Facial Emotion Detection using a GMK NucBox Mini PC

Submission by - Group 1

Nicholas Sadler, Parviz Ali & Aindrila Sil

School of Professional Studies, Northwestern University

MSDS 462: Computer Vision

Dr. Sreenidhi Bharadwaj

Introduction

Computer Vision problems prove challenging to even the most experienced practitioners and researchers, and productionalizing this work on edge devices provides an even greater challenge that we seek to implement in our work. Our work seeks to categorize facial images based on the emotion shown in a video feed from a GMK NucBox MiniPC (Win10, 8GB DDR4, 128GB SSD, Intel Celeron J4125, 2.7GHz) into seven categories: angry, disgust, fear, happy, sad, surprise and neutral.

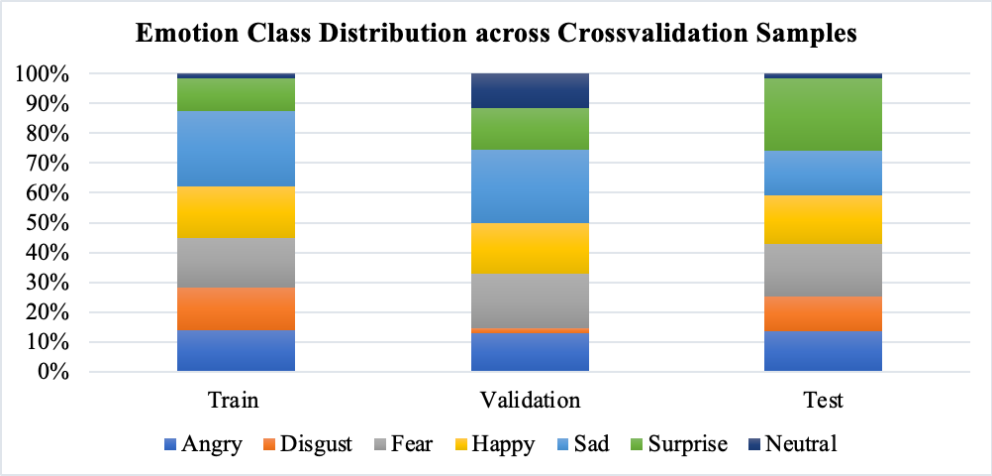
Methods

Data

Our experiments use the popular FER2013 dataset (Sambare, 2020), which consists of over 30,000 48x48 pixel images in grayscale, each belonging to one of the seven classes of emotions. We distributed these images into an 80:10:10 split for training, validation and test data, with 10% of the training data held out for validation. In order to make the images machine readable, we transformed the images in a few ways, via rotation, pixel scaling, adjusting width and height, and randomly flipping the images horizontally and vertically. The distribution of the emotion classes demonstrates that classes are not equally distributed, as the ‘Happy’ class and ‘Disgust’ class are over and under-represented, respectively.

	Train	Validation	Test
Angry	3,995	467	491
Disgust	4,097	56	416
Fear	4,830	653	626
Happy	4,965	607	594

Sad	7,215	895	528
Surprise	3,171	496	879
Neutral	436	415	55



Model Architecture

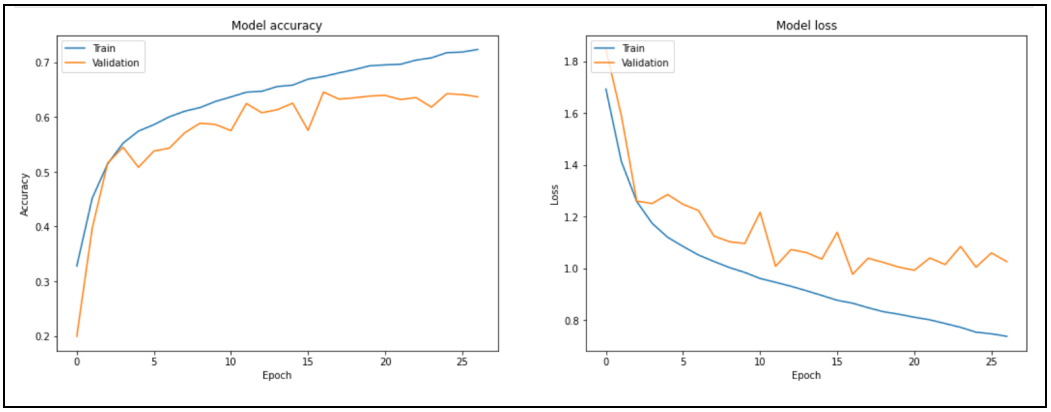
Our overall architecture involves three larger steps: model building in a Google Colab environment, importing the model onto our MiniPC edge device connected to a webcam, and using the device for inference via live feed. We chose a five layer CNN based Deep Learning architecture for our expression recognition model, using 2,137,991 total parameters trained over 27 epochs while utilizing the Adam optimizer and categorical cross-entropy for loss and accuracy metrics. The model github link and architecture details can be found in the appendix section 1.1, 1.2 and 1.3

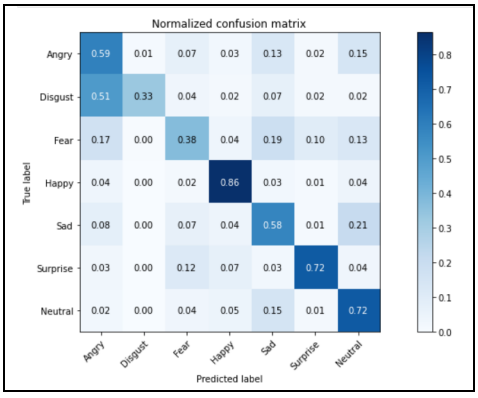
Discussion

Project Results

From a modeling perspective, our CNN model achieved 72% training accuracy, 63% validation accuracy and 65% test accuracy, where the ‘Happy’ class is correctly identified most often out of all seven classes.

