Emotion Detection with CNN Ensembles and CNN Autoencoders

Junda An
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
jundaa@andrew.cmu.edu

Yudong Liu
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
yudongl@andrew.cmu.edu

Tiancheng Zheng
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
tzheng2@andrew.cmu.edu

Abstract

Facial expressions contain valuable information about the emotions and thoughts of human beings, and emotion detection has wide applications in the healthcare, entertainment, and robotics industry. Experimenting on the FER-2013 database, we propose two different approaches to perform emotion detection: 1. CNN Ensembles: Using ensembles consisting of different high-performing Convolutional Neural Networks; 2. CNN Autoencoder: Using the encoder and the classifier to train the model and using the decoder to regularize the model. Our Top-13-Model Ensemble is able to achieve a 74.0% accuracy on the testing dataset. Our autoencoder model is able to achieve a 65.9% on the testing dataset.

1 Introduction

Facial expressions contain valuable information about the emotions and thoughts of human beings. Since emotion detection has wide applications in the healthcare, entertainment, and robotics industry, there have been numerous studies conducted on automatic facial expression analysis (Li & Deng, 2018). In the field of computer vision and machine learning, scholars have explored many different facial expression recognition (FER) systems to properly encode expression information from facial representations (Li & Deng, 2018). In particular, six facial expressions are identified to be the standard model for FER: anger, disgust, fear, happiness, sadness, and surprise (Ekman & Friesen, 1971).

FER systems are divided into two main categories, static image FER and dynamic sequence FER, based on different feature representation methods. Static image FER encodes only spatial information from a single image, whereas dynamic sequence FER encodes both spatial and temporal features from contiguous frames in the input sequence (Li & Deng, 2018). In this project, we are only concerned with static image FER. We aim to use machine learning techniques, Convolutional Neural Networks in particular, to detect and classify human emotions given real-world images of the human face.

2 Data

We used facial images from the FER-2013 database, which was introduced during the ICML 2013 Challenges in Representation Learning. It is a large-scale and unconstrained database collected automatically by the Google image search API, and all the images have been registered and resized to 48×48 pixels after rejecting wrongfully labeled frames and adjusting the cropped region (Li & Deng, 2018). The data consists of 35,887 grayscale images of faces, each with a dimension of $3 \times 48 \times 48$ (Goodfellow et al., 2013). There are 28,709 images in the training set, 3,589 images in the validation set, and 3,589 images in the testing set. Each image in the database belongs to one of the seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

3 Related Work

In the past, facial expression recognition was typically done using traditional machine learning methods such as local binary patterns, non-negative matrix factorization and sparse learning (Li & Deng, 2018). Principal component analysis, support vector machine (Ghayoumi & Bansal, 2016), and k-nearest neighbors (Tarnowski et al., 2017) are also among the frequently used methods. However, due to the dramatically improved computational capabilities of modern hardware and the increasingly well-designed neural network architectures, deep learning has taken over this field with various deep learning methods achieving state-of-the-art recognition accuracy in FER. With respect to the FER-2013 database, convolutional neural network (CNN) is often used as part of the state-of-the-art architecture, as demonstrated in various experiments by different scholars (Tang, 2015) (Devries et al., 2014) (Zhang et al., 2015) (Yanan Guo et al., 2016) (Kim et al., 2016) (Pramerdorfer & Kampel, 2016). With proper data-preprocessing and parameter tuning, the highest testing accuracy achieved by CNN models of this type is 75.2% (Pramerdorfer & Kampel, 2016).

4 Methods

We used two different methods to approach this problem: CNN ensemble and CNN autoencoder.

4.1 CNN Ensemble

4.1.1 State-of-the-Art Convolutional Neural Networks

We experimented with several state-of-the-art deep convolutional neural network architectures introduced in recent years that are known to have great performance on image recognition tasks: Residual Neural Networks (ResNet) (He et al., 2015), Densely Connected Convolutional Networks (DenseNet) (Huang et al., 2018), Wide Residual Networks (Wide ResNet) (Zagoruyko & Komodakis, 2017), ResNeXt (Xie et al., 2017), and VGG (Simonyan & Zisserman, 2015).

