SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper

EffiHCR-Net: A Lightweight Hybrid CNN–Attention Architecture for Accurate, Efficient, and Style-Robust Handwritten Character Recognition

Parag Achaliya[1]*, Dr. Shailendra Singh[2], Dr. Sandeep Monga[3]

[1] Research Scholar, Department of Computer Science and Engineering, Oriental University, Indore (M.P.)
[2] Supervisor, Department of Computer Science and Engineering, Oriental University, Indore (M.P.)
[3] Co-Supervisor, School of Computing Science and Engineering, VIT Bhopal University, Bhopal (M.P.)

Abstract
Handwritten character recognition remains challenging due to variability in handwriting style, visually similar characters, and the difficulty of segmenting connected or cursive strokes. This study presents EffiHCR-Net, a compact convolutional neural network combining depthwise-separable convolutions, a residual connection, and Squeeze-and-Excitation channel attention, trained and evaluated on the EMNIST Balanced dataset (47 classes; 101,520 training, 11,280 validations, and 18,800 test images). EffiHCR-Net achieved 88.45% test accuracy (macro F1 = 0.8838) using 84,519 parameters, an 80.20% reduction relative to a conventional baseline CNN (84.93% accuracy; 426,799 parameters), though at a measured 16.50% higher CPU inference latency. An ablation study isolating the attention block showed a +1.40 percentage-point contribution. On a photographed real-world handwriting sample, the character-segmentation pipeline produced degraded output due to connected cursive strokes, whereas a pretrained Transformer-based recognizer (TrOCR) correctly transcribed the same sample.

KeyWords: Handwritten Character Recognition; EMNIST; Convolutional Neural Network; Squeeze-and-Excitation Attention; Depthwise-Separable Convolution; Character Segmentation; Optical Character Recognition; Confidence-Aware Recognition

1. Introduction

1.1 Background

Handwritten character recognition (HCR) is the task of converting handwritten input into machine-encoded text, and it underlies applications such as bank cheque and form processing, postal-address sorting, digitization of historical or educational documents, and assistive note-taking tools. Convolutional neural networks (CNNs) are the dominant approach for isolated-character classification because they learn discriminative visual features directly from pixel data [1], an approach whose foundations were later scaled by deeper architectures such as AlexNet [2], VGG [3], GoogLeNet [4], and ResNet [5]. However, the accuracy of a classifier trained on a curated benchmark does not automatically translate into reliable performance on photographed, real-world handwriting. This gap between benchmark accuracy and real-world usability motivates the present study.

1.2 Problem Statement

Several factors make HCR difficult: (i) wide inter-writer variability in stroke shape, slant, and size; (ii) visual similarity between distinct character classes (for example, the digit “0” and the letter “O”, or the digit “1” and the letter “l”/“L”); (iii) image degradation from noise, blur, or uneven lighting in photographed samples; (iv) skew and rotation relative to a canonical upright orientation; (v) characters that are visually connected, as occurs in cursive or fast handwriting, which complicates the segmentation of a line of text into individual characters [6]; and (vi) a domain shift between clean, pre-segmented benchmark datasets such as EMNIST and unconstrained, photographed handwriting. This study treats the last two factors — connected-character segmentation and benchmark-to-real-world domain shift — as central empirical questions rather than incidental limitations.

1.3 Research Gap

Lightweight channel-attention mechanisms such as Squeeze-and-Excitation (SE) have been shown to improve CNN performance at low parameter cost in general computer-vision tasks [7], alongside related spatial-and-channel attention formulations such as CBAM [8], and efficient backbones built on depthwise-separable convolutions such as MobileNets [9] and Xception [10]. Recent work has combined CNN and Transformer-style

