EmoRec: An iOS Mobile App for Emotion Detection and Gamification to Improve Social Skills

Katherine Chen
Stanford University
kathchen@stanford.edu

Flora Huang
Stanford University
flora221@stanford.edu

Abstract

EmoRec is an mobile app that recognizes human emotions from user-provided images and helps users practice expressing specific emotions, designed for individuals with autism, social anxiety, or related disorders. The app's functionality lies in its use of a DenseNet model, which allows it to classify images into one of seven categories of emotions in real-time. We trained this model on the FER-2013 dataset [1] and employed data augmentation, regularization techniques, and hyperparameter tuning. We reached a final accuracy of 0.7281 on the test set. While this model offers significant improvement over random guessing, its accuracy indicate that there is still room for improvement. We developed an iOS mobile app and integrated the DenseNet into it using Apple's CoreML framework. From user tests, we found that EmoRec's friendly and intuitive interface made it effective in recognizing emotion and practicing expression. In the future, we plan to enhance the app's emotion detection capabilities and conduct further testing to gather feedback for a public release.

1. Introduction

We developed an iOS mobile app for interpreting human emotions from static images captured by the camera or uploaded from the phone's storage. The app has two modes: 1) a detection mode that interprets emotions from the uploaded images and 2) a practice mode that enables users to practice expressing and recognizing emotions. We intend to help people with autism, related disorders, or social anxiety to practice recognizing emotions better and expressing themselves more accurately, thereby improving social skills.

To provide context and background, we examined research papers related to emotion detection, facial expression analysis, and gamification as a means of improving social skills. For our data, we used the FER-2013 dataset [1] available on Kaggle, which contains 35,887 images with the 7 emotion labels to train and test our model. We also tested with photos of our own faces expressing different emotions.

To build the app, first, we trained and tested a CNN model using TensorFlow [2], and saved the model to a file. Next, we converted the trained model to a format that is compatible with iOS devices using Apple's Core ML framework [3]. Then, we integrated the converted Core ML model into an iOS app using Xcode and Swift, and used it to make predictions on images captured by the phone's camera or uploaded from storage. For gamification in the practice mode, our model scores users based on how well they express specific emotions. While mobile device implementations of emotion-recognizing CNNs do exist, they have not been adapted in a way that targets and benefits users with emotion-recognition difficulties.

We evaluated the accuracy of our algorithm by testing it on a separate test set and data directly collected from the app. While there was a decrease in accuracy for the app data due to factors like quality control, it did not deviate too much from the test set accuracy. We used plots and figures to qualitatively assess the app's performance for each of the 7 emotions. Quantitatively, we used metrics such as accuracy, precision, recall, and F1 score to compare our results with existing implementations. Furthermore, we gathered user feedback and engagement to assess the effectiveness of the practice mode in enhancing social skills.

2. Related Work

In this section, we examine research papers related to emotion detection, facial expression analysis, and gamification as a means of improving social skills.

In their paper, Gilligan and Akis [5] propose a CNN approach for real-time emotion detection, which aligns with our project's goal. They train their model using data from the Extended Cohn-Kanade dataset, Japanese Female Facial Expression dataset, and their own custom images, with pre-processing techniques such as re-scaling and Gaussian filters applied to enhance performance. The authors retrain LeNet and AlexNet models, achieving accuracy rates above

97%. However, they note that qualitative analysis of real-time images reveals limitations in the models' ability to accurately classify facial expressions, despite the high quantitative accuracy results.

In "Deep Facial Expression Recognition: A Survey" [12], Li et al. provide a comprehensive overview of deep learning techniques applied to facial expression analysis. The authors discuss datasets commonly used in the task, including FER-2013, which we use in our project. They present the standard pipeline of a deep facial expression system, including pre-processing, feature extraction, and classification, which we followed when creating our CNN. The survey paper also consolidates performances of several successful papers on emotion recognition, which we plan to compare our results to.

In "The challenger app for social anxiety disorder: New advances in mobile psychological treatment" [13], Miloff et al. focus on the design and features of the Challenger app, developed for the treatment of social anxiety disorder (SAD). The app utilizes cognitive-behavioral therapy principles and incorporates advanced features such as notifications, anonymous social interaction, and gamification techniques. The app's design inspires us to highlight the potential benefits of incorporating novel features in mental health applications.

