Emotion recognition through facial expression

Ana Maria Jaramillo
anamaria.jaramillorojas@edu.fh-kaernten.ac.at
Saghar Ghaffari
saghar.ghaffari@edu.fh-kaernten.ac.at
Kärnten University of Applied Sciences

June 25, 2023

Abstract

Emotions play a crucial role in human socialization and daily life. Facial gestures provide valuable cues for identifying human emotions, making facial expression recognition a widely studied field. This study represents our first implementation of an artificial neural network and utilizes a data set from Kaggle's "Facial Expression Recognition" challenge to develop a model for predicting human emotion based on facial gestures using a convolutional neural network. For the analysis, we present two principal architectures which were improved by tuning the parameters to get the best-performing model. These include data augmentation, convolutional layers, batch normalization, max pooling, kernel regularization, and dropouts. Resulting in 6 variations that were evaluated over the validation set. The best-performing model was also evaluated over a test data set, which was enriched with self–made data. To measure the performance of the model, metrics such as accuracy, loss, and confusion matrix are presented. The results demonstrate the effectiveness of the proposed model. In the end, we included a brief discussion analysis of the data set quality, the predicted probabilities, and the miss-classified samples.

***Keywords**: emotion recognition, facial expression, artificial neural networks, convolutional neural networks.*

I. Introduction

Emotions play a very important role in humans' socialization and therefore in their daily lives as mentioned by [1]. Emotions determine how we interact with the environment, reflect our thoughts, and are closely related to how we behave. Face gestures have been stated as a window to identify human emotions given that they contain important information to determine the emotion state [1].

Emotion recognition through facial expression has been widely studied and has its fundamental motivation on creating a recognition methodology for machines to identify human behavior and be able to respond to it [2], or even create smoothed responses to foment a stronger communication bridge between machines and humans [1].

In this study, we use an available data set provided by Kaggle for a competition called "Facial Expression Recognition" [3]. The data set is composed of 48x48 pixel gray-scale images of faces. The faces have been automatically registered so that the face is centered and occupies about the same amount of space in each image. The original data set was composed of 35,887 data points classified into 7 different emotions: 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral.

After conducting an initial exploratory data analysis, it was decided to delete the data corresponding to the emotion "Disgust" given that it was imbalanced in the number of appearances compared with the other emotions considered. With this pre-condition, we will develop a model to predict the human emotion label based on the gestures shown in the facial expression making use of an artificial neural network over 35,340 samples. The output of the model is a label (one of 6 different studied emotions: 0=Angry, 1=Neutral, 2=Fear, 3=Happy, 4=Sad, 5=Surprise) for each of the faces analyzed.

II. Emotion recognition in facial expressions through neural-network-based methods

In [1] an exhaustive revision of the bibliography about the topic has been made. Starting from the analysis of facial expressions, describing some implementation examples based on movements of the mouth, eyes, and eyebrows as well as the impact of facial muscles, lips, nasolabial furrow, and wrinkles in the detection of emotions. Based on this, it was stated that expressions of the face are a clear manifestation of the intensity of an involuntary emotion was drawn and stated as mentioned by [1] “The face is the window where the emotions are shown”.

Artificial neural networks (ANN) have been widely used for image recognition given the non-linearity nature of this type of analysis as we can see in [4], and also in their mentions about related work. Given that it has been a very well-studied topic, considerable increases in the predictive power of these models have been shown based on advances in deep learning techniques and its reinforcement with the use of hybrid approaches [1], [4]. The development of the topic and the implementation of the architectures have been improved and optimized to the point where models for facial expression classification in real-time have been developed [4].

The good performance on feature extraction allows these models to be robust and scalable, properties that support the use of neural networks for this type of analysis.

III. Implementation of the model

Looking forward to starting our path in ANN development, we have selected an existing data set published by Kaggle to be used as input in a challenge called: “Facial Expression Recognition” in 2013 [3] to predict the human emotion label based on the gestures shown in the facial expression making use of an artificial neural network.

The full data set available in [3] consists of 35,887 samples classified into 7 different emotions (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral) and with 3 different usage type specified: Training, PublicTest and PrivateTest. In Figure 1 we can see the distribution of the data set within the classes.