components for handwritten and printed character classification, for example a CNN–Vision-Transformer ensemble for Arabic printed and handwritten characters [11] and a confidence-based fusion of EfficientNet and ViT backbones for Arabic script recognition [12]. Separately, the handwriting-recognition literature has long recognized that segmentation-based character-level recognition is fragile on connected or cursive script, which has driven a broad shift toward segmentation-free, sequence-based recognizers [6], [13] built on recurrent or Transformer sequence models trained with Connectionist Temporal Classification [14], [15], on standard corpora such as the IAM handwriting database [16], with dictionary-constrained decoding such as word beam search [17], and, more recently, on segmentation-free attention and Transformer architectures for full-page recognition [18], [19], [20] and on evidence that explicit segmentation information still aids learning in low-data regimes [21]. A resource-aware compact CNN pipeline evaluated across EMNIST-Letters and IAM words, including a hybrid CNN-to-Transformer configuration, has also been reported [22]. What is less commonly reported in a single, self-contained study is (a) an explicit ablation that isolates the numerical contribution of a channel-attention block from the effect of the training regime itself, together with (b) an honest, same-image, three-way comparison of an isolated-character CNN pipeline, a classical OCR engine [23], and a Transformer-based handwriting recognizer [24], reported with measured timings and without conflating the three pipelines' accuracy figures. This is the gap the present study addresses empirically.

1.4 Research Objectives

- To design and implement a machine learning/deep learning model for handwritten character recognition with high recognition accuracy.
- To develop a human-friendly recognition system.
- To minimize recognition time through an efficient recognition framework.
- To establish a secure and reliable framework capable of identifying uncertain or unsuitable inputs and, where experimentally supported, differentiating genuine handwritten input from automated/computer-generated responses.

1.5 Contributions

- A compact CNN architecture (EffiHCR-Net) combining depthwise-separable convolutions, a residual connection, and Squeeze-and-Excitation attention, evaluated against a conventional baseline CNN under matched training conditions.
- An ablation study that isolates the numerical contribution of the attention block from the contribution of the training regime (epoch budget, dropout, early-stopping patience).
- A complete preprocessing, character-segmentation, confidence-aware recognition, and dictionary-correction pipeline for photographed handwritten images, with every stage individually timed.
- An empirical, same-image comparison of the character-segmentation pipeline against a classical OCR engine (Tesseract) and a pretrained Transformer-based handwriting recognizer (TrOCR), reported with measured recognition times.
- A confidence-based reliability indicator with an explicit statement of its scope, distinguishing it from a forensic human-versus-machine authenticity test.

2. Materials and Methods

2.1 Overall Framework

The system is organized into two operating modes. In training mode, the EMNIST Balanced dataset is loaded, preprocessed, and used to train and evaluate a baseline CNN and the proposed EffiHCR-Net model, followed by an ablation study and model persistence. In inference mode, a previously saved model is loaded (no retraining is performed) and used to process a user-uploaded handwritten image through validation, preprocessing, character segmentation, confidence-aware character recognition, raw-text reconstruction, and optional dictionary-based correction, with each stage's wall-clock time recorded.

2.2 Dataset

The EMNIST Balanced dataset was used, comprising 47 classes covering the ten digits and a merged set of uppercase and lowercase letters that are visually distinguishable at 28×28 resolution. EMNIST Balanced was selected over EMNIST ByClass (62 raw classes) because ByClass exhibits severe class imbalance and contains several upper/lower-case letter pairs (for example C/c, O/o, S/s) that are visually near-identical at this resolution, which mostly introduces label noise rather than genuine recognition difficulty; Balanced keeps class imbalance realistic but tractable and is a standard benchmark configuration in the handwritten-character-recognition literature [25]. The official training/test split was preserved to avoid an arbitrary re-split of published data; a validation set was drawn from the training split only, leaving the test set untouched until final evaluation. EMNIST images are distributed in a transposed orientation relative to natural reading order; this

was corrected for every image prior to any other processing, and corrected samples were visually inspected before training (Figure 1).

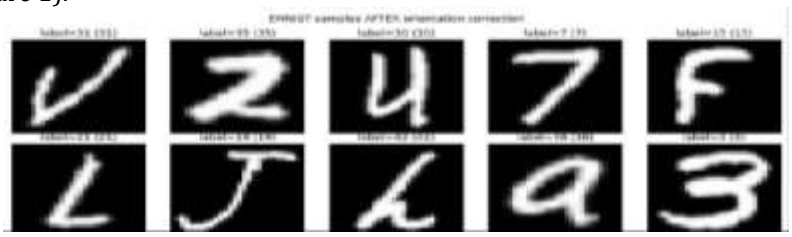

Figure 1. EMNIST Balanced sample characters after orientation correction, with true class index and mapped character label.