3. Methods

3.1. Baseline Model

The baseline model we chose was simple, dependent on making a random choice between the 7 categories of emotions. This random-guess algorithm, which was given no prior input training, was then evaluated on our test set, returning an overall accuracy of 0.1458 across all images. This is similar to our expected accuracy, which would have been $\frac{1}{7} \approx 0.1429$.

3.2. Our Model

We explored two different model architectures for emotion recognition: AlexNet and DenseNet.

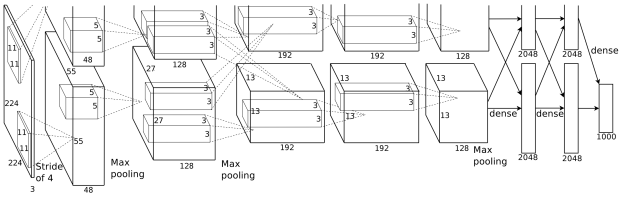


Figure 1. AlexNet model architecture.

AlexNet is a deep CNN architecture proposed by Krizhevsky et al. in 2012 [11]. It consists of a total of 8 layers: 5 convolutional layers with max pooling and batch normalization, followed by 3 fully connected layers, as shown in Figure 1. We chose AlexNet as our initial model architecture due to its effectiveness in general image classification tasks.

For pre-processing, in order to fit AlexNet's input dimensions, we resized our input images from the original 48x48x3 to 227x227x3 and down-scaled each pixel's color channel values from 0-255 to 0-1. We also used an Adam optimizer [10], an extension to stochastic gradient descent. However, ultimately, we chose not to use this model for our project. Our mobile app framework relied on our CNN to be exported in the Core ML format, and attempts to export AlexNet crashed due to its complexity and number of layers requiring far more storage than we had allocated to us.

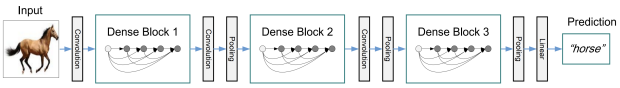


Figure 2. DenseNet model architecture.

DenseNet is a more recent architecture introduced by Huang et al. in 2017 [8]. It addresses the vanishing gradient problem and encourages feature reuse by introducing dense connections between layers. DenseNet uses densely connected blocks, which means that each layer receives feature maps from all preceding layers (Figure 2). This architecture has shown promising results in various computer vision tasks, including image classification. Specifically, our final model is a DenseNet with 3 Dense blocks, with Dropout layers in between. By exploring these different model architectures, we aimed to leverage their unique characteristics and assess their performance in emotion recognition.

3.3. Training Process

To train our models, we used the FER-2013 dataset, which contains 35,887 images with seven emotion labels: happiness, sadness, anger, fear, disgust, surprise, and neutral. The dataset was split into training and validation sets, with the latter used for evaluating the models' performance during training. We did not perform cross-validation. Additionally, we employed the following techniques to improve our model.

3.3.1 Data Augmentation

The FER-2913 dataset has limitations in terms of class imbalance, particularly with an over-representation of the happiness emotion category compared to other emotions. This bias can potentially impact the performance of our models, leading to challenges in accurately recognizing emotions that are underrepresented in the dataset [19].

As a result, we augmented our data by randomly applying transformations, namely translations and horizontal flipping, to the images belonging to the underrepresented

classes. By augmenting the training set with variations of the original images, we aimed to improve our models' ability to generalize to unseen images and enhance their performance on real-world data.

3.3.2 Regularization Techniques

We used three regularization techniques, namely early stopping [14], L2 regularization [15], and dropout [16]. Early stopping helped combat overfitting by allowing the training process to stop before our model started to overfit to our dataset. In addition, this strategy saved computational power and time. L2 regularization was added to the loss function. The regularization constant, also known as the weight decay parameter, is one of the hyperparameters that we tuned. Finally, we inserted a Dropout layer between each Dense block in the DenseNet, where we randomly set a portion of the neurons to zero during each training iteration, with a dropout rate of 0.5. This helped prevent overfitting and forced the CNN to rely on different subset of neurons for input, thereby learning more robust features.

