1 TABLE OF CONTENTS

2 ASD project 2
2.1 Introduction 2
2.2 Requirements 2
2.3 Approach 2
2.3.1 Deep learning 2
2.3.2 Model training 3
2.3.3 Transfer learning 4
2.4 Dataset 4
2.5 System architecture 5
2.5.1 Architecture 1 – EfficientNet 5
2.5.2 Architecture 2 – Yolo v3 and VGG-19 6
2.6 Experiments 9
2.6.1 With EfficientNet 9
2.6.2 With Yolov3 and VGG-19 10
2.7 Optimization and Future Directions 11
2.8 References 11

2 IMAGE PROCESSING FOR ASD PROJECT

2.1 INTRODUCTION

Problem:

To Detect and/or Predict emotional meltdowns of autistic signals based on simple signals that can be recovered by portable devices (e.g. mobile phone, smartwatch).

Use case:

Children and families of children with autism and other developmental disorders can use this app to detect meltdowns and handle them accordingly.

2.2 REQUIREMENTS

1. Fast response (fast execution of the detection algorithm)
2. Good accuracy in terms of sensitivity, ideally all meltdowns should be detected. Specificity and false positives are not necessarily an issue, since an unnecessary soothing process is not detrimental to the child.
3. The app should conform with private information protection and GDPR regulations.
4. Decision making based on multiple and varied signals including image, movement, and heartrate data.
5. The overall architecture and models should be small in size and cheap in computational power to be able to run on the processor of a mobile device.

2.3 APPROACH

2.3.1 Deep learning

Modern machine learning and artificial intelligence algorithms provide various solutions for image manipulation. Deep learning is a subfield of artificial intelligence that involves the creation and use of artificial neural networks to perform complex tasks. At its core, deep learning is a set of algorithms and techniques that allow machines to learn and improve their performance on a given task by analyzing vast amounts of data. These neural networks are modeled after the structure and function of the human brain, with multiple layers of interconnected nodes that can process and transform data in a highly nonlinear manner. By feeding large amounts of data into these networks, deep learning algorithms can automatically extract features and patterns from the data, and use them to make predictions, classifications, or generate new content. Deep learning has been successfully applied to a wide range of domains, including computer vision, speech recognition, natural language processing, and autonomous driving. Its ability to learn from complex and unstructured data has made it a powerful tool for solving real-world problems and advancing the frontiers of science and technology.

