Exploring CNN Hybrid Models for Facial Expression Recognition

Team: 20/20 computervision

Members: Yasmine Hemmati, Nizar Islah, Runqi Bi

1. Abstract

In this project, we implemented a convolutional neural network (CNN) that can recognize facial expressions, and then use Dlib's face detection to identify all faces in the image, and use the trained CNN model to output each person's facial expression in the image. We used the grayscale images in Kaggle's FER2013 to train the CNN model, and used the GPU in Kaggle to speed up the training process. In addition to using the original data set of FER2013, we also tried to use Face Detection to intercept every face in the data set and save it as a new data set, so as to remove the parts of the images that are not related to facial expressions and improve the accuracy of the model. We also implemented a combined ORB-CNN model using two inputs: the face-bounded images and a bag-of-features (BoF) from the ORB descriptors of the face-bounded images. The BoF was obtained by performing K-means clustering (K=100) on the ORB descriptors of the entire training set. ORB feature vectors represent the frequency of descriptors in each cluster. ORB-CNN performance was compared to the CNN-only model, achieving similar performance. Furthermore, we tested several data augmentation techniques and observed that random contrast resulted in an ~1.5% improvement compared to no augmentation. We investigated using classifiers such as Random Forest and SVM to classify the images instead of the softmax layer and using our trained CNN model for feature extraction only. We found that we can improve our accuracy by using a hybrid CNN-RF model and achieved a ~1% higher accuracy.

2. Introduction & Literature Review

2.1 Introduction

Facial Expression Recognition is a challenging and interesting problem in computer vision that has many applications such as building robots that can understand human emotion[8]. Facial Expression is defined to be the facial changes in response to a person's internal emotional state, intentions, or social communication[17]. Facial Expression Recognition consists of 3 main steps which are face detection, feature extraction and classification. Convolutional Neural Networks is a popular method used for facial expression recognition due to their success in learning features with a high level of abstraction[17]. In this report we explore different Convolutional Neural Network Hybrid models that incorporate ORB and SVM, Random Forest and XGBoost classifiers. In order to make the model applicable to any picture, we also implemented face detection. In addition, we used face detection to filter the original data set in order to improve the performance of the model.

2.2 Literature Review

The detection method of HOG plus SVM has been widely used in various places, such as face detection, face recognition, and pedestrian detection. Regarding the principle of detection based on HOG and SVM, we refer to Navneet Dalal and Bill Triggs's paper on pedestrian detection: "Histograms of Oriented Gradients for Human Detection"[4]. The

detection principle is similar to that of face detection, but the training data is different. The detection method based on HOG features first trains a large number of local SVM classifiers based on the HOG features, and then selects the better classification effect from these local classifiers. An overview of feature extraction and object detection chain is shown in Figure 1. The use of HOG local features combined with local classifiers has strong robustness to expressions and local uneven illumination. We used Dlib's face detection in this project, which is a detection method based on HOG and SVM.

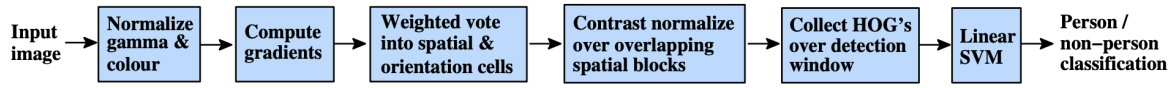

Figure 1. An overview of feature extraction and object detection chain mentioned in the article Histograms of Oriented Gradients for Human Detection(2005)[4]

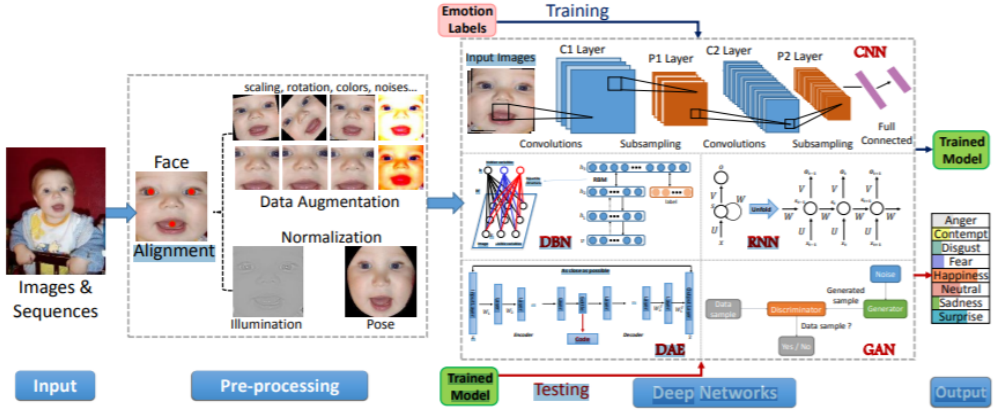

Figure 2. Standard pipeline for FER with deep learning (figure obtained from Li et al, "Deep Facial Expression Recognition: A Survey", 2020).

The traditional approach to FER consists of using handcrafted features as feature vectors extracted from raw images and implementing a shallow machine learning classifier such as an SVM. With facial expressions, it can be difficult to have a model that generalizes very well, due to large variation in face orientation as well as illumination/contrast, in addition to age, ethnicity, and gender differences. With the increased computational power with GPUs and larger datasets, deep learning methods have become the state-of-the-art for facial expression recognition, outperforming the traditional methods. To avoid overfitting, CNNs require large training sets, which is a problem for many of the available datasets for FER. Many modifications of the best CNN model architectures have been proposed for the specific problem of FER. Figure 2 shows the standard pipeline for end-to-end FER with CNNs, starting from unprocessed images to the final output, predicted probability of each facial expression class. The pre-processing block is important in this particular problem, and can drastically improve the success of deep networks for FER. Methods such as face alignment, normalization of illumination and pose, and data augmentation are common techniques used to pre-process the images. Furthermore, some recent papers have used pre-trained networks with other data sources to enhance model performance, to avoid the problem of overfitting on small datasets [20]. The winner of the FER-2013 challenge used a

CNN followed by a linear one-vs-rest SVM [8]. This model minimized margin-based loss instead of cross-entropy loss and trained all the layers of the deep networks by backpropagation gradients though the SVM classifier at the top of the network [13].