3.3.3 Hyperparameter Tuning

To optimize the performance of our models, we conducted hyperparameter tuning experiments. We explored different learning rates, batch sizes, and L2 regularization constants. Grid search was utilized to systematically evaluate different combinations of hyperparameters and to select optimal parameters for further training.

3.4. Mobile App Integration

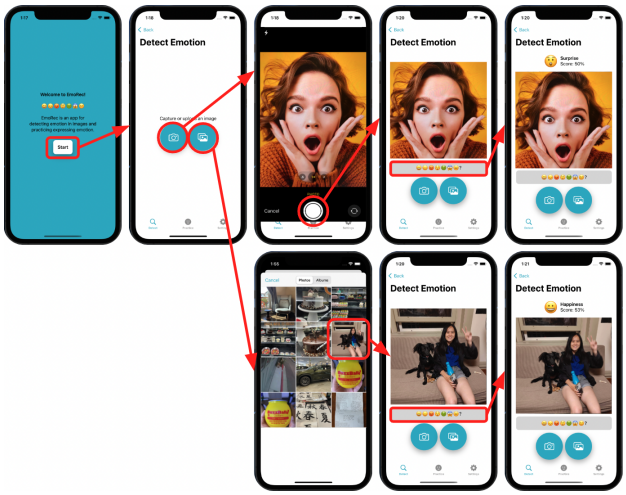


Figure 3. The EmoRec mobile app (detection mode).

After training and optimizing our model, we integrated it into an iOS mobile app using Apple's Core ML framework. Core ML allowed us to convert the TensorFlow models into a format compatible with iOS devices. We utilized Xcode and Swift to built the user interface and implement the functionality of the app.

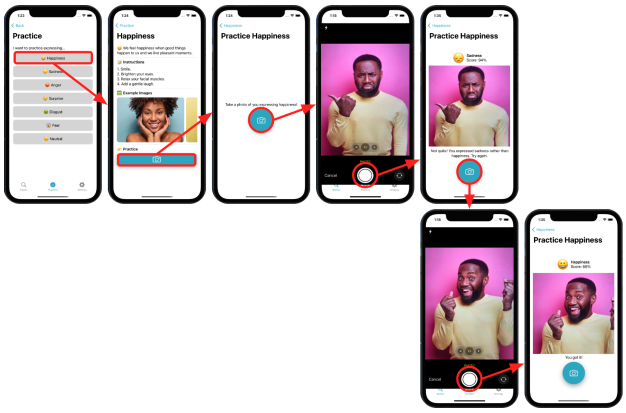


Figure 4. The EmoRec mobile app (practice mode).

The app has two modes: a detection mode and a practice mode. In the detection mode (Figure 1), users could capture images using the phone's camera or upload images from the device's storage. The app then utilizes the trained emotion detection model to analyze the captured/uploaded image and provide an interpretation of the emotions displayed in the image. The app classifies the emotions into one of the following categories: happiness, sadness, anger, fear, disgust, surprise, or neutral.

In the practice mode (Figure 2), users can engage in activities to improve their emotion recognition and expression skills. The app provides prompts for users to express specific emotions, along with step-by-step instructions and example images of the emotion, and users are able to capture images of their own facial expressions corresponding to the prompt. The app provides real-time feedback on the user's expression and compares it to the expected expression for the given emotion. Users receive scores based on how well they express the target emotion.

Throughout the development process, we focused on optimizing the app's performance and ensuring a seamless user experience. We conducted testing to ensure the accuracy and reliability of the emotion detection model within the mobile app environment, as well as evaluating the accuracy of our model on and off the mobile app platform. Additionally, we gathered user feedback to assess the effectiveness of the practice mode in enhancing social skills and make improvements based on the feedback received.

4. Dataset

The FER-2013 dataset [1] we used for training and testing our model contains 35,887 images with 7 emotion labels: anger, disgust, fear, happiness, sadness, surprise, and neutral. The dataset is pre-processed and standardized,

with each image resized to 48x48 pixels and converted to grayscale.

The dataset is divided into a training set and a test set. The training set consists of 28,709 images, while the test set contains 7,178 images. We used the training set to train our emotion detection model and the test set to evaluate its performance. We also split off a validation set from the original training set, accounting for about 20% of the original training set.

