TINYFER-LITE: A SMALL-DATA AND LIGHTWEIGHT ATTENTION-BASED ARCHITECTURE FOR REAL-TIME FACIAL EMOTION RECOGNITION

Nouman Khalid
Independent Researcher
Peshawar, Pakistan
nomi.jr@yahoo.com

ABSTRACT

Facial emotion recognition is useful for affective computing, education, human–computer interaction, and assistive systems, but practical models must balance accuracy with model size, latency, and reproducibility. This paper presents TinyFER-Lite, a compact PyTorch pipeline for FER2013-style small and imbalanced facial-expression datasets. The model combines a MobileNetV3-Small feature extractor, Efficient Channel Attention, ImageNet normalization, conservative augmentation, class-weighted focal loss, mixed precision training, and deployment-focused evaluation. On FER2013, the best checkpoint selected by validation macro-F1 obtains 67.07% accuracy, 67.18% macro-F1, and 67.09% weighted-F1 on the held-out PrivateTest split. The trained model has 932,202 trainable parameters, an estimated 3.61 MB size, and measured single-image latency of 8.85 ms on CPU and 7.04 ms on an RTX 3050 GPU in the local evaluation environment. The project does not claim state-of-the-art performance; it provides a reproducible lightweight baseline and a clear foundation for future FER ablations.

***K*eywords** Facial emotion recognition · FER2013 · lightweight neural networks · Efficient Channel Attention · imbalanced learning · real-time inference

1 Introduction

Facial expressions provide visible cues that are often used in human communication. Automatic facial emotion recognition aims to classify facial images into categories such as angry, disgust, fear, happy, sad, surprise, and neutral. The task is relevant to interactive tutoring, driver monitoring, social robotics, assistive interfaces, and affective computing. It remains difficult because expressions are ambiguous, annotations are noisy, and visual conditions vary across lighting, pose, occlusion, resolution, and subject identity. A model that works only under clean benchmark conditions is therefore less useful than a model whose data handling, training choices, failure modes, and deployment cost are reported clearly.

Large convolutional and transformer-based networks can improve benchmark accuracy, but they are often expensive to train and harder to deploy. Small FER datasets add another constraint: high-capacity models can overfit quickly, while imbalanced classes can make accuracy look acceptable even when minority-class recall is poor. FER2013 is a useful example of this tension. The dataset is large enough to support deep learning experiments, yet it contains low-resolution images, uncertain annotations, and strong class imbalance. Macro-F1, per-class recall, model size, and inference latency are therefore important companions to accuracy.

TinyFER-Lite addresses these constraints with a compact pretrained backbone, a lightweight attention module, imbalance-aware optimization, and a complete evaluation pipeline. The first complete experiment uses FER2013, a widely used low-resolution FER benchmark introduced through the Challenges in Representation Learning competition [1]. The contribution of this work is a reproducible PyTorch FER pipeline, a MobileNetV3-Small plus Efficient Channel Attention model, training support for class weighting and focal loss, and deployment metrics reported alongside

predictive performance. This work does not claim state-of-the-art performance. Instead, it provides a compact and inspectable baseline for lightweight FER research and future controlled ablation studies.

This paper studies three practical questions. First, can a pretrained mobile backbone provide a useful FER2013 baseline without exceeding a small deployment budget? Second, can a simple attention block and imbalance-aware loss improve the fit to minority and ambiguous classes without turning the system into a heavy architecture? Third, can the reported experiment be reproduced end to end, including training curves, checkpoint selection, per-class metrics, confusion analysis, model size, and latency? These questions shape the paper more than leaderboard comparison does.

The project is designed for modest hardware and for research workflows that need to move from dataset loading to training, evaluation, and real-time inference without hidden assumptions. It supports FER2013-style CSV files and folder-based datasets, uses ImageNet-compatible preprocessing, records training curves and checkpoints, exports confusion matrices and per-class reports, and includes a webcam inference script. This makes the work useful both as a research starting point and as a teaching-friendly reference implementation.

The remainder of the paper follows the same practical emphasis. Section 2 summarizes related FER, lightweight backbone, attention, and imbalance-learning work. Section 3 defines the system model mathematically, so the design can be read as a complete computational flow rather than as a collection of implementation choices. Section 4 describes the implemented method, Section 5 gives the experimental setup, Section 6 reports the results, and the final sections discuss limitations and conclusions.