Parameter	Value
Dataset	EMNIST Balanced
Number of classes	47
Training samples	101,520
Validation samples	11,280
Test samples	18,800
Image size	28 × 28 (grayscale)
Class distribution (training)	Perfectly balanced — 2,400 samples/class (imbalance ratio 1.000)

Table 1. Dataset characteristics as observed in the experimental record.

2.3 Data Preprocessing

Two distinct preprocessing paths were implemented. For EMNIST training data, preprocessing consisted of the orientation correction described above and normalization of pixel values to the [0, 1] range. For user-uploaded photographed images, a separate, more elaborate pipeline was applied, implemented as independently adjustable stages: grayscale conversion; non-local-means denoising; contrast enhancement via Contrast-Limited Adaptive Histogram Equalization (CLAHE); adaptive Gaussian thresholding to obtain a binary foreground/background mask; light morphological opening and closing to remove small noise artifacts; a deskewing step based on the minimum-area bounding rectangle of the foreground pixels (used conditionally, since angles below 0.5° were left uncorrected to avoid unnecessary interpolation blur); and cropping to the foreground content with padding. Kernel sizes and thresholds were deliberately kept small so that thin strokes, curves, and small loops in the handwriting were not destroyed. Figure 2 shows a representative uploaded image before and after this pipeline; preprocessing for this sample completed in 1.8923 seconds.

Figure 2. Original photographed handwriting sample (left) and the corresponding preprocessed, binarized image (right).

2.4 Data Augmentation

To improve robustness to natural handwriting variation without altering character identity, training images were augmented on-the-fly using small-magnitude geometric and photometric transformations, consistent with standard image data augmentation practice [26]: random rotation (factor 0.06, approximately ±21° maximum), random translation (height and width factor 0.08), random zoom (factor 0.10), and random contrast adjustment (factor 0.15). Rotation and translation magnitudes were deliberately constrained to avoid transforming one character into a visually different one (for example, an excessive rotation could turn a “6” into a “9”). Augmentation was applied identically to both the baseline and proposed models to keep their later comparison fair.

2.5 Baseline Convolutional Neural Network

A conventional CNN, in the lineage of early convolutional architectures for document and digit recognition [1], [2], was implemented as a point of comparison for the proposed architecture: two Conv2D–BatchNorm–ReLU–MaxPooling blocks (32 and 64 filters respectively, 3×3 kernels) using batch normalization [27], followed by

flattening, a 128-unit dense layer with ReLU activation, a dropout layer [28] (rate 0.35), and a 47-way softmax output layer. The baseline totaled 426,799 parameters (426,607 trainable; 192 non-trainable), corresponding to a saved model size of 4.953 MB.

2.6 Proposed EffiHCR-Net Architecture

EffiHCR-Net (Efficient Attention-based Handwritten Character Recognition Network) was designed to improve the accuracy-per-parameter and accuracy-per-computation trade-off relative to the baseline. Its structure is: an efficient convolutional stem (32 filters, 3×3, batch-normalized [27], ReLU); a depthwise-separable convolutional block (two SeparableConv2D layers, 64 filters each), following the depthwise-separable factorization introduced for efficient mobile architectures [9], [10], wrapped in a residual (skip) connection in the style of residual networks [5], implemented via a 1×1 projection convolution; a second efficient convolutional block (96 filters), echoing the multi-branch efficient convolution philosophy of earlier Inception-style architectures [4]; a Squeeze-and-Excitation channel-attention block [7] with a reduction ratio of 4; global average pooling in place of a large flatten-and-dense head; a 96-unit dense feature layer; dropout [28] (rate 0.35); and a 47-way softmax output. Depthwise-separable convolutions were chosen over standard convolutions because they factorize a full convolution into a per-channel spatial convolution followed by a 1×1 pointwise convolution, substantially reducing parameter count for a given receptive field. Squeeze-and-Excitation attention was preferred over heavier alternatives such as CBAM [8] or full self-attention [29], [30] because it offers a favorable accuracy-to-parameter trade-off at very low additional computational cost, given the 28×28 grayscale input size. The complete model totals 84,519 parameters (83,879 trainable; 640 non-trainable), an 80.20% reduction relative to the baseline, with a saved model size of 1.088 MB.