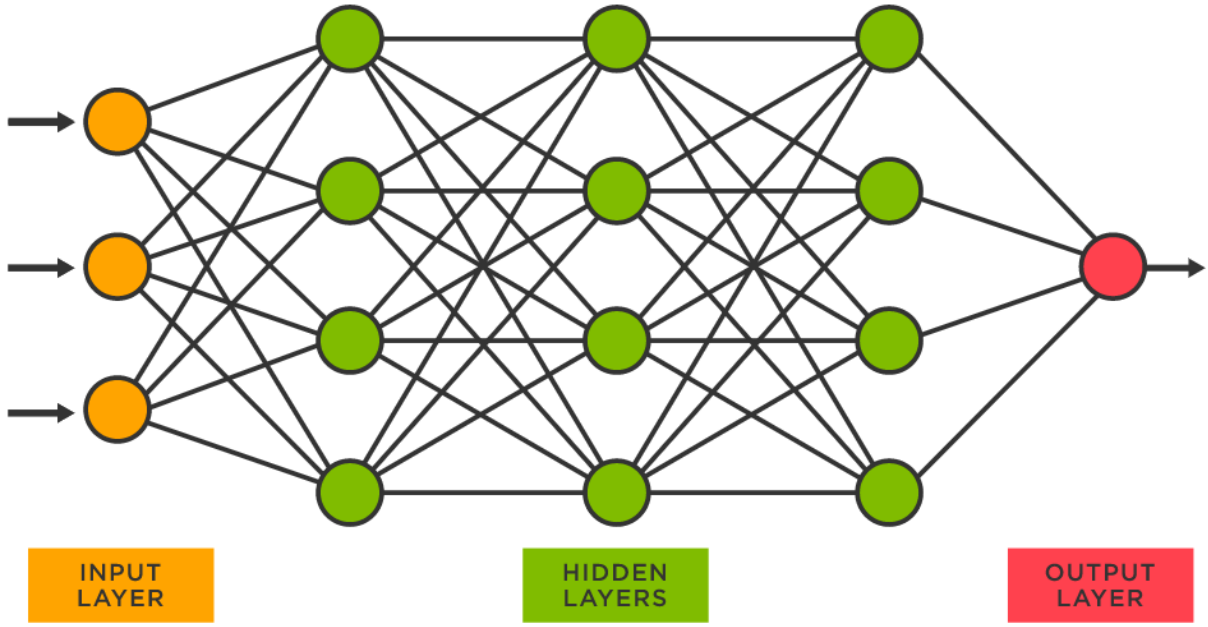


Figure 1 - Schematic of a typical neural network that contains an Input layer, three hidden layers and an output layer.

2.3.2 Model training

The process of building a neural network typically involves dividing the available data into three sets: the training set, the validation set, and the test set. Here is a description of each set and their respective roles in the training process:

1. Training set: This is the largest portion of the data and is used to train the neural network. The training process involves feeding the input data (features) into the neural network, which then computes an output (prediction) based on the current set of weights. The output is compared to the ground truth labels, and the weights are adjusted using an optimization algorithm (such as stochastic gradient descent) to minimize the difference between the predicted and actual outputs. This process is repeated for multiple epochs (passes over the entire training set) until the network converges to a set of weights that minimize the loss function.

2. Validation set: This is a smaller subset of the data that is used to evaluate the performance of the neural network during training. After each epoch, the neural network is evaluated on the validation set using the current set of weights. The validation set provides an estimate of the network's generalization performance, i.e., its ability to perform well on new, unseen data. The validation set can also be used for early stopping, a technique in which the training process is stopped when the performance on the validation set starts to degrade.

3. Test set: This is a separate subset of the data that is used to evaluate the final performance of the trained neural network. The test set should be completely independent of the training and validation sets and should not be used in any way during the training process. After the training is complete, the final set of weights is applied to the test set, and the performance metrics (such as accuracy, precision, recall, and F1 score) are computed. The test set provides an estimate of

the network's true generalization performance, i.e., its ability to perform well on completely new and unseen data.

2.3.3 Transfer learning

Modern machine learning and artificial intelligence algorithms provide various solutions for image manipulation. The current trend in image classification are deep neural networks with dozens of hidden layers and millions of trainable parameters. Most of the time, building and training a network from scratch and helping it “learn” the distribution of the problem’s target results in mediocre performance. In order to solve this problem, one can leverage on prior knowledge of solving other types of classification problems. Common technologies for the task of emotion detection include deep neural networks such as ResNet, VGG and EfficientNet. These networks can be pretrained on standard image classification tasks and then optimized to be able to perform classification with custom classes. This can be done using a technique called transfer learning, which “transfers” knowledge from one domain e.g., classification of images as [cat, dog, plane, e.t.c.] to another domain [angry, happy, sad]. A transfer learning approach reduces the time and complexity overhead that is required when building a model that targets a new problem, in this case detection of meltdowns for autistic children. However, the best performing and optimized model is not always suitable for use in production. There is a tradeoff regarding performance vs. size vs. speed of the model that needs to be carefully considered. In this regard, a good starting point is the family of EfficientNet_bX networks, which offers a good balance between size and performance for image classification tasks.

2.4 DATASET

An image database of faces of children, classified as autistic and non-autistic. The images were selected and downloaded manually from images available on the internet and cross-references with article content and descriptions. All images are center cropped around the child’s face. The images are of varied height and width but were resized to 225x225 pixels for further experimentation.

