Facial Expression Recognition

Jingyi Zhang, Haoxian Ruan, Andrew Nguyen and Adam Mhal
jyz0328@bu.edu, rhx2000@bu.edu, aynguyen@bu.edu , adammhal@bu.edu

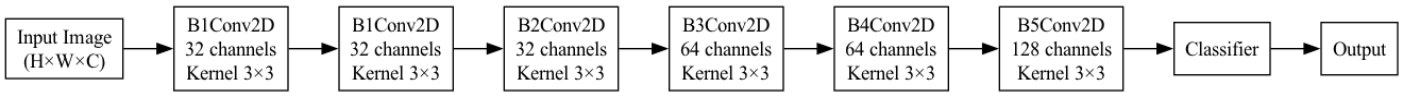

Figure1 . CNN improvement model flowchart

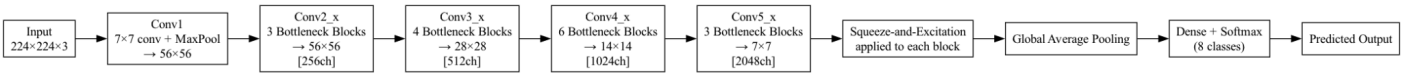

Figure2 . ResNet improvement model flowchart

Figure 3. An example of Categories of facial expressions[1]

1. Task

Our task is to develop a facial expression recognition (FER) system that can accurately classify human facial expressions such as: “angry”, “disgust”, “fear”, “happy”, “sad”, “surprise”, and “neutral”. The main challenges of this task include: (1) **Distinguishing similar expressions** such as "surprise" and "fearful." (2) **Handling different conditions:** The model needs to work well even when lighting, camera angles, or things like glasses and masks affect the face. (3) **Adapting to different people:** The model should recognize expressions correctly across various ages, genders, and ethnic backgrounds.

2. Related Work

We reviewed related works for the following models.

CNN, as a foundational approach in FER, has been widely employed as basic facial expression recognition. This architecture excels in extracting critical facial features, especially in areas like the nose and mouth [2]. However, despite strong performance on standard datasets, CNN models face limitations when generalizing across different databases and handling occluded facial features. These models also exhibit challenges in addressing pose variations.

ResNet enhances by introducing residual connections, which help avoid vanishing gradients and improve training in deep architectures [3]. It effectively extracts expressive features and performs well on standard FER benchmarks. However, it still struggles with generalization under extreme head poses, occlusions (e.g., masks), and subtle expressions, especially across datasets.

3. Approach

3.1 CNN Baseline

Convolutional Neural Networks (CNNs) extract features by applying small filters across input images. Local connectivity and weight sharing allow CNNs to efficiently learn spatial hierarchies.
The output of a convolutional layer is computed as:

$$z^{(l)} = \sigma(\text{Conv2D}(z^{(l-1)}; W^{(l)}, b^{(l)}))$$

where $z^{(l-1)}$ is the input feature map, $W^{(l)}$ and $b^{(l)}$ are the weights and bias, and σ is the ReLU activation function.

After flattening,the feature map is passed to a fully connected layer to produce a hidden vector **h**, followed by a softmax layer that computes the predicted class probabilities **y^**:

$$\hat{y} = \text{softmax}(W \cdot h + b)$$

Here is our **CNN baseline Model** Structure:

- Block 1: 2 convolutional layers + 32 channels
- Block 2: 1 convolutional layer with 32 channels + max pooling
- Block 3: 1 convolutional layer with 64 channels and max pooling

- Block 4: 1 convolutional layer with 64 channels and max pooling
- Classifier: Flattened feature maps are passed through a fully connected layer with 1024 units, followed by a softmax output layer

3.2 CNN Improvement

In the **CNN implementation**, we introduced the following improvements beyond the baseline, a flow chart is shown in figure 1:

- **Model enhancement**: We added a fifth convolutional layer with 128 channels and enhanced regularization by setting the dropout rate to 0.3 after this final convolutional layers.
- **Hyperparameter tuning**: We performed a grid search over batch sizes of 64, 128, and 256 and learning rates of 0.001, 0.005, and 0.01 to find the optimal hyperparameters
- **Seed configuration**: We use three random seeds to ensure that the results are reproducible. The final performance is reported as the average over these three runs to reduce randomness and enhance the credibility of the results.