3. Methodology, Experiments and Results

3.1 Methodology and Experiments

3.1.1 Preprocessing: Face Detection

In the process of face detection, we found that not all faces in the data set can be recognized. For example, those faces are severely masked. Among the pictures in FER2013 we used, many of them are not even human facial expressions. The defect of this data will lead to the low accuracy of the model we trained, so we thought of using face detection to intercept the recognized faces, and then resize the intercepted images to 48x48 and store them as a new data set, and then use the new data set to train CNN, we can expect a certain improvement on accuracy.

Thus, we used Dlib's face detection to traverse all the data in FER2013, including 28709 training set data and 7178 test set data, intercepted the detected faces and stored them as a new data set, so it can be said that the new data set is a subset of the original data set. The new data set contains 19,993 training sets and 4,965 test sets. Then we compared the accuracy and loss of the model trained with FER2013 and the newly generated data set and found that the accuracy of the new model rose slightly faster, and the accuracy rate was slightly higher, and the converge was faster.

3.1.2 Preprocessing: Data Augmentation

Data augmentation techniques can be used to improve the performance of a convolutional neural network, by transforming the images with some randomness to allow the model to learn robust features from the dataset. We implemented several augmentation techniques, including random rotation, zoom, and contrast. At each epoch, a random rotation, zoom, or contrast change is applied to each batch of images. Applying a random rotation allows the model to learn features that are rotation-invariant, which is important for identifying key points and regions. A random zoom results in learning features that are scale-invariant. The contrast function applies an increase or decrease in contrast of an image based on the mean contrast and the value specified to the function. The contrast augmentation improves the ability of the model to learn contrast-invariant features, which was especially important for this dataset because the images seemed to vary greatly in terms of illumination and contrast. In addition to these preprocessing techniques, we also implemented rescaling from 0-255 to 0-1.

3.1.3 CNN Only Model

Convolutional Neural Networks (CNN) is a feed-forward artificial neural network inspired by the visual cortex [5]. A CNN is composed of multiple components such as convolution layers, pooling layers and fully connected layers and uses the backpropagation algorithm for training. The convolution layers use filters that perform convolution operations to extract features of interest from the input and output a feature map . The pooling layers usually come after the convolutional layers and are responsible for downsampling and reducing the

dimension of the feature map [5, 6]. The fully connected layers compile the data from the previous layers to form the final output [7]. The classification layer is the last layer and it collects the final features and returns a column vector where each row points towards a class [5].

We initially wanted to base the parameters of our CNN on that in the paper “Facial Expression Recognition Using a Hybrid CNN–SIFT Aggregator” By Mundher Al-Shabi, Wooi Ping Cheah and Tee Connie. In the paper the number of convolution filters are 64, 128, and 256, respectively. Moreover they used categorical cross entropy as their cost function and Adam's optimizer. In addition, Leaky Relu is used as the activation function instead of relu to solve the “dying relu” problem. Leaky Relu provides a small negative slope when x<0 instead of setting y=0 when x<0. This makes the derivatives nonzero which makes the CNN learn faster than relu [8]. The network is trained for 300 epochs with a batch size of 128 [8]. Training the model with 300 epochs would’ve taken a very long time so we cut the batch size in half to 64 and trained the network for 50 epochs using the parameters and architecture layed out in the paper. We cut the batch size in half since smaller batch sizes converge faster even though they may not be able to achieve an optimum minimum that a larger batch size can reach [9]. This gave us a validation accuracy of 65.21% Next we tried increasing the number of convolution filters to 128, 256 and 512 respectively and retrained the model and this gave us a validation accuracy of 68.0%. Then we tried adding batch normalization normalization in between the layer since batch normalization accelerates training and provides some regularization alongside normalizing each batch [10]. This on it’s own did not improve our accuracy. Then we added L2 regularization to the first fully connected layer with a value of 0.01 to regularize the weights of the fully connected layer [8]. Since we added l2 regularization to the first fully connected layer we reduced the dropout parameter to 0.2 from 0.4 to prevent possibly adding too much regularization especially since smaller batch sizes have a regularizing effect due to their high variance [9]. We retrained the model with the first dense layer to consist of 512, 1024, 2048 and 4096 neurons. We achieved the highest accuracy of 68.75% on the original FER-2013 dataset when the first dense layer consisted of 512 neurons.

3.1.4 CNN-SVM/RF/XGBoost Hybrid Model

CNN’s both extract features and classify the images so we wanted to see if we can improve our accuracy by using CNN to extract features and another classifier such as the Random Forest Classifier or SVM to classify the images.

In order to extract the features of an image from a CNN we need to get the output of the layer of the CNN before the classification layer. Hence, we need a model that gives us the output of the fully connected layer prior to the classification layer. We created a model that has the same input as that of the CNN model but outputs the fully connected layer prior to the classification layer. So our model that gives us features is Model(input = model.input, output = model.layers[-3].output) [14] where model is the trained CNN model. Note model.layers[-3] is the fully connected layer prior to the classification layer for our CNN. As mentioned previously this layer compiles the data from the previous layers to form the final output and the previous layers extract features. Then when we use the predict method on this model we get the predicted features vector of the image not the predicted class.