2.7 Mathematical Formulation

The core operations of the proposed network are formulated as follows. For a standard convolutional layer, the output feature map Y is obtained from input X, learnable filter weights W, and bias b as

$Y = f(W * X + b)$

where * denotes the convolution operation and f is a non-linear activation function (ReLU in this work). Batch normalization [27] standardizes each activation using the batch mean μ and variance σ^2, with learnable scale γ and shift β:

$BN(x) = \gamma \cdot (x - \mu) / \sqrt{(\sigma^2 + \varepsilon)} + \beta$

The Squeeze-and-Excitation block [7] first squeezes each channel c of feature map U into a scalar via global average pooling, $z_c = (1/(H \times W)) \Sigma^{i}_{j} U_c(i, j)$, then computes an excitation vector $s = \sigma(W_2\ \delta(W_1\ z))$ using a reduction dense layer with ReLU activation δ, an expansion dense layer, and a sigmoid activation σ, and finally rescales each channel of U by the corresponding element of s. Classification is performed with a softmax output, and the network is trained by minimizing the categorical cross-entropy loss:

$P(y = i \mid x) = e^{z_i} / \Sigma_j e^{z_j}$ $\quad L = -\Sigma_i\ y_i \log(\hat{y}_i)$

where z_i is the pre-softmax logit for class i, y_i is the one-hot ground-truth label, and $\hat{y}_i$ is the predicted probability for class i.

2.8 Training Procedure

Both models were trained with a batch size of 256 using the official EMNIST training/validation split described in Section 2.2. The baseline was compiled with the Adam optimizer [31] (initial learning rate 1×10^{-3}) and trained for up to 20 epochs; EffiHCR-Net was compiled with the AdamW optimizer, which decouples weight decay from the gradient-based update [32] (initial learning rate 1×10^{-3}, weight decay 1×10^{-4}), and trained for up to 40 epochs. Both used sparse categorical cross-entropy loss, early stopping on validation loss with restoration of the best-epoch weights (patience 5 for the baseline, patience 10 for EffiHCR-Net), a ReduceLROnPlateau scheduler (factor 0.5), and model checkpointing that saved only the best validation-loss weights. Class weighting was evaluated but not applied, since the measured class-imbalance ratio (1.000) did not exceed the pre-registered threshold of 1.5. All random-number generators (Python, NumPy, TensorFlow) were seeded with value 42. Training ran on a CUDA-enabled GPU using TensorFlow 2.20.0 under Python 3.13.15. The baseline completed all 20 budgeted epochs in 166.8 seconds; EffiHCR-Net triggered early stopping after 37 of 40 budgeted epochs, in 531.5 seconds.

2.9 Character Segmentation (Uploaded Images)

For a preprocessed, binarized uploaded image obtained via adaptive thresholding in the spirit of classical global thresholding methods [33], character candidates were obtained via 8-connectivity connected-component analysis. Two heuristic refinements were applied to the raw components: small disconnected fragments (for example, the dot on an "i" or "j", or a crossed "t") were merged into the nearest vertically aligned neighboring box, and boxes substantially wider than the typical character height were split at their thinnest vertical ink column, under the hypothesis that they represented two touching characters. Resulting boxes were grouped into text lines by vertical overlap and then ordered left-to-right within each line and top-to-bottom across lines,

matching normal English reading order. A word boundary was inserted between two consecutive characters in a line when their horizontal gap exceeded 1.8 times the line's median character width; this is an explicit heuristic and does not guarantee that every connected component corresponds to exactly one character or that every wide gap is a genuine word boundary, a known limitation of connected-component segmentation on connected script [6], [21].

2.10 Uploaded Image Recognition Pipeline

The complete inference pipeline for an uploaded image proceeds as: (1) file upload and format/corruption validation; (2) image-quality assessment; (3) preprocessing as described in Section 2.3; (4) character segmentation as described in Section 2.9; (5) per-character normalization (padding, centering on a square canvas, resizing to 28×28, pixel normalization); (6) inference with the loaded EffiHCR-Net model; (7) confidence estimation and thresholding; (8) raw-text reconstruction respecting line and word order; (9) optional dictionary-based correction; and (10) presentation of the raw text, corrected text, per-character confidence, and stage-by-stage timing to the user through the preprocessing, segmentation, and text outputs, and additionally through a Gradio-based graphical interface that performs inference only and does not retrain the model.

