Abstract—There has been much research on facial affect detection, but many of them fall short on accurately identifying expressions due to changes in illumination, occlusion, or noise in uncontrolled environments. Also, not much research has been conducted on implementing the algorithms using multiple datasets, varying the size of dataset and the dimension of each image in the dataset. Our ultimate goal is to develop an optimized algorithm that can be used for real-time affect detection in automated vehicles. To this end, in this study we implemented the facial affect detection algorithms with various datasets and conducted a comparative analysis of performance across the algorithms. The algorithms implemented in the study included a Convolutional Neural Network (CNN) in Tensorflow, FaceNet using Transfer Learning, and Capsule Network. Each of these algorithms was trained using the three datasets (FER2013, CK+, and Ohio) to get the predicted results. The Capsule Network showed the best detection accuracy (99.3%) with the CK+ dataset. Results are discussed with implications and future work.

I. INTRODUCTION

Face detection applications are widely used in smart-phones for lock and payments to increase security. More advanced technologies have also been used to detect users’ facial affect expressions to facilitate social and emotional interactions between people and technologies. The resurgence of neural networks has improved such face detection and facial expression detection research. The artificial intelligence (AI) system can estimate users’ emotions by learning about each expression and applying its algorithm to a new set of facial expressions. Given that emotions can negatively influence on driving [1], [2], we have conducted research on emotion detection while driving [3]. A recent study has also shown that emotions like anger can have an impact on drivers’ takeover performance in semi-automated vehicles [4]. Emotional events can happen in the car that might increase the risk of an accident. Incidents that happen inside the car such as receiving a disturbing phone message, spilling the coffee, witnessing a car accident by the side of the road, or seeing an animal injured by a car can make them emotional and thus, distract drivers. The affect detection and mitigation system in semi-automated vehicles will detect drivers’ affective states as well as stress, confusion, or sleepiness. The vehicles can use AI algorithms to understand the drivers’ behavioral context by taking real-time data from cameras and other sensors to support their needs. Then, the system can take over the driving control to allow the driver to recover from the risky states.

To design an optimal affect detection system in the automated vehicle context, the present work implemented the currently available facial expression detection algorithms with varying data sets and quantitatively analyzed their performances.

II. RELATED WORK

There has been considerable research and development of facial affect detection systems using different algorithms with various datasets. The recent development of advanced neural network has stepped up these efforts. For example, most of the facial expression detection algorithm uses Convolutional Neural Network (CNN).

Mollahosseini et al. [5] trained a single CNN for FER dataset using two types of convolution layers with different range filters, one max pooling layer, and four “inception” layers i.e., sub-networks. The method achieved comparable accuracy but as the number of layers increased, the computing power needed increased. Minaee and Abdolrashidi [6] created an attentional convolutional network to classify the basic emotion in the face images, thereby attending to the important regions for detecting emotion. The algorithm achieved the accuracy of 70%. Pramerdorfer and Kampel [7] created an ensemble based deep CNN network and achieved a test accuracy of 75.2% on FER2013 dataset. The ensemble method determined the class by using the highest weighted mean along with the posterior class probabilities resulted from each individual with different weights. Fairly recently, Zhang and Xiao [8] reported implementing a Capsule Network for classifying CK+ dataset. The images were processed using data augmentation before sending it through the network. The accuracy reached up to 86.7%. Later, combining the classification results with the Nao robot makes it to visualize the emotion by changing its eye colors.

Some of them have tested multiple datasets. In a recent work, Ghaif [9] provided an architecture for a CNN of three layers. Each layer was followed by a pooling layer and three dense layers. The data were segregated as 90% for
training while 10% was used for testing. The dropout rate of the dense layer was 20%. Features were extracted with higher accuracy from the images using intensity normalization and contrast enhancement. Faces were detected using the pre-trained model of facial landmark detector. The next step of image enhancement and duplication was done by taking five copies of each image and applying the histogram equalization or bilateral filter to the cropped face image. Datasets used were JAFFED and KDEF, and the accuracy achieved by combining the datasets was 78%. Khorrami et al. [10] addressed the CK+ and the Toronto Face Dataset (TFD) by using zero-bias CNN and deep learning method based feature extraction to achieve advanced emotion detection results. It achieved an accuracy of around 95%, but it did not deal with non-posed images.

