Project Report - ECE 176

Laura Fleig
University of California, San Diego
Department of Cognitive Science
A16887351

Abstract

This project explores the application of Convolutional Neural Networks (CNNs) enhanced with squeeze-and-excite (SE) blocks for the task of facial expression recognition. With the aim of improving the accuracy and efficiency of existing models, we experimented with four different CNN architectures: a shallow network, a deeper network, an SE network based on the deep network, and an improved SE network. Each model was designed to tackle the complexities of recognizing human emotions through facial expressions, a task that finds extensive applications in sectors such as human-computer interaction, augmented reality, and online education. Our experiments demonstrated that the integration of SE blocks boosts model performance by focusing on informative features and suppressing less useful ones. The improved SE network achieved a test accuracy of 62% on the FER-2013 dataset and 76% on the RAF-DB dataset, underscoring the efficacy of attentional mechanisms in facial expression recognition models. This investigation sheds light on the potential of deep learning models enhanced with attentional mechanisms for emotion recognition and sets the stage for future research in optimizing CNN architectures for complex visual recognition tasks.

1 Introduction

Facial expression recognition (FER) represents a dynamic and fascinating subfield of pose estimation, situated at the intersection of computer vision and deep learning. This area of study is driven by the understanding that human faces convey a wealth of information, not only about identity but also regarding emotional states, intentions, and reactions. The ability to automatically recognize and interpret these expressions opens up a plethora of applications across various sectors. In human-computer interaction, FER paves the way for more intuitive and responsive systems that can adapt to the user's emotional state. Its applications extend into augmented reality and the gaming industry. Beyond commercial applications, this technology plays a crucial role in educational settings, particularly in online education, where it can be used to gauge student understanding and engagement, or in virtual meetings to assess participant reactions. The multifaceted utility of facial expression emotion recognition makes it not only an interesting area of research but also a critical technological advancement with widespread implications.

This project centers around leveraging Convolutional Neural Networks (CNNs), specifically tailored for the task of facial expression recognition. To augment the capability of traditional CNNs, this project incorporates attentional mechanisms (specifically squeeze-and-excite blocks) into the CNN architecture. This project examines 4 different network structures (a shallow network, a deep network, a deep network with SE blocks added, and an improved SE network) to explore how different CNN architectures impact facial expression recognition performance. We find that deeper networks with attention tend to do better at this task.

2 Related Work

Convolutional Neural Networks, or CNNs, excel in handling image data, automatically extracting and learning the most relevant features through their deep, hierarchical structure, making them particularly suited for recognizing the nuanced patterns in facial expressions. The strength of CNNs lies in their ability to process raw image pixels with minimal preprocessing, learning optimal representations directly from the data, which is crucial for capturing the subtleties of human emotions expressed through facial gestures.

Attentional Convolutional Neural Networks (ACNNs) (see, for example, [1]) introduce a dynamic weighting of the importance of different areas of the input image, allowing the model to prioritize regions most indicative of emotional expressions, such as the eyes and mouth. This addition aims to enhance the model's interpretability and accuracy by mimicking the human visual system's focus on salient features when identifying emotions.

One type of attentional mechanism is squeeze-and-excitation. "Squeeze and Excite" (SE) blocks represent a powerful architectural unit within convolutional neural networks (CNNs) that aim to recalibrate the channel-wise feature responses by explicitly modeling interdependencies between channels. This concept was introduced in the paper "Squeeze-and-Excitation Networks" [2], and it has since been widely adopted for enhancing the representational power of CNNs. The primary components of an SE block are the squeeze operation, which aggregates the global spatial information of each channel, and the excitation operation, which learns a channel-wise recalibration weight. SE blocks can enhance the model's ability to focus on the most informative features and suppress the less useful ones. They also pose a minimal increase in model parameters and computational cost, making them an efficient way to boost performance.

3 Methods

For preprocessing, we employed a multifaceted data augmentation strategy to enhance the diversity and robustness of our training dataset, leveraging Tensorflow's ImageDataGenerator function. Initially, all images were rescaled to normalize pixel values between 0 and 1. We allocated 20% of the dataset for validation purposes, ensuring the model's performance could be accurately assessed on unseen data. The augmentation techniques applied included horizontal flipping of images to simulate different perspectives, random rotations within a ±10-degree range to introduce variability in orientation, and shifts in both width and height by up to 8%, mimicking slight changes in the position of subjects within the frame. Additionally, we adjusted the brightness of the images within a range of 80% to 120% to account for varying lighting conditions, and applied random zooms between 92% and 108% to simulate cropping and focus variations. The 'nearest' fill mode was chosen for handling points outside the input boundaries, ensuring a natural look in the modified images. These strategies collectively aimed to make our model more adaptable and improve its generalization capabilities across varied imaging conditions. Most of these data augmentation strategies were adapted from [3].