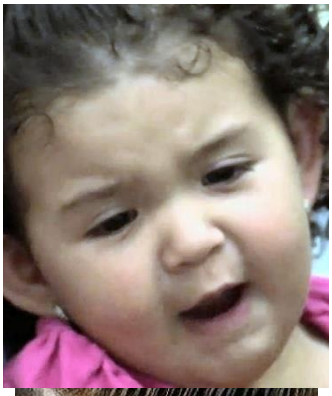
Table 1 - Breakdown of the Image Dataset

Classification	Number of images
Autistic	1463
Non-Autistic	1463
Total	2926

Note:

- There is a possibility that there are some cases of autism present in the non-autistic dataset, as it is not easy to discard the possibility as a result of the way the dataset was created, i.e., there isn’t a way to search for non-autistic children, one can only look up images of children.
- Image resizing introduces noise and may affect performance of the models.

Table 2 - Example of images included in the dataset.

Autistic			
Non-autistic			

2.5 SYSTEM ARCHITECTURE

2.5.1 Architecture 1 – EfficientNet

EfficientNet is a family of convolutional neural network (CNN) models that have been designed to achieve state-of-the-art performance on image classification tasks while being computationally efficient. The idea behind EfficientNet is to balance three main components of a CNN: depth, width, and resolution. The first component, depth, refers to the number of layers in the network. Deeper networks tend to have more capacity to capture complex features, but also require more computational resources to train and run. The second component, width, refers to the number of channels in each layer. Wider networks tend to capture more diverse features, but also require more memory to store the parameters. Finally, resolution refers to the size of the input images. Higher resolution images contain more details and can potentially improve the accuracy of the network, but also require more computational resources to process. EfficientNet models address this trade-off by using a novel compound scaling method that balances the three components. Specifically, the models are scaled up by increasing depth, width, and resolution in a systematic way. To do this, EfficientNet models use a compound coefficient that controls the scaling of each component. The optimal value of the coefficient is obtained through a grid search over a range of values. By combining compound scaling and other techniques, EfficientNet models achieve state-of-the-art performance on image classification tasks while being up to 8x smaller and more efficient than other models with similar accuracy. This makes them particularly useful in resource-constrained settings such as mobile devices and embedded systems.