2 Related Work

FER2013 remains a common benchmark despite its limitations: images are low resolution, grayscale, noisy, and imbalanced [1, 2]. FER+ revised FER2013 annotations using crowd-sourced label distributions, emphasizing that expression labels often contain genuine ambiguity rather than a single perfectly correct category [3]. This is important for FER research because benchmark accuracy alone can conceal uncertainty in the target labels. CK+ and JAFFE are smaller controlled datasets used in small-data FER analysis [4, 5], while RAF-DB provides a larger real-world facial-expression benchmark with diverse annotations [6]. These datasets motivate a pipeline that can support both noisy in-the-wild data and smaller controlled data in later experiments.

Efficient visual backbones are natural candidates for real-time FER. MobileNetV3 was designed for hardware-aware mobile vision [7], and EfficientNet introduced compound scaling across depth, width, and resolution [8]. Prior lightweight FER studies have explored MobileNet, EfficientNet, RexNet, improved MobileNetV3, and disequilibrium-aware compact networks for resource-constrained expression recognition [9, 10, 11]. These works support the premise that FER accuracy should be studied together with computational cost, especially when models are intended for webcam or edge-style inference.

Attention is also useful for FER because emotional cues are often concentrated around the eyes, brows, and mouth. Deep-Emotion demonstrated attentional convolution for expression recognition [12]; Efficient Channel Attention models cross-channel interactions with minimal overhead [13]; and Coordinate Attention encodes positional information into mobile-friendly channel attention [14]. TinyFER-Lite uses Efficient Channel Attention because it is simple, parameter-light, and easy to insert after the final convolutional feature map. Focal loss is relevant under class imbalance because it down-weights easy examples [15], and knowledge distillation provides a path for compressing stronger teachers into smaller students [16]. The current experiment focuses on direct supervised training, while the repository includes distillation support for future work.

The gap addressed in this work is therefore not a new large-scale FER benchmark result. The gap is a compact and reproducible research scaffold that connects model design, small-data training behavior, and deployment cost in one place. That focus is important for students and practitioners who need a baseline that can be inspected, modified, and run on accessible hardware before larger ablation studies are attempted.

3 System Model

TinyFER-Lite is modeled as an end-to-end mapping from a raw facial image to a probability distribution over emotion classes. Let the FER dataset be

$$\mathcal{D} = \{(r_i, y_i)\}_{i=1}^{N}, \qquad y_i \in \{1, \dots, C\}, \tag{1}$$

where r_i is a raw grayscale or RGB face image, y_i is the emotion label, and $C = 7$ for angry, disgust, fear, happy, sad, surprise, and neutral. The system first applies a deterministic preprocessing function $T(\cdot)$ that resizes the input, converts grayscale images to three channels, scales pixel values, and normalizes them with ImageNet statistics:

$$x_i = T(r_i), \qquad x_i \in \mathbb{R}^{3\times 224\times 224}. \tag{2}$$

During training, an augmentation operator $\mathcal{A}(\cdot)$ is sampled for each image, producing $\tilde{x}_i = \mathcal{A}(x_i)$. During validation and testing, $\tilde{x}_i = x_i$, so the evaluation path remains deterministic.

The visual encoder is a MobileNetV3-Small feature extractor $F_\theta(\cdot)$ parameterized by θ. It maps the processed image to a compact convolutional representation

$$H_i = F_\theta(\tilde{x}_i), \qquad H_i \in \mathbb{R}^{K \times h \times w}, \tag{3}$$

where K is the number of feature channels and h, w are the spatial dimensions of the final feature map. The Efficient Channel Attention block computes a channel descriptor through global average pooling,

$$s_{i,c} = \frac{1}{hw} \sum_{u=1}^{h} \sum_{v=1}^{w} H_{i,c,u,v}, \tag{4}$$

then estimates a lightweight channel gate with a one-dimensional convolution and sigmoid activation:

$$a_i = \sigma\left(\mathrm{Conv1D}_k(s_i)\right), \qquad a_i \in [0,1]^K. \tag{5}$$

The attended representation is formed by channel-wise reweighting,

$$\bar{H}_{i,c,u,v} = a_{i,c} H_{i,c,u,v}. \tag{6}$$

This design keeps attention inexpensive because it avoids a full channel-by-channel matrix and uses only local cross-channel interaction.

