Touchy Feely: An Emotion Recognition Challenge

Dhruv Amin
Stanford University
dhruv92@stanford.edu

Patrick Chase
Stanford University
pchase@stanford.edu

Kirin Sinha
Stanford University
ksinha@stanford.edu

Abstract

In this project, we built a classifier to detect basic human emotions from facial expressions. To train the model we used a dataset posted on Kaggle consisting of images labeled with the emotions angry, disgust, fear, happy, sad, surprise, and neutral. We experimented with different CNN architectures and hyperparameters and were able to achieve an accuracy of over 61% on the test set. Lastly, we examined the models from a qualitative perspective using tools such as saliency maps to get a better sense of how the model makes its predictions. In this paper we explain our motivation for this project, the dataset we used, how we iterated on our model to come up with the final version, and examine our results.

1. Introduction

As humans, we classify emotions all the time without knowing it. We can see if someone is happy or sad or frustrated and in need of help. However, this is a very complex problem that involves many subtleties about facial expression. Even just the tiniest change in someone's face can be a signal of a different emotion. Training models that understand human emotion will be critical to building truly intelligent machines that can interact with us as humans do. In the near future with the rise of augmented reality, there could also be many applications for an emotion classifier to help people who have trouble recognizing emotions interact in a world where this is an essential skill. Other applications include aiding federal interrogations and advertisement targeting based on emotional state.

Recent work suggests that convolutional neural networks may increase the accuracies of prior methodologies in identifying emotions, reaching approximately 60% accuracy. We will explore such an approach given an input of greyscale images of human faces. We then use our own CNN architecture and hyperparameters to output a predicted emotion, which can be anger, disgust, fear, joy, neutral, sadness and surprise. Among the metrics we will use to evaluate our success are percent accuracy, confusion matrices, loss curves, saliency maps, and the outputted images and classifications themselves.

1.1. Related Work

Human facial recognition, given its importance, has been a fixture of computer vision research over the past few decades [19]. Researchers have applied a variety of techniques ranging from AU aware facial features [18] to RNNs for video [10]. However, only in recent years have CNNs been applied to the problem.

As demonstrated by Szegedy et al. [16] through their GoogLeNet architecture, it is possible to train deep (27 layer) convolutional nets to reach a near human level accuracy for facial recognition. For emotion recognition specifically, in 2006, Anderson et al. [1] developed a face expression system for the six basic emotions before convolutional nets were popularly used. Their system combines face tracking, an optical flow algorithm, Support Vector Machines, and Multilayer Perceptrons to achieve a 81.82% recognition accuracy within their application, EmotiChat. Another approach by Happy et al. [7] uses facial region localization, feature extraction, and SVMs to achieve an average accuracy of 94.09% on the 593 image sequence Extended Cohn-Kanade (CKP) dataset. Both papers are particularly clever in terms data preprocessing to achieve better test results.

In the last decade, convolutional networks regained in popularity in research papers for the emotion recognition task. G. Levi et al. [13] utilized both Local Binary Patterns to transform images to overcome illumination issues and transfer learning from the large CASIA WebFace dataset [14] the smaller Static Facial Expressions in the Wild (SFEW) dataset to overcome data sparsity issues. Their system achieve 55.56% accuracy. Song et al. [15] created a deep convolutional neural network for learning facial expressions that is quite simple, combining 65k neurons in five layers to achieve an accuracy of 99.2%. Recently, Burkert et al. [2] proposed a state of the art CNN based system that

can achieve 99.6% for CKP and 98.63% for the MMI Facial Expression Database. Both datasets combined number roughly 30,000 color images of mixed age, gender, and ethnic participants. Their system is primarily built upon simple combinations of convolutional, ReLU, and pooling layers, with the novel creation of a Parallel Feature Extraction (FeatEx) layer that is repeated in blocks throughout the network. These related works demonstrate that it is possible to build a state of the art system with at least 60% accuracy using convolutional techniques and layers.

