# Facial Expression Recognition Competition Report

Manh Ha Nguyen

Le Thuy An Phan

Computer Vision - COMP SCI 3315

The University of Adelaide

# 1. Introduction and Understanding of the Task and Baseline Model

## 1.1 Overview

Facial expression recognition is a challenging yet essential task in computer vision, with applications spanning psychological research, human-computer interaction, and intelligent surveillance. In this report, our goal is to accurately classify facial expressions into seven distinct emotion categories using the FER2013 dataset. This dataset contains 48x48 grayscale facial images collected "in-the-wild," meaning the images exhibit natural variations in pose, illumination, and expression, closely representing real-world scenarios and adding complexity to the recognition task [1].

## 1.2 Understanding of the Baseline Model

A baseline implementation was supplied to anchor both methodological understanding and quantitative comparison. The system is a deliberately compact **seven-layer convolutional neural network (CNN)** trained with a minimal recipe [2]. Its structure is summarised below.

- **Convolutional feature extractor.** Three successive *{Conv → ReLU → MaxPool}* blocks learn an increasingly abstract hierarchy of visual primitives: from edges and local textures to higher-order facial structures.
- **Flatten stage.** The 3-D activation tensor produced by the final pooling layer is reshaped into a single 1-D feature vector, enabling dense processing.
- **Fully-connected classifier.** The flattened representation traverses **five** linear layers, each separated by a ReLU non-linearity, to map the learned features onto a compact latent space.
- **Output layer.** A final linear transform **(144, 7)** yields class logits for the seven emotion categories.

Training is performed for **10 epochs** using the Adam optimiser (learning rate of 1×10−3, default β-parameters). This baseline attains a **49.68 %** test accuracy (*val_acc = 0.4968*) while requiring **0.3275 GFLOPs**, corresponding to an efficiency score (accuracy / FLOPs) of **1.52**. These figures constitute the benchmark against which all subsequent enhancements are evaluated.

## 1.3 Introduction to the approach solution

While the baseline model serves as a critical reference point, and its accuracy of nearly 50% is a reasonable starting performance, it leaves significant room for improvement. Our primary goal was to improve both accuracy and efficiency. To achieve this, our approach involved exploring advanced techniques such as using pretrained models (e.g, EfficientNet b0, MobileNetV3 Large), reimplementation of SOTA models (e.g, ResEmoteNet, VGG19), implementing advanced data augmentation, training strategies and employing ensemble models [5][6][7]. These approaches are expected to help the model generalise better, reduce overfitting and achieve higher performance.

The table below summarizes all the models we implemented and tested to identify the best-performing architectures. Additionally, we employed ensemble methods to leverage the strengths of individual models and achieve optimal performance across training stages.

| Model name | Abalations<br><br>(Compared to Baseline) | Accuracy | Gflops | Efficiency (Acc/Gflops) |
|---|---|---|---|---|
| Baseline | - | 0.49675 | 0.32751 | 1.51678 |
| ResEmoteNet | Xavier initialization + AdamW + model checkpoint | 0.61365 | 0.24009 | 2.55596 |
| VGG19 (modified) | Xavier initialization + AdamW + model checkpoint | 0.62006 | 0.91341 | 0.67884 |
| EfficientNet b1 | Xavier initialization + AdamW + model checkpoint + parameter-wise weight-decay split + OneCycleLR + GradScaler | 0.56168 | 0.03544 | 15.84861 |
| MobileNet V3 Large | Xavier initialization + AdamW + model checkpoint + parameter-wise weight-decay split + OneCycleLR + GradScaler | 0.53548 | 0.01445 | 37.06665 |
| EfficientNetV2 S | Initialized from imagenet1k pretrained + AdamW+ model checkpoint + parameter-wise weight-decay split + OneCycleLR + Multi round training | 0.62929 | 0.15617 | 4.02954 |
| EfficientNet B0 (Standard)<br><br>(A) | Initialized from imagenet1k pretrained + AdamW+ model checkpoint + parameter-wise weight-decay split + OneCycleLR/RLOP + Multi round training + Focal Loss | 0.62250 | 0.02321 | 26.81963 |
| EfficientNet B0 (Crop Rotate Flip Train Dataset) | Initialized from imagenet1k pretrained + AdamW + model checkpoint + | 0.66658 | 0.02321 | 28.71854 |