Model Performance Output:





Beyond the modeling component of our work, we were able to successfully utilize our MiniPC edge device to use a USB connected webcam to perform frame by frame model

inferencing to first detect a face in the frame, then initiate a model prediction to detect the emotion present. The images from the model output are available in the appendix section 1.4.

Project Challenges and Considerations

Our original architecture used the same FER Dataset as our final architecture to classify seven classes of facial expressions, and we built an AlexNet model that achieved 63% test accuracy that we intended to deploy on AWS DeepLens. As we proceeded through development, we realized that there was a significant challenge to get a model deployed to DeepLens, and resources to help discover what we would need are sparse, DeepLens documentation references examples that are no longer compatible with the hardware altogether. We explored Sagemaker as a deployment option, but we discovered further compatibility issues when it came to integrating with DeepLens, so ultimately we decided to go in a different direction and explore using an iOS device as an edge device because of the various environmental compatibility challenges we faced. Our iOS architecture deployment was successful via XCode, but we saw high classification errors and photo size differences. Given more time, perhaps we could have a better dialogue with AWS and more thoroughly engage their support to end up using this avenue of deployment, but given our reasonably short timeline we decided to pivot away from DeepLens.

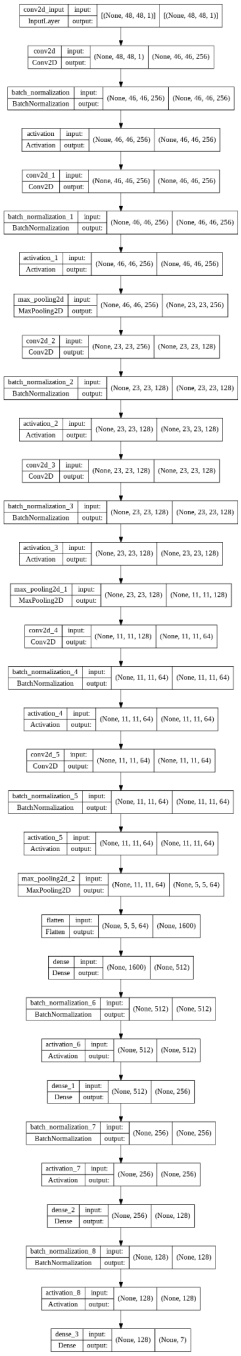
References

Sambare, M. (2020). FER-2013, Version 1. Retrieved Feb 15, 2022 from https://www.kaggle.com/msambare/fer2013.

Appendix

1.1. Model GitHub repository: https://github.com/parviza9999/MSDS462_Final_Grp1.git

1.2. Model Architecture



1.3. Model Architecture

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 46, 46, 256)	2560
batch_normalization (BatchNormalization)	(None, 46, 46, 256)	1024
activation (Activation)	(None, 46, 46, 256)	0
conv2d_1 (Conv2D)	(None, 46, 46, 256)	590080
batch_normalization_1 (BatchNormalization)	(None, 46, 46, 256)	1024
activation_1 (Activation)	(None, 46, 46, 256)	0
max_pooling2d (MaxPooling2D)	(None, 23, 23, 256)	0
conv2d_2 (Conv2D)	(None, 23, 23, 128)	295040
batch_normalization_2 (BatchNormalization)	(None, 23, 23, 128)	512
activation_2 (Activation)	(None, 23, 23, 128)	0
conv2d_3 (Conv2D)	(None, 23, 23, 128)	147584
batch_normalization_3 (BatchNormalization)	(None, 23, 23, 128)	512
activation_3 (Activation)	(None, 23, 23, 128)	0
max_pooling2d_1 (MaxPooling2D)	(None, 11, 11, 128)	0
conv2d_4 (Conv2D)	(None, 11, 11, 64)	73792
batch_normalization_4 (BatchNormalization)	(None, 11, 11, 64)	256
activation_4 (Activation)	(None, 11, 11, 64)	0

```
conv2d_5 (Conv2D)           (None, 11, 11, 64)        36928

batch_normalization_5 (Batc  (None, 11, 11, 64)       256
hNormalization)

activation_5 (Activation)   (None, 11, 11, 64)        0

max_pooling2d_2 (MaxPooling  (None, 5, 5, 64)         0
2D)

flatten (Flatten)           (None, 1600)              0

dense (Dense)               (None, 512)               819712

batch_normalization_6 (Batc  (None, 512)              2048
hNormalization)

activation_6 (Activation)   (None, 512)               0

dense_1 (Dense)             (None, 256)               131328

batch_normalization_7 (Batc  (None, 256)              1024
hNormalization)

activation_7 (Activation)   (None, 256)               0

dense_2 (Dense)             (None, 128)               32896

batch_normalization_8 (Batc  (None, 128)              512
hNormalization)

activation_8 (Activation)   (None, 128)               0

dense_3 (Dense)             (None, 7)                 903

=================================================================
Total params: 2,137,991
Trainable params: 2,134,407
Non-trainable params: 3,584
_________________________________________________________________
```

1.4. Model Output