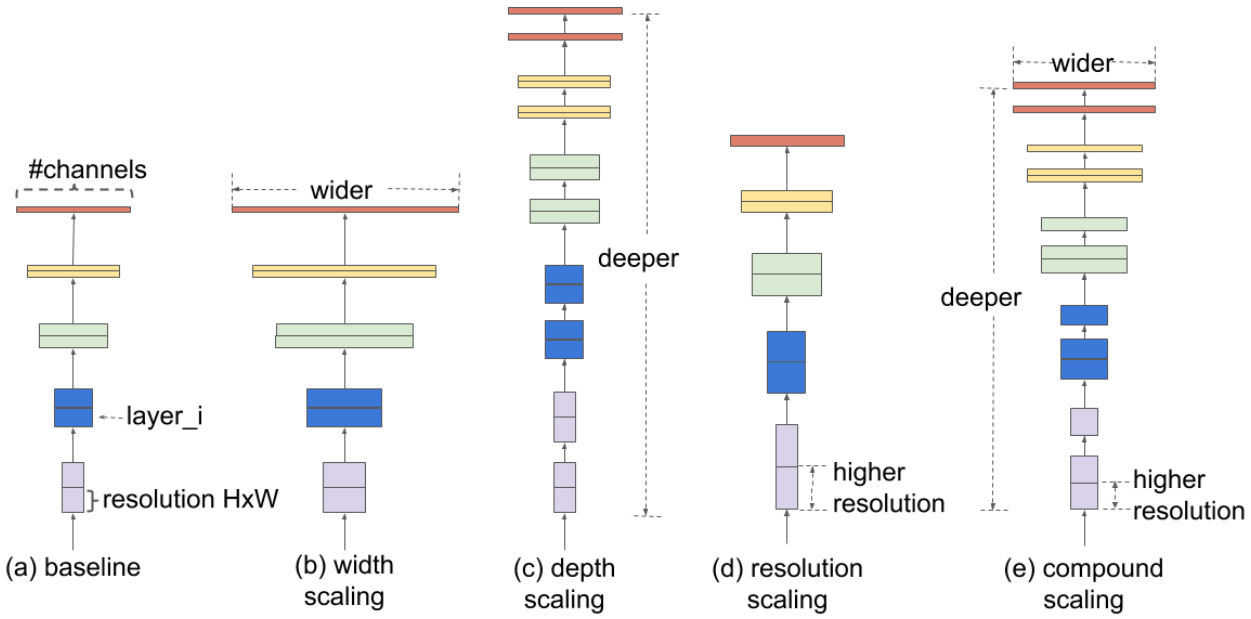


Figure 2- Comparison of different scaling methods. Unlike conventional scaling methods (b)-(d) that arbitrary scale a single dimension of the network and (e) compound scaling method that uniformly scales up all dimensions in a principled way.

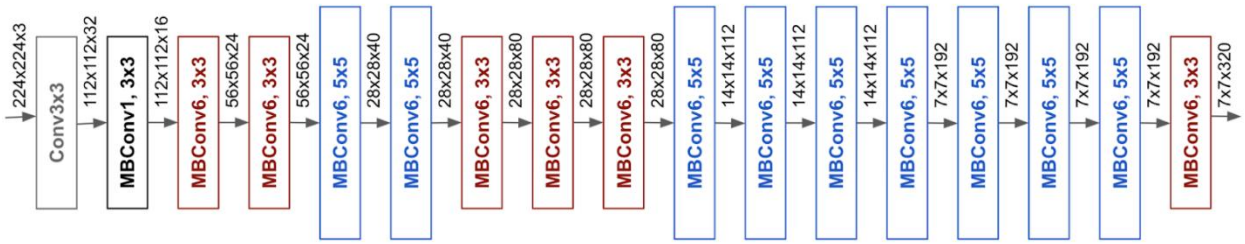


Figure 3 - The architecture for our baseline network EfficientNet-B0.

The model takes as input an image of a child's face. First we resize the image to a suitable dimension to match the model, specifically 224x224 pixels, and then feed it to the model. In our case, we used a pretrained EfficientNetB0 network from package [torchvision] with weights recovered from the best-performing scheme on the ImageNet classification dataset. Seven classes were used for prediction, namely "Happy", "Disgusted", "Suprised","Angry","Neutral","Sad","Fearful". The model takes the input image, passes it through the different layers of the network and returns a score value for each of the seven classes. The image is assigned to the emotion class with the highest score.

2.5.2 Architecture 2 – Yolo v3 and VGG-19

Ideally, we would want to use this software in a setting were multiple people are present. In this scenario, all human faces should be detected separately, the target child should be identified, and emotion should be detected. Live-time detection of objects and specifically faces can be achieved with YOLO (You Only Look Once), an object detection algorithm that uses a single convolutional neural network (CNN) to predict bounding boxes and class probabilities for objects in an image. The advantage of YOLO is that it observes the entire image instead of blocking regions of an image as done traditionally, thus it can make detections using all available information. Feature extraction and classification of the extracted objects at each predicted bounding box are recovered using Darknet.

Darknet is a similar to ResNet in terms of model architecture. It takes image data as input in the form of standard image formats such as JPEG, PNG, and BMP. Darknet is based on a deep convolutional neural network architecture that is designed for fast and efficient processing of images. The network is composed of several convolutional layers, followed by pooling layers and fully connected layers. The architecture is optimized for parallel processing using graphics processing units (GPUs) and is implemented in C and CUDA for maximum performance. During inference, Darknet takes an input image and feeds it through the trained network. The network outputs a set of bounding boxes and class probabilities for the objects detected in the image. The output is post-processed to filter out false positives and refine the bounding boxes. This way, an image depicting various objects can be processed and objects of the human class can be identified for further processing.