The classifier converts the attended feature map into a vector representation using global average pooling,

$$g_i = \mathrm{GAP}(\bar{H}_i), \tag{7}$$

followed by batch normalization, dropout, and a linear classifier:

$$z_i = W\,\mathrm{Dropout}\left(\mathrm{BN}(g_i)\right) + b. \tag{8}$$

The predicted class probabilities are

$$p_i = \mathrm{Softmax}(z_i), \qquad \hat{y}_i = \arg\max_c p_{i,c}. \tag{9}$$

The complete inference system can therefore be written compactly as

$$\hat{y} = \arg\max_c \left[f_\Theta(T(r))\right]_c, \tag{10}$$

where f_Θ includes preprocessing-compatible feature extraction, ECA refinement, pooling, dropout, and classification parameters.

Training minimizes an imbalance-aware empirical risk over the training split. For class weight w_{y_i} and focal parameter γ, the primary objective is

$$\min_\Theta \frac{1}{N} \sum_{i=1}^{N} w_{y_i} \left(1 - p_{i,y_i}\right)^\gamma \left(-\log p_{i,y_i}\right). \tag{11}$$

This objective links the system design to the small-data setting: class weights compensate for label-frequency imbalance, while the focal term reduces the influence of easy examples that can dominate training. At deployment time, the same learned function is evaluated under a latency and memory budget. The deployed model is considered practical when its average single-image latency $L(f_\Theta)$ is below the target real-time budget and its parameter memory $M(f_\Theta)$ remains small enough for the intended device. In this study, those quantities are reported directly rather than inferred from architecture names.

4 Method

The pipeline begins with dataset-agnostic preprocessing. Images are resized to 224×224, and FER2013 grayscale inputs are converted to three-channel tensors so ImageNet-pretrained backbones can be used without changing the first convolution. Inputs are normalized with ImageNet mean and standard deviation. Training augmentation uses random resized crop, horizontal flip, small rotation, affine translation and scale, brightness/contrast jitter, and random erasing. The exact values are reported in the experimental setup. These transformations are intentionally moderate: they encourage robustness to crop, illumination, and mild pose variation without aggressively distorting facial expression cues. Validation and test transforms are deterministic.

The repository implements four comparison architectures in addition to TinyFER-Lite: a SimpleCNN baseline, MobileNetV2, MobileNetV3-Small, and EfficientNet-B0. These models share the same preprocessing and evaluation interface, which keeps later baseline training and ablation work straightforward. The current paper reports complexity for all implemented architectures and predictive performance for the completed TinyFER-Lite run.

TinyFER-Lite uses MobileNetV3-Small as a compact convolutional feature extractor. The final feature map is refined by Efficient Channel Attention, then passed through global average pooling, batch normalization, dropout, and a linear classifier. The attention block applies a lightweight channel gate after feature extraction, allowing the classifier to emphasize channels that are more relevant for expression discrimination. The classifier is deliberately small so that most representational capacity remains in the pretrained convolutional backbone. Figure 1 shows the full system flow, including training and evaluation components.

$$\hat{y} = \mathrm{Softmax}\left(W \cdot \mathrm{Dropout}\left(\mathrm{GAP}\left(A(F(x))\right)\right)\right), \tag{12}$$

where x is the input image, F is the MobileNetV3-Small backbone, A is the attention module, GAP is global average pooling, and W is the final classifier.