2. Data

We used the dataset from the Kaggle challenge on Facial Expression Recognition, which gives 48x48 pixel grayscale images of faces and labels them using the established seven types of emotions: anger, disgust, fear, joy, neutral, sadness and surprise. We split between the data between training, validation, and test as follows: 28,709 - 3,589 - 3,589. The Kaggle dataset contains images that vary in viewpoint, lighting, and scale.

Figure 1 shows an example of an image in the dataset that is labeled happy. Figure 2 shows an example of an image in the dataset that is labeled with the emotion fear. By looking at these two photos we can see how complex the problem is. The faces are often facing different directions and are of different aged people. Making the data more challenging is the differing presence of items such as glasses or helmets, the possible obfuscation of the face by hair, and even the varying positioning of the hands and arms. We have also found some of the images like Figure 3 have water marks or other are quite blurry.

We did not extract any features from the images (PCA, ICA, etc.) before feeding them into the CNN, nor did we adjust in other ways such as subtracting the mean.

Figure 1. Image in training set with the label "happy."

3. Methods

Our approach was to experiment with different layer types and iterate on architecture design in response to training and validation results. Once we achieved satisfactory

Figure 2. Image in training set with the label "fear" and a water mark on it.

Figure 3. Image in training set with the label "happy" and a water mark on it.

performance solely from network design experiments we began running hyperparameter tuning experiments. The following is a brief overview of the network layers we experimented with:

Convolutional Layer: Convolutional Layers convolve the input by applying a filter with kernal size of n x m. Each CNN neuron has n x m connections to a local region of the input. The final output is a 3D volume of multiple filters.

$$x_{ij}^{l} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} w_{ab} y_{(i+a)(j+b)}^{l-1}$$

Max Pooling: Pool layers will perform a downsampling operation along the spatial dimensions of the input layer to reduce the size of the input

$$max\{x_{i+k,i+l} | k| \leq \frac{m}{2}, |l| \leq \frac{m}{2}\}$$

Batch Normalization: These layers help avoid issues in initialization of the input by forcing activations throughout the network to take on a unit Gaussian distribution.

Rectified Linear Unit: ReLU layers apply an element-wise activation function, such as thresholding at zero with max(0,x). The size of the input volume remains unchanged.

$$R(x) = max(0, x)$$

Fully Connected Layer: Fully connected layers have, as the name implies, neurons that are connected to every other

weight in the input volume. The final output results in a vector with dimensions the size of the number of classes.

Softmax and Cross Entropy Loss: The softmax classifier outputs intuitive normalized class probabilities by squashing the input between zero and one. The cross entropy between these probabilities and the true distribution, with all probability on a single label per sample, is minimized during back propagation of the error.

$$S(x)_j = \frac{e^{x_j}}{\sum_{i=1}^{N} e^{x_i}}$$

Adam Optimizer: Adam is used to descend down the gradient of the loss smoothly based upon the magnitudes of the previous gradients.

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 m_{t-1} + (1-\beta_2) g_t^2 \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\ \theta_{t+1} &= \theta_t \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \end{aligned}$$

Interestingly, existing research papers did not achieve exemplary performance on emotion recognition tasks from transfer learning, perhaps due to a lack of similarity between the dataset of similar models. Indeed, the fact that our dataset is in black and white compounds this problem as other datasets may have to be collapsed to work well. It would be very difficult to augment our dataset to allow it to work in conjunction with other colored images. Therefore, we instead experimented with building new architectures from baseline to max performance possible.

3.1. Architecture Experiments

We began with a simple baseline model with the following architecture in order to get a sense of our dataset:

Our most notable experiments are as follows:

1. **Baseline**
   Stacking multiple versions of this simple net on top of each other resulted in a max accuracy of 40% on the validation set before needing to introduce more powerful layers.

2. **[conv relu] x3 [maxpool] [fc relu] x2**
   Our next notable experiment was to add max pooling in order to allow more convolutional layers to exist without exploding the number of parameters. We also experimented with different forms of ReLU, such as Leaky ReLU, but did not notice any improvement. The deeper network increased performance to 51% validation accuracy.