3.3 ResNet Baseline

ResNet introduces residual connections, also known as skip connections, by adding the input back to the output:

$$z=F(x)+x$$

Here, x is the input feature map to the residual block, and z is the output after residual addition.F(x) is the residual mapping, defined as transformation of two convolutional layers(Conv1 and Conv2) and Batch Normalization (BN1 and BN2) and ReLU activation(σ):

$$F(x) = BN_2(Conv_2(\sigma(BN_1(Conv_1(x)))))$$

If the input and output dimensions differ, a 1×1 convolution is used to project the input and match dimensions before addition. This residual structure helps stabilize training in deep networks by preserving gradient flow and easing optimization.

In our **ResNet baseline**,we used residual blocks with skip connections to build a deep yet efficient expression classifier.Each residual block had two 3×3 convolutions, followed by batch normalization and ReLU. When dimensions changed, a 1×1 convolution adjusted the input for addition.

The network started with a 3×3 convolution (16 channels), followed by:

- 2 residual blocks with 16 channels
- 2 residual blocks with 32 channels (first uses stride 2)
- 2 residual blocks with 64 channels (first uses stride 2)
- A global average pooling layer reduces feature maps before a dense softmax layer outputs the expression class.

3.4 ResNet Improvement

For the **ResNet model implementation**, we introduced the following improvements,a flow chart is shown in figure 2:

- **Squeeze-and-Excitation Block**: To recalibrate channel responses, each residual block's output $U \in R^{H' \times W' \times C}$ was wrapped in an SE module:

  Squeeze: global average pooling

  $$z_c = \frac{1}{H'W'} \sum_{i=1}^{H'} \sum_{j=1}^{W'} U_{i,j,c} \quad c = 1,\ldots, C$$

  Excitation: two-layer MLP

  $$u = ReLU(W_1 z), \quad s = \sigma(W_2 u)$$

  Recalibration: $newU_{i,j,c} = s_c U_{i,j,c}$

- **Data Augmentation**: Training samples were randomly flipped horizontally to increase variation and reduce overfitting.
- **Label smoothing**: One-hot labels can make the model overconfident, so we replaced each one-hot $y \in \{0, 1\}^K$ by a soft target $y \in (0, 1)^K$:

  $$newy_c = (1 - \epsilon) y_c + \frac{\epsilon}{K} \quad c = 1,\ldots, K$$

  Where K is the number of classes, in this case 8, y_c = 1 is c is the true label and 0 otherwise, and ϵ is the smoothing factor, which we use ϵ = 0.1.
- **Adam Optimizer and Weight Decay:**We adopted the Adam optimizer with decoupled weight decay (AdamW), which prevents overfitting by directly applying weight decay to parameters, offering more consistent and

independent regularization than traditional Adam or SGD.

- **Data Input:** Input data was modified from 48×48 grayscale to 224×224 RGB to capture fine-grained facial features.

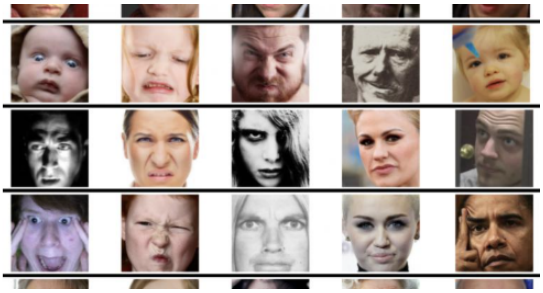

Figure 4. AffectNet samples

Emotion	Label	Training	Validation	Test	Total	Percentage
Neutral	0	52410	11233	11231	74874	26.03%
Happiness	1	94086	20166	20162	134414	46.73%
Sadness	2	17821	3819	3819	25459	8.85%
Surprise	3	9862	2114	2114	14090	4.90%
Fear	4	4464	957	957	6378	2.22%
Disgust	5	2662	571	570	3803	1.32%
Anger	6	17417	3733	3732	24882	8.65%
Contempt	7	2625	562	563	3750	1.30%
Overall		201347	43155	43148	287650	

Table1. Distribution of Emotion Labels in the Processed AffectNet Subset

4. Datasets

We chose the AffectNet dataset [5] as our source. AffectNet contains approximately 1 million facial images collected from the Internet by querying three major search engines using 1,250 emotion-related keywords in six different languages. Each image in AffectNet is manually annotated with one of eight emotion categories: **0: Neutral, 1: Happiness, 2: Sadness, 3: Surprise, 4: Fear, 5: Disgust, 6: Anger, 7: Contempt**.

