Abstract

This project proposes a, built from scratch, Convolutional Neural Network (CNN) model for the prediction of facial expressions from facial images. The proposed model gives a generalized performance and achieves a good accuracy and low loss on observed data. However, it comes with a number of limitations, including an imbalance in the distribution of the original data, lack of ethical consideration with respect to sourcing image data, and other factors such as the computational cost of hosting and a running a deep learning model. It is suggested that stakeholders take these variables into consideration when developing automatic facial expression recognition systems.

Problem Summary

It has been found that as much as 55% of communication takes place through facial expressions or other visual cues. Therefore, we want to leverage ML and AI techniques to accurately detect facial emotions from images and take steps towards the development of emotionally intelligent behavior in machines with AI capabilities. The key objective in this project is to build a **neural network model to perform multi-class classification and classify the expressions of facial images according to the associated emotion.** There are many use cases for this solution, such as, but not limited to: understanding human behavior, detecting mental disorders, and creating higher quality virtual assistants for customer-facing businesses.

Solution Summary

A number of CNN models were explored as part of the solution design, including basic architectural models, transfer learning models, and more complex models. CNN models were chosen over its counterpart Artifical Neural Network (ANN) because of its strengths when dealing with image data. The two main benefits of implementing a CNN over an ANN are for its local spatiality (the ability to identify similar clusters of pixels), and weight sharing (the ability to extract information from an entire image instead of just a particular region). Other parameters were also tried and tested when building the models such as: different color mode images, hyperparameter tuning, as well as different CNN architectures.

Figure 1 displays the best CNN model, which has a generalized performance (72% accuracy and 0.68 loss) and yields the highest accuracy on unseen data (83% on the test set). The model takes grayscale images as input as it performs better and at a lower computational cost when compared to RGB accepting models. It uses a Stochastic Gradient Descent (SGD) optimizer (learning rate of 0.05) and different iterations of the model with various architectures were tested (see Appendix 1 for final architecture).

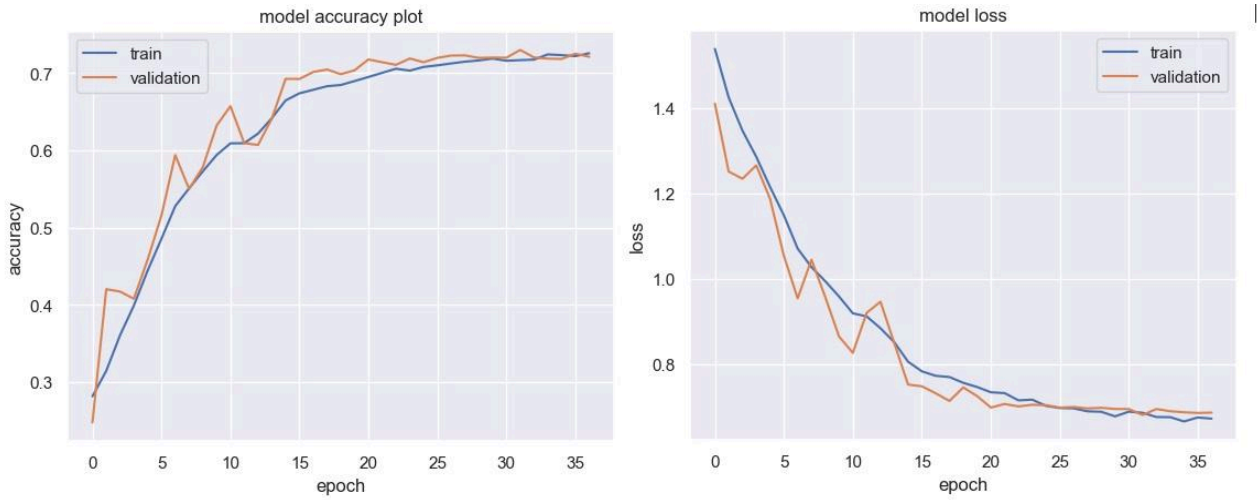


Figure 1: Chosen CNN model accuracy and loss plots

Furthermore, transfer learning architectures did not yield comparable results. Of the three tried, the VGG16 architecture performed the strongest (see Appendix 2). A possible explanation for their poor performance could be the differences in the data these models have been trained on and the data used for this project. However, a simpler CNN architecture outclassed all models in the accuracy on the training data but fell short on the validation data (see Appendix 3 and 4). Therefore, I would recommend the CNN with the specified final architecture given its performance respective to other models and structures.