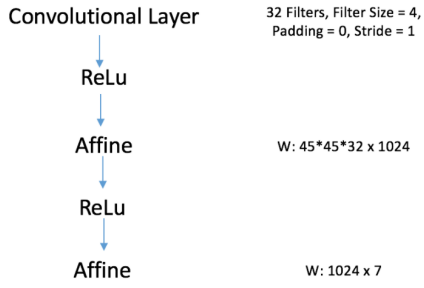

Figure 4. Inital model used to establish baseline training and validation accuracy.

3. **[conv relu bn maxpool] x4 [fc relu dropout] x3**
   Noticing overfitting in our training validation, we experimented with adding dropout to our final fully connected layers. In addition, to mitigate the impact of poor initialization or random distributions of outputs decreasing learning rate, we added batch normalization between convolutional blocks. Finally, with both changes, we were able to experiment with increasing the depth of the network once more and ultimately settled on four convolutional blocks followed by three fully connected layers for a 55% validation accuracy.

4. **[conv relu conv mp bn] [conv relu convx2 mp] x2 [fc]**
   In parallel to the above experiments, we explored slightly different opposite approaches to architectures we noticed in papers above. One such model was to avoid batch normalization in intermediate layers and to eliminate most fully connected networks. In addition, we began designing stackable intermediate block with pooling layers in between to increase the size of the network without exploding parameters. Ultimately architectures with this structure achieved max performance of 56.5% validation so we continued on from here.

5. **[conv bn relu drop] [conv bn relux2 mp drop] x3 [fc bn relu drop] x2**
   To prevent overfitting in order to increase the size of the net without decreasing performance, we introduced dropout on intermediate layers of the above architecture. This allowed us to increase both the number of intermediate blocks and fully connected layers. We likely could have experimented with removing inter-

mediate max pooling layers in order to increase the size of the network but were unable to find a configuration that didn't result in immediate overfitting. Our performance with the deepest configuration of this net reached 60.5% validation accuracy, in line with results we had seen in other papers.

Our final architecture is presented in figure 5.

3.2. Hyperparameter Tuning

Once we had established a successful base architechure that reached state-of-the-art accuracy, we sought to improve our performance through hyperparameter searches. The following represent the main hyperparameters adjusted.

Filter size: In an effort to increase training time while retaining accuracy, we experimented with slowly increasing the number of filters through successive layers instead of holding all convolutional layers at constant filter numbers. We were able to decrease the number of parameters in our final model by two orders of magnitude and still achieve final validation accuracy within 1% of our previous model at 59.5% on validation. As we had the luxury of training time, we kept our model with the best performing number of filters at each convolutional layer

Kernel sizes: After numerous runs to attempt a range between 1 and 7, we settled on a kernel size of 4 for each convolutional layer.

Padding: Interestingly, we found our best performance in intermediate layers occurred when we alternated padding between 3 and 1 on successive convolutional layers. We lifted the technique after noticing it in multiple papers and presume the better fit of the data results from applying a more appropriate padding size after the input has been shrunk from a maxpool.

Dropout: After experimenting with ranges between 10% and 50% we ultimately settled on 20% as best in mitigating some overfitting while still retaining accuracy.

Affine layer sizes: We determined after multiple experiments that 1024 neurons in our final fully connected layers led to the best performance for our model over larger or smaller layer sizes.

Learning rate: We determined starting learning by choosing the default rate recommended in the lecture notes. In the interest of time, we did not dive deep into finding the perfect learning rate because minor variations led to performance losses and our Adam update rule should adjust learning rate over time.

Number of Layers: Finally, after we settled on our final architecture, we varied the number of intermediate blocks and end fully connected layers, ultimately landing at 7 blocks and 2 fully connected output layers. We likely could increase the number of layers if we eliminated max pooling in our intermediate layers, and this conclusion is supported by our training and test plots. The tradeoff is the increase in the size of the model substantially hurting epoch time.