We used the official PyTorch implementations (Paszke et al., 2019) and downloaded their pretrained models on the ImageNet dataset (Deng et al., 2009) as a starting point. Afterwards, we made slight modifications to the architecture of these models to accommodate the FER-2013 database and performed fine-tuning on the training set. For the fine-tuning step, we used the SGD optimizer with momentum (Robbins & Monro, 1951), the Cross-Entropy Loss criterion, and the ReduceLROnPlateau learning rate scheduler. The initial learning rate is 1e-2, the weight decay is 5e-4, and the momentum is 0.9. The learning rate is reduced by a factor of 0.5 once the validation loss stopped improving. These values are chosen via empirical experiments as they allow the models to converge faster and smoother. We trained for 30 epochs with a batch size of 64.

The training objective is to minimize the cross-entropy loss function over all seven class labels using the stochastic gradient descent algorithm.

In order to counter the issue of overfitting, we used data augmentation on the training data to make our models more generalized. We applied the following transformations to the training data before passing it through the models:

Transformation	Effect
Resize(224)	The SOTA models require the input image to be of size 224×224
RandomRotation(45)	Allow the models to generalize to rotated images
RandomHorizontalFlip()	Allow the models to generalize to mirrored images

Table 1: Data Augmentation Transformations

4.1.2 Ensemble of Convolutional Neural Networks

To further improve the accuracy of our CNN classifiers, we used different ensemble models consisting of a varying number of CNNs. The ensemble model makes its prediction using the majority vote among its classifiers. We also tried to implement weighted votes by assigning larger weights to votes of more accurate classifiers, but it did not improve the model accuracy by much since all the classifiers have similar accuracies.

4.2 CNN Autoencoder

We also applied generative methods for extracting crucial information from images and performing emotion classification. An autoencoder is a specific type of a neural network, which is mainly designed to encode the input into a compressed and meaningful representation, and then decode it back such that the reconstructed input is similar as possible to the original one. (Bank et al., 2020) . Autoencoder could be used for denoising, dimensionality reduction or regularization in training.

Like fully connected multiple-layer perceptron networks, the conventional autoencoders introduce redundancy in parameters and are less efficient in capturing local features in the 2D images. Convolutional networks are known to have an ability to self-learn important features necessary for classification while preserving spatial information. Therefore, the CNN autoencoders, a variant of autoencoders are invented to deal with 2D images. In its encoder part, the linear layers are replaced with convolutional layers and pooling layers. Thus, the image is encoded in multiple local image features, which are flattened to become the latent features of the image. In the decoder part, deconvolutional layers and unpooling layers are used to reconstruct the image from local features. In the following parts, we are discussing two ways that wee use CNN autoencoders to perform the classification.

4.2.1 CNN Autoencoder as a Dimensionality Reduction Technique

The original image with size $48 \times 48 \times 3$ reside in a high dimentional space, which often leads to the curse of dimensionality (Friedman & Fayyad, 1997). The dimensionality reduction can be performed by the bottleneck layer. Therefore, the input is projected to a lower dimensional latent space by the encoder. This latent feature is feed to a classifier to perform the classification task.

Let E, D, and C denote the encoder, the decoder, and the classifier respectively. Let $\{x, y\}$ denote one sample of our training dataset, where x is the image and y is an one-hot vector denoting the class of x. We will get the following equations. $h = E(x)$, $\hat{x} = D(h)$, and $\hat{y} = C(h)$, where h is the latent feature, $\hat{x}$ is the reconstructed image, and $\hat{y}$ is a vector denoting probability of each class.

In the dimensionality reduction step, we use the Frobenius norm to measure the different between x and $\hat{x}$. Therefore, we are training E and D by minimizing the MSE loss defined as

$$L_{MSE}(x, \hat{x}) = |x - \hat{x}|_F^2 \tag{1}$$

Then, for each x, the encoder generates a hidden representation h.