We decided to iterate through increasingly complex network architectures to easily see how various changes impact performance.

3.1 Shallow Network

We first re-implemented a shallow network based on the descriptions in [4]. It starts with a convolutional layer of 32 filters of size 3x3, utilizing 'same' padding to preserve the spatial dimensions, and L2 regularization to prevent overfitting. Batch normalization and ReLU activation follow this layer to aid in faster convergence and introduce non-linearity. Dropout of 25% is applied to reduce overfitting further. The second convolutional layer increases the filter count to 64 and includes a max-pooling layer with a pool size of 2x2 to reduce the spatial dimensions by half, followed by another dropout layer. The output is then flattened to transition from the 2D feature maps to a 1D vector, making it suitable for dense layers. A dense layer with 512 units follows, employing L2 regularization and ReLU activation, along with a higher dropout rate of 50% to combat overfitting in the fully connected portion of the network. The final layer has 7 units corresponding to the number of classes and uses a softmax activation function for multi-class classification. The model is compiled with an Adam optimizer with a learning rate of 0.0001, categorical cross-entropy as the loss function, and tracks accuracy as a performance metric. See Table 1.

Layer	Output Shape	Param #
Conv2D	(48, 48, 32)	320
BatchNormalization	(48, 48, 32)	128
ReLU	(48, 48, 32)	0
Dropout	(48, 48, 32)	0
Conv2D	(48, 48, 64)	18496
BatchNormalization	(48, 48, 64)	256
ReLU	(48, 48, 64)	0
MaxPooling2D	(24, 24, 64)	0
Dropout	(24, 24, 64)	0
Flatten	(36864)	0
Dense	(512)	18874880
ReLU	(512)	0
Dropout	(512)	0
Dense	(7)	3591

Table 1: Shallow Network Architecture

3.2 Deeper Network

Next, we structured a deep network, also based on descriptions in [4]. It starts with a convolutional layer equipped with 64 filters of size 3x3, employing same padding and L2 regularization to combat overfitting, followed by batch normalization and ReLU activation for non-linearity and faster convergence. It incorporates a max pooling layer to reduce the dimensionality and a dropout of 25% for further regularization. In the second convolutional layer, the filter size is increased to 128 with a larger kernel size of 5x5, maintaining the same padding, L2 regularization, batch normalization, ReLU activation, followed by max pooling and dropout. The third and fourth convolutional layers increase the complexity even more, both employing 512 filters of size 3x3, each followed by batch normalization, ReLU activation, max pooling, and drop out.

After the convolutional layers, the network flattens the output to transition from 2D feature maps to a 1D vector suitable for dense layers. It then moves through two fully connected layers, first with 256 units and then with 512 units, each incorporating L2 regularization, batch normalization, ReLU activation, and a higher dropout rate of 50%.

The architecture concludes with a dense output layer of 7 units with a softmax activation function, aimed at multi-class classification among seven possible categories. Compiled with an Adam optimizer set at a learning rate of 0.001, the model employs categorical cross-entropy as the loss function and tracks accuracy as a metric. See Table 2.

3.3 Squeeze-and-Excite Network

Subsequently, we wanted to add another layer of complexity and decided to experiment with squeeze-and-excite networks. One challenge this posed was moving from Sequential to Functional API. We based our Squeeze-and-Excite (SE) network on our previous deep network. It starts with a convolutional layer followed by ReLU, batch normalization, and an SE block to emphasize important features. The architecture deepens with more convolutional layers, each followed by batch normalization, SE blocks, and either max pooling/dropout. After the convolutional layers, the network includes a flattening step, two dense layers with ReLU and batch normalization, and dropout for regularization. It concludes with a softmax output layer for classification. Like above, the Adam optimizer with an exponential decay learning rate schedule is used for training. See Table 3.