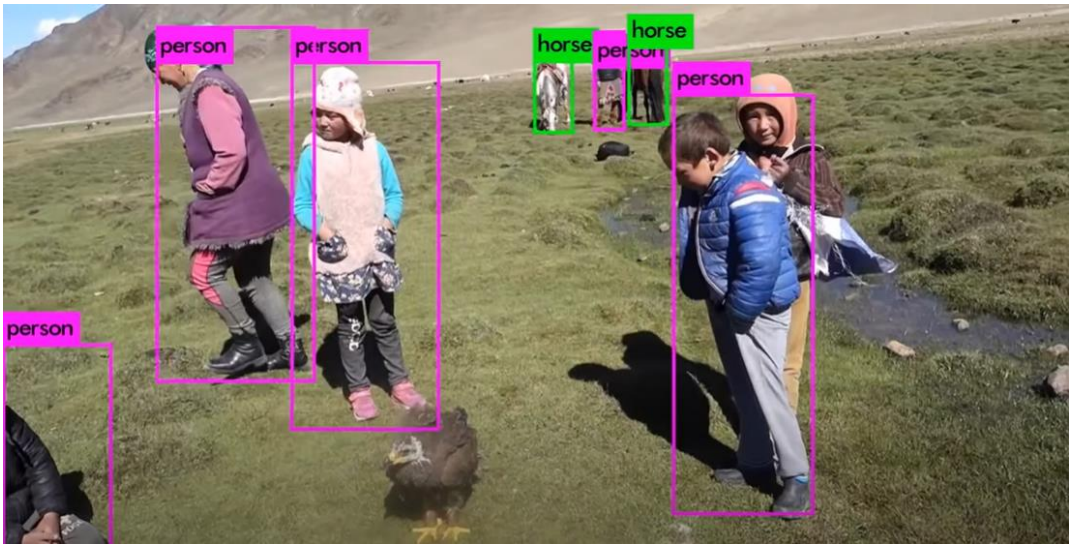
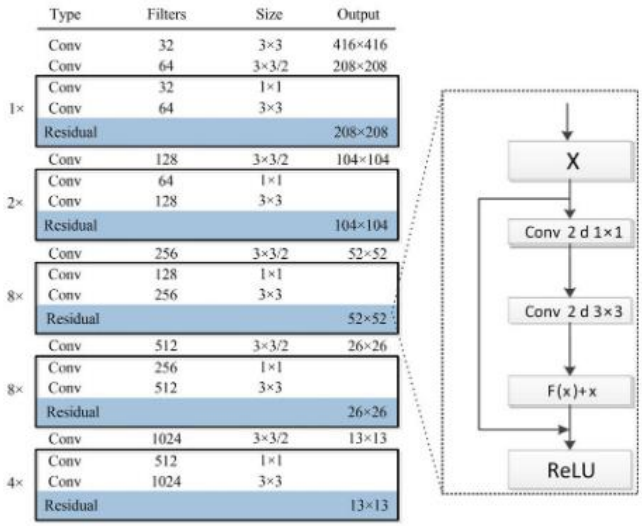


Figure 4 - Example of object detection and class prediction using YOLO



	Type	Filters	Size	Output
	Conv	32	3×3	416×416
	Conv	64	3×3/2	208×208
1×	Conv	32	1×1	
	Conv	64	3×3	
	Residual			208×208
	Conv	128	3×3/2	104×104
2×	Conv	64	1×1	
	Conv	128	3×3	
	Residual			104×104
	Conv	256	3×3/2	52×52
8×	Conv	128	1×1	
	Conv	256	3×3	
	Residual			52×52
	Conv	512	3×3/2	26×26
8×	Conv	256	1×1	
	Conv	512	3×3	
	Residual			26×26
	Conv	1024	3×3/2	13×13
4×	Conv	512	1×1	
	Conv	1024	3×3	
	Residual			13×13

Figure 5 - Darknet-53 architecture

In our implementation, we used the updated version of YOLOv3, which predicts bounding objects at three scales and uses Darknet-53 with 53 convolutional layers as a backbone classifier model. After a human or person class is detected, then another model is used for sentiment classification. The area inside the detected bounding box is used as input for the sentiment classification model. A pretrained VGG-19 model trained on the [FER-2013 dataset] is used for this task. Same as before, seven classes were used for prediction, namely "Happy", "Disgusted", "Surprised", "Angry", "Neutral", "Sad", "Fearful". The complete system architecture is shown in Figure 7.