| | | | | |
|---|---|---|---|---|
| (B) | parameter-wise weight-decay split + OneCycleLR/RLOP + Label Smoothing + Multi round training + Crop Rotate Flip Augmented Train Dataset | | | |
| EfficientNet B0 (Crop Rotate Flip Random Brightness Contrast)<br><br>(C) | Initialized from imagenet1k pretrained + AdamW + model checkpoint + parameter-wise weight-decay split + OneCycleLR/RLOP + Multi round training + Crop Rotate Flip with Random Brightness and Constrast Augmented Train Dataset | 0.64514 | 0.02321 | 27.79509 |
| EfficientNet B0 (Crop Rotate Flip Random Cut Out)<br><br>(D) | Initialized from imagenet1k pretrained + AdamW + model checkpoint + parameter-wise weight-decay split + OneCycleLR/RLOP + Multi round training + Crop Rotate Flip with Random Cut Out Augmented Train Dataset | 0.65216 | 0.02321 | 28.09729 |
| EfficientNet B0 (MixUp CutMix)<br><br>(E) | Initialized from imagenet1k pretrained + AdamW + model checkpoint + parameter-wise weight-decay split + OneCycleLR/RLOP + Multi round training + Label Smoothing + MixUp Augmented Train Dataset | 0.64685 | 0.02321 | 27.86866 |
| Best Ensemble Model | Combined all EfficientNetB0 models [A, B, C, D, E] using weighted average probability with weight | 0.69880 | 0.11605 | 6.02138 |

| | | | | |
|---|---|---|---|---|
| | accordingly [0.1, 0.3, 0.1, 0.2, 0.3] | | | |
| Submitted Ensemble Model Result for Competition | Combined model [B, E] using weighted average probability with weight accordingly [0.45, 0.55] | 0.68738 | 0.04642 | 14.80736 |

# 2. Description of System Modifications for Performance Enhancement

To significantly improve upon the baseline model, we adopted a multifaceted approach emphasizing robust architecture, advanced data augmentation techniques, refined training strategies, and ensemble methods. Our primary objective was to simultaneously enhance both accuracy and computational efficiency.

## 2.1 Model Architecture: EfficientNet B0

The EfficientNet-B0 model is pretrained on the large-scale ImageNet-1k dataset, and serves as a powerful feature extractor. We adapted it for the FER2013 dataset by making two crucial modifications:

- **Input layer adjustment:** The original EfficientNet-B0 expects a 3-channel (RGB) input. As our FER2013 dataset consists of grayscale images, we replaced the initial convolutional layer's input channels from 3 to 1 (grayscale) [5]. Doing this, we can correctly process our 48x48 grayscale images.
- **Classifier Head Replacement:** The final classification head was replaced with a new nn.Linear layer to output 7 classes (according to the 7 emotion categories in the FER2013 dataset). This ensures the model's output is aligned with our specific task.

This architectural shift provided a strong foundation, from which we benefit from the learned features on a vast dataset, yet can tailor to our specific input and output requirements.

## 2.2 Advanced data augmentation

To enhance the model's generalisation capabilities and reduce overfitting, we use diverse data augmentation techniques. Beyond simple transformations, we explored more advanced methods to create a richer training dataset.

Two primary sets of augmentations we used for different model variants:

- **Standard transformation (for "Crop, Rotate, Flip" model):** for one branch of our EfficientNet-B0 training, we applied:
    - Random horizontal flip: with p=0.5: mirrors images horizontally
    - Random crop: with padding=4: simulate variations in facial positioning within the image.
    - Random rotation: degree=10: introduce slight rotational variances.

    - Grayscale conversion and normalisation: standard preprocessing steps to match input requirements and scale pixel values.

- **MixUp Augmentation (for "MixUp" model):** for another branch, we used MixUp, a technique that linearly combines 2 training examples and their labels. This encourages the model to predict smoothly between classes and improves robustness. MixUp implementation we used:

    - Alpha parameter *(MIXUP_ALPHA=0.4)*: control the strengths of interpolation between samples, drawn from a Beta distribution.