3.4 Improved SE Network

Finally, we moved away from the original structures introduced in [4], changing the filter numbers in the convolutional layers. The first block uses 32 filters of size 3x3, while subsequent blocks increase the number of filters to 64 and then 128 and 128. Each block consists of a convolutional layer with ReLU activation and batch normalization, enhanced by a SE block. After the second, third, and fourth convolutional layers, max pooling reduces spatial dimensions, and dropout is applied after 2, 4, and dense layers to prevent overfitting. We did not add more than 4 convolutional layers, as

Layer	Output Shape	Param #
Conv2D	(48, 48, 64)	640
BatchNormalization	(48, 48, 64)	256
ReLU	(48, 48, 64)	0
MaxPooling2D	(24, 24, 64)	0
Dropout	(24, 24, 64)	0
Conv2D	(24, 24, 128)	204928
BatchNormalization	(24, 24, 128)	512
ReLU	(24, 24, 128)	0
MaxPooling2D	(12, 12, 128)	0
Dropout	(12, 12, 128)	0
Conv2D	(12, 12, 512)	590336
BatchNormalization	(12, 12, 512)	2048
ReLU	(12, 12, 512)	0
MaxPooling2D	(6, 6, 512)	0
Dropout	(6, 6, 512)	0
Conv2D	(6, 6, 512)	2359808
BatchNormalization	(6, 6, 512)	2048
ReLU	(6, 6, 512)	0
MaxPooling2D	(3, 3, 512)	0
Dropout	(3, 3, 512)	0
Flatten	(4608)	0
Dense	(256)	1179904
BatchNormalization	(256)	1024
ReLU	(256)	0
Dropout	(256)	0
Dense	(512)	131584
BatchNormalization	(512)	2048
ReLU	(512)	0
Dropout	(512)	0
Dense	(7)	3591

Table 2: Deeper Model Architecture

research suggests that adding more layers for facial expression recognition on similar datasets does not improve accuracy [4].

The network transitions to fully connected layers after flattening the convolutional output. A dense layer with 1024 units is followed by batch normalization and dropout, then a second dense layer with 256 units continues the pattern of batch normalization and dropout. The architecture concludes with a softmax output layer with 7 units, indicating the model is designed for classification into seven categories. As with the previous models, an Adam optimizer with an exponential decay learning rate schedule is used to compile the model, targeting categorical cross-entropy as the loss function and aiming to optimize accuracy. See Table 4.

4 Experiments

4.1 Datasets

FER-2013: The Facial Expression Recognition 2013 (FER-2013) dataset is widely utilized for the automatic recognition of facial expressions. It contains grayscale images of faces, each labeled with one of seven emotion categories: anger, disgust, fear, happiness, sadness, surprise, and neutral. The images are 48x48 pixels. The training set includes 28,708 images, while the public test set includes 3,589 images [5].

RAF-DB: The Real-world Affective Faces Database has nearly 30,000 images of faces from the Internet. There is a variety of ages, genders, ethnicities, head poses, occlusions, filters, and lighting conditions, sorted into seven categories. Half of the dataset is available publicly, so this project uses 15,000 images [6] [7]. Out of curiosity (inspired by [8] and [9]) and for educational purposes, we

Layer	Output Shape	Param #
InputLayer	(48, 48, 1)	0
Conv2D	(48, 48, 64)	640
ReLU	(48, 48, 64)	0
BatchNormalization	(48, 48, 64)	256
GlobalAveragePooling2D	(64)	0
Dense	(4)	256
Dense	(64)	256
Reshape	(1, 1, 64)	0
Multiply	(48, 48, 64)	0
Conv2D	(48, 48, 128)	73856
ReLU	(48, 48, 128)	0
BatchNormalization	(48, 48, 128)	512
GlobalAveragePooling2D	(128)	0
Dense	(8)	1024
Dense	(128)	1024
Reshape	(1, 1, 128)	0
Multiply	(48, 48, 128)	0
MaxPooling2D	(24, 24, 128)	0
Dropout	(24, 24, 128)	0
Conv2D	(24, 24, 512)	590336
ReLU	(24, 24, 512)	0
BatchNormalization	(24, 24, 512)	2048
GlobalAveragePooling2D	(512)	0
Dense	(32)	16384
Dense	(512)	16384
Reshape	(1, 1, 512)	0
Multiply	(24, 24, 512)	0
MaxPooling2D	(12, 12, 512)	0
Conv2D	(12, 12, 512)	2359808
ReLU	(12, 12, 512)	0
BatchNormalization	(12, 12, 512)	2048
GlobalAveragePooling2D	(512)	0
Dense	(32)	16384
Dense	(512)	16384
Reshape	(1, 1, 512)	0
Multiply	(12, 12, 512)	0
MaxPooling2D	(6, 6, 512)	0
Dropout	(6, 6, 512)	0
Flatten	(18432)	0
Dense	(256)	4718848
BatchNormalization	(256)	1024
Dropout	(256)	0
Dense	(512)	131584
BatchNormalization	(512)	2048
Dropout	(512)	0
Dense	(7)	3591

Table 3: SE Model Architecture

explored semi-supervised learning by artificially creating an unlabeled subset of the RAF-DB dataset and creating pseudo labels. We first did an initial training of 25 epochs on just the labeled data, and then trained another 10 epochs on the combined labeled and pseudo labeled data.