Limitations and Recommendations for Further Analysis

There are data and model related recommendations for further analysis as well as ethical and privacy concerns related to the collection of images. With respect to the training data, **Figure 2** displays a clear imbalance in the validation set and a small sample size.

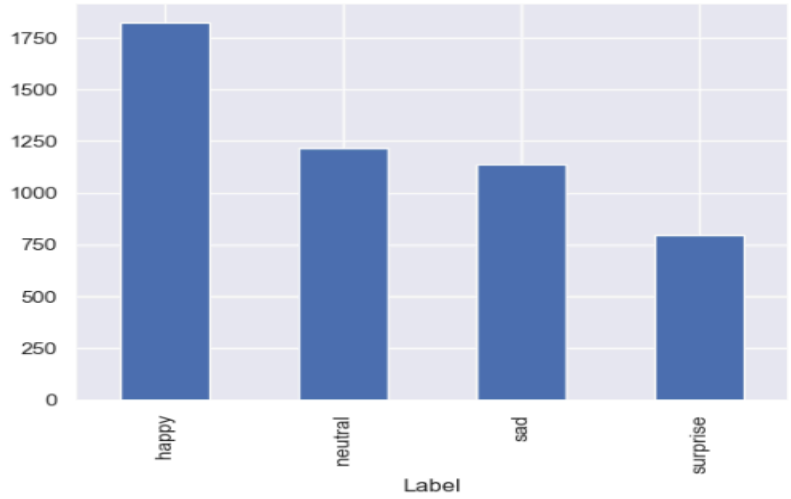


Figure 2: Class distribution of validation set

It is recommended that the chosen CNN model be trained on a larger and more evenly distributed dataset before being deployed. There are two reasons behind this suggestion: (1) our training and validation accuracies did not break an 80% accuracy mark and therefore the model still has a lot of room to improve, (2) the model struggled the most in distinguishing the Neutral and Sad emotion classes from one another. The bigger sample size would allow the model a better opportunity to identify the unique features of each class.

Earlier, the parameters of the model were briefly mentioned, but it is also recommended that further hyperparameter tuning be evaluated. For example, a few different SGD learning rates were tested, but different types of optimizers were not evaluated against SGD. There is nearly an infinite amount of hyperparameter tuning that can be done, but using the proposed model as a baseline, I strongly believe improvement can be found with minimal time commitment and changes.

Regarding the process of collecting and using image data, stakeholders should be aware of the ethical and privacy concerns that arise. The model requires facial images to be collected at all stages from the training data all the way to the new images it is classifying. There is a wide range of ethical concerns with facial recognition systems which include, but not limited to: user privacy and consent, security behind the collected biometric data, and misuse of collected data, among many other factors. It is strongly encouraged that stakeholders take the time to identify solutions to the inherent risks and ethical concerns behind the development of an automatic facial expression recognition system.

Therefore, for more accurate predictions and to stay above privacy issues, it is necessary to include **more data into training the model, further tuning of hyperparameters, and setting up a plan to avoid ethical breaches**.

Recommendations for Implementation

The above analysis portrays an effective CNN model at predicting facial emotions, the relative improvements that can be made, and makes note of the privacy concerns that may arise from deploying such a model, but it does not dive into the associated costs of training and deploying the model. Model training was initialy done on Google Colab, but it quickly ran out of RAM and was moved onto my local machine (64GB RAM versus Google Colab's 12GB). Even then, runtimes for certain models still took an erroneous amount of time. When training and deploying the model, stakeholders will need to allocate a budget to cloud computing software, data collection, and model monitoring tools. Cloud computing can be broken down into three main categories: compute, storage, and data transfer with the associated cost equal to usage multiplied by unit price. For example, the cost of compute is charged by time (seconds or hours), whereas storage and data transfer is usually charged per GB. There are essentially two trade-offs to take into account when choosing a cloud instance: high cost with high performance, or lower cost with slower performance. In this specific project and at its current state a low cost option would be suitable. If an estimated cost had to be given, for a project of this scope, I'd expect an

hourly cost of 80$ per hour and approximately 300 hours of work from start to finish, which comes out to 24000$ (excluding post deployment model monitoring). Note that the estimated cost does not include the involvement of other teams as machine learning operates in conjunction of various elements within a bigger umbrella. However, if the time comes to scale or deploy more models, I would recommend stakeholders to come back to this and reevaluate.