To get the classes corresponding to our training and validation data we originally used train_data.classes where train_data = train_data_generator.flow_from_directory(…) and train_data_generator = ImageDataGenerator(…). Then when we trained the various Random Forest and SVM classifiers the accuracy was 22-25% which was very low. Then when we used model.predict_class(test_data) and found the accuracy compared to test_data.classes was only around 16%. This was odd since the model had a validation accuracy of 68.75% so the accuracy we were supposed to get was supposed to be 68.75%. Hence it seemed that the classes method was not giving us the correct classes. Therefore, we found the csv version of our dataset on Kaggle which had a column indicating the class of all the images in the data set so we did not need to use the classes method and redid all the steps using the csv data set. Using the csv version of the data set for this model fixed the problem which is why the results of this hybrid model is generated using the csv version of the dataset.

To classify the feature vectors we experimented with the Random Forest classifier, SVM classifier and XGBoost classifier. SVM's goal is to find the optimal hyperplane to separate two classes where every member of a class is represented as a point in an n dimensional space. Random Forest is an ensemble learning method that makes a large number of decision trees and outputs the mode of the classes of the individual trees[11]. XGBoost (Extreme Gradient Boosting) is a boosting algorithm based on gradient boosted decision trees algorithm but applies better regularization techniques to prevent overfitting and this is the difference between XGBoost and gradient boosting [12].

To use SVM for a multi classification problem we set the decision_function_shape ='ovo' where 'ovo' stands for one-vs-one instead of the default 'ovr' which stands for one vs rest since SVM is normally used for binary classification. One-vs-One considers each binary pair of classes instead of considering one class vs all the rest like 'one vs rest'[18]. We fit an SVM classifier with both linear and rbf (radial basis function) kernels. When we used the decision_function_shape ='ovr', the validation accuracy for both the SVM with the linear kernel and SVM with an rbf kernel was lower than the CNN only validation accuracy. Hence our results only include the SVM classifiers where decision_function_shape ='ovo'. We also fit different random forest classifiers with different parameters. Moreover we used the same n_estimators parameter for the XGBoost classifier as the Random Forest classifier that performed the best since they are both ensemble learning methods. The n_estimators parameter is the number of trees in the forest.

3.1.5 ORB-CNN Hybrid Model

ORB ("Oriented FAST and Rotated BRIEF") is a feature extraction method that is a computationally efficient alternative to SIFT [2]. ORB works by first detecting keypoints with FAST, then using Harris corner detection to select the top N keypoints. It also computes an intensity weighted centroid of every patch with a corner at the center. The vector from the centroid to the corner provides a direction (hence "oriented FAST"). Since FAST does not provide multi-scale features, ORB calculates FAST at multiple scales using a scale pyramid. For descriptors, ORB uses a modification of the BRIEF descriptors. First, it calculates the BRIEF keypoints and applies a rotation matrix R using the orientations computed from the keypoints. In order to maintain the property of BRIEF (features have high variance and a

mean of around 0.5), a subset of all the features are chosen by a greedy search which looks for features with mean close to 0.5 and low correlation. The high variation is a desirable quality because a feature will have different responses based on the input. Together, the oFAST and rBRIEF methods make up the ORB feature.

Combining deep and shallow features into one model has been explored in different problems in the past, with one notable example being a SIFT-CNN hybrid model for facial expression recognition [3]. Given that we had not encountered any paper which uses the ORB feature for facial expression recognition in a hybrid model, we decided to implement an ORB-CNN model. ORB features can be combined into a CNN model through the following steps. First, we applied face detection on the raw images as mentioned in the previous section. Then, we used the orb function from the cv2 library to compute keypoints and descriptors for every image. Since every ORB descriptor has 32 features and each descriptor has different features, we used a K-means clustering algorithm (from scikit-learn library) to group descriptors into one of K clusters. This is known as a bag-of-visual-words or bag-of-features algorithm. The bag-of-features was then used to obtain a histogram for every image corresponding to the frequency of occurrences of each visual word in the image, resulting in a K-dimensional feature vector per image. K is a hyperparameter which can be tuned with cross-validation, but due to time constraints we used constant value of 100. The paper used K=300, but they resized the images (70x70 from 48x48) Next, we passed the 100-dimensional feature vectors into a fully-connected (FC) layer, followed by a dropout layer. The output of this layer was concatenated with the output of an intermediate FC layer of the CNN model. The merged vector was then passed into an FC layer, which in turn was inputted to a dropout layer and the final FC layer with a softmax activation function for the class probability prediction. The exact model architecture is provided in the following figure.