4.2 Results

In general, we found that a batch size of 32 (vs. 128 mentioned in [4]) worked best for this. We also experimented with placing BatchNormalization before and after activation, as this seems to be a

Layer	Output Shape	Param #
InputLayer	(48, 48, 1)	0
Conv2D	(48, 48, 32)	320
ReLU	(48, 48, 32)	0
BatchNormalization	(48, 48, 32)	128
GlobalAveragePooling2D	(32)	0
Dense	(2)	64
Dense	(32)	64
Reshape	(1, 1, 32)	0
Multiply	(48, 48, 32)	0
Conv2D	(48, 48, 64)	18496
ReLU	(48, 48, 64)	0
BatchNormalization	(48, 48, 64)	256
GlobalAveragePooling2D	(64)	0
Dense	(4)	256
Dense	(64)	256
Reshape	(1, 1, 64)	0
Multiply	(48, 48, 64)	0
MaxPooling2D	(24, 24, 64)	0
Dropout	(24, 24, 64)	0
Conv2D	(24, 24, 128)	73856
ReLU	(24, 24, 128)	0
BatchNormalization	(24, 24, 128)	512
GlobalAveragePooling2D	(128)	0
Dense	(8)	1024
Dense	(128)	1024
Reshape	(1, 1, 128)	0
Multiply	(24, 24, 128)	0
MaxPooling2D	(12, 12, 128)	0
Conv2D	(12, 12, 128)	147584
ReLU	(12, 12, 128)	0
BatchNormalization	(12, 12, 128)	512
GlobalAveragePooling2D	(128)	0
Dense	(8)	1024
Dense	(128)	1024
Reshape	(1, 1, 128)	0
Multiply	(12, 12, 128)	0
MaxPooling2D	(6, 6, 128)	0
Dropout	(6, 6, 128)	0
Flatten	(4608)	0
Dense	(1024)	4719616
BatchNormalization	(1024)	4096
Dropout	(1024)	0
Dense	(256)	262400
BatchNormalization	(256)	1024
Dropout	(256)	0
Dense	(7)	1799

Table 4: Improved SE Model Architecture

currently debated topic. We found that having BatchNormalization after activation creates a more stable and faster-converging validation loss.

Our shallow network reached 41% validation accuracy in 30 epochs on the FER-2013 dataset (Figure 1). This was lower than the accuracy reached in [4]; however, the paper uses a different dataset.

After 30 epochs, the deeper network reached 56% validation accuracy on the FER-2013 dataset (Figure 2). This represents a significant improvement compared to the shallower network, which aligns with expectations: A deep architecture is designed to handle more nuanced distinctions between

Figure 1: Shallow Model Plots

Figure 2: Deeper Model Plots

image categories by extracting a broader and more complex set of features and employing more aggressive regularization techniques to manage the increased model complexity.

Incorporating squeeze-and-excitation blocks increased accuracy to 61% (Figure 3).

Our improved SE network reached 63% accuracy on the FER-2013 dataset (Figure 4). We incorporated Early Stopping, which set in at epoch 52, determining that epoch 47 was the best.

After completing training, we then evaluated our final model, the improved SE network, on the FER-2013 test set and achieved 62.3% test accuracy. While this is below current benchmarks, we believe this is a decent result for the computational and time limitations of this project.

Finally, we trained the improved SE network on the RAF-DB dataset. Here, we got 77% validation accuracy on the initial training on the labeled dataset, and nearly 80% validation accuracy on the combined training round (Figure 5). When running on the RAF-DB test set, we achieved 76.5% accuracy.

5 Supplementary Material

Please refer to https://github.com/Greencouch82/ECE176-finalproject for my code and video.

