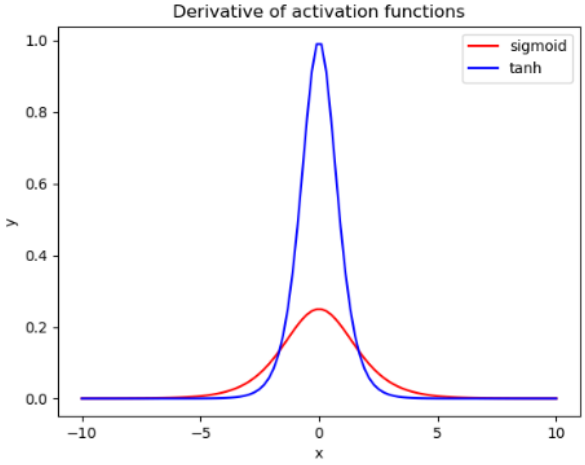

Figure 9: Tanh and sigmoid gradients

Evidently the results show that using tanh has improved accuracy significantly. However, the Model 1 using RELU activation provides much better accuracy and reduced loss values when compared. This is due to the vanishing gradient problem. We will discuss more about this in the discussion and analysis phase.

3. Results

The data has been split into train test and validation data points. The data sets thus will be stored in the image batches (ibatch), to be fed to the model along with their labels. The Image classification is binary classifier that takes in the Boolean values of 0 and 1. 1 being sad and 0 being happy. Model 1 has comparatively better results where the loss has been gradually decreasing to 0. The image batches are then fed into the model with the specified hyper parameters. The summary of operations would be stored in the logs using TensorFlow call backs. I have run the model for 45 epochs with a default batch size of 32. Learning rate would be 0.01. Interpolation for the image dataset would be bilinear interpolation. Interpolation helps in scaling the data for better learning and prediction capabilities. Model1 steadily improves in accuracy for 45 epochs and reaches a stagnation. Model 2 performs well but fails to achieve accuracy greater than 60%. This is due to the vanishing gradient issue discussed in the analysis section. Model 3 has improved accuracy and performance with a precision of over 70% due to the tanh activation yet does not exceed model 1 accuracy scores.

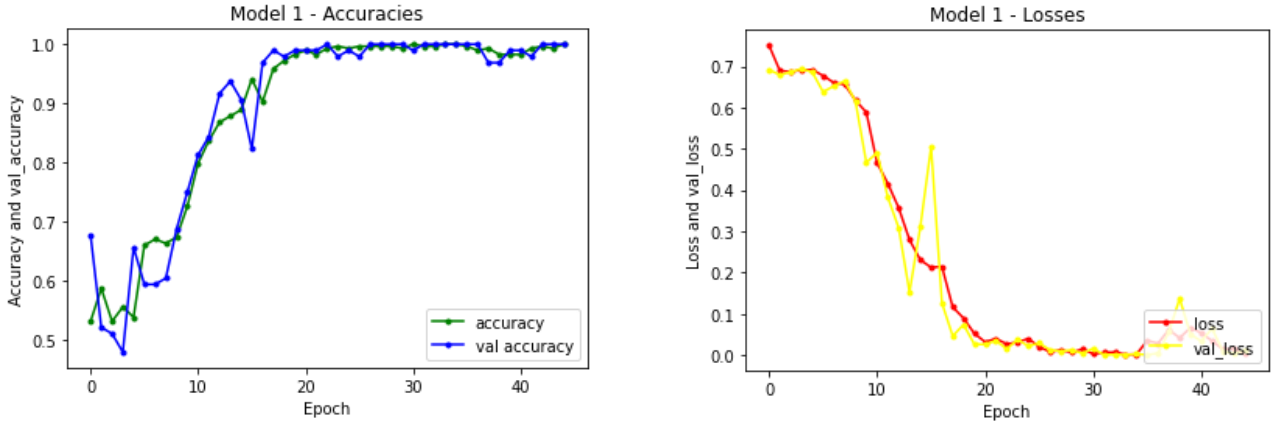

Figure 10: Model 1 plotted Accuracy and loss tendencies.