Others have implemented their systems using different models at the same time. Quinn et al. [11] built SVM and CNN models capable of recognizing seven emotions using the one vs. all (OVA) [12] sklearn’s linear kernel SVM model. Different methods were tried on the SVM model to increase the accuracy. The first method was using the scaled pixel values as the new features. The scaled pixels were created such that each image had a mean pixel value of 0 and a variance of 1. Principal Component Analysis (PCA) was used as the next method to segregate the most important components, thereby reducing the dimensionality of the training set. Then, the Histogram of Oriented Gradients (HOG) described the distribution of gradients because different emotions had different gradients around the mouth and eye areas. Next, the CNN model was made with an increased number of layers, decreased filter sizes and an increased number of parameters for a correct prediction on images which had smaller facial features. Overfitting in the model was handled by using dropout layers, early stopping around 100 epochs, and augmenting the training set. It obtained an accuracy of around 65%. The datasets used for training both the models were FER2013 and CK+. Kaur and Gandhi [13] used various pre-trained deep CNN (DCNN) models such as AlexNet, Resnet50, GoogLeNet, VGG-16, Resnet101, VGG-19, Inceptionv3, and InceptionResNetV2. The last few layers of each of the models were replaced to accommodate the brain Magnetic Resonance (MR) images. The test accuracy on the pre-trained AlexNet yielded the best performance of more than 90%.

On the other hand, Zhi et al. [14] confirmed that using different datasets would lead to distinct detection performance. They examined whether the Support Vector Machine (SVM) classifiers trained on children or adults dataset work on vice versa groups due to the difference in facial features and expression. For this, they used the constrained local neural fields method for extracting facial features along with normalization. As expected, the affect detection accuracy increased when the dataset was trained with the target population (i.e., children’s data for children’s facial affect detection). In other words, the detection accuracy of the same algorithm can be significantly different depending on the datasets used.

In sum, despite many developments, most of the previous algorithms dealt with neither smaller datasets nor datasets having higher resolution images such as the Ohio dataset. Also, the detection accuracy was not high enough to use the algorithm in the real-world settings, such as driving. Based on this background, we tried to achieve higher accuracy than these algorithms by incorporating some of the unique properties of each research work.

III. DATASETS

As a first step to design a better system, we implemented three different algorithms using three facial expression datasets. These datasets have been chosen because these datasets are well managed with emotion labels in separate files. Each dataset has been used to implement the systems a few times, but little research has focused on direct comparisons of the performance of these datasets.

1) FER2013

The dataset consists of grayscale images with pixel values of 48*48. There are images of seven categories and each facial expression falls in these categories which are six basic emotions (anger, disgust, fear, happiness, sadness and surprise) [15] + neutral. Each emotion is labelled from 0 to 6 sequentially. The train.csv file contains 2 columns, “emotion” and “pixels”. The emotion column contains the numeric code for emotion in which the image lies into. The pixels column contains the string separated pixel values of each image. The dataset contains 35887 images. Emotion labels in the dataset are classified as follows:

- 0: 4593 images Anger
- 1: 547 images Disgust
- 2: 5121 images Fear
- 3: 8989 images Happiness
- 4: 6077 images Sadness
- 5: 4002 images Surprise
- 6: 6198 images Neutral

FER2013 is the baseline dataset for the methods implemented in section IV, as the algorithms we included in the present study have already been implemented or constructed using the FER dataset. The dataset is available in the following link [16].