After data preprocessing, we constructed a mini version of AffectNet with a total of **287,650** images. The dataset was turned into a CSV file easier reading with the image, label, valence, arousal, and landmarks each in their respective columns. The images were then cropped based on the landmark coordinates, converted to grayscale images, and all resized to 48×48 pixels for consistency and computational efficiency. The dataset is split as follows:

- **Training set**: 201,347 samples
- **Validation set**: 43,155 samples
- **Test set**: 43,148 samples

The distribution of emotion labels across the training, validation, and test sets is shown in Table 1. This subset preserves the class imbalance present in the original AffectNet dataset, with Happiness and Neutral accounting for the majority of samples, while Disgust and Contempt remain underrepresented. We use this preprocessed AffectNet Subset as our dataset.

5. Evaluation Metrics

We evaluate model performance using **accuracy, loss, and the confusion matrix**.

Accuracy shows how many predictions are correct, while loss (using categorical cross-entropy) tells us how far off the predictions are from the true labels. During training, we track both training and validation accuracy and loss at each epoch to understand how the model is learning.In general, higher accuracy and lower loss mean better performance.

The confusion matrix helps us analyze per-class performance. Ideally, values along the diagonal—representing the proportion of correctly predicted samples for each class—should be close to 1. Higher diagonal values indicate better recognition accuracy for the corresponding emotion.