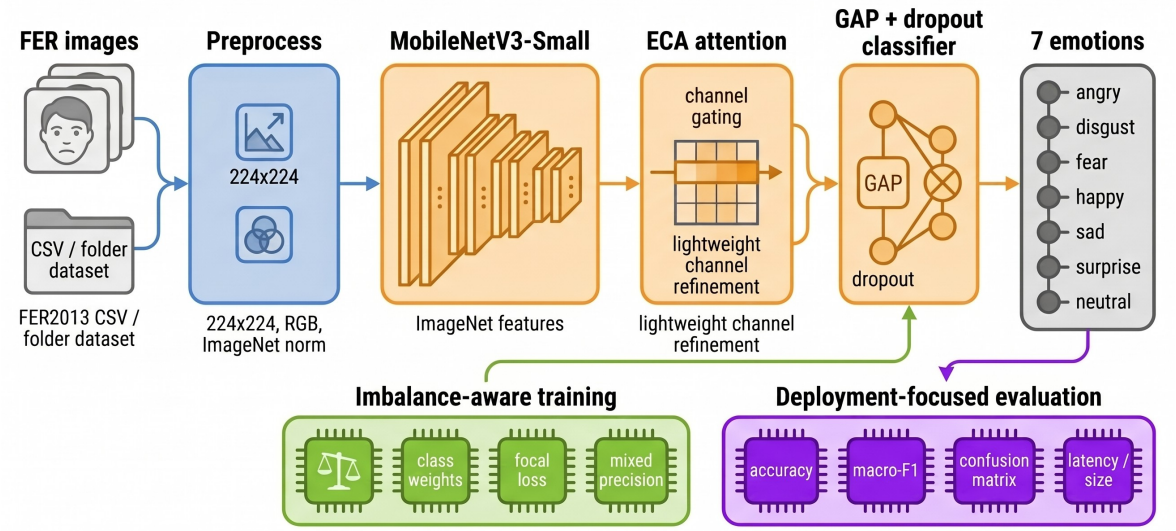

Figure 1: TinyFER-Lite system overview, including preprocessing, MobileNetV3-Small feature extraction, ECA refinement, imbalance-aware training, and deployment-focused evaluation.

The reported run uses class-weighted focal loss. For logits z, target y, focal parameter γ, and class weight w_y, the loss is

$$\mathcal{L}_{focal} = w_y(1 - p_y)^{\gamma}\left(-\log p_y\right), \tag{13}$$

where p_y is the predicted probability of the ground-truth class. The experiment uses $\gamma = 2$.

The implementation also supports standard cross entropy, class-weighted cross entropy, label smoothing, weighted sampling, ReduceLROnPlateau scheduling, cosine scheduling, early stopping, and mixed precision training. These options are exposed through command-line arguments so experiments can be reproduced without editing source files. Optional knowledge distillation is implemented through a separate training script that combines hard-label cross entropy with temperature-scaled KL divergence from a teacher model.

The design intentionally separates the research components from the deployment components. Dataset loaders handle CSV and folder-based inputs, model builders instantiate comparable architectures through a shared interface, and evaluation utilities compute predictive and deployment metrics from the same checkpoint. This separation helps prevent a common small-project problem: changing preprocessing, labels, or evaluation scripts between training and testing. The same class names, image size, normalization, checkpoint metadata, and attention setting are carried through the pipeline.

5 Experimental Setup

The experiment uses FER2013 with the standard Training, PublicTest, and PrivateTest splits. PublicTest is used for validation and checkpoint selection; PrivateTest is held out for final reporting. Table 1 gives the class distribution

and shows the severe minority-class imbalance for disgust. This imbalance is one reason macro-F1 is emphasized in addition to accuracy.

Table 1: FER2013 split distribution.

Emotion	Train	Public	Private	Total
Angry	3995	467	491	4953
Disgust	436	56	55	547
Fear	4097	496	528	5121
Happy	7215	895	879	8989
Sad	4830	653	594	6077
Surprise	3171	415	416	4002
Neutral	4965	607	626	6198
Total	28709	3589	3589	35887

Table 2 summarizes the main training and reproducibility details. The run was executed locally with CUDA-enabled PyTorch on an NVIDIA GeForce RTX 3050 GPU. The best checkpoint is selected by PublicTest macro-F1 rather than by final epoch or validation loss. This choice is aligned with the small-data setting: macro-F1 rewards balanced performance across the seven emotion classes and reduces the chance that the selected checkpoint is dominated by frequent classes.

Table 2: Training and reproducibility configuration.

Setting	Value
Input	RGB conversion when needed, 224×224 resolution, ImageNet normalization
Model	ImageNet-pretrained MobileNetV3-Small, ECA, GAP, BatchNorm, dropout 0.3, linear classifier
Objective	Class-weighted focal loss with $\gamma = 2$, no label smoothing, no weighted sampler
Optimizer	AdamW, learning rate 10^{-4}, weight decay 10^{-4}
Scheduler	ReduceLROnPlateau, factor 0.5, patience 3, monitored by validation macro-F1
Run	100 epochs, batch size 64, seed 42, early-stopping patience 100
Selection	Best PublicTest macro-F1; selected checkpoint at epoch 67
Software	Python 3.10.0, PyTorch 2.7.1+cu128, torchvision 0.22.1+cu128, CUDA runtime 12.8
Hardware	AMD Ryzen 5 3600 CPU; NVIDIA GeForce RTX 3050 GPU with 6144 MiB VRAM; NVIDIA driver 595.79
Artifacts	Best and last checkpoints, per-epoch metrics, training curves, PrivateTest report, confusion matrix, and complexity outputs