2) Extended Cohn-Kanade (CK+)

The CK+ database has the number of subjects increased by 27% and the number of sequences by 22% from the original database [17]. The target expression for each sequence is fully FACS (Facial Action Coding System) coded and emotion labels have been validated [17]. The CK+ comprises a total of 593 sequences across 123 subjects. Sequences range from neutral to peak expression as shown in Figure 1. Emotion labels in the dataset are classified as follows:

- 0: 135 images Anger
- 1: 177 images Disgust
- 2: 75 images Fear
- 3: 207 images Happiness
• 4: 84 images Sadness
• 5: 249 images Surprise
• 6: 54 images Contempt

Fig. 1. CK+ Image Sequence

The dataset is available at the following link [17].

3) Ohio Compound Facial Expressions

The dataset consists of images with 21 compound emotion categories. Basic component categories are combined to create newly constructed compound emotions [18]. The creation of these 21 categories is different and consistent with the sub categories as per the FACS analysis. For example, a sadly disgusted combines the muscle movements seen in sadness and disgust. The basic component categories consist of six basic emotions + neutral. Total images with pure emotions are 1610.

Emotion labels in the dataset are classified as follows:

• 0: 230 images Anger
• 1: 230 images Disgust
• 2: 230 images Fear
• 3: 230 images Happiness
• 4: 230 images Sadness
• 5: 230 images Surprise
• 6: 230 images Neutral

The database is obtained at the following link [19].

IV. METHODS

In the present study, we compared three main algorithms. Each of the algorithms has a convolution as one of the layers because research has shown that CNN models can provide higher accuracy than the traditional methods, like SVM (Support Vector Machine) or Random Forest [20].

a) Convolutional Neural Network in Tensorflow

A CNN consists of two types of layers:

i) Feature Extraction: Convolutions and Pooling

ii) Classifier

CNN uses a filter to get a feature map over the input images. Pooling technique is used for dimensionality reduction to reduce the number of parameters. There are numerous types of pooling techniques but the popular ones are max, average, and median pooling. Average pooling takes the average in each window which decreases the feature map size while keeping the average of all the information. Dropout is a technique where randomly selected neurons are dropped during training to reduce overfitting. We have selected the ReLU activation function over sigmoid because ReLU is computationally more efficient and it has better convergence performance than sigmoid [21]. Categorical cross-entropy is chosen as the loss function because cross-entropy calculates a score that summarizes the average difference between the actual and predicted probability distributions and categorical values ensure the output as multi-class.

The following Figure 2 is the CNN architecture which has been made. The last dense layer outputs a seven dimensional vector for the seven emotions.

| Layer (type) | Output Shape | Param # |
|--------------|--------------|---------|
| conv2d_1 (Conv2D) | (None, 48, 48, 6) | 456 |
| max_pooling2d_1 (MaxPooling2D) | (None, 24, 24, 6) | 0 |
| conv2d_2 (Conv2D) | (None, 24, 24, 16) | 2416 |
| activation_1 (Activation) | (None, 24, 24, 16) | 0 |
| max_pooling2d_2 (MaxPooling2D) | (None, 12, 12, 16) | 0 |
| conv2d_3 (Conv2D) | (None, 10, 10, 64) | 9280 |
| max_pooling2d_3 (MaxPooling2D) | (None, 5, 5, 64) | 0 |
| flatten_1 (Flatten) | (None, 1600) | 0 |
| dense_1 (Dense) | (None, 128) | 264928 |
| dropout_1 (Dropout) | (None, 128) | 0 |
| dense_2 (Dense) | (None, 7) | 98 |

