ISSN: 2583-9292(ONLINE)
DOI: 10.xx/ijdsml.2023.xx

FACIAL EMOTION RECOGNITION AND HANDLING DATA IMBALANCE IN MACHINE LEARNING

Diya Elizabeth[1] and Siria Sadeddin[2]
[1]*School of Mathematics, Indian Institute of Science Education and Research, Thiruvananthapuram, India*
[2]*Department of Mathematics, Universidad Nacional de Colombia, Colombia*

Abstract

In this paper, we use deep learning to make an emotion recognition convolution neural network by customizing the EfficientNet model pretrained on the ImageNet dataset. We used the FER-2013 dataset available in the Wolfram repository. Seven classes of emotions are considered in the dataset: happy, sad, angry, surprise, disgust, fear, and neutral. We try out different methods to tackle class imbalance.

Keywords:

Undersampling, Weighted Sampling, Focal Loss

1. INTRODUCTION

Facial expressions are crucial non-verbal communication in our everyday lives. They indicate feelings, allowing people to express their emotions. Most humans have the ability to recognise the facial expressions of a person instantly. Here, we will train a neural network to do the same. This model can be applied in teaching autistic children, young children in general, and people blind to facial expressions. It can also be used to train robots to interact more efficiently with humans.

2. CONVOLUTION NETWORKS

Neural networks or artificial neural networks (ANNs) are inspired by the human brain, mimicking the way that biological neurons signal to one another. Neural networks are comprised of node layers; each node connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network.

A multilayer perceptron (MLP) is a fully connected class of feedforward neural network that consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. For basic binary images, using an MLP would be sufficient, but for more complex images having pixel dependencies throughout the model might be unsatisfactory. Using a convolutional neural network, we can capture the spatial and temporal dependencies in the image.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks; that is, each neuron in one layer is connected to all neurons in the next layer. The “full connectivity” of these networks makes them prone to overfitting data. A CNN consists of different layers, such as an input layer, an output layer and a hidden layer consisting of multiple further convolutional layers, pooling layers, fully connected layers and normalisation layers.

3. TRANSFER LEARNING

Transfer learning is a technique where a model trained for one task is reused for a second related task. Rather than creating a new network from scratch, which may take days or weeks to train, we can use a pre-trained model from the Wolfram Neural Net Repository as a starting point and customize it accordingly.

For this project, EfficientNet trained over the dataset ‘ImageNet’ was modified. EfficientNets demonstrate a significant performance gain both in accuracy and inference time compared to the existing classification models trained on ImageNet. Furthermore, the models are successfully used in transfer learning on datasets such as CIFAR-100, Flowers, Birdsnap, Stanford Cars and others, still outperforming the existing state-of-the-art nets.

3.1 CREATING THE NEURAL NETWORK

We start by calling EfficientNet model pretrained on the dataset ‘ImageNet’. In order to train a new model, we can use this architecture. We remove the last Linear and Softmax layers and add our own layers as required to customize the net to our dataset.

3.2 CUSTOMIZING THE NEURAL NETWORK

After the last 2 layers of the model EfficientNet are removed, we construct a customized neural network by adding new layers. This allows the model to learn new features from our dataset.

The layers contribute as follows:

- LinearLayer: Adds complexity to the neural network.
- DropoutLayer: Prevents overfitting by randomly putting off the neurons while training.
- BatchNormalizationLayer: Normalizes its input data by learning the data mean and variance.
- SoftMaxLayer: Contributes the activation function layer for multiclass classification.

4. FER-2013 DATASET

We used the FER-2013 dataset available in the Wolfram repository. It has 35887 greyscale images of faces with expressions categorized into one of the seven categories: happy, sad, angry, surprise, disgust, fear, neutral. It was made under challenging conditions, such as varying lighting, different head movements, and variances in facial features due to ethnicity, age, gender, facial hair, and glasses. This is important to not create a bias in our model.

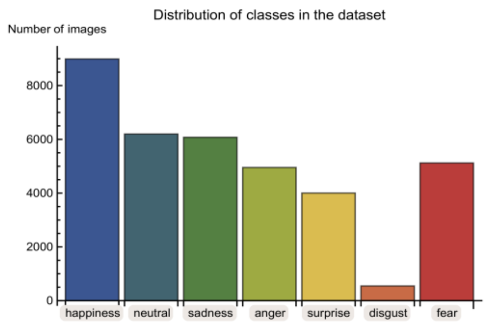


Fig.1. Distribution of classes in the dataset

4.1 DATA DISTRIBUTION OF CLASSES OF TRAIN AND TEST SETS

We chose 4000 random images from each class for our sample dataset. 70% of this sample dataset is chosen randomly as train dataset and the other 30% as test dataset. The function Random Sample is used to ensure that the data was chosen randomly to ensure an accurate representation of the original dataset in the train and test sets.