Since the "Disgust" emotion has 1.52% of the whole samples which are 547 out of 35,887, it makes the distribution imbalance and could affect the model efficiency, therefore we will remove it from the samples and work with the data set corresponding to the 6 emotions left. The data set was re-labeled as 0=Angry, 1=Fear, 2=Happy, 3=Sad, 4=Surprise, and 5=Neutral.

i. Train data preparation:

The training data set that we found available was in the shape of a CSV file with no missing values. It consisted of objects with three components: the label or emotion, the usage, and the pixels. The label is a number between 0 and 6, representing the emotion as described before. The usage will be ignored, given that we will work with the data set as a whole, to make use of the `train_test_split` available in TensorFlow to create our own Training set, validation set, and test set. The split will be 80% training, 10% validation, and 10% test data. The pixels are stored as a string, where the pixels are space-separated values.

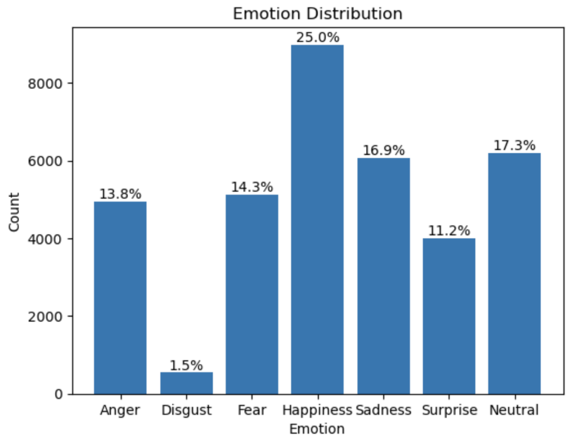

Figure 1: *Percentage of data per each emotion.*

To be able to train the model, we started by iterating through the whole data set to obtain the 48x48 arrays of pixels, as well as the label for each array. We checked that the values of the data set were varying between 0 and 255. The image pixel information was reshaped to match the input shape of a convolutional neural network (CNN) model.

The labels were one-hot encoded to create an array of six columns, with each column representing one target emotion. In this encoding, only the column representing the label for that row has a value of 1, while the rest of the values are zero. "This type of encoding creates a new binary feature for each possible category and assigns a value of 1 to the feature of each sample that corresponds to its original category" [5].

In Figure 2, we will show five examples of each emotion to exemplify the images that compose the data set we are using.

ii. Building the model:

In the following link, you will find the notebook developed for this project: www.colab.com/ANN-Project

ii.1 Architecture 1:

For the first model, we proposed a CNN, which is widely used for image-based tasks, including emotion recognition [4]. CNNs are effective in capturing spatial patterns and hierarchical representations from images. They typically consist of convolutional layers, pooling layers, and fully connected layers. For the first model architecture, we implemented the following:

1. We initialized a sequential model, which allowed us to build the model layer by layer in a sequential manner.

2. We added a convolutional layer with 32 filters, each of size 3x3 as the input layer. For selecting the sizes of the kernels, we took [6] into consideration and concluded that smaller odd-sized kernel filters should be a popular choice over larger and even sizes, for this reason, 3x3 was selected.

   The activation function used is "ReLU". The `input_shape` parameter specifies the shape of the input images, which is (48, 48, 1), indicating a grayscale image of size 48x48. The "ReLU"

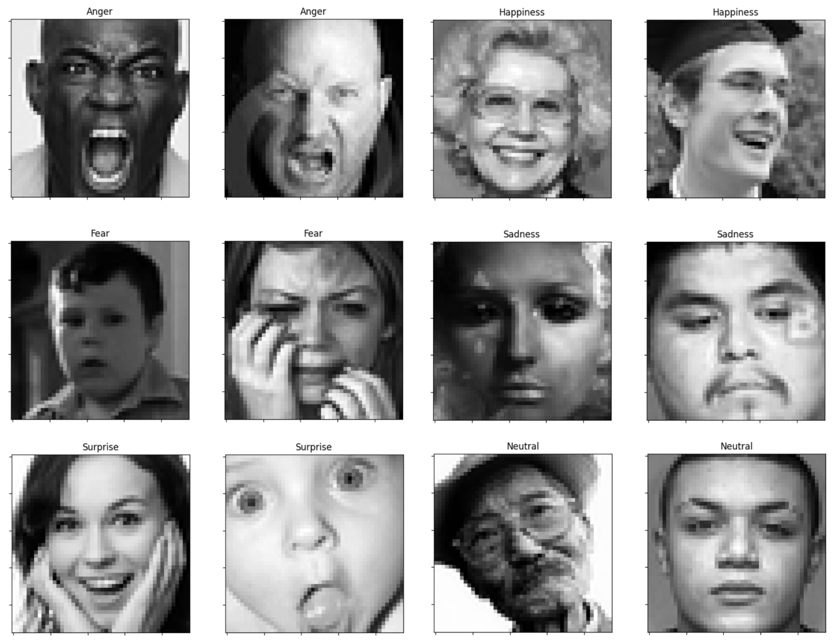

Figure 2: *Examples of each emotion.*

activation function is commonly used in CNNs since it introduces non-linearity, allowing the network to learn complex representations [7].