    - Combined inputs and labels: the input image was a weighted sum of 2 images, and the loss was a weighted sum of losses from their respective labels.

    - Simpler base transforms: We intentionally kept the base augmentations simpler (only *RandomHorizontalFlip* and *Resize*).

## 2.3 Optimised training strategies

Beyond the architecture and data, our training methodology was significantly refined from the baseline to achieve optimal performance and robust convergence.

- **Multi-stage progressive training:** Instead of a single training run, we implemented a multi-stage fine-tuning approach. Each stage loaded the best-performing checkpoint from the previous stage and continued training with adjusted hyperparameters. This allowed for gradual optimisation, moving from broader learning rates to more precise tuning. For instance, the "Crop, Rotate, Flip" model went through three such stages, and the "MixUp" model underwent two. This strategy allowed us to progressively maximise each model's potential.

- **Optimiser choice: AdamW:** we moved from the baseline's Adam optimiser to AdamW. AdamW decouples weight decay from the L2 regularisation in the optimiser, which often leads to better generalisation by preventing overfitting more effectively, especially in deep learning models. We also implemented a parameter-wise weight decay split, skipping weight decay for biases and normalisation layers for further optimisation.

- **Learning rate schedulers:** Instead of a fixed learning rate with the baseline, we experimented with different learning rate scheduling techniques:

    - **OneCycleLR**: used in the initial training stages, this scheduler dynamically adjusts the learning rate throughout an epoch in a cyclical manner, often helping faster convergence.

    - **ReduceLROnPlateau (RLOP)**: applied in later fine-tuning stages, RLOP monitors a performance metric, such as validation accuracy, and reduces the learning rate when the metrics stop improving. This helps the model settle into a good local minimum.

- **Loss functions:** We incorporated advanced loss functions to guide the training more effectively:

    - **Label smoothing**: beyond using standard Cross-Entropy Loss, we applied Label Smoothing. This technique prevents models from becoming overly confident in their predictions by slightly regularising the one-hot encoded labels. This generally improves generalisation.

    - **Focal loss**: In some later stages of training, the default EfficientNet-B0, we explored Focal Loss (with $\gamma$=2.5 and class weights). This loss function helps to address class imbalance by

down-weighting the loss contribution of well-classified examples. And hence, training the harder, misclassified examples was more focused on.

- **Gradient clipping**: We consistently applied gradient clipping (e.g., grad_clip_norm = 2.0 down to 0.5) to prevent exploding gradients, especially during the initial training phases and also when experimenting with aggressive learning rates.

### 2.4 Ensemble Modeling for Peak performance

We recognise that individual models capture different aspects of data, and we leverage ensemble methods to combine their strengths and achieve a higher overall accuracy.

- **Reciprocal Rank Fusion (RRF):** We initially explored RRF, a method commonly used in information retrieval, which aggregates ranked lists based on their inverse ranks. For an ensemble of two models (*model_efficientnetb0_v2_tuned_2.pth and model_efficientnetb0_v5_tuned.pth*), RFF achieved a best test accuracy of **0.6773** with best weight of **[0.55, 0.45]**. However, it was outperformed by a simpler yet more effective approach.
- **Weighted Average Probabilities (WAP):** This method is crucial for our task. It combines the predicted class probabilities (logits passed through softmax) from multiple models by taking a weighted average. We performed a grid search to find the optimal weights for each model in the ensemble.

For our submitted results, we employs a *weighted-average probability* (WAP) fusion of two high-performing EfficientNet-B0 variants:

| **Variant** | **Core augmentations** | **Single-model accuracy** |
|---|---|---|
| EfficientNetB0-CRF | Random Crop + Rotate + Horizontal Flip | 0.6666 |
| EfficientNetB0-MixUp | MixUp (α = 0.4) with minimal base transforms | 0.6469 |

A grid search over weighting coefficients identified the optimal vector **w = [0.45, 0.55]**, assigning the larger share to the MixUp model. The resulting ensemble achieves **68.74 % accuracy** while maintaining an aggregate computational budget of **≈ 0.046 GFLOPs** (2 × 0.0232 GFLOPs), thus preserving the efficiency advantages of the underlying lightweight architecture.