3.3. Implementation Details

We built our model entirely using PyTorch. For development, we used an iPython Notebook similar to those in our previous assignments, and we developed using GPUs on Google Cloud. We would run each of our models for 10 epochs on the cloud after which the performance seemed to

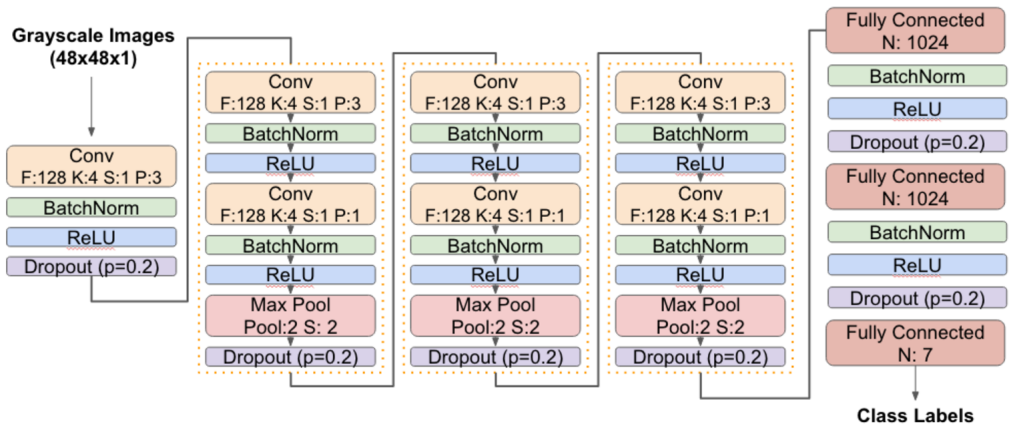

Figure 5. Final architechure used for task of emotion recognition.

level off. The 10 iterations would take anywhere from 20 to 45 minutes, so we were able to iterate fairly quickly with all of us testing parameters and architectures at the same time.

4. Results

4.1. Accuracy

We were able to achieve good results even with our baseline model. For our training accuracy we got 18147 / 28709 images classified correctly an for validation we had 1429 / 3589 images classified correctly. From the baseline, we were able to achieve a higher validation accuracy using deeper models with dropout without changing the training accuracy and causing additional overfitting. Test and validation set accuracies for our final model were averaged over 4 runs. For the test set, only the last run was used. Our final model performed well on the test set, reaching state-of-the-art accuracy levels using CNNs.

Model	Training	Validation	Test
Baseline	.6321	.3982	
Intermediate	.6305	.5542	
Final	.7849	.6037	.6105

Table 1. Accuracies achieved using various models

4.2. Confusion Matrix

We examine the confusion matrix for the test set to understand better where misclassifications occur. We see that on the whole, our classification is fairly good, as seen by the strong presence along the diagonal. The model is most successful at classifying “happy correctly. While the saliency maps may offer more clarity on this issue, it may be that happy has clear indicators, such as a smile, that make it easier for the model to identify. The model most struggles to determine between “sad and “fear. Again, intuitively this might make sense as those two emotions often present similarly. These results from the confusion matrix suggest that removing one of “sad” or “fear” from the list of possible emotions may significantly improve our classification.

4.3. Loss and Accuracy Over Time

Looking at graphs in figure 7, we see a healthy loss curve. The loss decreases substantially in the first epoch and then slowly after that. We do see the model is learning fairly slowly in the later epochs, so there might be a way to set the learning rate to achieve more efficient learning, but by running for more epochs we were able to get the algorithm to converge nevertheless. In the second plot in figure 7, we can see the training and validation accuracy as at the end of each epoch for our best model. While there is a gap between the training accuracy and the validation accuracy which suggests overfitting, we found that when we

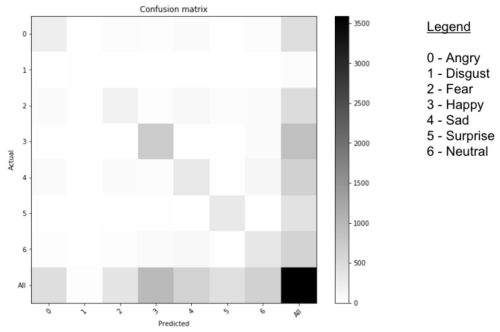

Figure 6. Confusion matrix on the test set using the final model.

increased the regularization parameter to bring the validation accuracy closer to the training accuracy, the training performance decreased substantially and overall the validation performance was worse. So even though there is the gap, this model has the best performance of any of the models we tried. However, the gap does suggest that we could potentially improve performance even more by exploring a finer granularity of regularization hyperparameters, adding more dropout or changing the CNN architecture to bring the validation accuracy closer to the training accuracy.