In terms of data pre-processing, the images were already pre-processed and standardized in the dataset, so we did not perform any additional pre-processing steps. Each image is already resized to a consistent resolution of 48x48 pixels and converted to grayscale to simplify the input data. Regarding normalization, we scaled the pixel values of the images from the original range of 0-255 to a normalized range of 0-1, which helped stabilize the training process and improve the convergence of the model.

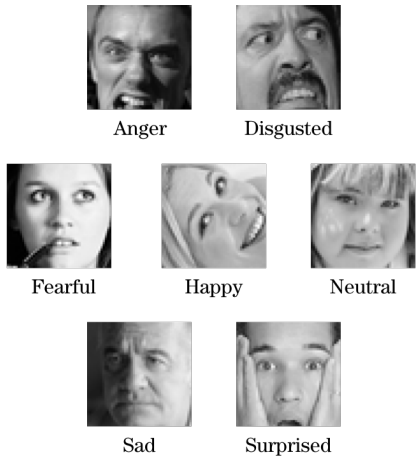


Figure 5. Examples of an image from each emotion category, randomly pulled from the FER-2013 dataset.

Table 1. Distribution of emotions in the FER-2013 dataset.

	Train Examples	Test Examples
Anger	3,995	958
Disgust	436	111
Fear	4,097	1,024
Happiness	7,215	1,774
Neutral	4,965	1,233
Sadness	4,830	1,247
Surprise	3,171	831
Total	28,709	7,178

To address the issue of limited data, we applied data augmentation techniques as described in the section above with translations and horizontal flipping. This helped increase the diversity and size of the training set, allowing the model to learn more robust features, as well as generalize better to unseen data.

For our features, we directly used the pixel intensities of the grayscale images as the input to our CNN model. This is because the CNN model is capable of extracting relevant features from the images as part of the training process, without the need for manual feature extraction techniques.

5. Experiments

5.1. Testing the Model

In this section, we present the experimental evaluation of our model. We experimented with two model architectures, AlexNet and DenseNet, for our emotion recognition task. However, in the testing phase, we focused on the DenseNet architecture due to practical reasons. AlexNet required significantly longer training time and had over 200 million learnable parameters [11], making it computationally expensive and difficult to convert to the Core ML format. Attempted conversions of AlexNet to Core ML would frequently crash due to memory limitations. In contrast, DenseNet had around 2 million learnable parameters and faster training time [8], making it a more suitable choice given our computational requirements. Thus, we chose DenseNet to overcome the challenges faced with AlexNet.

5.1.1 Quantitative Results & Discussion

We divided the FER-2013 dataset into training, validation, and testing sets, and our DenseNet model was trained using various hyperparameters without cross-validation. To enhance the performance and generalization of our model, we first attempted data augmentation as a pre-processing step. Specifically, we applied random translations and horizontal flips to underrepresented classes in our training data in order to introduce variations in image position and orientation. Table 2 shows the accuracy comparison between our DenseNet models trained with and without data augmentation. This experiment used a learning rate of 0.005 and batch size of 32, hyperparameters that were tuned later.

Table 2. Training accuracies of DenseNet with and without data augmentation (lr=0.005, bs=32).

	Without	With
Epoch 1	0.2692	0.2249
Epoch 2	0.3724	0.3025
Epoch 3	0.4326	0.3725
Epoch 4	0.4711	0.3985
Epoch 5	0.4976	0.4157
Final Training Accuracy	0.5711	0.5396

The model trained without data augmentation (early stopped at 17 epochs) consistently achieved higher accuracies compared to the model trained with data augmentation (early stopped at 18 epochs) in the first five epochs. Based on these results, contrary to our hypothesis, data augmentation did not improve emotion prediction accuracy.

Several factors could have contributed to this outcome. We speculate that the augmented images introduced too much variation. Our translation distance of 12 pixels (1/4 of the image's side length) may have been too large for the model to learn meaningful patterns, through cutting off important facial features. It is also possible that our DenseNet with 2 million parameters may be too shallow, causing it to struggle to capture the increased diversity introduced by data augmentation. If we had more time, we could experiment with deeper or more complex models to better leverage the augmented data.