Total params: 217,983
Trainable params: 217,885
Non-trainable params: 0

Fig. 2. Implemented CNN Architecture

b) FaceNet using Transfer Learning

FaceNet is a unified embedding for Face Recognition and Clustering. FaceNet manages to learn a mapping of face images to Euclidean distances where the distances denote the measure of face similarity. FaceNet embeddings can be used as feature vectors for face recognition, verification, and clustering. For this work, the method was used to extract the faces using the FaceNet_pytorch python library and applying transfer learning for expression detection. Transfer learning focuses on storing and applying the knowledge gained while solving one problem and applying it to a different but related problem [22]. Once the face is extracted, it is transformed to single channel grayscale for emotion classifier. The goal is to finetune a pre-trained ImageNet model for the datasets of CK+ and Ohio. The pre-trained model provides the training a good starting point. RGB images (Ohio or CK+) are fed as input to the pre-trained Residual Networks (ResNet)-50 model. ResNet tries to resolve the vanishing gradient
problem in deep neural networks. The problem is tackled by using skip connections. The skip connection creates an alternate shortcut path for the gradient flow, thereby it ensures to learn an identity function that makes the higher network layer to work as good as the lower one.

A new pre-trained model is created by swapping the first convolution layer of ResNet-50 with a new initialized 1 channel convolution layer. This is the Gray ResNet-50 model. The difference between the output embeddings of the two ResNet models is computed as the L2 loss used in backpropagation to update the Gray ResNet-50 model as shown in Figure 3. The model is further finetuned using FER dataset and the output is seven-dimensional vector classifying emotions. Gray images from CK+ or Ohio dataset again finetunes the model with last eleven layers as frozen. The layers are kept as frozen as the number of images are limited and the number (eleven) gave the highest accuracy. The model thereby learns from richer features present in the “wild” (our) dataset. Grayscale images are used here because the model should focus on the actual facial expression and not learn any biases that may come with color.

c) Capsule Network

i) Problem with Convolutional Neural Network

The use of pooling layer in CNN loses a lot of valuable information and it ignores the relation between part and the whole. For instance, a CNN is used as a face detector in images. A CNN would have learnt that there are five fundamental features that make up a face (two eyes, nose, mouth, and the oval shape of a face). However, there could be an image that has an oval shaped figure, with two eyes, a nose and a mouth all lying outside this oval shaped figure. Then, the CNN will classify it as a face, too. This issue of max-pooling causes incorrect results related to the application for image classification. The Capsule Network maintains the hierarchical pose between the parts/features of the object in an image. Capsule Network has a base on the CNN, but the neuron form is converted to the vector from scalar which contains highly informative outputs. [24].

ii) Difference Between Neurons and Capsules

The length of the output vector represents the probability that the entity exists, and its orientation represents the instantiated parameters. The direction that the output vector points to encodes the particular detected feature state. The probability (length of the vector) is independent of the position of the detected feature within the image, however its orientation is dependent on eye orientation.

In a CNN, a neuron receives the scalar inputs from other lower layer neurons. The inputs are multiplied by scalar weights which then get summed up. The sum is passed to the activation function which outputs another scalar value according to its functionality. The weights are later finetuned using the backpropagation algorithm, to match the target output values. The calculations happening in the Capsule Network vs. the neurons are shown in Table I.

Table I

| Input from low-level capsule/neuron | vector($u_j$) | scalar($x_i$) |
|------------------------------------|--------------|---------------|
| **Operation**                      | **Affine Transform** | $u_{ij} = W_{ij} a_i$ | - |
| **Weighting**                      | $s_j = \sum_i c_{ij} u_{ij}$ | $a_j = \sum_i w_i x_i + b$ |
| **Sum**                            | $v_j = \frac{|x|}{||x||} x_i$ | $h_j = f(a_j)$ |
| **Nonlinear Activation**           | $\|x\| = \sqrt{x_i^2 + x_j^2}$ | $||x|| = \sqrt{x_i^2 + x_j^2}$ |
| **Output**                         | **vector($v_j$)** | **scalar($h_j$)** |

Table 1: Capsule vs. Traditional Neuron

Fig. 3. Fine-tuning Resnet-50 using grayscale versions of ImageNet [23].
C) Sum of weighted Input Vectors
All of the weighted input vectors are added
to produce a single sum.