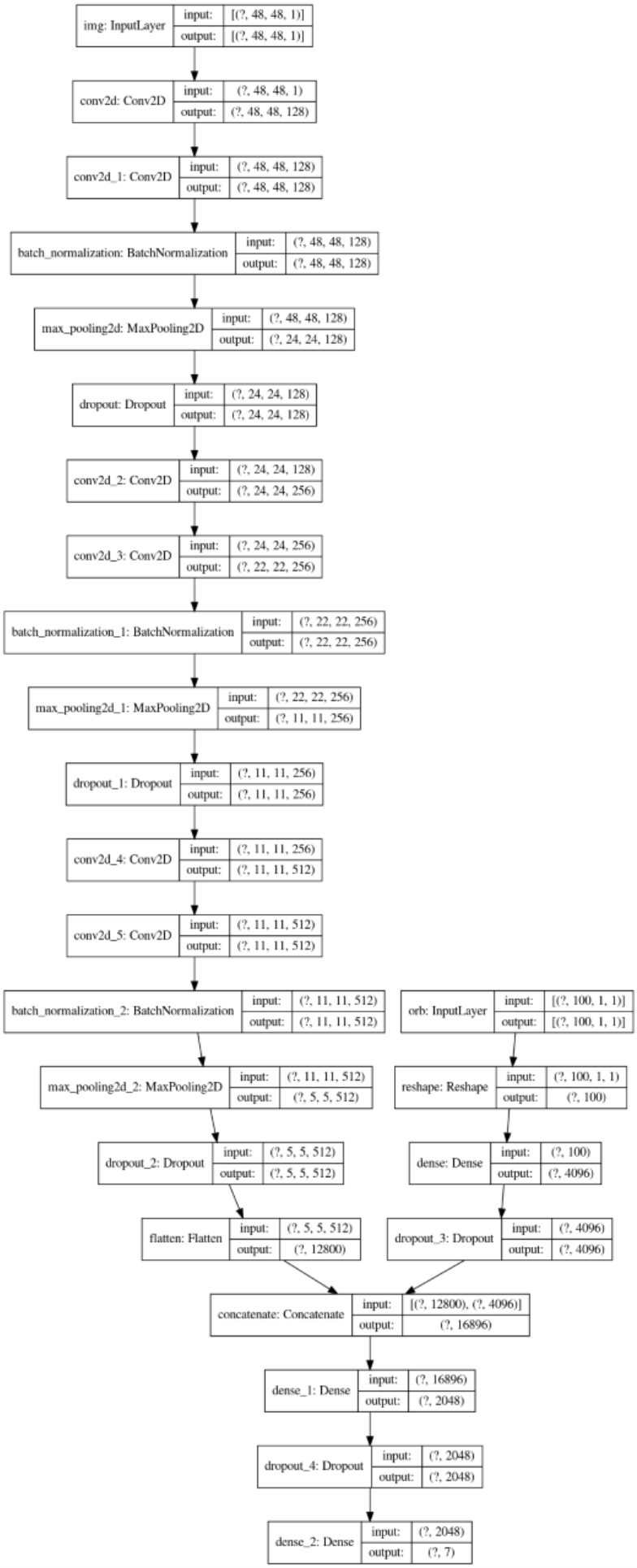

Figure 3. ORB-CNN Architecture

3.1.6 Face Detection To Output The Facial Expression

We hope that the model we trained can not only be used to detect images in FER2013 or similar images, for example, the entire photo has only one face, and the size is a grayscale image of 48x48. But we hope it can be used more widely in real life. In the real-life, the photos are commonly colorful, with various complex backgrounds, and even multiple faces in the photos. In this case, we need to use face recognition to find all the faces in the photos and use our pre-trained CNN model to predict the person's expression.

In this project, we first used Dlib's get_frontal_face_detector function to recognize faces, then we cut out the faces, and resize them to a size of 48x48. Finally, we use the pre-trained CNN model to predict the expressions of people in the photo and print these predicted expressions on their faces as outputs.

Dlib's get_frontal_face_detector function uses SVM plus HOG for face detection. HOG is called Histogram of Oriented Gradient. HOG feature is a feature descriptor used for object detection in computer vision and image processing. It constructs features by calculating and counting the histogram of the gradient direction of the local area of the image. The HOG feature extraction method is: 1) Grayscale the image; 2) Normalize the color space of the input image; the purpose is to reduce the impact of local shadows and lighting changes in the image, while suppressing noise interference; 3) Calculate the gradient of each pixel of the image, including size and direction; 4) Divide the image into cells; 5) Count the gradient histogram of each cell; 6) Combine every few cells into a block, and concatenate the feature descriptors of all cells in a block to obtain the HOG feature descriptor of the block. 7) Connect the HOG feature descriptors of all blocks in the image to get the HOG feature descriptor of the image[1]. This is the final feature vector available for classification. Finally, The SVM model is trained using the HOG vectors for multiple faces.

We tried many different pictures with Dlib's face detection, we found that Dlib's face detection is very stable for frontal and slightly non-frontal faces, most faces can be successfully detected. In addition, slight facial occlusion will not affect the detection of human faces, as shown in Figure 7. Except for the face in the lower right corner, the nose and mouth are completely blocked and therefore cannot be detected, the other faces wearing masks can be detected. But considering that if the face is largely obscured, the facial expression itself is difficult to judge, and even the human eye cannot judge it with certainty, so this restriction is actually irrelevant. Otherwise, we can see the bounding box often excludes part of forehead and even part of chin sometimes. But the forehead and chin are very decisive in judging expressions, so this is actually okay. But this method has a limitation, that is, it does not work for side face and extreme non-frontal faces, like looking down or up, for example, the person in the upper left corner of Figure 7.

3.2 Results

3.2.1 Face Detection

We use Dlib's face detection to cut out the detected faces and store them as a new database, using the same CNN model to train a new data set, and train FER2013 data When comparing the accuracy rate obtained with loss, it can be found that using the data set filtered by face detection, within 20 epochs, the learning rate is 0.0001 and the decay is 1e-6, the accuracy rate is from 64.43% It rose to 65.60%. Figure 4 is the model accuracy

and model loss trained 20 times using the FER2013 data set, and Figure 5 is the model accuracy and model loss trained 20 times using the new data set. Through the comparison of the two pictures, we can see that with the new modem, the accuracy and loss converge faster and the accuracy is higher and the loss is lower.