We decided to not use data augmentation, and proceeded to tune 3 hyperparameters: learning rate of stochastic gradient descent (lr), batch size (bs), and L2 regularization constant (λ), with results shown in Tables 3 and 4. Prior to hyperparameter tuning, we used lr=0.005, bs=32, and λ=0.01. For each combination of hyperparameters we explored, we did a first-pass training until the process early-stopped, then fine-tuned for 20 epochs. We found that the optimal hyperparameter values are lr=0.001, bs=32, and λ=0.01.

Table 3. Training accuracies of hyperparameter tuning: learning rate and batch size (λ=0.01).

bs \ lr	0.001	0.005	0.01
16	0.2513	0.5711	0.2513
32	0.6452	0.6065	0.6225
64	0.6570	0.6361	0.6445

Table 4. Training accuracies of hyperparameter tuning: L2 regularization constant (lr=0.001, bs=64).

λ	Training Accuracy
0.005	0.6298
0.01	0.6361
0.05	0.6107

To evaluate the performance of our model, we measured several key metrics. The primary metric we focused on was accuracy, which indicates the overall percentage of correctly classified images. Additionally, in Table 5, we computed the precision, recall, and F1 score for each emotion category to assess the model's performance on individual emotions.

Our model achieved high accuracy (72.81% on the test set) and performed strongly for the categories of happiness and surprise, as indicated by high precision, recall, and F1 scores. However, the model's performance was relatively low for disgust and fear, with the lowest recall and F1 scores. There could be several reasons why. Firstly, as seen in Table 1, the dataset is heavily skewed towards happiness, with 7,215 happiness train examples, in contrast to only 436 disgust train examples. Another reason could be the inherent subjectivity and ambiguity of these emotions. For example, disgust and fear can sometimes be similar because they are both associated with potential threats or dangers, and their facial expressions can share certain visual cues such as widened eyes or open mouths. In such cases the model may struggle to capture the subtle nuances that differentiate these emotions.

Table 5. Performance metrics of our final DenseNet model.

	Precision	Recall	F1 Score
Anger	0.54	0.59	0.56
Disgust	0.70	0.32	0.43
Fear	0.52	0.46	0.49
Happiness	0.87	0.86	0.87
Neutral	0.58	0.66	0.62
Sadness	0.53	0.51	0.52
Surprise	0.80	0.77	0.79

In Table 6, we generated a confusion matrix to analyze the distribution of predicted emotions and identify any patterns of misclassifications between different emotions. We see that 44% of disgust examples were misclassified as anger, more than the 35% that were correctly classified as disgust. Furthermore, anger was often misclassified as neutral (12%) or sadness (13%), fear was often misclassified as sadness (19%), neutral was often misclassified as sadness (14%), and sadness was often misclassified as neutral (19%). Again, these patterns could be attributed to class imbalance in the training data, as well as the inherent overlap in the facial cues of certain emotions, making them more prone to confusion during the classification process.

Table 6. Confusion matrix of our final DenseNet model.

5.1.2 Qualitative Results & Discussion

Figure 6 shows examples of correctly and incorrectly classified images for each emotion category, providing a visual representation of the model's predictions. From these examples, we examined failure cases where the model struggled to accurately predict the emotions. One key failure case was in that the model struggled to tell apart certain emotions from each other.

Figure 6. Test set images sorted in rows by emotion category.

Interestingly, a vast majority of incorrect classifications involved the misclassification of negative emotions from each other. In tandem with the information provided in the confusion matrix, we see that a high number of images demonstrating disgust were categorized as anger. This informs us that although our model is better at identifying the facial feature divide between positive and negative emotions, nuances within the negative emotion categories may require our model to complete additional training to learn.

Another interesting observation involves how misclassified images are oriented. For instance, many of misclassified images were taken from a side-profile or odd angle, had a lack of centering which meant some facial features were not visible, and/or had complex visual features such as bad lighting or distracting backgrounds. These may also explain disparities within the model's ability to classify faces.

5.2. Testing the Mobile App

In addition to evaluating the model's performance, we conducted user testing sessions with two participants to test the practical application of our emotion recognition system using the mobile app. The participants provided valuable feedback on the user interface, ease of use, and their perception of the accuracy of the emotion predictions. Figures 7 and 8 show examples of how the app predicted a user's emotions correctly and incorrectly.