Bibliography

- "Machine Learning (ML) Costs: Price Factors and Real-World Estimates." ITRex, 20 June 2024, itrexgroup.com/blog/machine-learning-costs-price-factors-and-estimates/?ref=hackernoon.com
- Li, Yuqi. "How to Estimate the Time and Cost to Train a Machine Learning Model." Medium, 9 June 2022, towardsdatascience.com/how-to-estimate-the-time-and-cost-to-train-a-machine-learning-model-eb6c8d433ff7
- Bogusch, Kevin. "Cloud Computing Costs in 2024." Oracle.com, Oracle, 24 Jan. 2024, www.oracle.com/cloud/cloud-computing-cost/
- Almeida, Denise et al. "The ethics of facial recognition technologies, surveillance, and accountability in an age of artificial intelligence: a comparative analysis of US, EU, and UK regulatory frameworks." AI and ethics vol. 2,3 (2022): 377-387. doi:10.1007/s43681-021-00077-w
- "CS231n Convolutional Neural Networks for Visual Recognition." Cs231n.github.io, cs231n.github.io/understanding-cnn/

Appendix

Appendix 1: Chosen CNN Architecture

Model: "sequential"

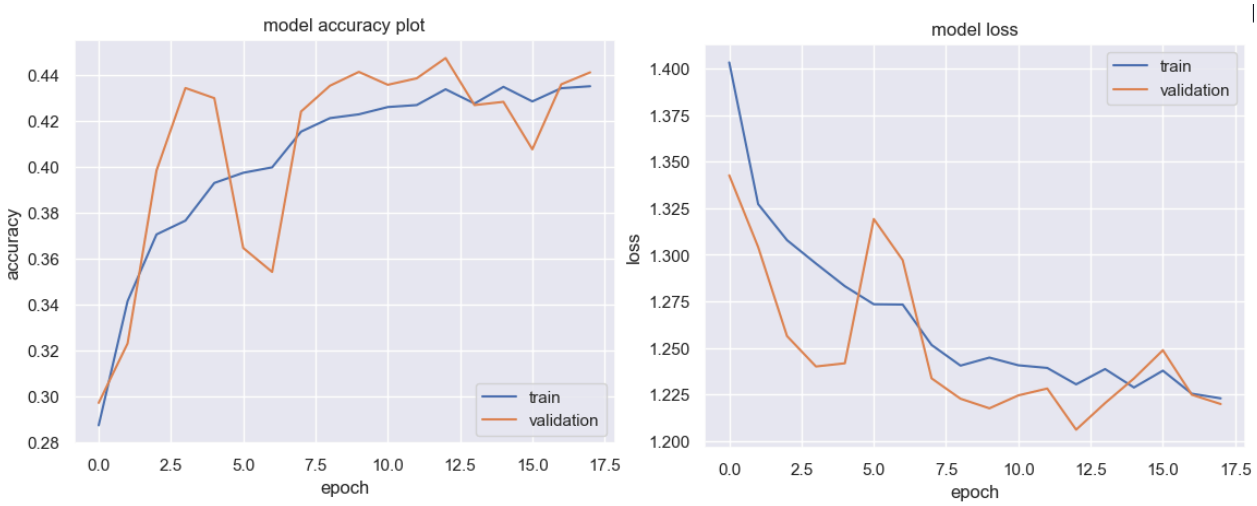
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 32)	320
batch_normalization (BatchNormalization)	(None, 48, 48, 32)	128
activation (Activation)	(None, 48, 48, 32)	0
max_pooling2d (MaxPooling2D)	(None, 24, 24, 32)	0
conv2d_1 (Conv2D)	(None, 24, 24, 128)	36,992
batch_normalization_1 (BatchNormalization)	(None, 24, 24, 128)	512
activation_1 (Activation)	(None, 24, 24, 128)	0
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 128)	0
dropout (Dropout)	(None, 12, 12, 128)	0
conv2d_2 (Conv2D)	(None, 12, 12, 256)	295,168
batch_normalization_2 (BatchNormalization)	(None, 12, 12, 256)	1,024
activation_2 (Activation)	(None, 12, 12, 256)	0
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 256)	0
conv2d_3 (Conv2D)	(None, 6, 6, 512)	1,180,160
batch_normalization_3 (BatchNormalization)	(None, 6, 6, 512)	2,048
activation_3 (Activation)	(None, 6, 6, 512)	0
max_pooling2d_3 (MaxPooling2D)	(None, 3, 3, 512)	0
dropout_1 (Dropout)	(None, 3, 3, 512)	0
conv2d_4 (Conv2D)	(None, 3, 3, 512)	2,359,808
batch_normalization_4 (BatchNormalization)	(None, 3, 3, 512)	2,048
activation_4 (Activation)	(None, 3, 3, 512)	0
max_pooling2d_4 (MaxPooling2D)	(None, 1, 1, 512)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 256)	131,328
batch_normalization_5 (BatchNormalization)	(None, 256)	1,024
activation_5 (Activation)	(None, 256)	0
dropout_2 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 512)	131,584
batch_normalization_6 (BatchNormalization)	(None, 512)	2,048
activation_6 (Activation)	(None, 512)	0
dropout_3 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 4)	2,052