The exact augmentation settings were fixed in the training script: resize to 257 pixels before crop, random resized crop to 224×224 with scale $[0.85, 1.00]$ and aspect ratio $[0.90, 1.10]$, horizontal flip with probability 0.5, rotation within 10°, affine translation up to 5%, scale $[0.95, 1.05]$, shear of 5°, brightness and contrast jitter of 0.2, and random erasing with probability 0.2, area scale $[0.02, 0.12]$, and aspect ratio $[0.30, 3.30]$. Class weights are computed from the training labels by inverse frequency,

$$w_c = \frac{N_p}{C_p n_c}, \tag{14}$$

where n_c is the number of samples in class c, C_p is the number of classes present in the split, and N_p is the number of samples belonging to present classes. All seven FER2013 classes are present in this experiment. The executed training command used FER2013, TinyFER-Lite, ECA attention, focal loss, class weighting, 100 epochs, batch size 64, learning rate 10^{-4}, ImageNet pretraining, and CUDA mixed precision. The 100-epoch run took approximately 2 h 16 min according to local training-log timestamps.

All reported artifacts are written to the experiment results directory. The training run saves the best checkpoint, the last checkpoint, per-epoch metrics, and the training-curve figure. Evaluation reloads the best checkpoint and exports aggregate metrics, a per-class report, a confusion matrix, model-size estimates, and latency measurements. This separation keeps validation-based model selection distinct from held-out PrivateTest reporting.

Accuracy is reported as the fraction of correct predictions. Macro-F1 is the unweighted mean of per-class F1 scores and is used as the primary selection metric because it treats rare and frequent emotions equally. Weighted-F1 is also reported because it reflects the empirical class distribution of the held-out split. Latency is measured with a batch size of one using a synthetic $3 \times 224 \times 224$ tensor, ten warmup passes, and 50 timed forward passes. CUDA synchronization is used for GPU timing.

The experiment is reported as a single completed run rather than a multi-seed benchmark. This choice is stated explicitly because FER2013 results can vary with seed, augmentation, and checkpoint selection. The validation split is used only to select the checkpoint; the PrivateTest split is used once for the final reported performance. This keeps the reported PrivateTest metrics separate from model-selection decisions.

6 Results

Figure 2 shows the training dynamics over 100 epochs. The best validation checkpoint occurs at epoch 67, with PublicTest accuracy 65.84% and PublicTest macro-F1 65.76%. The final epoch reaches PublicTest accuracy 65.42% and macro-F1 65.44%. Validation loss rises after early improvements, suggesting overfitting even though macro-F1 remains relatively stable. The learning-rate scheduler reduces the learning rate over training, and the late-epoch curves show that the model has largely converged rather than continuing to gain generalization performance.

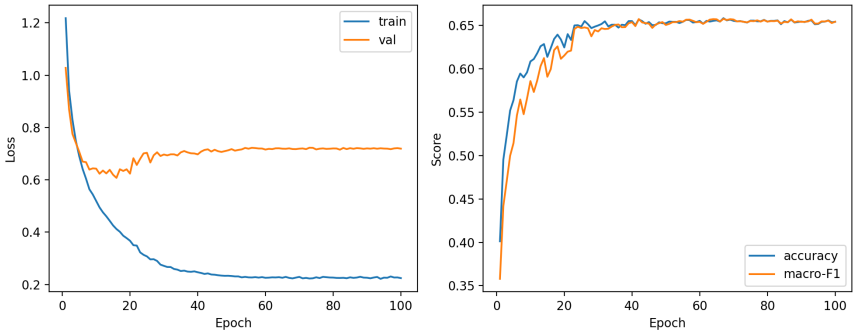

Figure 2: Training loss and PublicTest validation metrics across 100 epochs.

The best validation checkpoint is evaluated once on FER2013 PrivateTest. Table 3 reports aggregate performance and deployment metrics. Figure 3 shows class-level confusion, and Table 4 gives the detailed per-class report.

Table 3: PrivateTest performance and deployment metrics.