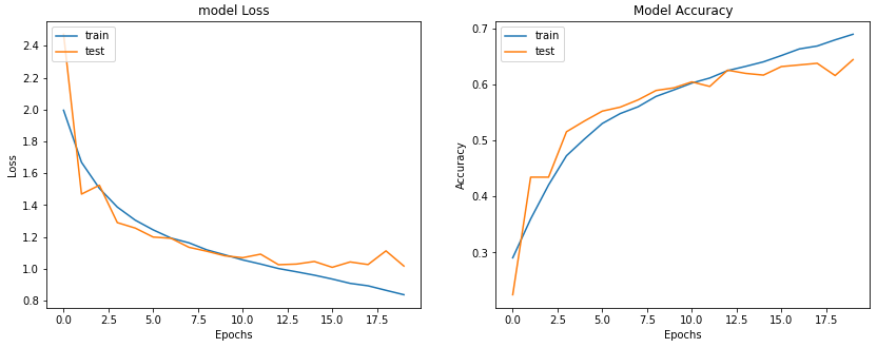

Figure 4

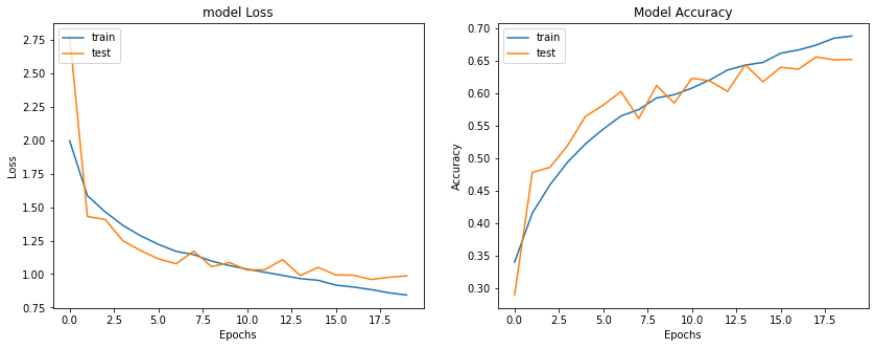

Figure 5

We use the pre-trained CNN model to test the following four pictures Figure 6, Figure 7, Figure 8 and Figure 9. We can see that most of the faces in the picture are drawn with square circles. This square is actually the return value of Dlib's face detection equation and represents the detected face. The expressions predicted by the trained CNN model were also printed on the faces of these people. And through observation, it can be seen that the facial expression recognition printed on the face makes sense.

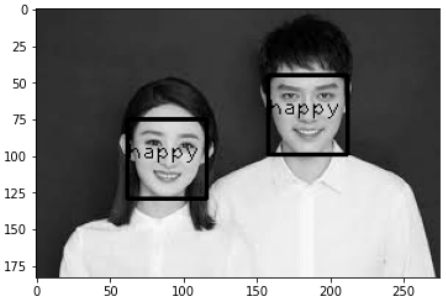

Figure 6

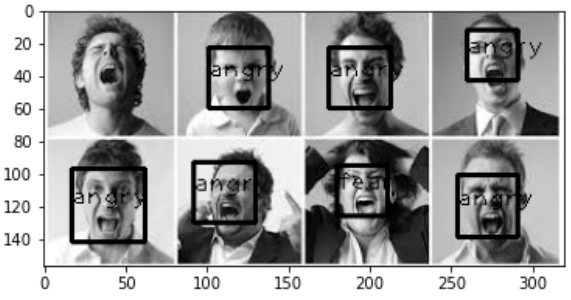

Figure 7

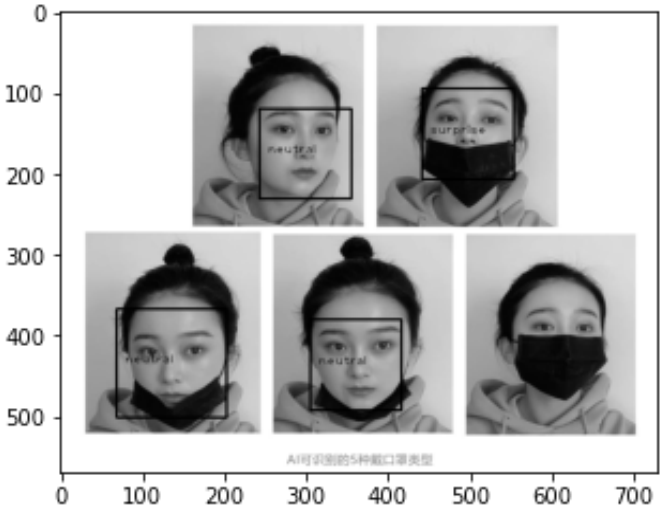

Figure 8

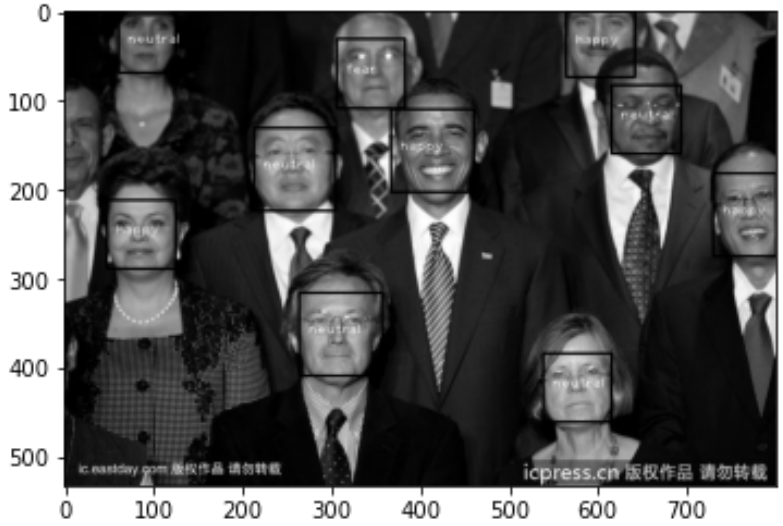

Figure 9

3.2.2 CNN Only

The final CNN architecture that achieved a 68.75% validation accuracy on the FER-2013 dataset consists of six convolution layers, three Max-Pooling layers, followed by two dense fully connected layers. Each time Max-Pooling is added, the number of the next convolution filters doubles [8].The number of convolution filters are 129, 256, and 512, respectively. The window size of the filters is 3x3. Max pooling layers with a stride of size 2x2 is placed after every two convolutional layers [8]. The dense layer consists of 512 neurons linked as a fully connected layer. Each Max-pooling and dense layer is followed by a dropout layer [8] and a batch normalization layer is added after each convolutional layer before the max pooling layer. Moreover, l2 regularization was added to the dense layer.