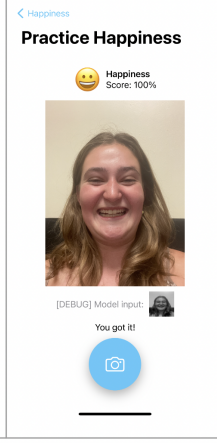
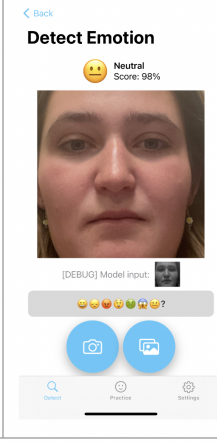
	Example #1	Example #2
Actual	Happiness	Neutral
Predicted	Happiness	Neutral
Screenshot		

Figure 7. App predicted correct emotions.

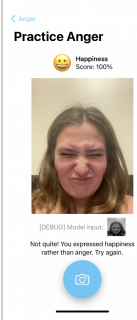
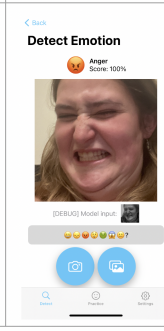
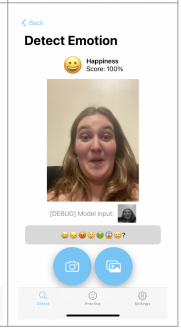
	Example #1	Example #2	Example #3
Actual	Anger	Disgust	Surprise
Predicted	Happiness	Anger	Happiness
Screenshot			

Figure 8. App predicted incorrect emotions.

Here are the key observations from the testing sessions:

- **Skew Towards Happiness**: We observed that the the predictions of user-provided images showed a noticeable skew towards the happiness category. As Participant A pointed out, "Why does it say I'm happy? I'm clearly sad in this picture!"
- **Feedback on User Interface:** Both participants found the user interface to be visually appealing and intuitive. They appreciated the clean layout and easily understandable icons and buttons.
- **Feedback on Ease of Use:** Both participants found the app to be straightforward to use, with clear instructions and simple navigation. They were able to capture images, upload them, and receive emotion predictions without any difficulties.
- **Feedback on Detection Mode:** While both participants found the detection mode visually appealing and easy to use, they occasionally experienced confusion due to inaccuracies in the emotion predictions. They noted that the app sometimes misinterpreted their facial expressions, leading to unexpected predictions. Despite these occasional inaccuracies, they found the tool amusing and appreciated the real-time feedback provided by the detection mode.
- **Feedback on Practice Mode:** Both participants found the practice mode to be engaging and helpful in improving their emotion recognition and expression skills. Participant A pointed out the "thoughtful and clear instructions for each emotion," while Participant B appreciated the "timely and encouraging feedback" which allowed them to adjust their facial expressions accordingly.

Overall, the testing of the mobile app yielded valuable insights. While there was a skew towards the happiness category in the predictions, participants appreciated the user-friendly interface, ease of use, and effectiveness of the app in enhancing emotion expression skills. The negative perceptions of accuracy indicate areas for potential improvement to ensure more consistent and nuanced predictions across various emotions of diverse people.

6. Conclusion

Overall, we found that EmoRec is an effective way to help train users emotional expressivity and identification. Test participants found the app to be friendly and easy to navigate and understand, which meets our goal of a seamless user experience. EmoRec's underlying DenseNet model, with an accuracy of 0.7281 on the test set, provided the app with its emotion identifying ability. Throughout the model's development, we experimented with techniques such as data augmentation, regularization, and hyperparameter tuning to boost its accuracy. However, its low accuracy rate and frequent misclassifications indicate that there is further room for improvement.

6.1. Future Work

In the future, we plan to focus on optimizing the accuracy of our model, for instance through testing different model types. Due to memory limitations, we had to switch from the AlexNet model to the DenseNet model to store it in the Core ML format. In order to compare the DenseNet and AlexNet models side by side within the EmoRec app, we intend to increase our memory allocation for further experimentation. While AlexNet and DenseNet are good starting points, we want to explore more advanced architectures specifically designed for image classification tasks, such as VGGNet [4], ResNet [7], or InceptionNet [17]. These architectures often have more layers and utilize techniques like skip connections or residual blocks to improve accuracy in emotion recognition.