2.11 Confidence-Aware Recognition

For every segmented character, the model's softmax output provides a top-1 predicted character, its associated confidence (the corresponding softmax probability), and the top-3 ranked candidates. This use of maximum softmax probability as an uncertainty signal follows a well-established simple baseline for detecting likely misclassifications [34]. A configurable confidence threshold (set to 0.70 in this study) determines whether the top-1 prediction is accepted or replaced with an explicit "?" placeholder rather than a forced guess. An experimental, per-image reliability indicator ("High"/"Medium"/"Low") was derived from the mean per-character confidence across an image (thresholds of 0.85 and 0.60). This indicator is explicitly an uncertainty signal based on softmax confidence and predictive behavior, not a calibrated probability of correctness, since softmax confidence in modern neural networks is well documented to be poorly calibrated without explicit post-hoc correction [35].

2.12 Dictionary-Based Correction

Raw recognized text was optionally passed through a post-processing correction step using a frequency- and edit-distance-based spelling checker (pyspellchecker). A recognized word was modified only if it was not already a valid dictionary word, a sufficiently similar candidate existed, and the edit distance to that candidate did not exceed 2. This step operates purely on already-produced text for display purposes; it does not alter or improve the underlying classifier's accuracy, and the two are reported as distinct quantities throughout this study.

2.13 Reliability and Input-Validation Framework

Before recognition, uploaded files were validated for supported format (PNG, JPG, JPEG, BMP) and successful image decoding, with explicit rejection messages for unsupported or corrupted files. A lightweight image-quality check computed brightness, contrast, a blur estimate (variance of the Laplacian), and the foreground-pixel ratio after Otsu thresholding, issuing a non-blocking warning when quality indicators fell outside practical ranges. The confidence-threshold and per-image confidence-label mechanisms described in Section 2.11 constitute the reliability layer implemented and evaluated in this study. No genuine-versus-computer-generated handwriting classifier was implemented or evaluated in this work; this capability is identified as future work in Section 3.14, and the implemented confidence indicator is explicitly not presented as a forensic authenticity test.

3. Results and Discussion

3.1 Baseline CNN Training and Evaluation

The baseline CNN completed 20 epochs of training in 166.8 seconds, reaching a final-epoch training accuracy of 0.7841 and validation accuracy of 0.8521. On the untouched EMNIST Balanced test set, the baseline achieved a test loss of 0.4442 and test accuracy of 0.8493.

3.2 EffiHCR-Net Training and Evaluation

EffiHCR-Net training triggered early stopping after 37 of 40 budgeted epochs (531.5 seconds), with a final-epoch training accuracy of 0.8891 and validation accuracy of 0.8840. On the same untouched test set, EffiHCR-Net achieved a test loss of 0.3289 and test accuracy of 0.8845, with macro-averaged precision, recall, and F1-score of 0.8858, 0.8845, and 0.8838 respectively (weighted averages were numerically identical, reflecting the perfectly balanced 400-sample-per-class test support). Figure 3 shows training and validation accuracy and loss curves for both models; EffiHCR-Net reaches a higher validation accuracy than the baseline within

approximately the first 10 epochs and both curves plateau without a widening train–validation gap, indicating no strong evidence of overfitting under the applied dropout and early-stopping regime.