Below are two confusion matrices to visualize the performance of the CNN-only model. The entries of the confusion matrix of figure 10 are the number of classifications and the entries

of the confusion matrix of figure 11 is the decimal percentage for each classification.

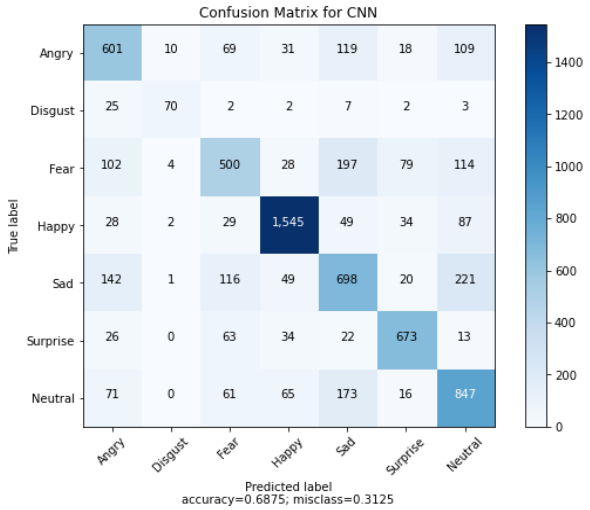

Figure 10

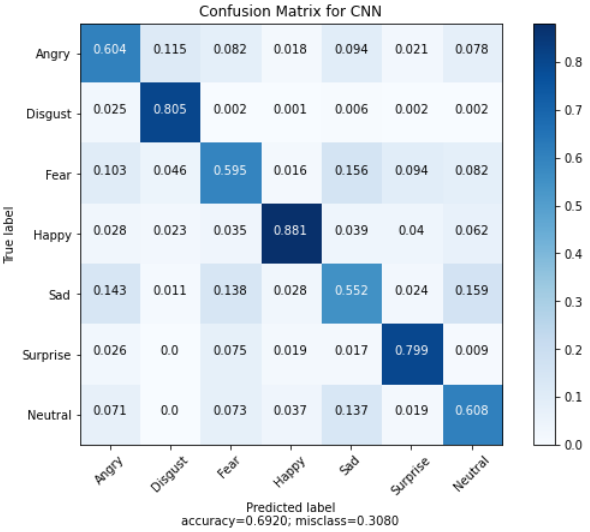

Figure 11

In the confusion matrix we see that the emotions happy, disgust and surprise were classified the most accurately. This seems consistent with how accurately humans tend to recognize these emotions. Humans recognize happiness and surprise after very short exposure times (10–20 ms), even at low resolution [15]. In addition humans recognize fear and anger the slowest (100–250 ms), even in high-resolution images[15]. Therefore our confusion matrix does seem to correspond with how humans recognize emotions since fear and anger have a higher misclassification percentage. Moreover in the confusion matrix we see that sadness is the most misclassified emotion. If sadness is misclassified, it is mostly misclassified as anger and this makes sense since people tend to confuse these emotions.

3.2.3 CNN-SVM/RF/XGBoost Hybrid Model

The validation accuracy of each model is displayed in the table below.

Model	**Validation Accuracy**
RandomForestClassifier(random_state=42, n_estimators=300)	69.68%
RandomForestClassifier(random_state=42, n_estimators=700)	69.70%
RandomForestClassifier(random_state=42, n_estimators=1200)	69.37%
RandomForestClassifier( )	69.61%
svm.SVC(kernel=rbf, decision_function_shape=ovo)	69.12%
svm.SVC(kernel=linear ,decision_function_shape=ovo)	67.42%
XGBClassifier(n_estimators = 700, learning_rate=0.01)	69.11%

We see every hybrid model outperforms the CNN only model which achieves a 68.75% validation accuracy except the svm classifier with a linear kernel. The highest validation accuracy is achieved with the RandomForestClassifier(random_state=42,n_estimators=700). This model achieves an accuracy of 69.70% on the validation set hence has a higher validation accuracy than the CNN only model by about 1%.