Figure 5.CNN baseline accuracy and loss plot

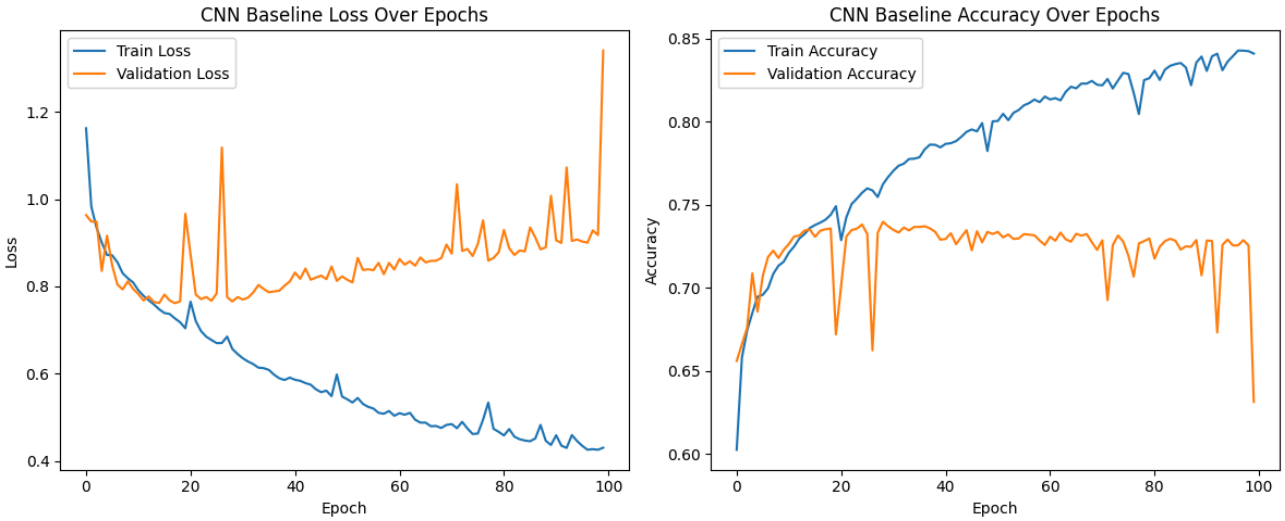

Figure 6.CNN implement accuracy and loss plot

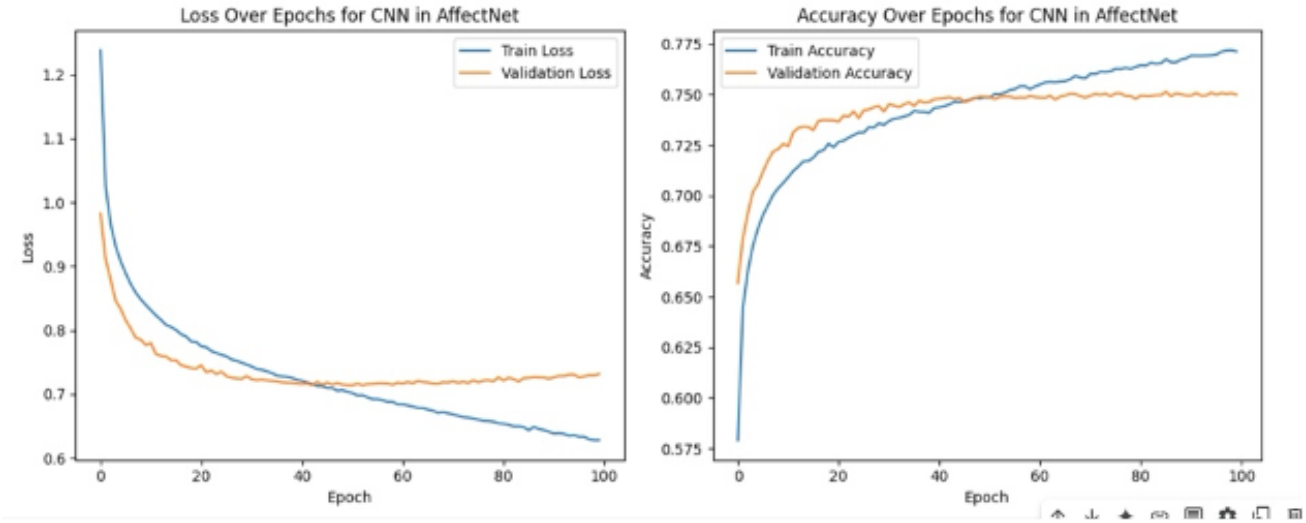

Figure 7.ResNet baseline accuracy and loss plot

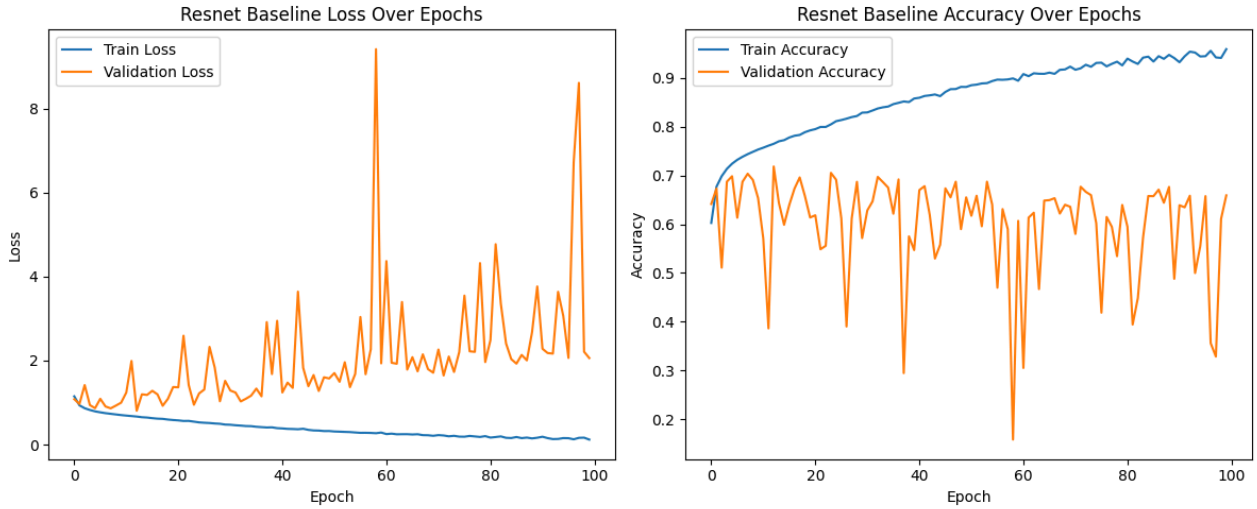

Figure 8. ResNet implement accuracy and loss plot

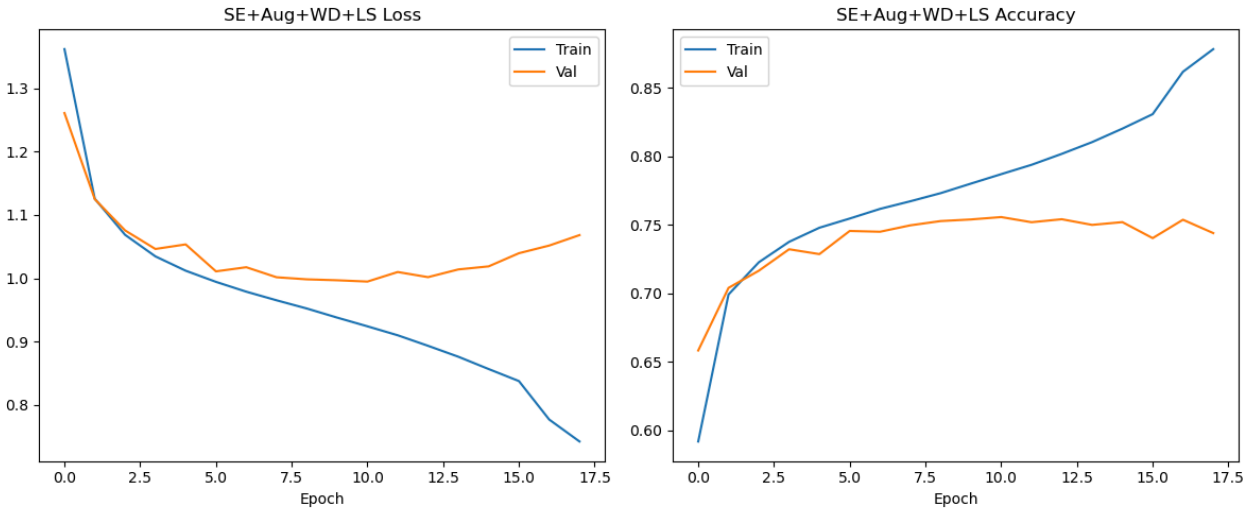

Table 2. Loss and Accuracy Comparison Across All Models

MODEL/DATA	VAL LOSS	VAL ACCURACY	TEST LOSS	TEST ACCURACY
CNN Baseline	**1.35**	**62.98%**	**1.35**	**62.98%**
CNN Improvement	**0.75**	**77.01%**	**0.63**	**75.14%**
RenNet Baseline	**2.06**	**65.91%**	**2.05**	**66.39%**
RenNet Improvement	**1.01**	**74.49%**	**1.02**	**75.28%**

Table 3. Test Set Per-Class Accuracy (Confusion Matrix Diagonals)

Model/Emotion	0 Neutral	1 Happiness	2 Sadness	3 Surprise	4 Fear	5 Disgust	6 Anger	7 Contempt
CNN Baseline	**0.31**	**0.92**	**0.78**	**0.2**	**0.32**	**0.09**	**0.36**	**0**
CNN Improvement	**0.76**	**0.89**	**0.46**	**0.35**	**0.27**	**0.17**	**0.51**	**0.04**
RenNet Baseline	**0.54**	**0.89**	**0.45**	**0.3**	**0.42**	**0.24**	**0.47**	**0.07**
RenNet Improvement	**0.71**	**0.93**	**0.55**	**0.53**	**0.26**	**0.19**	**0.59**	**0.02**

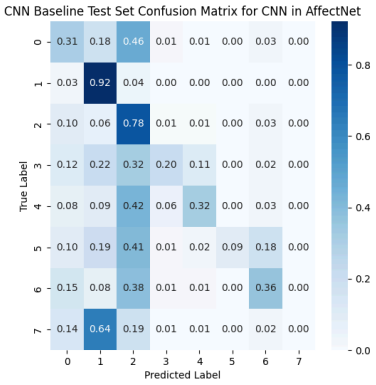

Figure 9 CNN baseline Test Set confusion matrix

6. Results

6.1 CNN Baseline Result

Our CNN baseline model was trained for 100 epochs with a batch size of 64 and a learning rate of 0.001, using SGD with momentum. Model details have been explained in the approach section.