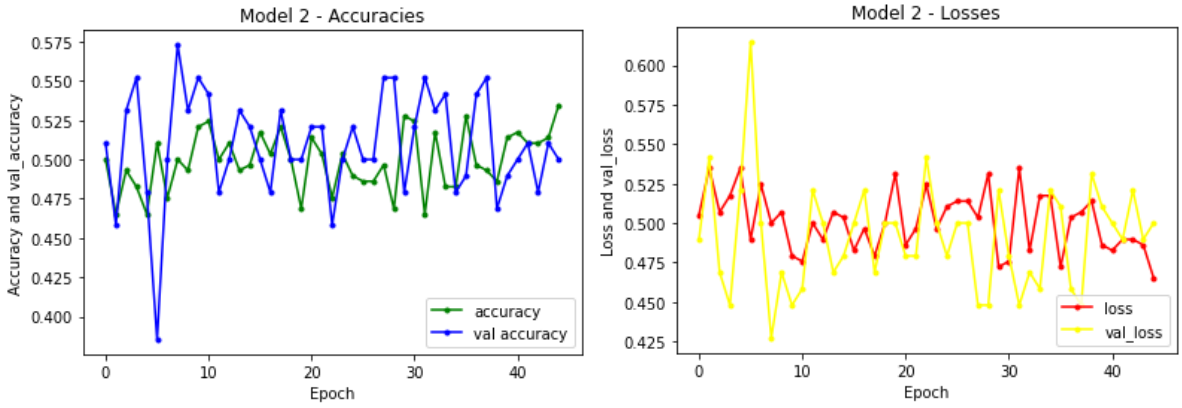

Figure 11: Model 2 plotted Accuracy and loss tendencies.

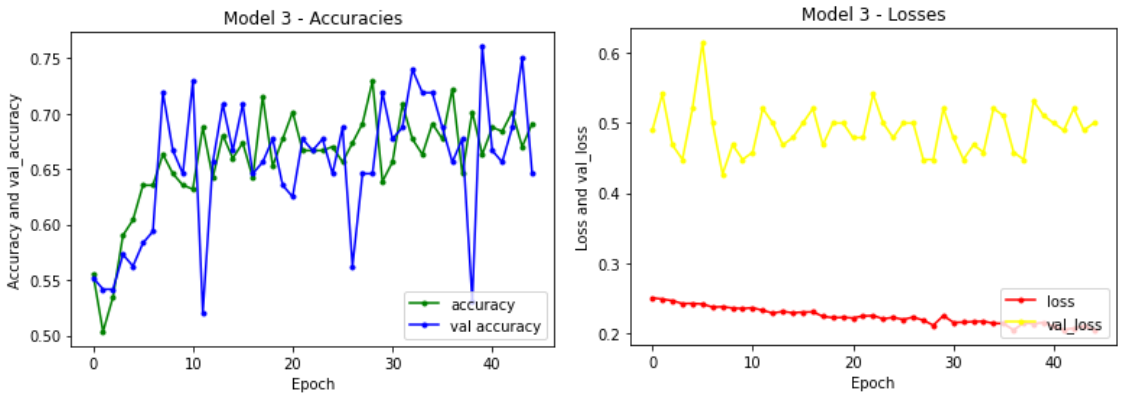

Figure 12: Model 3 plotted Accuracy and loss tendencies.

4. Discussion and Analysis

The deep learning neural networks are improved neural networks with more hidden layers. These perform exceptionally well with substantial amounts of data. However, the activation functions such as sigmoid still suffer from a phenomenon known as vanishing gradient problem.

The neural networks will take in the input and pass them to hidden layers and gives out an output, through this process the forward propagation uses varied weights and biases. The weights are multiplied with the inputs and the aggregate sum is added to bias which in turn goes through the activation function to give us the output. Backward propagation is the next step after forward propagation. A loss function gets calculated using the prediction from the forward propagation learnings as shown in figure 13. Here the y depicts the prediction and mx+b would be the output with m being weight and b the bias.

$$f(m,b) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + b))^2$$

Figure 13: The function for calculating the loss function.

The weights and biases are updated accordingly for a reduced loss function. At each iteration, the loss function is calculated using the derivative of earlier input and learning rate. The formula for calculating the gradient descent can be seen in figure14.

$$f'(m,b) = \begin{bmatrix} \frac{df}{dm} \\ \frac{df}{db} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum -2x_i(y_i - (mx_i + b)) \\ \frac{1}{N}\sum -2(y_i - (mx_i + b)) \end{bmatrix}$$

Figure 14: The function for calculating the loss function.

All the new weights are dependent on the old weights, since the main criterion for gradient descent is finding the minima and reducing the loss function, the gradient of loss function based on the derivatives of earlier inputs would be steadily decreasing till it's close to zero. This phenomenon is known as vanishing gradient problem. This problem has a significant impact for deep layers resulting in reduced accuracy. This problem persists for tanh and can be fixed only through the use Rectified linear unit function. Relu only has the derivative values of 0 and 1 thus giving us 0 only if the results are less than 0, this helps us prevent vanishing gradient problem.