Metric	Value
Accuracy	0.6707
Macro-F1	0.6718
Weighted-F1	0.6709
Trainable parameters	932,202
Estimated size	3.61 MB
CPU latency	8.85 ms/image
GPU latency	7.04 ms/image

Table 4: PrivateTest per-class report.

Emotion	Precision	Recall	F1	Support
Angry	0.5660	0.6375	0.5996	491
Disgust	0.7586	0.8000	0.7788	55
Fear	0.5503	0.4659	0.5046	528
Happy	0.9103	0.8316	0.8692	879
Sad	0.5523	0.5152	0.5331	594
Surprise	0.7627	0.7957	0.7788	416
Neutral	0.5892	0.6965	0.6384	626

The held-out PrivateTest results show a close match between accuracy, macro-F1, and weighted-F1. This indicates that the trained checkpoint is not relying entirely on the most frequent classes, although the per-class table still reveals uneven difficulty. Happy and surprise obtain the strongest F1 scores. Fear and sad remain the weakest categories, which is consistent with FER2013 ambiguity and the visual overlap between negative or neutral expressions. Disgust obtains high F1 despite low support, but its support is small enough that this number should be interpreted cautiously.

The confusion matrix provides a complementary view of the same behavior. Happy is recognized strongly, while negative expressions show more cross-confusion. Fear is often confused with sad, angry, and neutral, reflecting the

fact that FER2013 images are low resolution and sometimes visually ambiguous. Sad also has a meaningful number of predictions assigned to neutral, which suggests that the model sometimes captures expression intensity but not the specific negative category. These errors are useful for future ablation design because they point to the classes most likely to benefit from better annotation, calibration, or face-region attention.

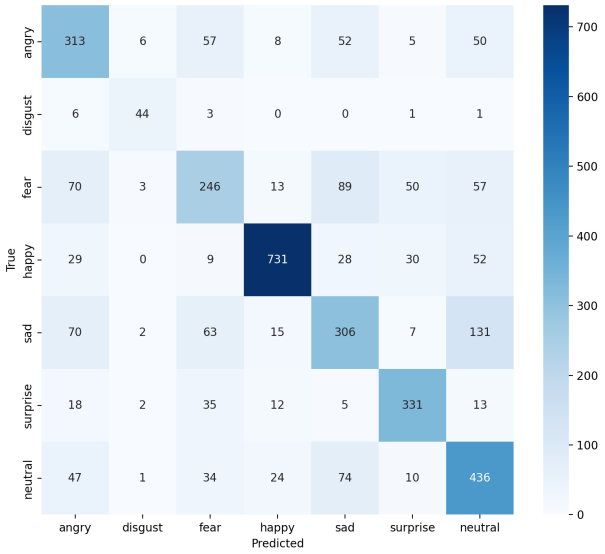

Figure 3: FER2013 PrivateTest confusion matrix.

Table 5 and Figure 4 compare the implemented architectures by parameter count, estimated size, and single-image latency. These measurements describe deployment cost only. The TinyFER-Lite row uses the evaluated checkpoint latency from Table 3; the untrained comparison architectures use the same benchmarking utility with their implemented model definitions. The baselines were not trained under the same 100-epoch FER2013 protocol in this run, so the paper does not report baseline accuracies. TinyFER-Lite has the smallest estimated model size among the implemented architectures and remains below one million trainable parameters.

Table 5: Implemented model complexity.

Model	Params	MB	CPU ms	GPU ms
SimpleCNN	1,175,015	4.49	29.72	2.55
MobileNetV2	2,232,839	8.65	18.97	8.08
MobileNetV3-Small	1,525,031	5.86	11.42	8.01
EfficientNet-B0	4,016,515	15.48	31.62	11.72
TinyFER-Lite	932,202	3.61	8.85	7.04