In our CNN baseline, Training accuracy steadily increased and reached around **0.85**, while training loss dropped steadily below **0.4**, indicating effective learning on the training set. Validation accuracy peaked at **0.74**, and although it showed some fluctuations in later epochs, both validation accuracy and loss remained relatively controlled — with

accuracy never dropping below **0.6** and loss never exceeding **1.5** (see Figures 5), suggesting the model maintained decent generalization.

On the test set, the CNN baseline model achieved an accuracy of **62.98%** with a loss of **1.35**. As shown in the confusion matrix (Figure 9 and table 3), the CNN baseline performs well on common emotion categories like Happiness (1) and Sadness (2), with diagonal values reaching 0.92 and 0.78 respectively. However, it struggles more on less frequent or visually subtle emotions like Fear (4), Disgust (5), and Contempt (7), which are often confused with Neutral or Happiness. This suggests that while the CNN Baseline model handles dominant classes reliably, it has limitations in fine-grained emotion recognition.

6.2 CNN Improvement Result

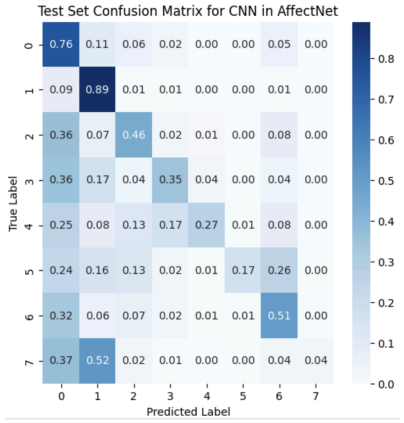

Figure 10 CNN implement Test Set confusion matrix

For CNN improvement, we performed a grid search over batch sizes of 64, 128, and 256 and learning rates of 0.001, 0.005, and 0.01 to find the optimal hyperparameters.The final model was trained with a batch size of **256** and a learning rate of **0.01** with 100 epoches, selected based on validation performance. Reported metrics are averaged over three random seeds for stability and reproducibility. Model details have been explained in the approach section.