3. We added a batch normalization layer. This normalizes the input of each mini-batch, which helps in stabilizing and accelerating the training process. Normalizing the inputs of each mini-batch helps in stabilizing the learning process, improving the speed of convergence and generalization performance [8].
4. We added a max pooling layer with a pool size of 2x2. Max pooling is an operation that reduces the spatial dimensions of the input, helping to extract important features while reducing computational complexity. Reducing the spatial dimensions through max-pooling helps in downsampling the input, extracting important features while reducing the computational complexity [9].
5. Then we added another convolutional layer with 64 filters of size 3x3 and "ReLU" activation. We also added its corresponding batch normalization layer and another max pooling layer.
6. After that, we added another convolutional layer with 128 filters of size 3x3 and "ReLU" activation, the batch normalization layer, and the max pooling layer.
7. Next, the output was flattened from the previous layer into a 1D array, preparing it for the fully connected layers.
8. We added a fully connected layer with 128 units and "ReLU" activation.
9. Dropout was applied to the previous layer with a rate of 0.5. Dropout is a regularization method that randomly sets a fraction of input units to 0 during training, which helps prevent over-fitting [10].

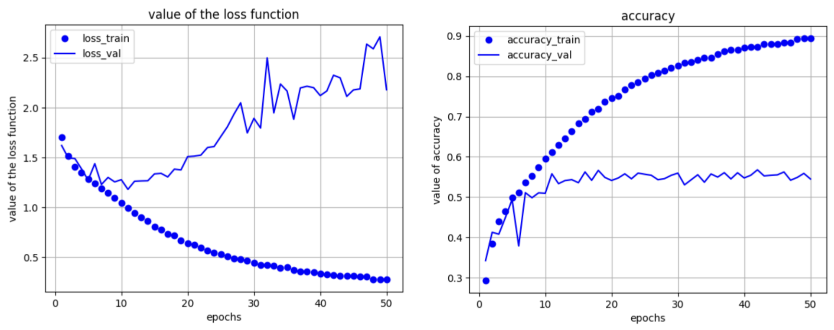

Figure 3: *Loss function and accuracy architecture 1.*

10. The final fully connected layer is added with 6 units, representing the 6 evaluated emotions: Anger, Fear, Happiness, Sadness, Surprise, and Neutralness, with "softmax" activation. "Softmax" ensures that the output probabilities sum up to 1, allowing us to interpret the outputs as class probabilities.

After the architecture building process, the compilation of the model was made using the optimizer "Adam", the loss function "categorical cross-entropy", and the metrics set to "accuracy".
for this approach, the architecture was selected by many trials of parameter variation, such as layers, number of units, activation functions, and the optimizer to come out with the best-performing solution. The loss function and accuracy are presented in Figure 3.

ii.2 Architecture 2:

For the second architecture, we varied some parameters in the formulation of the model as shown below.

1. We initialized a sequential model.
2. We added a convolutional layer with 32 filters, each of size 3x3 as the input layer. The activation function used is "ReLU", the input images stay the same (48, 48, 1) grayscale images of size 48x48.
3. We added another convolutional layer with 32 filters, each of size 3x3, with its corresponding batch normalization layer.
4. Then we added two convolutional layers with 64 filters of size 3x3 and "ReLU" activation. We also added its max pooling layer. After this, we included a max pooling of size 2x2.
5. Next, the output was flattened from the previous layer into a 1D array.
6. We added a fully connected layer with 256 units and "ReLU" activation.
7. Dropout was applied with a rate of 0.5.
8. At the end, the output layer is a fully connected one, with 6 units, representing the 6 evaluated emotions, with "softmax" activation. Here we evaluated the model with a batch size of 64 in the compilation.