These modifications from architectural choices and complex data handling to training and intelligent model combination contributed to a substantial improvement in facial expression recognition performance compared to the baseline.

## 3. Methods for reducing Computational Cost and Improving Accuracy-Cost Tradeoff

Achieving an optimal balance between recognition accuracy and computational expenditure was a central theme throughout this study. Although deeper, more expressive networks can yield higher accuracies, their elevated FLOP counts and memory footprints often render them unsuitable for deployment in resource-constrained settings. Consequently, each design choice was evaluated not only for raw performance but also for its impact on efficiency.

## 3.1 EfficientNet-B0 Adaptations for Cost Reduction

Our decision to use the EfficientNet-B0 architecture, as detailed in Section 2.1, contributed to cost reduction. By adapting this pretrained model for FER2013 dataset, we introduced key optimisation that directly minimises computational overhead:

- **Optimised input handling:** the conversion of input images to 48x48 pixels (original is 224x224) and to 1-channel grayscale directly reduces the data volume processed at each layer. This significantly cut down the number of mathematical operations the model performs, resulting in substantial computational savings compared to typical 224x224 RGB inputs.
- **Streamlined classification head:** replacing the large 1000-class ImageNet classifier with a compact 7-class linear layer reduces the overall parameter count and simplifies the final inference step, further contributing to efficiency.
- **Mixed-precision training (AMP)**: Forward and backward passes run inside torch.cuda.amp.autocast with a GradScaler. Although this does not alter the theoretical FLOPs reported by fvcore, it cuts memory traffic and speeds execution on modern GPUs.

These fundamental adaptations ensured that our chosen model was computationally lean for our specific task, even before considering advanced training strategies.

## 3.2 Strategic Model Selection and Ensemble for Efficiency

Our strategy was heavily influenced by observations regarding model size and performance:

- **Balancing depth and learning capacity:** Custom re-implementations of ResEmoteNet and VGG-19 delivered modest accuracy gains but demanded FLOP budgets 4-30 times larger than EfficientNet-B0. Given diminishing returns, we elected to treat these larger networks as auxiliary experiments rather than deployment candidates.
- **Focusing on compact architectures:** With judicious hyper-parameter tuning (One-Cycle LR scheduling, label smoothing, gradient clipping), a single EfficientNet-B0 variant exceeded 66 % test accuracy at a negligible cost. This finding motivated a “small-model majority” strategy.
- **Weighted-average ensembling.** Instead of relying on one monolithic model, we fused complementary EfficientNet-B0 variants via Weighted Average Probabilities (WAP). A grid search over weight vectors produced an optimal allocation [0.45, 0.55] for the *Crop–Rotate–Flip* and *MixUp* branches, respectively. The resulting ensemble attained **68.74 %** accuracy while consuming only **0.046 GFLOPs**—a compelling accuracy-to-cost ratio.
- **Early stopping and Reduce-on-Plateau.** Validation accuracy was monitored every epoch; training halted after 20 epochs without improvement, and the learning rate was decayed automatically when progress stalled. These safeguards curtailed needless GPU cycles.

## 3.3 Deliberate architectural choices

Efficiency considerations likewise influenced what we **did not** deploy:

- **Vision Transformers (ViTs):** Although state-of-the-art on large-scale benchmarks, ViTs incur quadratic self-attention costs and require high-resolution inputs—counter to the low-FLOP mandate of this competition [8].
- **Excessive weight decay on scale/bias parameters:** Biases and normalisation parameters were excluded from AdamW weight decay, improving convergence without additional compute.
- **Oversized data augmentations:** All geometric and photometric transforms (MixUp, CutMix, random crop/flip/rotate) were tuned to preserve the 48×48 canvas, avoiding costly up- or down-sampling operations.

# 4. Limitations and Conclusions

This project aimed to develop an efficient and accurate facial emotion recognition system. While significant advancements were made, it is important to acknowledge certain limitations and draw key conclusions from our methodology and results.

## 4.1 Limitations

- **State-of-the-Art (SOTA) Performance is not achieved:** while our system shows competitive performance, it did not reach the absolute highest reported actuaries in the academic literature for the FER2013 dataset. Achieving those numbers generally demands considerably larger backbones, extensive pre-training on auxiliary datasets, and long, finely tuned training schedules—all of which exceed our practical compute envelope [9].