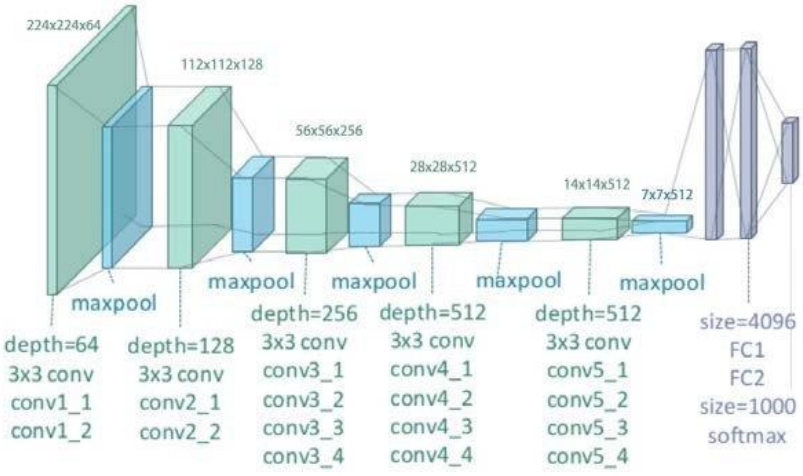


Figure 6 - Architecture of the VGG-19 model used for sentiment analysis

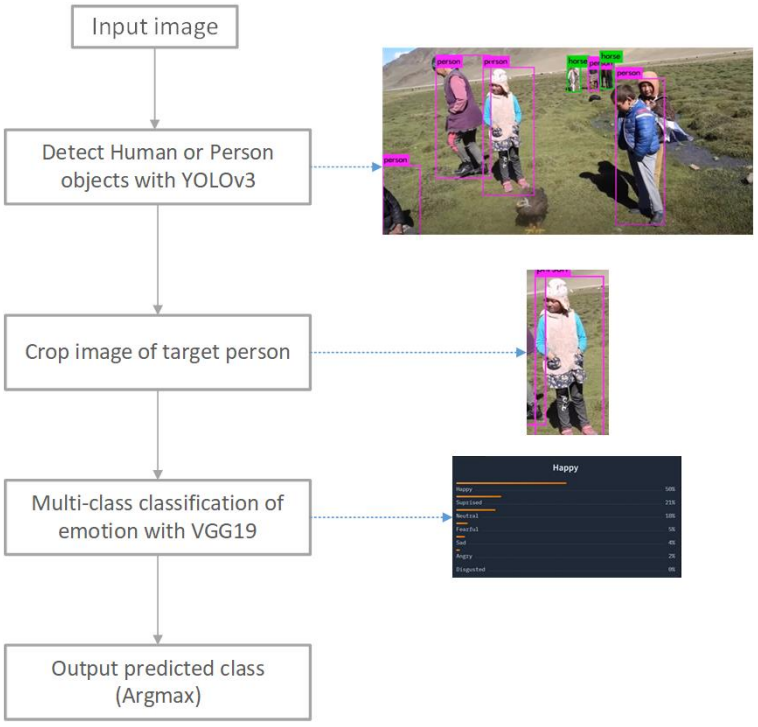
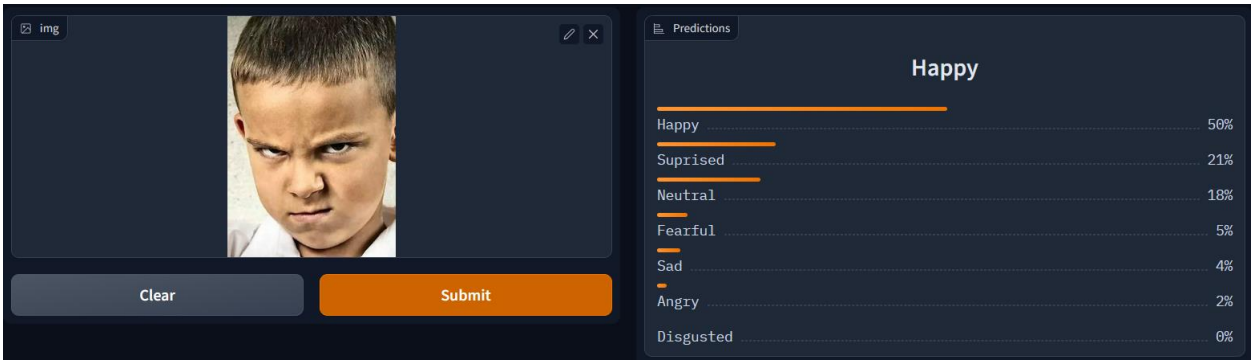