As shown in Figure 6, the CNN improvement model exhibited smoother learning behavior. Training accuracy steadily increased and reached around **0.77**, while training loss decreased and converged to around **0.6**. Validation accuracy plateaued at approximately **0.75**, and validation loss remained stable around **0.7** throughout training. Compared to the CNN baseline, the improved model achieved **lower validation loss**, **higher validation accuracy**, and **more stable optimization curves**, suggesting better convergence and generalization. These results indicate that the improved model delivers more reliable prediction performance.

On the test set, the CNN improvement model achieved an accuracy of **75.14**% with a loss of **0.63**. As shown in the confusion matrix (Figure 10 and table 3), the model performs more robustly across emotion categories. It maintains strong diagonal values on dominant classes like Happiness (1) at 0.89 and Neutral (0) at 0.76, while showing improvements on previously weaker categories. Notably, compared to lower scores in the CNN baseline, Surprise (3) improved to 0.35, Disgust (5) to 0.17, and Anger (6) to 0.51. Although Contempt (7) remains difficult to distinguish (0.04), the confusion across visually similar emotions is significantly reduced. These results demonstrate that CNN improvement model has better balance across classes and stronger fine-grained emotion recognition.

6.3 ResNet Baseline Result

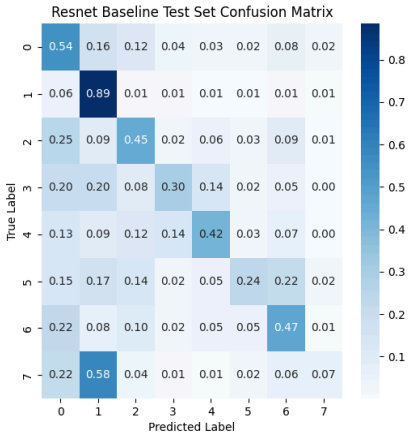

Figure 11 CNN baseline Test Set confusion matrix

Our ResNet baseline model was trained for 100 epochs with a batch size of 64 and a learning rate of 0.001, using SGD with momentum. Model details are provided in the *Approach* section.

In our ResNet baseline, the training accuracy gradually increased and eventually reached around **0.94**, while the training loss steadily decreased to below 0.1, indicating effective fitting on the training set. However, validation performance showed large fluctuations: the validation loss spiked multiple times above **5.0**, and the validation accuracy fluctuated significantly with an overall downward trend in the later stages (see Figure 7). These patterns suggest that the model may suffer from overfitting.

On the test set, the model achieved an accuracy of **66.39%** with a loss of **2.05**. In the confusion matrix (Figure 11 and table 3), the diagonal value for

Happiness (1) remains strong at 0.89, similar to the CNN baseline. Most categories showed better performance than CNN baseline: for example, Neutral (0) and Anger (6) reached 0.54 and 0.47, both higher than in the CNN baseline. Only Sadness(2) shows a poor performance compared to CNN baseline. Overall, the ResNet baseline model demonstrates stronger learning capacity than CNN baseline.

6.4 ResNet Improvement Result

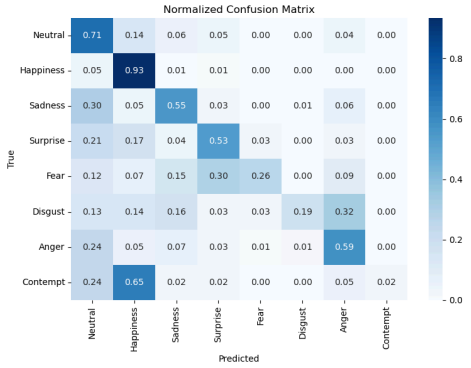

Figure 12 ResNet implement Test Set confusion matrix

For ResNet improvement, we trained a ResNet-50 using 224×224 RGB inputs, with SE blocks, label smoothing (ε = 0.1), and data augmentation (horizontal flip with 15% margin crop). The model was trained for 50 epochs with a batch size of **64**, a fixed learning rate of **1e-3**, and weight decay of **1e-5** using the AdamW optimizer. We applied ReduceLROnPlateau (factor = 0.5, patience = 5) and EarlyStopping (patience = 7), and restored the best weights based on validation loss. More model details have been explained in the approach section.