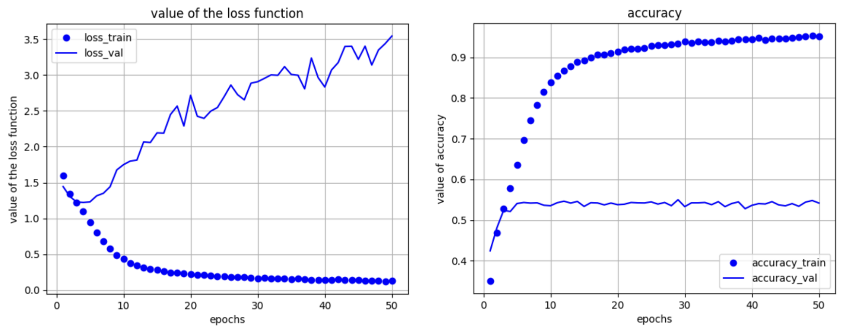

Figure 4: *Loss function and accuracy architecture 2.*

ii.3 Architecture 3:

Given that we wanted to evaluate the performance of the models under many different scenarios, we included some variations of the first 2 architectures. In the third model, we kept the same architecture as model 2, but the following changes were considered:

1. The fully connected layer was changed from 256 units to 512, with the activation function "ReLU".
2. Dropout was applied with a rate of 0.25.
3. We evaluated the model with a batch size of 128 in the compilation.

ii.4 Architecture 4:

For this approach, we took the base of Architecture 3 but we decided to include data augmentation in the process. This technique is used to increase the diversity and size of the training data set, by applying various random transformations to the existing images. We used the TensorFlow function `ImageDataGenerator`, which specifies different types of transformations that can be applied to the images. Below we present a brief description of each considered parameter, available in the package documentation [11]:

rotation_range: Specifies the range of random rotation angles (in degrees) that can be applied to the images. By setting it to 10, the images can be randomly rotated up to 10 degrees clockwise or counterclockwise.

width_shift_range and height_shift_range: Specify the range of horizontal and vertical shifts that can be applied to the images. These parameters define the maximum proportion of the total width or height by which the image can be shifted horizontally or vertically. In this case, the images can be randomly shifted up to 10% of their width or height.

shear_range: Specifies the range of shearing transformations that can be applied to the images. Shearing involves shifting one part of the image along a direction while keeping the other part fixed. This parameter defines the maximum shear angle in degrees.

zoom_range: Specifies the range of random zooming that can be applied to the images. This parameter defines the maximum zoom level, where a value of 0.1 means the images can be zoomed in or out by up to 10%.

horizontal_flip: Specifies whether horizontal flipping should be applied to the images. If set to True, the images can be randomly flipped horizontally.

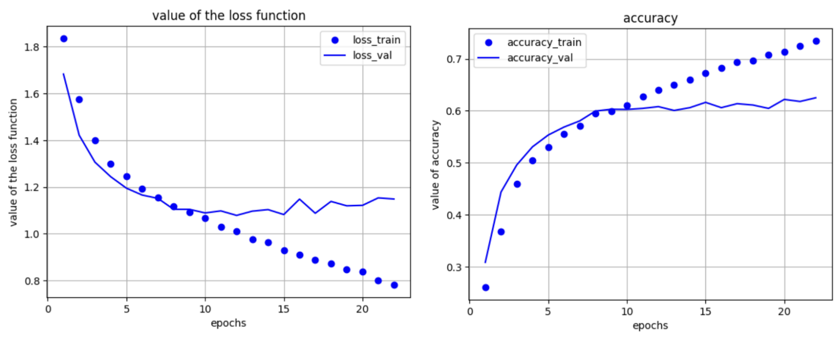

Figure 5: *Loss function and accuracy architecture 5.*

By using data augmentation, the model is exposed to a greater variety of training examples, which helps improve its ability to generalize and perform well on unseen data. It introduces variations in the training data, making the model more robust to different orientations, positions, and scales of the input images. This can help prevent over-fitting and improve the model's performance and ability to handle real-world scenarios with diverse image conditions [12]. Early stopping was set as 10 patience on the validation loss. Early stopping is an optimization technique used to reduce over-fitting without compromising on model accuracy. The main idea behind early stopping is to stop training before a model starts to over-fit [13].

ii.5 Architecture 5:

Approach 5 has architecture 3 as the base, but we have also added "L2" regularization to the dense layer using *kernel_regularizer* to control model complexity and prevent over-fitting. "L1" was also used but we did not get any improvement not even in training learning. "L2" was set to 0.001. For the compilation, we used "Adam" as the optimizer with a learning rate of 0.001. Early stopping was set as 10 patience on the validation loss.