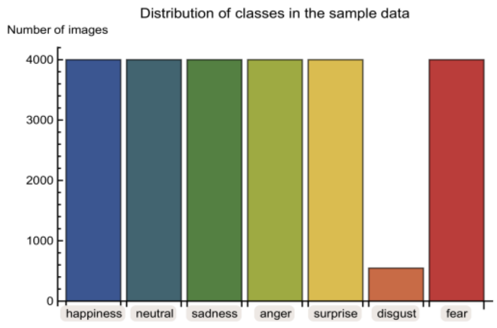


Fig.2. Distribution of classes in the sample data of train and test set

5. DATA IMBALANCE

It is evident from the graphs that the emotion class ‘disgust’ has a significantly less amount of data compared to the other classes. Most machine learning algorithms are based on the assumption that data is equally distributed among all classes in the data set. A model trained on a dataset with a bias will also be biased towards the majority classes; when training, the model learns more from the majority classes and develops a bias. The model would perform poorly in real-time. We will look at a few different ways in which we can resolve this issue.

5.1 UNDERSAMPLING

In the undersampling method, we remove the excess number of samples from the majority classes so all classes end up with about the same amount of sample data. Here, the class ‘disgust’ only has 547 samples. By taking 500 samples from each class for our sample data, each of our classes ended up with the same amount of data.

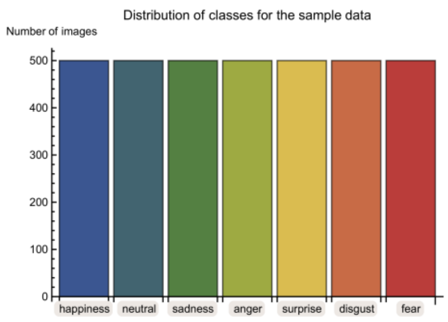


Fig.3. Distribution of classes for the sample data after undersampling

The model is trained with a batch size 24 and 2 training rounds.

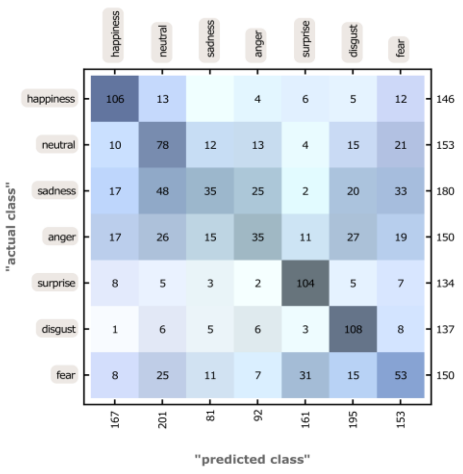


Fig.4. Confusion Matrix Trained Model

5.2 WEIGHTED CLASS TRAINING

In this method, we weigh the loss computed for different samples depending on whether they belong to the majority or minor classes.

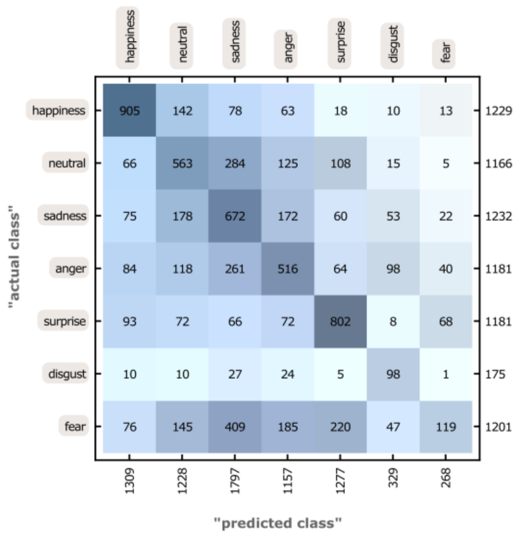


Fig.5. Confusion Matrix – Weighted Class Training

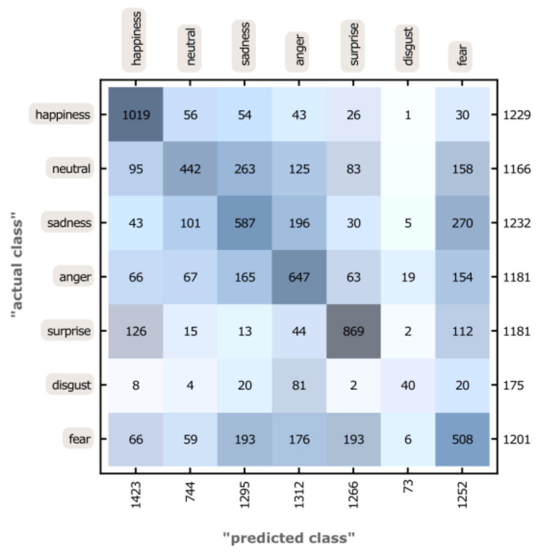
$$l_c(x,y)=L_c=\{l_{1,c},\ldots,l_{N,c})\}^T, \text{ where } N \text{ is the batch size} \quad (1)$$

$$l_{1,c}=-w_{n,c}[p_c y_{n,c}\log\sigma(x_{n,c})+(1-y_{n,c})\log(1-\sigma(x_{n,c})] \quad (2)$$

where, we use Inverse of Number of Samples: we weigh the samples as the inverse of the class frequency for their respective classes and then normalize them over the seven classes. Sample weight, $w_{n,c}$=1/(Number of samples in class c) , where c is the class number and n is the number of samples in the batch. The net is then trained with the custom loss.