In the classification step, we use the accuracy rate to measure the performance of our classifier. Therefore, we are training C by minimizing the cross entropy loss defined as

$$L_{CE} = \sum_{c=1}^{7}(-y_c log(\hat{y}_c) + (1 - y_c) log(1 - \hat{y}_c)) \tag{2}$$

The architectures of the encoder, decoder, and the classifier are shown in Table 2, Table 3, and Table 4.

We use Adam optimizer with 0.05 learning rate to train the autoencoder and Adam optimizer with 0.001 learning rate to train the classifier. We decrease the learning rate by 25%, if the validation loss increases for 2 consecutive epochs.

4.2.2 CNN Autoencoder as a Regularization Technique

CNN autoencoders could be applied as a regularization technique in classification. In an attempt to improve our CNN emotion recognition models and reduce overfitting, we created a multitasking network shown in Figure 1 to perform emotion classification and also image reconstruction at the same time. Our network consists of a CNN encoder, a CNN decoder , and a linear classifier, whose structures are shown in Table 2, Table 3, and Table 4.

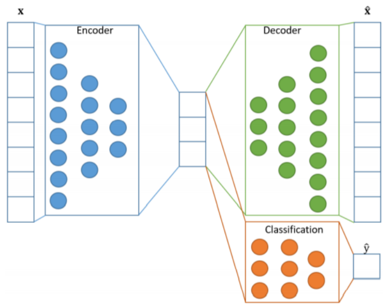


Figure 1: Regularization CNN Autoencoder

Layers	Input Channel	Output Channel	Kernel Size	Normalization	Activation	Stride	Padding
Conv2D	3	32	(3,3)	BatchNorm2d	ReLU	1	1
Conv2D	32	32	(3,3)	BatchNorm2d	ReLU	1	1
MaxPool2d			(2,2)			0	0
Conv2D	32	64	(3,3)	BatchNorm2d	ReLU	1	1
Conv2D	64	64	(3,3)	BatchNorm2d	ReLU	1	1
MaxPool2d			(2,2)			0	0
Conv2D	64	128	(3,3)	BatchNorm2d	ReLU	1	1
Conv2D	128	128	(3,3)	BatchNorm2d	ReLU	1	1
MaxPool2d			(2,2)			0	0
Conv2D	128	256	(3,3)	BatchNorm2d	ReLU	1	1
Conv2D	256	256	(3,3)	BatchNorm2d	ReLU	1	1
Linear	256*3*3	686					

Table 2: Encoder Structure

Layers	Input Channel	Output Channel	Kernel Size	Normalization	Activation	Stride	Padding
Linear	686	256*3*3					
ConvTranspose2D	256	256	(3,3)	BatchNorm2d	ReLU	1	1
MaxUnPool2d			(2,2)			0	0
ConvTranspose2D	256	128	(3,3)	BatchNorm2d	ReLU	1	1
MaxUnPool2d			(2,2)			0	0
ConvTranspose2D	128	128	(3,3)	BatchNorm2d	ReLU	1	1
ConvTranspose2D	128	64	(3,3)	BatchNorm2d	ReLU	1	1
MaxUnPool2d			(2,2)			0	0
ConvTranspose2D	64	64	(3,3)	BatchNorm2d	ReLU	1	1
ConvTranspose2D	64	32	(3,3)	BatchNorm2d	ReLU	1	1
MaxUnPool2d			(2,2)			0	0
ConvTranspose2D	32	32	(3,3)	BatchNorm2d	ReLU	1	1
ConvTranspose2D	32	3	(3,3)			1	1

Table 3: Decoder Structure

Layers	Input Dimension	Output Dimension	Activation
Linear	686	512	ReLU
Dropout(0.5)			
Linear	512	7	Sigmoid

Table 4: Classifier Structure

When performing the training, image data tensors are passed into the encoder to extract crucial features from the data. The latent space representation is passed into the linear classifier to generate a cross-entropy loss between our emotional prediction and training label $L_{CE}(y, \hat{y})$. The latent space representation is also passed into the decoder to reconstruct the input image data and generate a MSE-Loss $L_{MSE}(x, \hat{x})$ between the original image and the decoder output. We perform back-propagation on the weighted combination of the classification loss and reconstruction loss

$$\mathbb{L} = L_{CE}(y, \hat{y}) + \lambda L_{MSE}(x, \hat{x}). \tag{3}$$

Therefore, for weight w_C in the linear classifier, we have

$$\frac{\partial \mathbb{L}}{\partial w_C} = (\hat{y} - y)(\frac{\partial \hat{y}}{\partial w_C}) \tag{4}$$