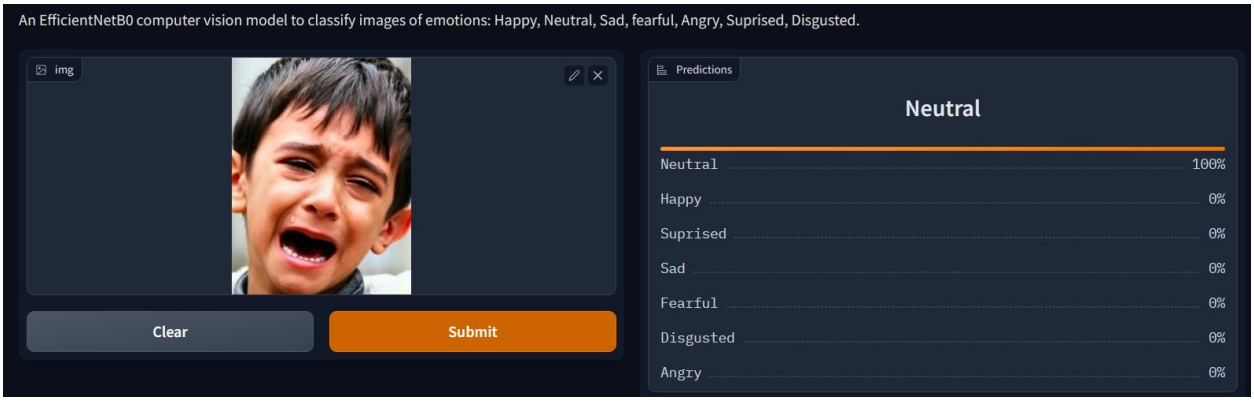


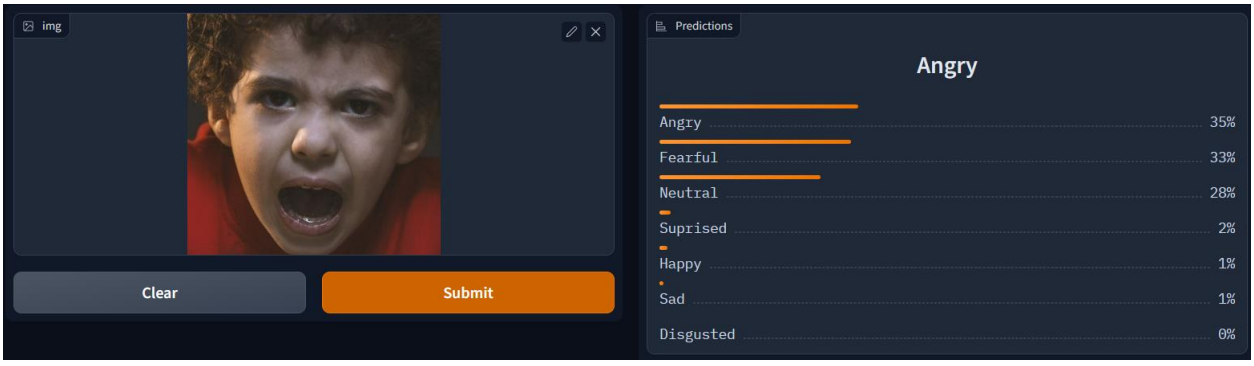
Figure 7 – The architecture of the complete sentiment detection pipeline