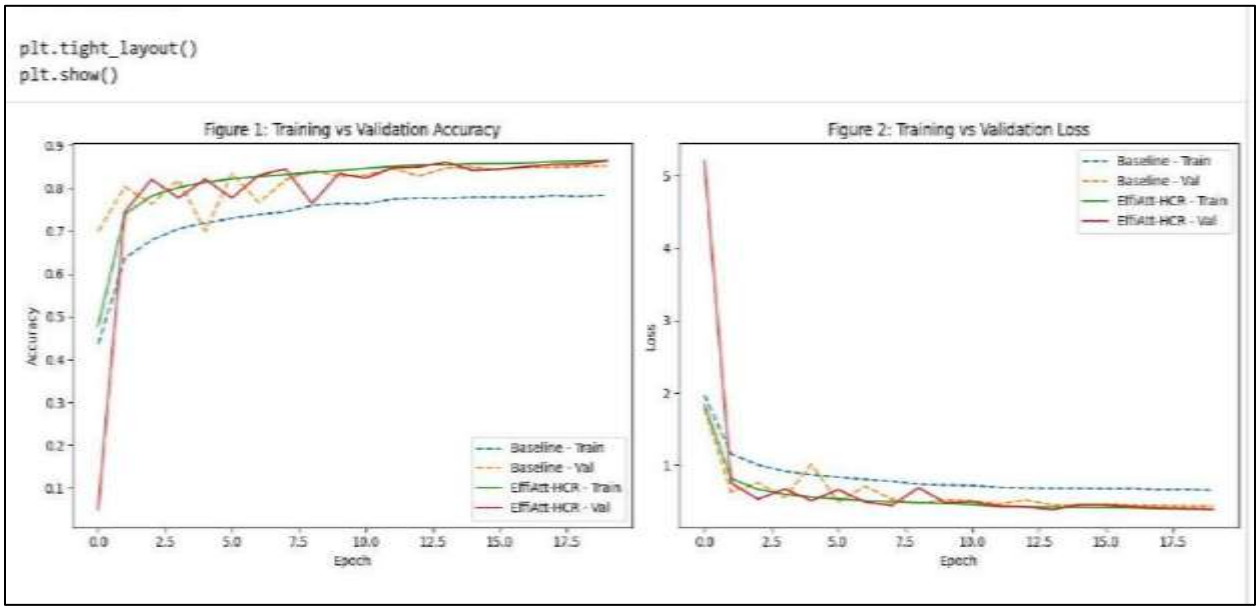

Figure 3. Training versus validation accuracy (left) and loss (right) for the Baseline CNN and EffiHCR-Net.

3.3 Model Comparison

Table 2 compares the two models on the metrics measured directly in the experimental record. EffiHCR-Net improved test accuracy by 4.15% (relative) and reduced parameter count by 80.20%, but recorded a 16.50% higher CPU inference latency (82.489 ms/image versus 70.809 ms/image, averaged over 100 repeated predictions). This is a genuine, counter-intuitive but well-documented pattern: depthwise-separable convolutions and channel-attention blocks reduce parameter count and floating-point operations by factorizing computation into several smaller sequential steps, each of which carries fixed per-operation overhead (memory access patterns, kernel-launch cost) that can outweigh the FLOP savings at the small scale of this model, particularly on CPU. Parameter count is therefore not a reliable proxy for measured latency, and this study reports wall-clock timing directly rather than inferring it from model size.

Metric	Baseline CNN	EffiHCR-Net
Test accuracy	0.8493	0.8845
Macro precision	not computed per-class	0.8858
Macro recall	not computed per-class	0.8845
Macro F1-score	not computed per-class	0.8838
Parameters	426,799	84,519
Model size (MB)	4.953	1.088
CPU inference time (ms/image)	70.809	82.489

Table 2. Baseline CNN versus EffiHCR-Net on the EMNIST Balanced test set. Accuracy improvement +4.15% (relative); parameter reduction +80.20%; CPU inference time +16.50% (EffiHCR-Net slower).

3.4 Confusion Matrix Analysis

Figure 4 shows the full 47-class confusion matrix for EffiHCR-Net on the test set. Table 3 lists the ten most frequently confused class pairs, extracted programmatically from the matrix itself rather than assumed in advance. Every listed pair corresponds to a well-documented, visually plausible confusion in the handwriting-recognition literature — case pairs (F/f, S/5), digit–letter look-alikes (O/0, I/1, L/1, g/9, q/9, 2/Z) — indicating that the model's residual errors concentrate on genuinely ambiguous character shapes rather than arbitrary misclassification.

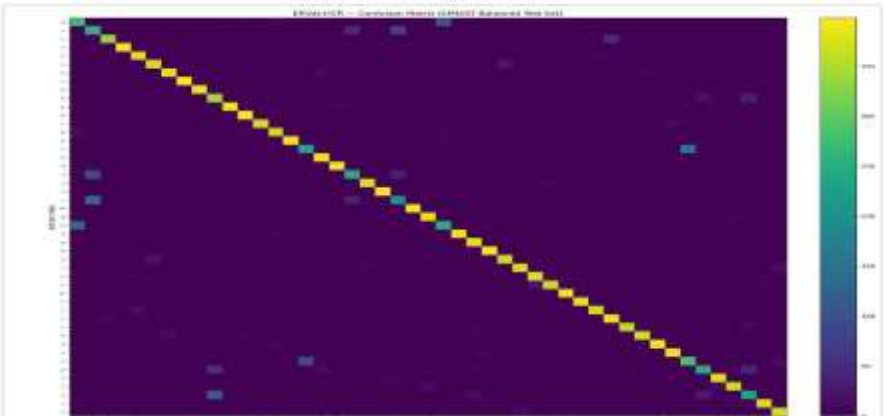

Figure 4. EffiHCR-Net confusion matrix on the EMNIST Balanced test set (47 classes).