Denote the ith feature in the reconstructed output D from the decoder as D_i. Consider weight w_D in the decoder network, we have:

$$\frac{\partial \mathbb{L}}{\partial w_D} = \frac{\lambda}{n} \sum_{i=1}^{n} (\hat{x}_i - x_i)(\frac{\partial \hat{x}_i}{\partial w_D}) \tag{5}$$

By chain rule, define a permutation of the entries in the output image $\hat{x}_i$ and denote x_i as the ith entry, we could obtain the derivative with respect to the encoder parameter w_E as

$$\frac{\partial \mathbb{L}}{\partial w_E} = \frac{\partial h}{\partial w_E}[(\hat{y} - y)\frac{\partial \hat{y}}{\partial h} + \frac{\lambda}{n} \sum_{i=1}^{n} (\hat{x}_i - x_i)(\frac{\partial \hat{x}_i}{\partial h})] \tag{6}$$

We use Adam optimizer with 0.001 learning rate to train multitasking model. We decrease the learning rate by 25%, if the validation loss increases for 2 consecutive epochs.

5 Results

5.1 CNN Ensemble

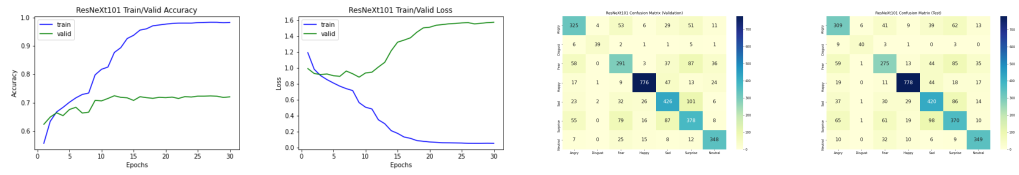
5.1.1 State-of-the-Art Convolutional Neural Networks

The training, validation, and testing accuracies of the state-of-the-art CNN models after 30 epochs of fine-tuning are shown below. We used the model's accuracy on the testing set as the metric to evaluate how well the model performed. 17 different models from the 5 major model architectures are evaluated in our experiment.

Model	Training Accuracy	Validation Accuracy	Testing Accuracy
ResNet18	0.872	0.688	0.691
ResNet34	0.912	0.706	0.699
ResNet50	0.933	0.708	0.693
ResNet101	0.936	0.701	0.692
ResNet152	0.961	0.719	0.693
ResNeXt50	0.975	0.725	0.694
ResNeXt101	0.983	0.720	0.708
Wide ResNet50	0.970	0.709	0.703
Wide ResNet101	0.970	0.711	0.706
DenseNet121	0.952	0.720	0.698
DenseNet161	0.971	0.727	0.705
DenseNet169	0.956	0.713	0.700
DenseNet201	0.970	0.722	0.703
VGG11	0.900	0.696	0.683
VGG13	0.886	0.698	0.683
VGG16	0.899	0.707	0.682
VGG19	0.925	0.701	0.685

Table 5: Accuracies of State-of-the-Art CNN Models after 30 Epochs

From the table we can see that ResNeXt (Xie et al., 2017), Wide ResNet (Zagoruyko & Komodakis, 2017) and DenseNet (Huang et al., 2018) have a slight edge in terms of model performance, which is reasonable considering that all three architectures are built on top of the original ResNet (He et al., 2015) architecture to improve its performance. The VGG (Simonyan & Zisserman, 2015) architecture come in last in terms of overall performance since it is one of the earlier models that does not utilize the technique of shortcut connection (Liu et al., 2019).