5.3 FOCAL LOSS TRAINING

Focal Loss can be used when there is an extreme imbalance between the majority and minority classes. It is a modification of alpha-balanced Cross Entropy Loss. $CE(p_t)=-\alpha_t \log(p_t)$, where p_t is the probability of belonging to the class (0, 1) and α_t is the weight for each class. Large class imbalances overwhelm cross-entropy loss and dominate the gradient. Though α balances the importance of positive/negative examples, it does not differentiate between easy/hard examples. We add a modulating factor, $(1-p_t)^\gamma$ and a focusing parameter, γ to the cross entropy loss to produce the focal loss: $FL(p_t)=-(1-p_t)\log(p_t)$.



"actual class" \ "predicted class"	happiness	neutral	sadness	anger	surprise	disgust	fear	
happiness	1019	56	54	43	26	1	30	1229
neutral	95	442	263	125	83		158	1166
sadness	43	101	587	196	30	5	270	1232
anger	66	67	165	647	63	19	154	1181
surprise	126	15	13	44	869	2	112	1181
disgust	8	4	20	81	2	40	20	175
fear	66	59	193	176	193	6	508	1201
	1423	744	1295	1312	1266	73	1252	

Fig. Fig.5. Confusion Matrix – Focal Loss Training

6. CONCLUSION

By comparing the results from the three methods we used to address the data imbalance in the dataset, the accuracies obtained are as follows: Undersampling gives us 0.494286 accuracy, using FocalLoss gives us 0.558316 and weighted sampling, 0.498982. We can see that using FocalLoss gave us the highest accuracy and F1 scores. The weighted class model performed the next best. The accuracies could be improved in the future with more training rounds and network layers.

ACKNOWLEDGEMENT

There are many people I would like to thank for their support. I would first like to thank Siria Sadeddin for being my mentor, her immense help and guidance allowed me to learn and use many new skills which I could use to complete this project. I would also like to express my sincere gratitude to Tuseeta Banerjee and Mads Bahrami for their time and consideration, the TAs for being available whenever I was in trouble, and Aravind Hanasoge and the entire Wolfram team for conducting this program.

REFERENCES

[1] S. Sadeddin, "Face Mask Detection: Classifying Image Data", Available at https://community.wolfram.com/groups/-/m/t/2139499, Accessed on 2021.

[2] S. Sadeddin, "Loss Focal: Una Solucion Para El Desbalance De Datos. AI En Espanol", Available at https://siriasadeddin.wixsite.com/siriaai/post/loss-focal-una-soluci%C3%B3n-para-el-desbalance-de-datos, Accessed on 2020.

[3] S. Chandler, "Machine Learning with Weighted Data", Available at https://community.wolfram.com/groups/-/m/t/1223968, Accessed on 2018.

[4] I. Shrivastava, "Handling Class Imbalance by Introducing Sample Weighting in the Loss Function. GumGum Tech Blog", Available at https://medium.com/gumgum-tech/handling-class-imbalance-by-introducing-sample-weighting-in-the-loss-function-3bdebd8203b4, Accessed on 2020.

[5] T.Y. Lin and P. Dollar, "Focal Loss for Dense Object Detection", Available at https://arxiv.org/pdf/1708.02002.pdf, Accessed on 2018.

[6] S. Saxena, "Introduction to Softmax for Neural Network", Available at https://www.analyticsvidhya.com/blog/2021/04/introduction-to-softmax-for-neural-network/, Accessed on 2021.

[7] K. Singh, "How to Improve Class Imbalance using Class Weights in Machine Learning", Available at https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/, Accessed on 2020.

[8] D. Godoy, "Understanding Binary Cross-Entropy/Log Loss: A Visual Explanation", Available at https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a, Accessed on 2018.

[9] L. Hoeltgen, "Designing Neural Networks in Mathematica", Available at https://laurenthoeltgen.name/post/ml-nn-mathematica/, Accessed on 2015.

[10] G. Singh, "Facial Emotion Classification using Deep Learning", Available at https://medium.com/analytics-vidhya/facial-emotion-classification-using-deep-learning-d08dd02a2d38, Accessed on 2019.

[11] A. Chowdhry, "Emotion Recognition With Deep Learning On Google Colab". Available at https://blog.clairvoyantsoft.com/emotion-recognition-with-deep-learning-on-google-colab-24ceb015e5, Accessed on 2021.