However, from the figure (15) below we can see that the model 1 which uses relu quickly overfits the results to 1. This is due to the convergence, the very concept of relu is giving us a 0 or 1 output instead of some fraction between 0 and 1 like sigmoid and tanh. This means that the model will learn much faster and takes all the positive values and nullifies negative values. Thus, the model using relu converges much quicky than sigmoid and tanh giving us an accuracy of 100%.

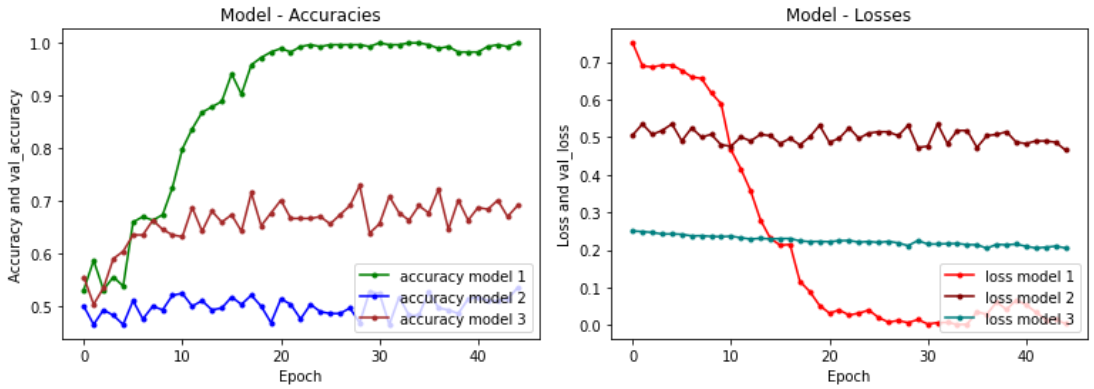

Figure 15: Accuracy and loss comparisons between the 3 models.

7. Conclusion

To Summarize, we have extracted datasets from the internet using beautiful soup, a web scraping tool and used preprocessed the data through normalizing and resizing it to train our three different models of convolution neural networks. The data has been further split into training, testing and validation splits and were fed into the models. The model 1 performed much better than model 2 and 3. The reason for the reduced prediction accuracy of the latter models is due to the phenomenon known as vanishing gradient descent. We have generated an ROC curve to visualize the best model.

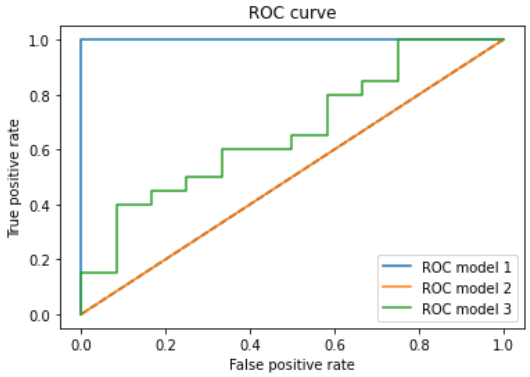

Figure 16: Accuracy and loss comparisons between the 3 models.

From Figure 16, the best model is Relu model and thus we have used it for the prediction of a test image. The outputs have proven that the images have been predicted correctly.

Sad Image prediction

Happy Image prediction

Figure 16: Test images and the model output.

8. References

1. Hubel, DH; Wiesel, TN (October 1959). "Receptive fields of single neurones in the cat's striate cortex". J. Physiol. 148 (3): 574–91. doi:10.1113/jphysiol. 1959.sp006308. PMC 1363130. PMID 14403679.

2. LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner (1998). "Gradient-based learning applied to document recognition" (PDF). Proceedings of the IEEE. 86 (11): 2278–2324. CiteSeerX 10.1.1.32.9552. doi:10.1109/5.726791. S2CID 14542261

3. Fukushima, Kunihiko (1980). "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position"

4. Yamaguchi, Kouichi; Sakamoto, Kenji; Akabane, Toshio; Fujimoto, Yoshiji (November 1990). A Neural Network for Speaker-Independent Isolated Word Recognition. First International Conference on Spoken Language Processing (ICSLP 90). Kobe, Japan.

5. Tomasz Szandała, Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks, arXiv:2010.09458 [cs.LG]

6. https://www.baeldung.com/cs/sigmoid-vs-tanh-functions

7. https://analyticsindiamag.com/most-common-activation-functions-in-neural-networks-and-rationale-behind-it/

8. https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html

9. https://github.com/nicknochnack/ImageClassification