As shown in Figure 8, the ResNet improvement model showed smoother and more reliable training behavior, with **lower validation loss, higher validation accuracy, and more stable optimization curves** compared to the baseline. Training accuracy steadily increased to around **0.86**, while training loss decreased consistently to below **0.8**. Validation accuracy plateaued near 0.75 with minimal fluctuation, and validation loss remained stable around 1.0. Throughout training, validation accuracy never dropped below **0.65**, and validation loss stayed under **1.5** — suggesting effective regularization and reduced overfitting.

On the test set, the ResNet improvement model achieved an accuracy of **75.28%** with a loss of **1.02**. As shown in the confusion matrix (Figure 12 and table 3), the model performs more robustly across emotion categories. It maintains strong diagonal values for **Happiness (0.93)** and **Neutral (0.71)**, and also shows clear improvements on more challenging emotions compared to the ResNet baseline — such as **Sadness (from 0.45 → 0.55)**, **Surprise (0.30 → 0.53)**, and **Anger (0.47 → 0.59)**. Overall, the improved model demonstrates a more balanced performance across all eight emotion classes and stronger fine-grained recognition ability compared to the ResNet baseline.

6.5 Brief summarize

Across all four models, the improved versions of CNN and ResNet both showed clear improvements over their baselines. As shown in Table 2, both models achieved over 75% test accuracy, with lower validation loss, more steadily curve and fewer signs of overfitting. Table 3 also highlights better accuracy on harder emotions like Surprise, Disgust, and Anger, suggesting stronger fine-grained recognition. These results confirm that our model and training adjustments were effective.

7.additional topic:GAN

In our expression recognition task, we additionally introduce a conditional GAN to help the model better understand how emotions appear in facial features. Instead of generating images for their own sake, the GAN teaches the model to transform neutral faces into expressive ones—highlighting emotion-specific changes while ignoring identity. This leads to more robust and generalizable classification.

Here is a detailed explanation of how we build our GAN model . Given a neutral image I_N and an expression image I_E, the **generator G** produces a transferred image $\hat{I}=G(I_N,I_E)$. Instead of relying on paired data, we randomly sample I_N and I_E from different identities, forcing the model to focus on emotional content rather than facial identity.

A patch-based discriminator D evaluates the realism of $\hat{I}$, and training is guided by three losses:

- Adversarial loss (PatchGAN) :

  $\mathbf{L_{adv} = MSE(D(G(I_N,I_E)), 1)}$

- Feature matching loss (from intermediate layers of D):

  $\mathbf{L_{fm} = L1(Features\_D(I_E), Features\_D(G(I_N,I_E)))}$

- Expression classification loss :
  $\mathbf{L_{expr} = CrossEntropy(Classifier(G(I_N,I_E)), 1)}$

The total generator loss is a weighted sum, where λ_{adv} is 1,λ_{expr} is 20, λ_{fm} is 10:

$$L_G = \lambda_{adv} * L_{adv} + \lambda_{expr} * L_{expr} + \lambda_{fm} * L_{fm}$$

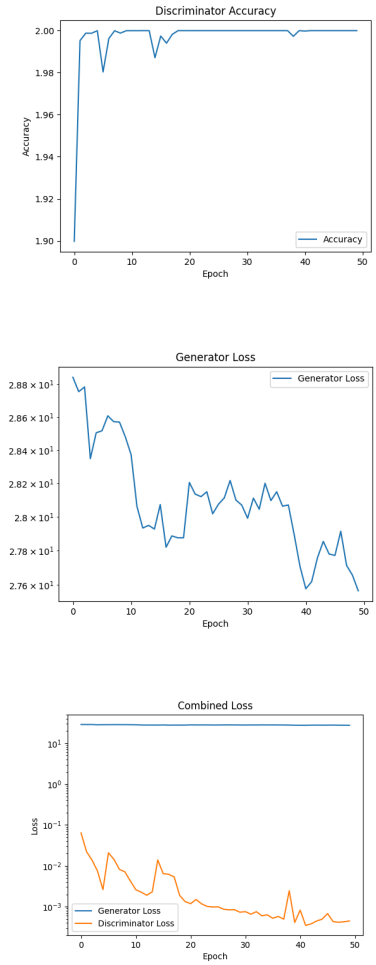

Figure 13-15 GAN Discriminator Accuracy(up), Generator Loss(middle), Combined Loss(down)

Figure 16 Reference Image, Expression Image, and Inference Image (left to right)

As shown in Figure 13, the discriminator accuracy rapidly climbs above 98% within the first 10 epochs and stabilizes near 100%, indicating strong real-vs-fake separation. Over 50 epochs, the generator loss steadily decreases on a log scale (from ≈10 down to ≈1, see Figure 14), while the discriminator loss fluctuates but trends downward (from 10^{-1} to ≈10^{-4},see Figure 15), confirming both networks' convergence.