- **Restricted external data usage:** to maintain the integrity of the competition, we explicitly ruled out any additional dataset (e.g., the relabelled FER+ corpus) [10]. Although such data could have alleviated label noise and boosted generalisation, its inclusion would have blurred the provenance of results and compromised the integrity of our ablations. Consequently, the attainable accuracy ceiling is lower than what might be possible with broader data.

- **Tight hardware budget.** All experiments were executed either on the free-tier Google Colab GPU allocation (volatile availability, 12 GB VRAM or less) or on a MacBook laptop using Apple's integrated M-series GPU via the Metal Performance Shaders (MPS) backend. These modest environments limited batch sizes, curtailed exploration of very deep or high-resolution architectures, and forced conservative hyper-parameter sweeps. Potentially fruitful, but compute-hungry, directions—such as larger EfficientNet variants, Vision Transformers, or exhaustive augmentation searches—were therefore omitted to keep training times and memory usage practical.

## 4.2 Conclusions

Despite the limitations listed above, our project successfully demonstrated a highly efficient and accurate approach to facial emotion recognition:

- **Superior single model efficiency:** our optimised EfficientNEt-B0 model (Crop, Roate, Flip variant) achieved 0.6666 accuracy with exceptional efficiency, and required only 0.0232 GFLOPs.

- **Effective ensemble for competition:** our two-model EfficientNEt-B0 ensemble delivered 0.6874 accuracy for the final result, maintaining a combined cost of just 0.0464 GFLOPs. This shows the power of ensembling efficient models for a strong accuracy-cost tradeoff.

- **Balance accuracy and cost:** Ultimately, we met our objective of balancing performance and computational expense. Our strategic adaptations—from EfficientNet-B0 customisation and advanced augmentation to multi-stage training and weighted ensembling—allowed us to achieve robust recognition within a practical computational budget.

In conclusion, our system offers a practical and efficient solution for FER-2013, demonstrating the viability of optimising smaller models and ensemble techniques over more resource-intensive approaches.

# 5. Availability of Resources

- Training was done on both MPS and CUDA, for loading a checkpoint, first load on CPU, and then move to CUDA.
- For all the model checkpoints, please access: https://tinyurl.com/CV3315-ManhHa-ThuyAn

# Bibliography


[1] I. J. Goodfellow *et al.*, "Challenges in representation learning: A report on three machine learning contests," *Neural Networks*, vol. 64, pp. 59–63, Apr. 2015, doi: https://doi.org/10.1016/j.neunet.2014.09.005.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: https://doi.org/10.1038/nature14539.

[3] A. K. Roy, K. H. Kumar, A. Sharma, A. Dey, and Ansari, "ResEmoteNet: Bridging Accuracy and Loss Reduction in Facial Emotion Recognition," *arXiv.org*, 2024. https://arxiv.org/abs/2409.10545

[4] K. Simonyan and A. Zisserman, "VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION," Apr. 2015. Available: https://arxiv.org/pdf/1409.1556

[5] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," Sep. 2020. Available: https://arxiv.org/pdf/1905.11946

[6] M. Tan and Q. V. Le, "EfficientNetV2: Smaller Models and Faster Training," *arXiv:2104.00298 [cs]*, Jun. 2021, Available: https://arxiv.org/abs/2104.00298

[7] A. Howard *et al.*, "Searching for MobileNetV3." Available: https://arxiv.org/pdf/1905.02244

[8] A. Dosovitskiy *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *arXiv:2010.11929 [cs]*, Oct. 2020, Available: https://arxiv.org/abs/2010.11929

[9] "Papers with Code - FER2013 Benchmark (Facial Expression Recognition)," *paperswithcode.com*. https://paperswithcode.com/sota/facial-expression-recognition-on-fer2013

[10] E. Barsoum, C. Zhang, C. C. Ferrer, and Z. Zhang, "Training Deep Networks for Facial Expression Recognition with Crowd-Sourced Label Distribution," *arXiv:1608.01041 [cs]*, Sep. 2016, Available: https://arxiv.org/abs/1608.01041