(a) Epoch vs. Accuracy Plot (b) Epoch vs. Loss Plot (c) Confusion Matrix (Validation Dataset) (d) Confusion Matrix (Testing Dataset)

Figure 2: ResNeXt101 Training Plots and Confusion Matrices

The training and validation epoch vs. accuracy plot, the training and validation epoch vs. loss plot, and the confusion matrices on the validation and testing dataset of the best performing model among the 17 models, ResNeXt101 (Xie et al., 2017), are shown above. The training plots and confusion matrices of the other models all have roughly the same shape.

As we can see from the epoch vs. loss plot, the model starts to slightly overfit after around 15 epochs, but it has no negative effect on the performance of our model as the validation accuracy stays at roughly 70% throughout the training process.

From the confusion matrices we can see that our model performs equally well on the validation and testing datasets. It should be noted that the label "Disgust" contains way fewer images than the other labels, but our model shows robustness towards this imbalance of data, predicting most of the images under this label correctly.

5.1.2 Ensemble of Convolutional Neural Networks

To further improve the performance of the CNN models, we put them into ensembles based on their accuracy on the testing set. Starting from the model with the highest accuracy, ResNeXt101, we gradually increase the number of models in the ensemble until all 17 models are included. The evaluation results are shown in the plot below.

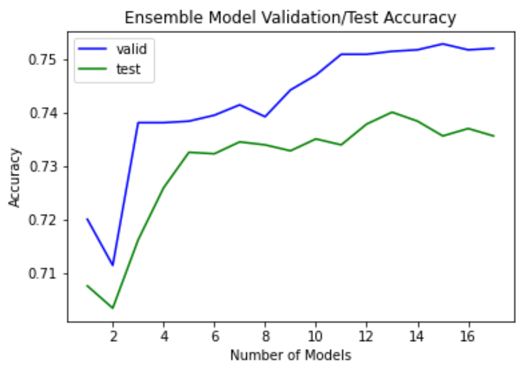
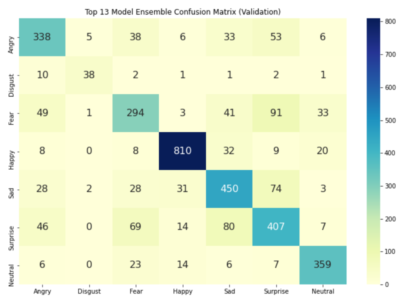


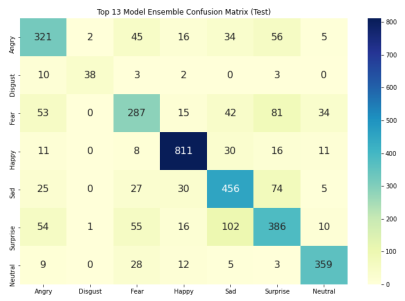
Figure 3: Number of Models vs. Accuracy Plot

As expected, the ensemble model achieved a higher accuracy than any of the individual CNN models. In particular, the ensemble consisting of the top 13 models achieved the highest testing accuracy of 74.0%. Interestingly, all the VGG models are not included in this ensemble as they have the 4 lowest testing accuracies. This continues to show that CNN models with shortcut connections perform significantly better than CNN models that do not utilize this technique.

The confusion matrices of the best performing ensemble model is shown below. The overall pattern is similar to the confusion matrices of the best performing individual CNN, ResNeXt101, but more accurate in terms of the classification of nearly every label.



(a) Confusion Matrix (Validation Dataset)



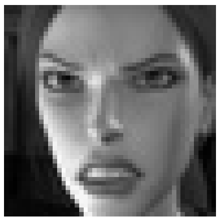
(b) Confusion Matrix (Testing Dataset)

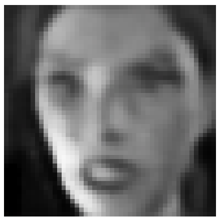
Figure 4: Top-13-Model Ensemble Confusion Matrices

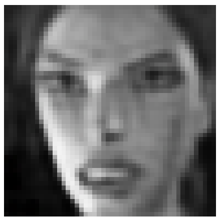
5.2 CNN Autoencoder

5.2.1 Dimensionality Reduction

We experiment with different hidden representation dimensions from 200 to 1000. We find out 1×500 is a reasonable dimension, since it has low MSE loss, which indicates that it preserves important features, while keeping the dimensionality low. The autoencoder converges in 50 epochs. According to Figure 4, 1×200 hidden dimension is not able to reconstruct the eyes and the month clearly, while 1×500 hidden dimension and 1×1000 hidden dimension are able to capture most of the facial features.