Figure 3: SE Model Plots

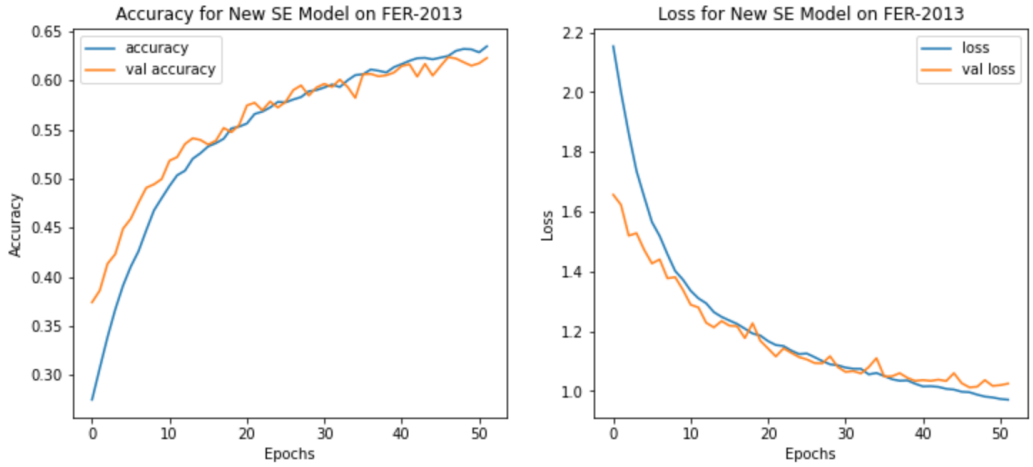

Figure 4: Improved SE Model Plots on FER-2013

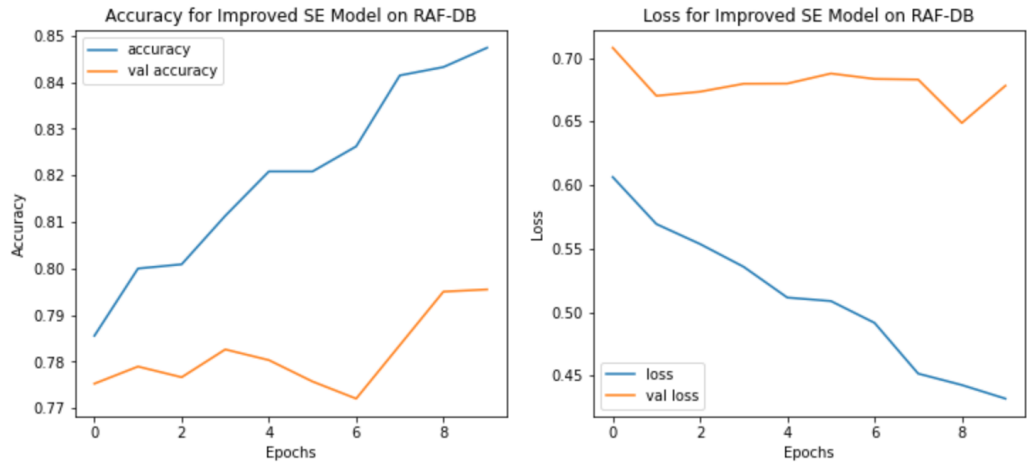

Figure 5: Improved SE Model Plots on RAF-DB (just combined training epochs)

Figure 6: Improved SE Model Prediction Visualization

References

[1] S. Minaee and A. Abdolrashidi, "Deep-emotion: Facial expression recognition using attentional convolutional network," 2019.

[2] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[3] Hujie-Frank, "Hujie-frank/senet: Squeeze-and-excitation networks." [Online]. Available: https://github.com/hujie-frank/SENet

[4] S. Alizadeh and A. Fazel, "Convolutional neural networks for facial expression recognition," 2017.

[5] M. Sambare, "Fer-2013," Jul 2020. [Online]. Available: https://www.kaggle.com/datasets/msambare/fer2013

[6] S. Li, W. Deng, and J. Du, "Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 2584–2593.

[7] S. Li and W. Deng, "Reliable crowdsourcing and deep locality-preserving learning for unconstrained facial expression recognition," *IEEE Transactions on Image Processing*, vol. 28, no. 1, pp. 356–370, 2019.

[8] Z. Min, Q. Ge, and C. Tai, "Why the pseudo label based semi-supervised learning algorithm is effective?" 2023.

[9] J. Yu, Z. Cai, R. Li, G. Zhao, G. Xie, J. Zhu, W. Zhu, Q. Ling, L. Wang, C. Wang *et al.*, "Exploring large-scale unlabeled faces to enhance facial expression recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 5803–5810.