The loss function and accuracy for the fifth model are presented in Figure 5. Given that this model was presenting the best permanence among the 6 evaluated.

ii.6 Architecture 6:

Approach 6 has architecture 5 as the base, but we added an RMSprop compiler optimizer and set the learning rate to 0.001.

The 6 variations were trained and tested over the validation data set. In the table below, you can visualize the total number of true positives (TP) obtained for each model as well as the history loss and accuracy.

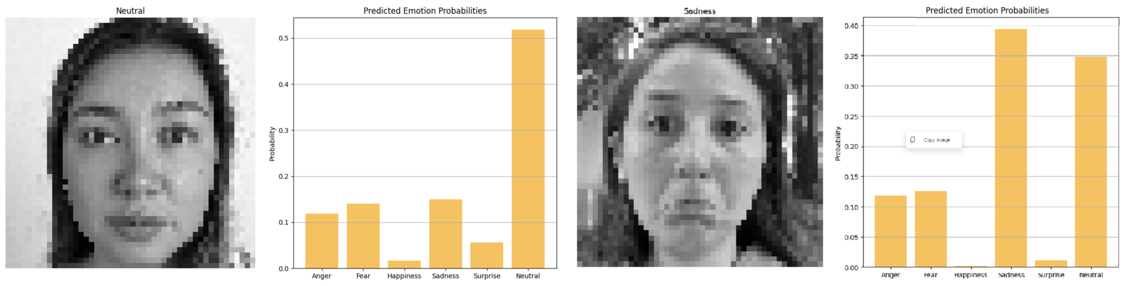

Figure 6: *2 Prediction examples.*

	M 1	**M 2**	**M 3**	**M 4**	**M 5**	**M 6**
Anger TP:	0.39	0.35	0.50	0.55	0.54	0.50
Fear TP:	0.45	0.35	0.42	0.27	0.25	0.29
Happiness TP:	0.71	0.75	0.80	0.79	0.80	0.83
Sadness TP:	0.55	0.41	0.56	0.54	0.53	0.42
Surprise TP:	0.69	0.72	0.73	0.75	0.73	0.82
Neutral TP:	0.46	0.52	0.53	0.68	0.60	0.63
Loss-Train set:	0.29	0.12	0.39	1.16	0.78	0.78
Accuracy-Train set:	0.88	0.95	0.85	0.55	0.73	0.72
Loss-Val set:	2.28	3.54	1.38	1.01	1.14	1.20
Accuracy-Val set:	0.55	0.54	0.62	0.61	0.62	0.59
Accuracy-Test set:	0.55	0.53	0.56	0.61	0.59	0.59

iii. Test data:

looking forward to testing our model, and with the idea of personalizing the test data set we added 60 labeled face images to be part of the test data. The images were recorded from family and friends. Here are a couple of the results obtained with this data:

IV. Test and results

In order to show the results for the test set of the best-performing model, we present the confusion matrix as you can see in Figure 7.

V. Discussion

To conclude our project there are some points that we would like to highlight from the implementation journey.

As we can see in Figure 6, there are some emotions that show a straightforward result in the probabilities (a predominant behavior of only one emotion), but there are some other cases where the prediction shows a split probability behavior. This can also be compared with the difficulty that some of the emotions represent for us when trying to mimic them.

The more accurate results are obtained for happiness and surprise and the least accurate for fear and sadness. Fear is the one with the higher number of wrongly classified cases, which were confused with anger and surprise.