(a) Original


(b) 1×200 hidden dimension
$\mathbb{L}_{MSE} = 0.0398$


(c) 1×500 hidden dimension
$\mathbb{L}_{MSE} = 0.0364$

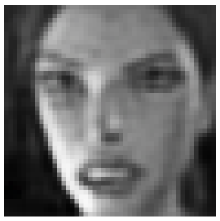
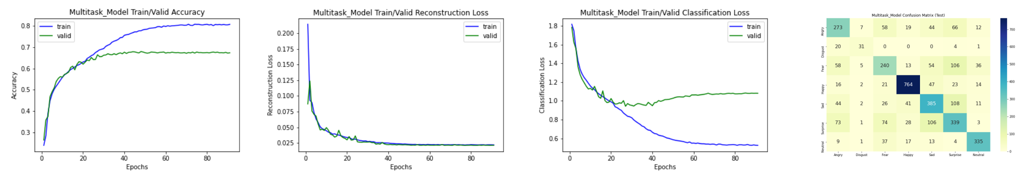

(d) 1×1000 hidden dimension,
$\mathbb{L}_{MSE} = 0.0335$

Figure 5: Reconstructed Image Comparison and MSE loss after 80 epochs

Despite the autoencoder's success in reconstructing images, the classifier is not able to make correct predictions. Even using 1×1000 hidden dimension, the training accuracy and validation can only achieve 25.1% and 24.7% after the model converges. The model does not improve when we add more linear layers to the classifier.

5.2.2 Multitasking Model



(a) Epoch vs. Accuracy (b) Epoch vs. Reconstruction Loss (c) Epoch vs. Classification Loss (d) Confusion Matrix (Testing Dataset)

Figure 6: Multitask Model Training Plots and Confusion Matrix

The training performance of our multitasking model is satisfying in terms of both image reconstruction and classification. We conducted experiments choosing different latent space representation dimensions ranging from 500 to 1000. The multitasking model performed best when choosing the latent space representation dimension as 686. Classification performance reached a validation accuracy of 67.4% , which is close to a lot of the state of art models. The reconstruction loss of the autoencoder also converged during training to 0.024 and reduced overfitting for classification. Unlike ordinary CNN classifiers, our training accuracy slowly converged to 80% instead of boosting to over 95% during training. The confusion matrix is shown above. The multitasking model performed well when predicting emotion Happy and Surprise, but had a weaker performance on Angry and Disgust compared to state of the art models.

6 Discussion and Analysis

For ensemble models, it is often difficult to determine the exact number of models to include in the ensemble. Based on the empirical result in 5.1.2, we can see that including at least 3 models in the ensemble will achieve an improved accuracy over the best individual CNN model. The Top-2-Model Ensemble does not perform well because a majority vote cannot be attained when the two models vote differently, which will result in a random decision. However, since both models are usually accurate, we can still achieve a decent accuracy. We can also see that the testing accuracy does not improve by much when we include more than 5 models, reaching a point of diminishing returns. Therefore, if we have limited computing power, choosing the top 5 individual models in the ensemble model should give a good enough result.

According to the result in 5.2.1, our CNN autoencoder is able to extract important features for the reconstruction. However, the low training accuracy for the classification indicates that the model is underfitting. One of the reasons for the bad performance is that training the autoencoder is unsupervised, so the hidden representation does not extract sufficient information for the emotional detection, even though it learns enough features for the reconstruction. Therefore, it has much worse performance than the multitasking model, even though they have the same encoder, decoder and classifier.