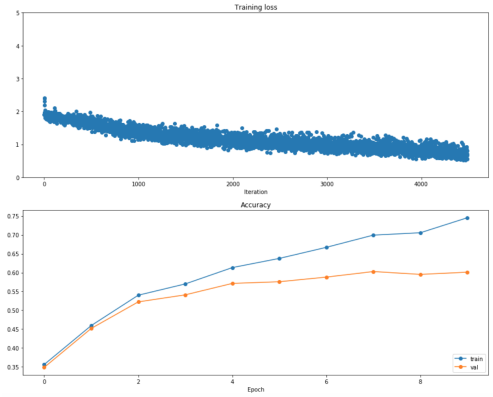

Figure 7. Loss and training and validation accuracy after each epoch.

4.4. Saliency Maps and Qualitative Assessment

Saliency maps show which pixels contribute the most to the classification, helping us to gain insight into how our convolutional net is making classifications. Interestingly, when examining the saliency maps we see that the model does seem to be influenced by pixels we would expect for a variety of emotions. For example, in the saliency maps in figure 8, we can see that for the images classified as “happy” the pixels around the smile or presence of dimples contribute the most to the classification of happy. In the image on the left, even the squint in the child’s eye is

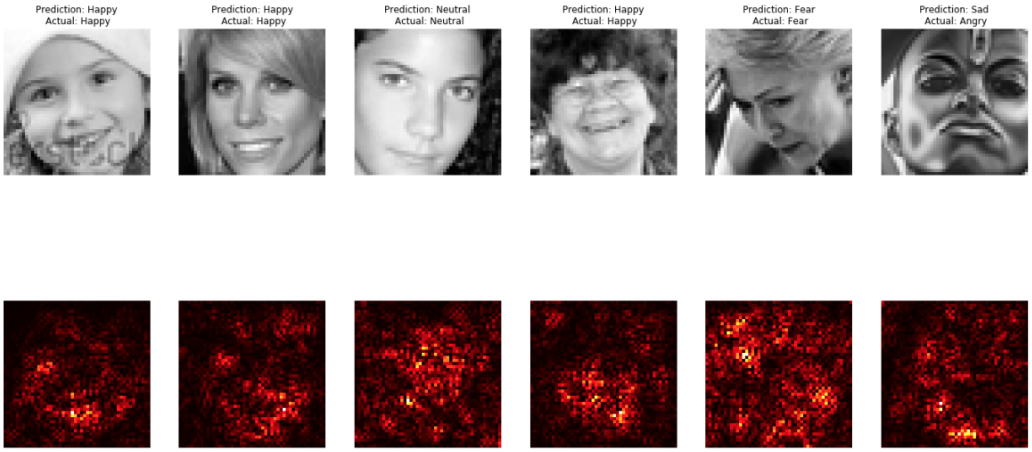

Figure 8. Classifications and saliency maps for examples in the validation set.

contributing more than the rest of the child's face to the classification. This particularly interesting because the CNN has learned to pick up on the subtle features that we as humans use to classify emotions subconsciously. In addition, we saw the the position of the eyebrows became salient for "surprise.

Given this information, we gain better insight into how some misclassifications may be occurring. For example, in the final picture on the right, "sad rather than "angry may have been predicted due to the emphasis on the pixels relating to the position of the mouth. The pixels on the chin are contributing a lot to the classification of "sad". It may be that the algorithm mistakenly thought the chin was the mouth for this photo due to the angle of the picture. Finally, when we manually examine our examples, we were surprised how often we even had disagreement over whether an emotion should truly be considered "surprised" or "fear", "happy" or "surprised", etc. One potential extension for future work could be to collect multiple labels per photo from Amazon Turk and then modify our net to instead output probabilities to see how often the computer is unsure when multiple humans are unsure.