Total params: 4,146,244 (15.82 MB)

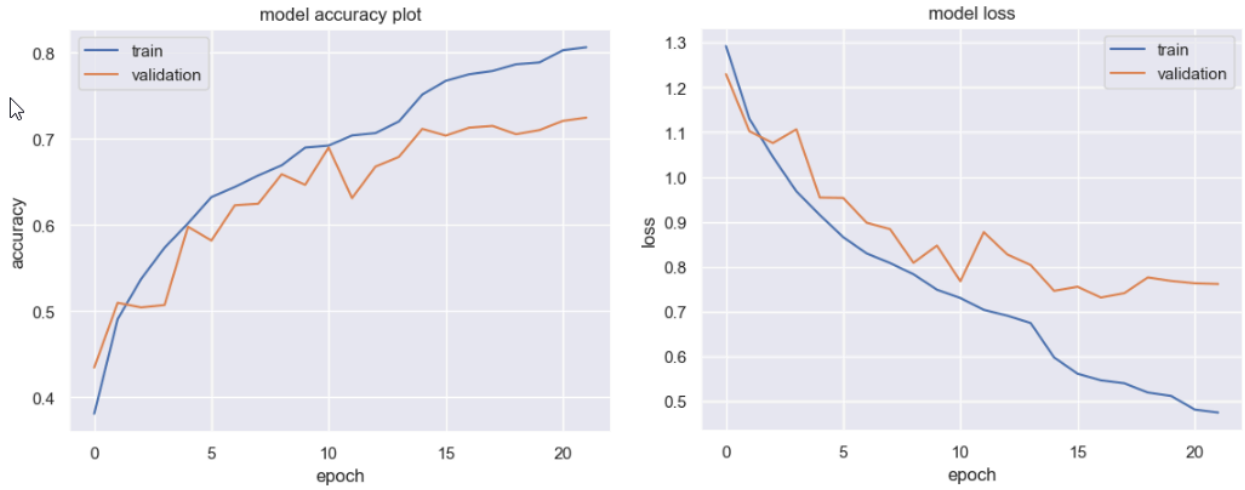
Trainable params: 4,141,828 (15.80 MB)

Non-trainable params: 4,416 (17.25 KB)

Appendix 2: Accuracy and Loss Graphs for VGG16 Transfer Learning



Appendix 3: Accuracy and Loss Graphs for CNN with Simpler Architecture



Appendix 4: Simpler CNN Architecture

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 256)	2,560
batch_normalization (BatchNormalization)	(None, 48, 48, 256)	1,024
leaky_re_lu (LeakyReLU)	(None, 48, 48, 256)	0
max_pooling2d (MaxPooling2D)	(None, 24, 24, 256)	0
conv2d_1 (Conv2D)	(None, 24, 24, 128)	295,040
batch_normalization_1 (BatchNormalization)	(None, 24, 24, 128)	512
leaky_re_lu_1 (LeakyReLU)	(None, 24, 24, 128)	0
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 128)	0
conv2d_2 (Conv2D)	(None, 12, 12, 64)	73,792
batch_normalization_2 (BatchNormalization)	(None, 12, 12, 64)	256
leaky_re_lu_2 (LeakyReLU)	(None, 12, 12, 64)	0
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_3 (Conv2D)	(None, 6, 6, 32)	18,464
batch_normalization_3 (BatchNormalization)	(None, 6, 6, 32)	128
leaky_re_lu_3 (LeakyReLU)	(None, 6, 6, 32)	0
max_pooling2d_3 (MaxPooling2D)	(None, 3, 3, 32)	0
flatten (Flatten)	(None, 288)	0
dense (Dense)	(None, 512)	147,968
dense_1 (Dense)	(None, 128)	65,664
dense_2 (Dense)	(None, 4)	516

Total params: 605,924 (2.31 MB)

Trainable params: 604,964 (2.31 MB)

Non-trainable params: 960 (3.75 KB)