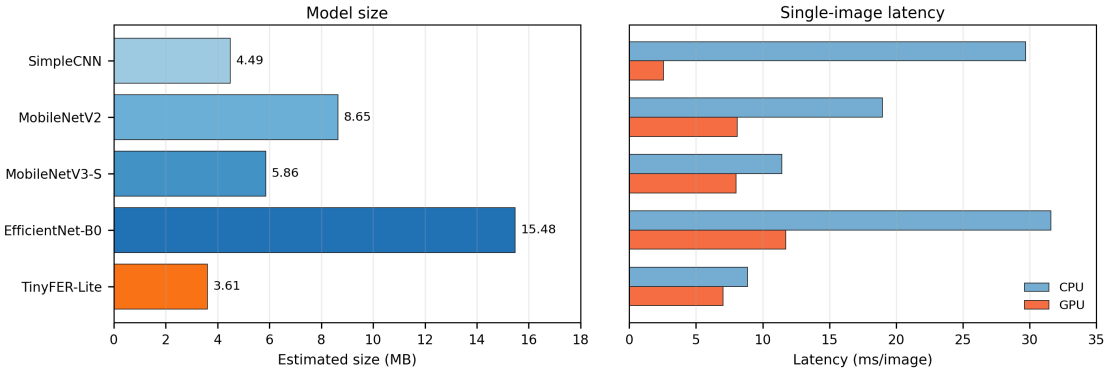

Figure 4: Size and latency comparison for implemented architectures.

7 Discussion

The PrivateTest accuracy of 67.07% and macro-F1 of 67.18% show that a small attention-based model can provide a useful FER2013 baseline with fewer than one million trainable parameters. The result is modest compared with heavily optimized FER systems, but it is meaningful for a transparent architecture and reproducible training script. It also shows why macro-F1 should accompany accuracy: class-level behavior is uneven, and the hardest categories matter beyond a leaderboard summary.

The model is deployment-friendly by size, with a 3.61 MB parameter-and-buffer estimate. The measured latency supports real-time webcam use locally, although batch-size-one GPU latency is affected by preprocessing overhead and CUDA launch costs. Latency should therefore be measured on the intended target rather than inferred from parameter count alone. TinyFER-Lite is the smallest implemented model, but not the fastest GPU model under synthetic batch-size-one timing, showing that small models can be limited by framework overhead as much as arithmetic cost.

The validation curves show overfitting pressure. Training loss decreases while validation loss rises after early epochs, and macro-F1 remains more stable than validation loss. The next necessary experiment is a controlled baseline suite that trains SimpleCNN, MobileNetV2, MobileNetV3-Small, EfficientNet-B0, and TinyFER-Lite under the same split, seed policy, augmentation, optimizer, epoch budget, and checkpoint-selection metric. That table should report accuracy, macro-F1, weighted-F1, parameter count, model size, CPU latency, and GPU latency for every trained model. Until then, the current complexity table is only a deployment-cost comparison.

The most informative ablation would isolate each design choice under identical conditions. A clean table should compare MobileNetV3-Small alone, MobileNetV3-Small with ECA, ECA with cross entropy, ECA with focal loss, ECA with class weighting, the full TinyFER-Lite recipe, and a distilled student if a teacher is trained. A class-frequency versus recall plot would also show the relationship between imbalance and errors more directly than the confusion matrix alone.

The system model clarifies where improvements should be attached. Noisy labels suggest label smoothing, soft labels, or distillation; weak localization suggests spatial-channel or coordinate attention; deployment limits require attention to preprocessing, batch-size-one execution, and runtime overhead. Reporting the system as a mathematical flow makes these intervention points explicit.

8 Limitations and Ethical Considerations

This paper reports one completed FER2013 training run for TinyFER-Lite. The implemented baselines are measured for complexity and latency, but their FER2013 accuracies are not reported because they were not trained under the same protocol. The study should therefore be read as a reproducible project report and a lightweight baseline, not as a full benchmark study. External validity on CK+, JAFFE, RAF-DB, and live webcam settings requires additional experiments. The latency numbers are also hardware- and software-dependent; they should be re-measured on the target machine, camera pipeline, and inference runtime.

Facial emotion recognition systems can be misused when predictions are interpreted as reliable evidence of a person's internal emotional state. Expressions are culturally and contextually dependent, and dataset labels are imperfect. TinyFER-Lite should be treated as an experimental image classifier, not as a diagnostic, psychological, surveillance, or identity system. Any real deployment should include consent, transparency, bias analysis, and clear limitations. If the model is used for demonstration, predictions should be presented as uncertain visual classifications rather than as conclusions about a person.

Another limitation is that the current system assumes a cropped or detected face image at inference time. Webcam inference depends on the quality of the face detector, camera lighting, subject pose, and preprocessing latency. These factors are outside the FER2013 classification benchmark but matter strongly in a live deployment. A production-facing version should therefore evaluate the full camera-to-prediction path rather than only the classifier forward pass.