D) Vector-to-Vector Non-linearity.
Vector-to-Vector non-linearity, also known as Squash, modifies a vector and squeezes
it to a unit length without changing its direction.

iii) Dynamic Routing
Dynamic routing decides where each capsule output goes. As shown in Figure 4, the lower
capsules make the decision about which higher level capsule the output is to be sent to. The
lower capsule’s output is multiplied with the weights (C) before passing it to J or K (left or right). These are the higher level capsules. These capsules receive many input vectors from
other lower capsules. As shown in Figure 4, the inputs are shown by red and blue points.
The predictions of lower level capsules come together when these points cluster together.
Thus, after multiplying with matrix W, it lands away from prediction in capsule J and close
to prediction in capsule K. Thus, capsule K accommodates the target result well, thereby
adjusting its weights.

![Figure 4. Lower level capsule sends its input to the higher level one that “agrees” with its input.][25]

iv) Capsule Architecture
The architecture has 2 components: Encoder and Decoder.

A) Encoder
The encoder takes an image as input and encodes it into vector of instantiation parameters through learning. It consists of three layers which are the Convolutional Layer, PrimaryCaps Layer and the DigitCaps Layer. The PrimaryCaps Layer creates combinations of the features, which are detected by the convolutional layer. The DigitCaps Layer outputs a 16x10 matrix. It has seven digit capsules (one for each emotion). For each of the seven vectors denoted for
each emotion, loss value is calculated as shown in equations below [25]:

\[
L_C = T_C \max(0, m^+ - ||v_c||^2) + \lambda (1-T_C) \max(0, ||v_c|| - m^-)^2
\]

where;
\(L_C\) = loss term for one DigitCap
\(T_C\) = 1 when correct DigitCap, 0 when incorrect
\(m^+\) = zero loss when correct prediction with probability greater than 0.9, non-zero otherwise
\(m^-\) = L2 norm
\(\lambda\) = 0.5 constant used for numerical stability

B) Decoder
The decoder decodes the DigitCaps output vector into an image (reconstruction). It uses the output of the correct DigitCap layer to learn and reconstruct a 48*48 pixel image. The Euclidean distance between the reconstructed image and the input image is used as the loss function. It has three fully-connected layers which produce an output vector of length 2304 which is then reshaped to give back a 48x48 decoded image.

V. Experiment
Each dataset has been divided into training and validation sets. Training accuracy denotes accuracy of the network to correctly predict emotions for the images in the training set. Validation accuracy is the accuracy for predicting emotions for images in the validation set. Training accuracy turns out to be higher compared to the validation. This is due to the network learning patterns from the training set which may not be present in the validation one. The data were split into the training set and the validation set in the ratio of 85:15. The results and the graphs obtained by applying the three algorithms for the (CK+) Extended Cohn-Kanade and the Ohio dataset were described below. FER2013 is the base dataset for the methods implemented in section IV. For the classification report, precision, recall and f1-score
were calculated. Precision is the ratio of true-positives to the total predicted positives. Recall is the ratio of correctly predicted positives to the total observations in the class. F1-score is weighted average of Precision and Recall.

A. Convolutional Neural Network in Tensorflow

a) CK+ Dataset

The training accuracy obtained is 82.35% and the validation accuracy is 65.5%.

The following Figure 5 shows the predictions on the images.

![Output predicted emotions on Cohn-Kanade Dataset](image)

Fig. 5. Output predicted emotions on Cohn-Kanade Dataset

The graphs of Training loss vs Epochs and Validation accuracy vs Epochs are shown in Figure 6.

![Training loss and Validation accuracy graphs with CK+ dataset](image)

Fig. 6. Training loss and Validation accuracy graphs with CK+ dataset

The low validation accuracy versus the high training accuracy in this case can be related to the reason of overfitting. Machine learning models suffer from a problem of overfitting, where the model learns the parameters so well that it is unable to adjust to the new scenarios resulting in high variance. The overfitting can be dealt by fine-tuning the architecture of the CNN; some examples include finding and removing redundant parameters, adding new parameters in more useful places in the structure, adjusting the learning rate decay schedule, adapting the location and probability of dropout and experimenting to find ideal stride sizes.