2.6 EXPERIMENTS

2.6.1 With EfficientNet







For the total of images of autistic children, most images were classified as ‘Happy’ or ‘Neutral’. Those two classes also had the highest average probability for the predicted emotion, at 71% and 60% respectively.

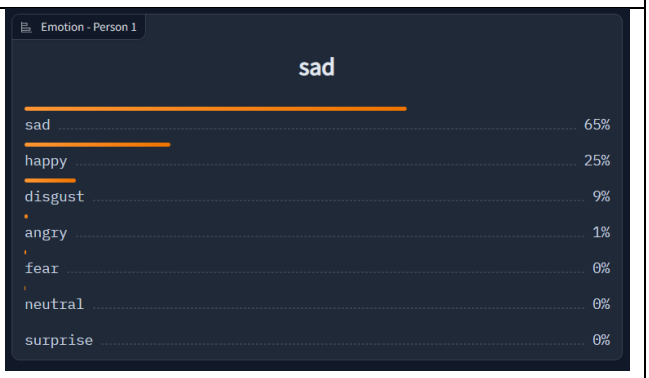
```
print(counts)
```

```
{'Happy': 827, 'Disgusted': 4, 'Suprised': 82, 'Angry': 71, 'Neutral': 434, 'Sad': 3, 'Fearful': 42}
```

```
print(prob_level)
```

```
{'Happy': 71.758455585828, 'Disgusted': 58.72436314821243, 'Suprised': 54.8237734633248, 'Angry': 52.15450789726955, 'Neutral': 60.6723445351772, 'Sad': 34.99150673548381, 'Fearful': 47.08987481537319}
```

2.6.2 With Yolov3 and VGG-19

For the total of images of autistic children, most images were classified as ‘Happy’ or ‘Neutral’, but with different distribution than before. Those two classes also had the highest average probability for the predicted emotion, at 91% and 78% respectively. This model performed much better, because it make

predictions with higher score i.e. more certain, than the previous EfficientNet model. However, this comes at a cost of speed and processing power.

```
print(f'{counts}')
```

```
{'angry': 121, 'disgust': 1, 'fear': 12, 'happy': 882, 'neutral': 304, 'sad': 131, 'surprise': 12}
```

```
print(f'{prob_level}')
```

```
{'angry': 68.13274471720389, 'disgust': 51.080673933029175, 'fear': 55.291188011566796, 'happy': 91.1547351586305, 'neutral': 7
7.7939995456683, 'sad': 62.11384459761263, 'surprise': 68.1723989546299}
```

2.7 Optimization and Future Directions

The architectures provided above can be further optimized for the task at hand. A head-on approach would be the application of transfer learning on labeled images of autistic children specifically and relevant emotions. Furthermore, model performance can be improved, in combination with other signals and additional context.

2.8 References

Joseph Redmon, A. F. (2018). *YOLOv3: An Incremental Improvement.* doi:https://doi.org/10.48550/arXiv.1804.02767

Joseph Redmon, S. D. (2016). You Only Look Once: Unified, Real-Time Object Detection. *CVPR 2016*. doi:https://doi.org/10.48550/arXiv.1506.02640

Karen Simonyan, A. Z. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *3rd International Conference on Learning Representations (ICLR 2015).* doi:https://doi.org/10.48550/arXiv.1409.1556

Mingxing Tan, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *ICML 2019*. doi:https://doi.org/10.48550/arXiv.1905.11946