3.2.4 ORB-CNN Hybrid

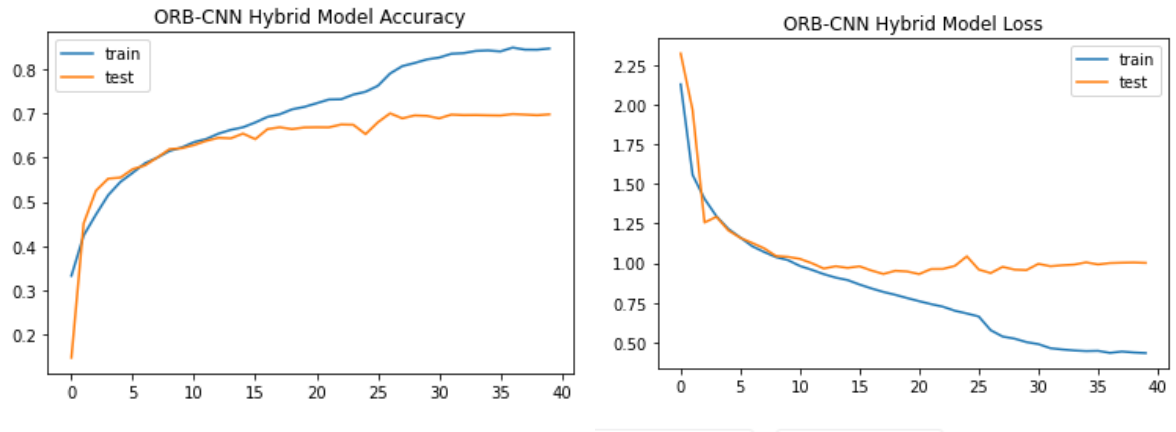

Figure 12

The highest accuracy with the ORB-CNN hybrid model on the test set after 40 epochs was 69.97%. Our final CNN model in comparison achieved 69.99%, essentially resulting in a similar performance with the inclusion of ORB features in the model. This result disagrees with the results found in the CNN-SIFT paper, which reported an improvement over the CNN-only model. This was an unexpected result, as the combination of deep and shallow features allows the model to perform better with small datasets such as the one we used, since CNNs often require larger training sets, whereas ORB works well with a small number of images. It is possible that ORB-CNN could perform better than CNN-only if we did hyperparameter tuning for ORB (number of clusters K, or threshold for edges). Moreover, CNN-only may be better than hybrid models for this dataset, especially with the right model architecture. Our reasoning is that the state-of-the-art models currently outperform hybrid models in the FER-2013 dataset. It would be interesting to see the performance with an even smaller dataset. One advantage, however, of using ORB over SIFT is the computational efficiency, as the original ORB paper stated that it is over 2 magnitudes faster than SIFT. Although we cannot directly compare our result with the CNN-SIFT paper (since we used a different dataset), we generated a table comparing our performance on FER-2013 against the state-of-the-art models.

Table 3.2.4 Comparison of our best model and state of the art on the FER-2013 dataset (our model in bold):

Model	Accuracy (%), rounded to 3 sig. figures
ORB-CNN	**70.0**
CNN Ensemble [16]	75.2
VGG [16]	72.7
Res-Net [16]	72.4

3.2.5 Data Augmentation

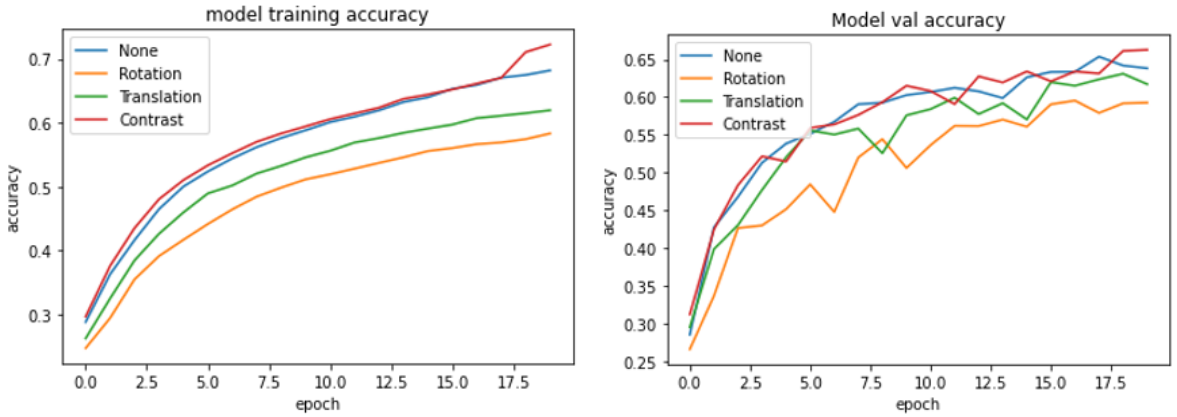

Figure 13. Effect of data augmentation on model performance.

The data augmentation results are shown in Figure 13 above. The random contrast (in red) showed the largest improvement compared to the baseline (no data augmentation), whereas the random translation and rotation performed worse than the baseline. One possible explanation for this is that the model was sensitive to illumination and contrast differences in the images, and with the addition of random contrast, it provided more training examples under different contrast to learn robust features that are contrast-invariant, instead of overfitting on the training set. Furthermore, we did not extensively test the parameters that decide the range of randomness (+/- 10% was what we used). In the future, it would be more informative to test several increments for each augmentation method and compare them.

4. Conclusion

We have developed various CNNs to recognize facial expressions, and compared their performance using data visualization. We also used face detection with CNN to realize facial expression recognition for any number of faces in any images outside the FER2013 data set, and use face detection to remove parts of the images in the data set that are not related to facial expressions, like forehead and parts of the chin. You can see that the performance of the model has been improved. However, although face detection can recognize frontal and

slightly non-frontal faces or slight shelter faces, it's limitations are that it can't recognize side faces and extreme non-frontal faces poorly.

We implemented a hybrid ORB-CNN model that performed similarly to our final CNN-only model. The best CNN architecture may still be better than a hybrid model due to its ability to extract higher level features, however with smaller datasets, such as the one we used, the performance should be similar or higher for hybrid models. We believe further testing with hyperparameter tuning would result in a higher performance for the ORB-CNN model. Moreover, we implemented hybrid CNN-SVM/RF/XGBoost models and found that most of these hybrid CNN models produce more accurate results than the CNN only models. In addition, data augmentation is a very useful technique that we implemented, and further experiments should use random contrast as an augmentation method to improve the performance of the final model. These results may be useful to guide an investigation in other problems in computer vision, as CNNs benefit from augmentation regardless of the content of the image. In addition, hybrid models should be considered when working with very small datasets.

5. Authors' Contributions

Runqi Bi:
I first imported the FER2013 dataset and wrote a CNN model for facial expression estimation using Convolutional Layer, Pooling Layer, Fully-Connected Layer and Drop-Out Layer with an original validation accuracy of about 64.43% with 20 epoches. Then, I used Dlib's face detection to perform face detection on the entire FER2013 dataset and store the detected results in a new folder and use CNN to train the model with the new data set and compare the training results including model accuracy and model loss with the training result of the original FER2013 dataset with exactly the same model. I also used Dlib's face detection function which is based on HOG and SVM to realize the input of any photo, no matter how many people are on the photo and what the background is, it can recognize the face, and use the CNN pre-trained model to predict the person's expression, and print the expression of the person on the face as output.