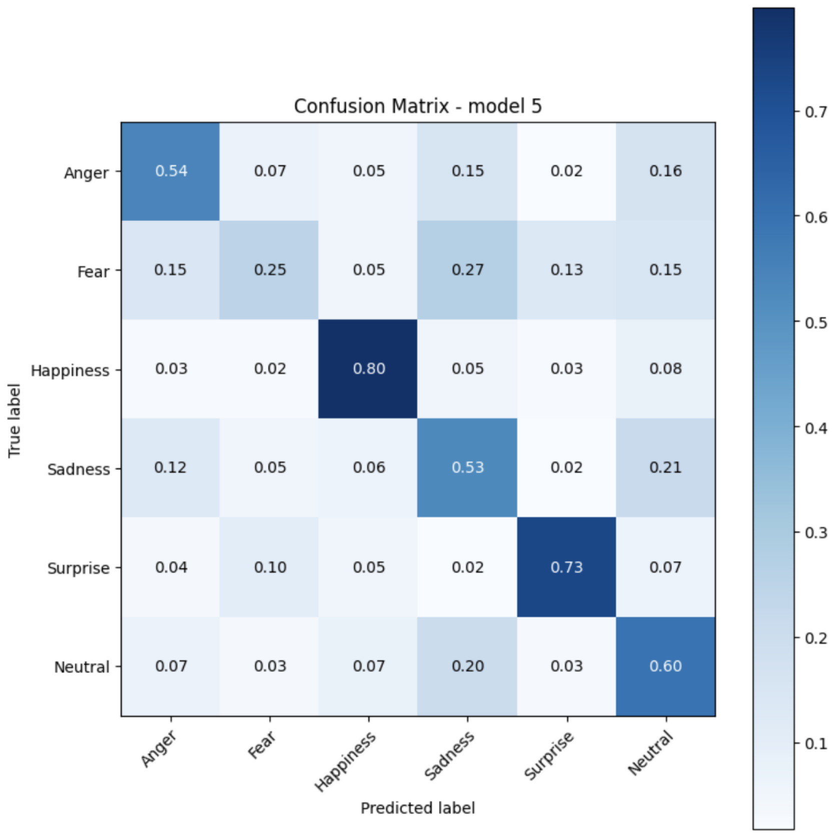

Figure 7: *Confusion matrix best-performing model.*

During the implementation of the model, we saw that some of the images that are part of the training set were wrongly classified (from our point of view), this is adding uncertainty to the model and is directly affecting the obtained accuracy.

The resolution of the images included in the training set is poor, but according to [4], facial expression classification "can be performed even at low image resolutions of 14 x 19 pixels, which can further reduce the number of operations required for inference by a large margin" as a strategy for designing highly efficient architectures.

References

[1] J. G. Rázuri, D. Sundgren, R. Rahmani, and A. M. Cardenas, "Automatic emotion recognition through facial expression analysis in merged images based on an artificial neural network," in *2013 12th Mexican International Conference on Artificial Intelligence*. IEEE, 2013, pp. 85–96.

[2] J. G. Rázuri, P. G. Esteban, and D. R. Insua, "An adversarial risk analysis model for an autonomous imperfect decision agent," *Decision Making and Imperfection*, pp. 163–187, 2013.

[3] Dumitru, I. Goodfellow, W. Cukierski, and Y. Bengio, "Challenges in representation learning: Facial expression recognition challenge," 2013, published by Kaggle. [Online]. Available: https://www.kaggle.com/competitions/challenges-in-representation-learning-facial-expression-recognition-challenge/data

[4] J. R. Lee, L. Wang, and A. Wong, "Emotionnet nano: An efficient deep convolutional neural network design for real-time facial expression recognition," *Frontiers in Artificial Intelligence*, vol. 3, p. 609673, 2021.

[5] V. Dey, "When to use one-hot encoding in deep learning?" 2021, last accessed 18 June 2023. [Online]. Available: https://analyticsindiamag.com/when-to-use-one-hot-encoding-in-deep-learning/

[6] S. Sahoo, "Deciding optimal kernel size for cnn," 2018, last accessed 24 June 2023. [Online]. Available: https://towardsdatascience.com/deciding-optimal-filter-size-for-cnns-d6f7b56f9363

[7] C. B, "Why rectified linear unit (relu) in deep learning and the best practice to use it with tensorflow," 2021, last accessed 24 June 2023. [Online]. Available: https://towardsdatascience.com/why-rectified-linear-unit-relu-in-deep-learning-and-the-best-practice-to-use-it-with-tensorflow

[8] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. pmlr, 2015, pp. 448–456.

[9] P. Kumar, "Max pooling, why use it and its advantages." 2021, last accessed 18 June 2023. [Online]. Available: https://medium.com/geekculture/max-pooling-why-use-it-and-its-advantages-5807a0190459

[10] N. McNealis, "A simple introduction to dropout regularization (with code!)," 2020, last accessed 18 June 2023. [Online]. Available: https://medium.com/analytics-vidhya/a-simple-introduction-to-dropout-regularization-with-code-5279489dda1e

[11] TensorFlow, “Tensorflow v2.12.0 documentation for python,” 2023, last accessed 24 June 2023. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

[12] K. Maharana, S. Mondal, and B. Nemade, “A review: Data pre-processing and data augmentation techniques,” *Global Transitions Proceedings*, 2022.

[13] A. Z. Mustafeez, “What is early stopping?” 2023, last accessed 25 June 2023. [Online]. Available: https://www.educative.io/answers/what-is-early-stopping#