Qualitatively (see Figure 16), the GAN produces visually coherent transfers: neutral inputs acquire the intended expressions (e.g. happiness, anger) with preserved facial structure and minimal artifacts, demonstrating the model's ability to generalize expression mapping across identities.

8.Conclusion

In this project, we explored both CNN and ResNet architectures for facial expression recognition. CNN offered a simpler, faster structure that performed reliably on common emotions like happiness and neutrality. In contrast, ResNet—with its residual connections—demonstrated stronger feature extraction power, especially for subtle or low-frequency emotions, but also required careful tuning to avoid overfitting.

Throughout the process, we learned that building an effective FER system is not just about using deeper models. There's a tradeoff between architectural complexity and training stability. Techniques like label smoothing, data augmentation, and balanced loss handling turned out to be just as important as the model itself.

We also experimented with a conditional GAN that transforms neutral faces into expressive ones. While still in the early stages, this provided us with new insight into the facial structure of emotions and opened possibilities for future data augmentation or generative learning.

In summary, a robust FER system results from the combined power of model architecture, training strategy, and deep understanding of data—not any single technique alone.

Appendix A. Detailed Roles

See table 4 Below. Note: Some tasks we worked on are not included in this report, as we only presented the most meaningful and representative methods.

Appendix B. Code repository

GitHub code repository:
https://github.com/jyz0328/Facial-Expression-Recognition-
Google Drive link:
https://drive.google.com/drive/u/1/folders/1MdR9WIQ8Kbv_Aq7Bg2KntPREfKzp0Utv

References

1) The Definitive Guide to Reading Microexpressions (FacialExpressions)

https://www.scienceofpeople.com/microexpressions/
2) Wang, Z. J., Turko, R., Shaikh, O., Park, H., Das, N., Hohman, F., Kahng, M., & Chau, D. H. (2020). CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization. . https://arxiv.org/abs/2004.15004
3) Hasani, B., Negi, P. S., & Mahoor, M. H. (2020). BReG-NeXt: Facial Affect Computing Using Adaptive Residual Networks With Bounded Gradient.. https://arxiv.org/abs/2004.08495
4) arXiv:2004.11823[cs.CV] https://arxiv.org/abs/2004.11823
5) Ali Mollahosseini, Behzad Hasani, and Mohammad H. Mahoor, "AffectNet: A New Database for Facial Expression, Valence, and Arousal Computation in the Wild", IEEE Transactions on Affective Computing, 2017. https://www.mohammadmahoor.com/pages/databases/affectnet/

Table 4. Team member contributions

Name	**Task**	**File names**	**No. Lines of Code**
Jingyi Zhang	1.Build ResNet baseline for FER2013 and AffectNet 2.Build VGG baseline for FER2013 and AffectNet 3.Modify and implement CNN, ResNet, and VGG baselines to unify evaluation metrics and extend training epochs. 4.Wrote report and prepare presentations 5. Build readme for project	ResNet_FER2013.ipynb, VGG_FER2013.ipynb, ResNet_Baseline_AffectNet.ipynb. VGG_Baseline_AffectNet.ipynb.	~700
Haoxuan Ruan	1.Build Data preprocessing code for FER2013. 2.Build CNN baseline for FER2013 and AffectNet 3.Build and implement CNN model improvements 4.Wrote report and prepare presentations	FERPreporcess.ipynb, CNN-fer2013.ipynb, CNN_Baseline_AffectNet.ipynb. CNN_Affectnet_hypertune.py	~700
Andrew Nguyen	1.Build Data preprocessing code for AffectNet. 2.Modify CNN, ResNet, and VGG Baselines to make it fit with Affectnet 3.Build and implement ResNet model improvements 4.Wrote report and prepare presentations	affectnetCSV.py, Final_resnet.py, Process_affectnet.py, make_bbox_csv.py	~700
Adam Mhal	1.evaluation code for dataset Y 2.Build and evaluate GAN model 3.Wrote report and prepare presentations	GAN_FER2013.py, transformEmotionsSCCAffectNet.py, transformEmotionsSCC.sh, GAN_FER2013.ipynb	~700