B. FaceNet using Transfer Learning

a) CK+ Dataset

The training accuracy obtained is 62.46% and the validation accuracy is 41.17%. The graphs of Training loss vs Epochs and Validation accuracy vs Epochs are shown in Figure 8.

![Training loss and Validation accuracy graphs with CK+ dataset](image)

Fig. 8. Training loss and Validation accuracy graphs with CK+ dataset

b) Ohio Dataset

The training accuracy obtained is 89.25% and the validation accuracy is 50.91%. The graphs of Training loss vs Epochs and Validation accuracy vs Epochs are shown in Figure 9.

As can be seen from the graph of the validation accuracy, the validation accuracy reached relatively high accuracy at an epoch of around 20. This is called the optimum model complexity. After this point, the model started overfitting.
C. Capsule Network

Capsule Network causes underfitting when lesser dataset images are used for training [8]. To overcome this issue, data augmentation is done by rotating, amplifying and adding the salt and pepper noise. After augmenting, the CK+ dataset had a total number of images to 157885 and the Ohio dataset to 21134.

a) CK+ Dataset

The training accuracy obtained is 99.7% and the validation accuracy is 99.3%. The classification report using the ground truth and the predicted labels of neutral + six basic emotions are shown in Figure 10.

![Classification Report for the CK+ Dataset](image)

b) Ohio Dataset

The training accuracy obtained is 97.3% and the validation accuracy is 96.2%. The classification report using the ground truth and the predicted labels of neutral + six basic emotions are shown in Figure 12.

![Classification Report for the Ohio Dataset](image)

VI. CONCLUSION

Rigorous work is needed to implement and evaluate a facial affect detection system which can be used in real-time. If well implemented, such a system can play a vital role in semi-automated vehicles and many other application domains. However, most algorithms in literature were unable to tackle smaller datasets and datasets having high resolution images. They also did not achieve accuracy as high as 90%. To tackle these
issues, we conducted a comparative analysis of the facial expression detection methods using the validation accuracy on the FER2013, CK+, and Ohio datasets as can be seen in Table II. The important result is that the capsule network shows higher accuracy than other methods. This is partly due to the advantage of the Capsule Network taking vectors as input instead of scalar values as in a fully connected convolution layers. The issues due to max pooling are also resolved in the Capsule Network. Algorithms trained and validated on CK+ (extended cohn-kanade) dataset is shown to have higher accuracy than others due to its grayscale images, which capture the features better. Also, CK+ dataset has a small number of subjects (123) which are being posed in a centered form with good lighting. On the other hand, Ohio and FER-2013 datasets more accurately reflect real-time conditions due to its automatically captured, non-posed photos. Drivers would sit in the fixed seat, but for the real-time use inside the vehicle, the algorithm should be improved with more natural datasets. Despite the successful results, there are limitations in the implemented algorithms. Running FaceNet using Transfer learning required considerable computational power. The Capsule Network provided better performance on a larger dataset, but the data augmentation was required for the smaller datasets. The algorithms were still not tested with real human participants and in the actual vehicle seat. In the next iteration, we would like to combine the affect detection system with the driving simulator and study the feasibility of the real-time affect detection, similar to the work done in [8].