9 Conclusion

TinyFER-Lite provides a reproducible lightweight FER pipeline with training, evaluation, distillation, and webcam inference. On FER2013 PrivateTest, the reported checkpoint reaches 67.07% accuracy and 67.18% macro-F1 with 932,202 trainable parameters and a 3.61 MB estimated model size. These results support compact pretrained backbones plus lightweight attention for small-data FER, while also showing that broader baseline training and cross-dataset evaluation are needed before stronger claims. The system model frames the pipeline as image preprocessing, lightweight feature extraction, channel attention, classification, imbalance-aware training, and deployment measurement, making

future ablations easier to define. TinyFER-Lite should therefore be viewed as a compact research platform for small-data FER rather than as a final closed model.

References

[1] Ian J. Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, Yingbo Zhou, Chetan Ramaiah, Fangxiang Feng, Ruifan Li, Xiaojie Wang, Dimitris Athanasakis, John Shawe-Taylor, Maxim Milakov, John Park, Radu Ionescu, Marius Popescu, Cristian Grozea, James Bergstra, Jingjing Xie, Lukasz Romaszko, Bing Xu, Zhang Chuang, and Yoshua Bengio. Challenges in representation learning: A report on three machine learning contests. *Neural Networks*, 64:59–63, 2015. https://doi.org/10.1016/j.neunet.2014.09.005.

[2] Yousif Khaireddin and Zhuofa Chen. Facial emotion recognition: State of the art performance on FER2013. *arXiv preprint arXiv:2105.03588*, 2021. https://arxiv.org/abs/2105.03588.

[3] Emad Barsoum, Cha Zhang, Cristian Canton Ferrer, and Zhengyou Zhang. Training deep networks for facial expression recognition with crowd-sourced label distribution. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, pages 279–283, 2016. https://doi.org/10.1145/2993148.2993165.

[4] Patrick Lucey, Jeffrey F. Cohn, Takeo Kanade, Jason Saragih, Zara Ambadar, and Iain Matthews. The extended Cohn-Kanade dataset (CK+): A complete dataset for action unit and emotion-specified expression. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 94–101, 2010. https://doi.org/10.1109/CVPRW.2010.5543262.

[5] Michael J. Lyons, Shigeru Akamatsu, Miyuki Kamachi, and Jiro Gyoba. Coding facial expressions with Gabor wavelets. In *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 200–205, 1998. https://doi.org/10.1109/AFGR.1998.670949.

[6] Shan Li, Weihong Deng, and JunPing Du. Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2852–2861, 2017. https://doi.org/10.1109/CVPR.2017.277.

[7] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019. https://doi.org/10.1109/ICCV.2019.00140.

[8] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 2019. https://proceedings.mlr.press/v97/tan19a.html.

[9] Andrey V. Savchenko. Facial expression and attributes recognition based on multi-task learning of lightweight neural networks. *arXiv preprint arXiv:2103.17107*, 2021. https://arxiv.org/abs/2103.17107.

[10] Xunru Liang, Jianfeng Liang, Tao Yin, and Xiaoyu Tang. A lightweight method for face expression recognition based on improved MobileNetV3. *IET Image Processing*, 17(8):2375–2384, 2023. https://doi.org/10.1049/ipr2.12798.

[11] Lifang Zhou, Siqin Li, Yi Wang, and Junlin Liu. SDNET: Lightweight facial expression recognition for sample disequilibrium. In *ICASSP 2022 – 2022 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2415–2419, 2022. https://doi.org/10.1109/ICASSP43922.2022.9746695.

[12] Shervin Minaee and Amirali Abdolrashidi. Deep-emotion: Facial expression recognition using attentional convolutional network. *Sensors*, 21(9):3046, 2021. https://doi.org/10.3390/s21093046.

[13] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. ECA-Net: Efficient channel attention for deep convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11534–11542, 2020. https://doi.org/10.1109/CVPR42600.2020.01155.

[14] Qibin Hou, Daquan Zhou, and Jiashi Feng. Coordinate attention for efficient mobile network design. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13713–13722, 2021. https://doi.org/10.1109/CVPR46437.2021.01350.

[15] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2999–3007, 2017. https://doi.org/10.1109/ICCV.2017.324.

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. https://arxiv.org/abs/1503.02531.