The multitasking model is able to achieve 65.9% testing accuracy, beating some baseline models. One key characteristic of this model is that it does not overfit too much. Unlike the complicated CNN models discussed in 4.1.1, the validation loss does not increase much for the last several epochs. The difference between the training accuracy and the testing accuracy is not too significant, either. Therefore, we conclude that the decoder is able to regularize our model and prevent it from overfitting. In the future, we will try to make our encoder more complicated by applying the ResNet structure or the DenseNet structure and use decoder to regularize. In that way, may the model achieve both low variance and low bias.

Another way to improve our model is to apply more data preprocessing. For example, use the Viola-Jones face detector (Viola & Jones, 2001) to detect the face in the image and remove background and non-face areas. We can also preprocess the data by using a GAN-based face frontalization framework (Lai & Lai, 2018) to frontalize input face images while preserving the identity and expression characteristics.

References

Bank, D., Koenigstein, N., & Giryes, R. (2020). Autoencoders.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. *CVPR09*.

Devries, T., Biswaranjan, K., & Taylor, G. W. (2014). Multi-task learning of facial landmarks and expression. *2014 Canadian Conference on Computer and Robot Vision*, 98–103. https://doi.org/10.1109/CRV.2014.21

Ekman, P., & Friesen, W. (1971). Constants across cultures in the face and emotion. *Journal of personality and social psychology*, *17*(2), 124–129. https://doi.org/10.1037/h0030377

Friedman, J. H., & Fayyad, U. (1997). On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, *1*, 55–77.

Ghayoumi, M., & Bansal, A. K. (2016). Unifying geometric features and facial action units for improved performance of facial expression analysis. *CoRR*, *abs/1606.00822*. http://arxiv.org/abs/1606.00822

Goodfellow, I. J., Erhan, D., Carrier, P. L., Courville, A., Mirza, M., Hamner, B., Cukierski, W., Tang, Y., Thaler, D., Lee, D.-H., Zhou, Y., Ramaiah, C., Feng, F., Li, R., Wang, X., Athanasakis, D., Shawe-Taylor, J., Milakov, M., Park, J., … Bengio, Y. (2013). Challenges in representation learning: A report on three machine learning contests.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition.

Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2018). Densely connected convolutional networks.

Kim, B., Dong, S., Roh, J., Kim, G., & Lee, S. (2016). Fusing aligned and non-aligned face information for automatic affect recognition in the wild: A deep learning approach. *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1499–1508. https://doi.org/10.1109/CVPRW.2016.187

Lai, Y., & Lai, S. (2018). Emotion-preserving representation learning via generative adversarial network for multi-view facial expression recognition. *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*, 263–270. https://doi.org/10.1109/FG.2018.00046

Li, S., & Deng, W. (2018). Deep facial expression recognition: A survey. *CoRR*, *abs/1804.08348*. http://arxiv.org/abs/1804.08348

Liu, T., Chen, M., Zhou, M., Du, S. S., Zhou, E., & Zhao, T. (2019). Towards understanding the importance of shortcut connections in residual networks.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library.

Pramerdorfer, C., & Kampel, M. (2016). Facial expression recognition using convolutional neural networks: State of the art.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *Ann. Math. Statist.*, *22*(3), 400–407. https://doi.org/10.1214/aoms/1177729586

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.

Tang, Y. (2015). Deep learning using linear support vector machines.

Tarnowski, P., Kołodziej, M., Majkowski, A., & Rak, R. J. (2017). Emotion recognition using facial expressions [International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland]. *Procedia Computer Science*, *108*, 1175–1184. https://doi.org/https://doi.org/10.1016/j.procs.2017.05.025

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, *1*, I–I. https://doi.org/10.1109/CVPR.2001.990517

Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks.

Yanan Guo, Dapeng Tao, Jun Yu, Hao Xiong, Yaotang Li, & Dacheng Tao. (2016). Deep neural networks with relativity learning for facial expression recognition. *2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, 1–6. https://doi.org/10.1109/ICMEW.2016.7574736

Zagoruyko, S., & Komodakis, N. (2017). Wide residual networks.

Zhang, Z., Luo, P., Loy, C. C., & Tang, X. (2015). Learning social relation traits from face images.