REFERENCES

[1] M. Jeon, B. N. Walker, and J.-B. Yim, “Effects of specific emotions on subjective judgment, driving performance, and perceived workload,” Transportation research part F: traffic psychology and behaviour, vol. 24, pp. 197–209, 2014.
[2] M. Jeon, “Don’t cry while you’re driving: sad driving is as bad as angry driving,” International Journal of Human–Computer Interaction, vol. 32, no. 10, pp. 777–790, 2016.
[3] E. Vasey, S. Ko, and M. Jeon, “In-vehicle affect detection system: Identification of emotional arousal by monitoring the driver and driving style,” in Adjunct Proceedings of the 10th International Conference on Automotive User Interfaces and Interactive Vehicular Applications, 2018, pp. 243–247.
[4] H. Sanghavi, “Measuring the influence of anger on takeover performance in semi-automated vehicles,” 2020, unpublished Master’s Thesis.
[5] A. Mollahosseini, D. Chan, and M. H. Mahoor, “Going deeper in facial expression recognition using deep neural networks,” in 2016 IEEE Winter conference on applications of computer vision (WACV). IEEE, 2016, pp. 1–10.
[6] S. Minace and A. Abdolrashidi, “Deep-emotion: Facial expression recognition using attentional convolutional network,” arXiv preprint arXiv:1902.01019, 2019.
[7] C. Pramerdorfer and M. Kampel, “Facial expression recognition using convolutional neural networks: state of the art,” arXiv preprint arXiv:1612.02903, 2016.
[8] J. Zhang and N. Xiao, Capsule Network-Based Facial Expression Recognition Method for a Humanoid Robot. Springer, 2020, ch. 16, pp. 113–122.
[9] F. Ghaffar, “Facial emotions recognition using convolutional neural net,” arXiv preprint arXiv:2001.01456, 2020.
[10] P. Khorrami, T. Paine, and T. Huang, “Do deep neural networks learn facial action units when doing expression recognition?” in Proceedings of the IEEE International Conference on Computer Vision Workshops, 2015, pp. 19–27.
[11] M. Quinn, G. SIVESIND, and G. REIS, “Real-time emotion recognition from facial expressions,” 2015.
[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg et al., “Scikit-learn: Machine learning in python,” the Journal of machine Learning research, vol. 12, pp. 2825–2830, 2011.
[13] T. Kaur and T. K. Gandhi, “Deep convolutional neural networks with transfer learning for automated brain image classification,” Machine Vision and Applications, vol. 31, pp. 1–16, 2020.
[14] Z. Zheng, X. Li, J. Barnes, C.-H. Park, and M. Jeon, “Facial expression recognition for children: Can existing methods tuned for adults be adopted for children?” in International Conference on Human-Computer Interaction. Springer, 2019, pp. 201–211.
[15] P. Ekman, “An argument for basic emotions,” Cognition & emotion, vol. 6, no. 3–4, pp. 169–200, 1992.
[16] “Challenges in representation learning: Facial expression recognition challenge,” https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data/
[17] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, “The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression,” in 2010 IEEE computer society conference on computer vision and pattern recognition-workshops. IEEE, 2010, pp. 94–101.
[18] S. Du, Y. Tao, and A. M. Martinez, “Compound facial expressions of emotion,” Proceedings of the National Academy of Sciences, vol. 111, no. 15, pp. E1454–E1462, 2014. [Online]. Available: https://www.pnas.org/content/111/15/E1454
[19] ——, “Compound facial expressions of emotion,” Proceedings of the National Academy of Sciences, vol. 111, no. 15, pp. E1454–E1462, 2014.
[20] S. Raschka, “When does deep learning work better than svm or random forests?” 2016, https://www.kdnuggets.com/2016/04/deep-learning-vs-svm-random-forest.html.
[21] R. (https://stats.stackexchange.com/users/41749/rockthestar), “What are the advantages of relu over sigmoid function in deep neural networks?” Cross Validated, uRL:https://stats.stackexchange.com/q/126238 (version: 2019-10-24). [Online]. Available: https://stats.stackexchange.com/q/126238
[22] T. George Karimpanal and R. Bouffanais, “Self-organizing maps for storage and transfer of knowledge in reinforcement learning,” Adaptive Behavior, vol. 27, no. 2, pp. 111–126, 2019.
[23] M. Chobanyan, “Training an emotion detector with transfer learning,” https://towardsdatascience.com/training-an-emotion-detector-with-transfer-learning-91dea84adeed.
[24] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic routing between capsules,” in Advances in neural information processing systems, 2017, pp. 3856–3866.
[25] M. Pechyonkin, “Understanding hinton’s capsule networks. part 1. intuition,” https://pechyonkin.me/capsules-1/.