Another crucial limitation we have identified involves the lack of diversity within the FER-2013 dataset. We observed a lack of spatial variation between dataset images, which created difficulties when integrating the model into our mobile app, especially when dealing with busy backgrounds and variably sized faces. This makes it challenging to accurately detect emotions if the faces being evaluated are not the main feature in the image. Moreover, the FER-2013 dataset, despite covering a broad range of emotions, predominantly consists of images showcasing facial features of young Caucasians. To address this issue, we plan to augment our training and testing datasets with images of faces from all ages and racial identities, perhaps by including other external emotion detection datasets, or collecting our own data through the mobile app. This augmentation will not only enhance the accuracy of our model but also ensure equity for EmoRec's users.

Finally, we propose conducting a user study aimed at addressing the effectiveness of EmoRec in assisting our target users with emotion recognition and expression. In particular, our study will focus on individuals with autism, social anxiety, and related disorders. By conducting this study, we aim to gain valuable insights into how EmoRec can be enhanced to better support the unique learning needs of these individuals.

7. Appendix

Python code and related files used for the development of EmoRec's DenseNet model may be found on Github: https://github.com/flowers-huang/cs231n-project/.
EmoRec's overall app framework, built on Swift and

Xcode, may also be found on GitHub: https://github.com/katchen1/EmoRecApp.

8. Contributions & Acknowledgements

The two authors, Flora Huang and Katherine Chen, contributed equally to this work.

Huang implemented, trained, and tested the emotion detection model using TensorFlow. She conducted experiments with different CNN architectures, hyperparameter tuning, and data augmentation techniques to improve the accuracy of the model. Huang also converted the trained model to the Core ML format, making it compatible with iOS devices. Additionally, she conducted evaluations and gathered performance metrics to assess the model using matplotlib [9], pandas [18], and numpy [6].

Chen designed and developed the EmoRec mobile app using Figma, Xcode, and Swift. She integrated the Core ML model into the app, enabling real-time emotion detection from images captured by the phone's camera or uploaded from the device's storage. Chen implemented the user interface, including the detection mode and practice mode, and conducted user testing and gathered feedback to evaluate the effectiveness of the app.

We would like to acknowledge the FER-2013 dataset [1], made available on Kaggle, which provided the labeled images for training and testing our model. Additionally, we would like to thank Katy Werwath and Alice Wang for their contributions to EmoRec through their user testing sessions. Finally, we thank the CS231N course instructors and teaching staff for providing the knowledge and materials that contributed to our completion of this project.

References

[1] FER - 2013 dataset with 7 emotion types. https://www.kaggle.com/datasets/ananthu017/emotion-detection-fer. 1, 3, 8

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 1

[3] Apple. coretoolsml. 1

[4] Yoshua Bengio and Yann LeCun, editors. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 7

[5] Tanner Gilligan and Baris Akis. Emotion AI, Real-Time Emotion Detection using CNN. https://web.stanford.edu/class/cs231a/prev_projects_2016/emotion-ai-real.pdf. 1

[6] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. 8

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 7

[8] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018. 2, 4

[9] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. 8

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 2

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*, 60(6):84–90, May 2017. https://doi.org/10.1145/3065386. 2, 4

[12] Shan Li and Weihong Deng. Deep Facial Expression Recognition: A Survey. *IEEE Transactions on Affective Computing*, 13(3):1195–1215, Jul 2022. https://doi.org/10.1109%2Ftaffc.2020.2981446. 2

[13] Alexander Miloff, Arvid Marklund, and Per Carlbring. The challenger app for social anxiety disorder: New advances in mobile psychological treatment. *Internet Interventions*, 2(4):382–391, 2015. https://doi.org/10.1016/j.invent.2015.08.001. 2

[14] Garvesh Raskutti, Martin J. Wainwright, and Bin Yu. Early stopping and non-parametric regression: An optimal data-dependent stopping rule, 2013. 3

[15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *CoRR*, abs/1404.7828, 2014. 3

[16] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 3

[17] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014. 7

[18] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. 8

[19] Suorong Yang, Weikang Xiao, Mengcheng Zhang, Suhan Guo, Jian Zhao, and Furao Shen. Image data augmentation for deep learning: A survey, 2022. 2