Yasmine Hemmati:
I chose and tuned the parameters and the layers for the final CNN architecture which was originally inspired by the paper "Facial Expression Recognition Using a Hybrid CNN– SIFT Aggregator " by Mundher Al-Shabi, Wooi Ping Cheah and Tee Connie . I re-trained the model using a variety of different parameters and regularization methods to find the model that achieves the highest validation accuracy. The highest validation accuracy I was able to achieve was 68.75%. Moreover the paper "Deep Learning using Support Vector Machines" by Yichuan Tang shows that using linear SVM as the top layer rather than softmax improves results for their CNN model. Therefore I wanted to try using the CNN model only for feature extraction and to see if different classification algorithms, in addition to SVM, would also improve our results. I decided to experiment using the SVM, Random Forest and XGBoost Classifiers. When comparing the validation accuracy of the CNN/Random-forest classifier model with the CNN only model we see that the CNN/Random forest classifier model surpasses the results of the CNN only model. In addition, the CNN/Random Forest classifier with n_estimaters= 700 surpasses the CNN-SVM and CNN/XGBoost models we have trained for our project. Moreover using a random forest classifier rather than softmax

achieves better results than the CNN only model. In addition, I wrote the sections of the report corresponding to my contributions mentioned above.

Nizar Islah:
I incorporated several preprocessing steps for raw images before input to the CNN model, including random rotation, random zoom, and random contrast. Then, I tested the performance of the model in terms of accuracy and loss on the training and test set with each preprocessing method, with random contrast achieving the greatest performance improvement. Next, inspired by a recent paper on a hybrid model combining deep and shallow features (CNN-SIFT), I decided to implement a CNN-ORB hybrid model, especially since the authors had not attempted this. I used the ORB function from OpenCV for feature extraction. I also implemented code to obtain feature vectors from ORB descriptors using the bag-of-features approach with K-means clustering (used scikit-learn for K-means). In addition, I wrote the code for the combined CNN-ORB model. I compared my hybrid model to the CNN-SIFT model from the paper and our CNN model. It achieved better test accuracy on the FER 2013 dataset than our original CNN model, and comparable accuracy to our final CNN model. Lastly, I wrote the sections in the report corresponding to my contributions stated above.

6. References

[1] Mallick, S. (2016, December 06). Histogram of Oriented Gradients. Retrieved from https://www.learnopencv.com/histogram-of-oriented-gradients/

[2] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011, November). ORB: An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision* (pp. 2564-2571). IEEE.

[3] Sun, X., Lv, M. (2019). Facial Expression Recognition Based on a Hybrid Model Combining Deep and Shallow Features. *Cogn Comput* 11, 587–597. https://doi.org/10.1007/s12559-019-09654-y.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, CA, USA, 2005, pp. 886-893 vol. 1, doi: 10.1109/CVPR.2005.177.

[5] Liu, Y. H. (2018). Feature Extraction and Image Recognition with Convolutional Neural Networks. https://iopscience.iop.org/article/10.1088/1742-6596/1087/6/062032/pdf

[6] Convolutional Neural Networks cheatsheet Star. (n.d.). https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks

[7] Singh, S. (2019, March 02). Fully Connected Layer: The brute force layer of a Machine Learning model. from https://iq.opengenus.org/fully-connected-layer/

[8] Al-Shabi, M., Cheah,, W. P., & Connie, T. (n.d.). Facial Expression Recognition Using a Hybrid CNN– SIFT Aggregator. https://arxiv.org/pdf/1608.02833.pdf

[9] Kandel, I., & Castelli, M. (2020, May 05). The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. https://www.sciencedirect.com/science/article/pii/S2405959519303455

[10] Brownlee, J. (2019, December 03). A Gentle Introduction to Batch Normalization for Deep Neural Networks. https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/

[11] Random forest. (2020, December 15). https://en.wikipedia.org/wiki/Random_forest

[12] DataTechNotes. (2019, July 04). Classification Example with XGBClassifier in Python. https://www.datatechnotes.com/2019/07/classification-example-with.html

[13 Tang, Y. (2013) Deep Learning using Support Vector Machines.]http://www.cs.toronto.edu/~tang/papers/dlsvm.pdf

[14] Brownlee, J. (2020, August 18). Transfer Learning in Keras with Computer Vision Models. https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/

[15] Du, S., & Martinez, A. (2013, March 01). Wait, are you sad or angry? Large exposure time differences required for the categorization of facial expressions of emotion. https://jov.arvojournals.org/article.aspx?articleid=2121397

[16] Pramerdorfer, C., & Kampel, M. (2016). Facial expression recognition using convolutional neural networks: state of the art. *arXiv preprint arXiv:1612.02903*.

[17] Liu, Y. H. (2018). Feature Extraction and Image Recognition with Convolutional Neural Networks. Retrieved December, =, from https://iopscience.iop.org/article/10.1088/1742-6596/1087/6/062032/pdf

[18] Brownlee, J. (2020, September 07). One-vs-Rest and One-vs-One for Multi-Class Classification.https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/

[19] Li, S., & Deng, W. (2020). Deep facial expression recognition: A survey. *IEEE Transactions on Affective Computing*.

[20] Knyazev, B., Shvetsov, R., Efremova, N., & Kuharenko, A. (2017). Convolutional neural networks pretrained on large face recognition datasets for emotion classification from video. *arXiv preprint arXiv:1711.04598*.