5. Conclusion

In this project, we sought to classify the image of a face into the seven basic human emotions. We developed and experimented with the architecture of a deep convolutional neural network ourselves, and performed a hyperparameter search to optimize our results. We were able to reach the state-of-the-art test accuracy of approximately .61. However, we believe if we addressed the overfitting of the training data, we could reach even higher test accuracies.

A possible extension of our work would be to enable a real-time facial emotion recognition. Such a task could be accomplished by using YOLO v2 and training it to identify faces in the real time. From those faces, we could capture still images and then use our convolutional net to identify an emotion. A final emotion would be outputted if either a majority of the still frames captured matched one emotion, or alternatively, if we knew it was that emotion with some confidence interval.

References

[1] K. Anderson and P. W. Mcowan, A real-time automated system for recognition of human facial expressions, IEEE Trans. Syst., Man, Cybern. B, Cybern, pp. 96105, 2006.

[2] P. Burkert, F. Treir, M. Afzal, "DeXpression: Deep Convolutional Neural Network for Expression Recognition" in German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany.

[3] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, Return of the devil in the details: Delving deep into convolutional nets, in British Machine Vision Conference, 2014.

[4] CS231n. Lecture Notes. http://cs231n.github.io/convolutional-networks/

[5] A. Dhall, R. Goecke, S. Lucey, and T. Gedeon, Static facial expression analysis in tough conditions: Data, evaluation protocol and benchmark, in Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, pp. 21062112, Nov 2011.

[6] Facial Expression Research Group Database. http://grail.cs.washington.edu/projects/deepexpr/ferg-db.html

[7] S. Happy and A. Routray, Automatic facial expression recognition using features of salient facial patches, Affective Computing, IEEE Transactions on, vol. 6, no. 1, pp. 112, Jan 2015.

[8] S. Ouellet, "Real-time emotion recognition for gaming using deep convolutional network features," CoRR, vol. abs/1408.3750, 2014.

[9] Kaggle. "Challenges in Representation Learning: Facial Expression Recognition Challenge." https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/leaderboard

[10] S. Kahou, V. Michalski, K. Konda, R. Memisevic, and C. Pal, Recurrent neural networks for emotion recognition in video, ICMI, pp. 467474, 2015.

[11] T. Kanade, J. F. Cohn, and Y. Tian, "Comprehensive database for facial expression analysis," in Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on, pp. 46-53, 2000.

[12] A. Kolakowska, A. Landowska, M. Szwoch, W. Szwoch, and M. R. Wrobel, Human-Computer Systems Interaction: Backgrounds and Applications 3, ch. Emotion Recognition and Its Applications, pp. 51-62. Cham: Springer International Publishing, 2014.

[13] G. Levi and T. Hassner, Emotion recognition in the wild via convolutional neural networks and mapped binary patterns, in Proc. ACM International Conference on Multimodal Interaction(ICMI),November 2015.

[14] M. Pantic, M. F. Valstar, R. Rademaker, and L. Maat, Web-based database for facial expression analysis, in Proceedings of IEEE Intl Conf. Multimedia and Expo (ICME05), Amsterdam, The Netherlands, July 2005, pp. 317321.

[15] I. Song, H.-J. Kim, and P. B. Jeon, Deep learning for real-time robust facial expression recognition on a smartphone, in Consumer Electronics (ICCE), 2014 IEEE International Conference on . IEEE, 2014, pp. 564567

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, Going deeper with convolutions, CoRR, vol. abs/1409.4842, 2014. [Online].

[17] Visual Geometry Group. "Very Deep Convolutional Networks for Large-Scale Visual Recognition." University of Oxford. http://www.robots.ox.ac.uk/ vgg/research/very_deep

[18] A. Yao, J. Shao, N. Ma, and Y. Chen, Capturing au-aware facial features and their latent relations for emotion recognition in the wild, in Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, ICMI 15, (New York, NY, USA), pp. 451458, ACM, 2015.

[19] D. Yi, Z. Lei, S. Liao, and S. Z. Li, Learning face representation from scratch, CoRR, vol. abs/1411.7923, 2014.

[20] Z. Yu and C. Zhang, "Image based static facial expression recognition with multiple deep network learning," in Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, ICMI 15, (New York, NY, USA), pp. 435-442, ACM, 2015.