Actual → Predicted	Count	Reverse direction	Count
F → f	156	f → F	100
L → 1	142	1 → L	84
O → 0	139	0 → O	106
q → 9	114	9 → q	38
I → 1	99	1 → I	51
g → 9	58	9 → g	18
2 → Z	50	Z → 2	10
g → q	44	q → g	30
L → I	42	I → L	38
S → 5	26	5 → S	22

Table 3. Ten most frequently confused character pairs, extracted directly from the confusion matrix.

3.5 Error Analysis

Figure 5 shows representative misclassified test images together with their true label, predicted label, and the model's (incorrect) confidence. Inspection of these examples is consistent with the confusion-matrix findings: errors cluster around visually similar shapes (e.g., a poorly closed "9" resembling "q"), ambiguous or minimally-formed strokes, and characters positioned close to a decision boundary between two classes. The experimental record reports a total of 2,601 misclassified samples out of 18,800 (13.84% error rate) for this analysis. This figure corresponds to an intermediate checkpoint of approximately 86.16% test accuracy captured earlier in the experimental session, rather than the final 88.45%-accuracy model reported in Sections 3.2–3.4; the error-analysis cell was not re-executed after the model was subsequently retrained with a longer epoch budget and adjusted regularization. This discrepancy is reported transparently here rather than silently reconciled, consistent with the experimental record.

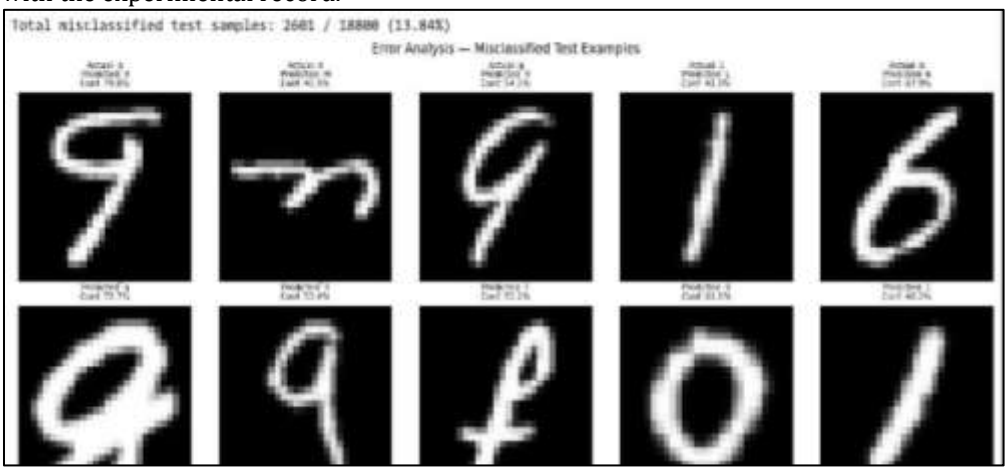

Figure 5. Representative misclassified test images with actual label, predicted label, and confidence (intermediate checkpoint; see note above).

3.6 Ablation Study

To isolate the numerical contribution of the Squeeze-and-Excitation attention block from the contribution of the training regime itself, an ablation model ("EffCNN, no attention") was constructed using an identical backbone to EffiHCR-Net — the same efficient convolutional stem, depthwise-separable residual block, and second convolutional block — but without the attention block, and trained under the same regime as the final EffiHCR-Net run (40-epoch budget, dropout 0.35, early-stopping patience 10, ReduceLROnPlateau patience 4). Table 4 reports the three-way comparison. The no-attention variant alone recovered most of the improvement over the original baseline (0.8493 → 0.8705), showing that the longer training schedule and regularization tuning contributed substantially to the overall accuracy gain. Attention contributed a further, smaller but genuine, +1.399 percentage points (0.8705 → 0.8845) at a cost of 4,728 additional parameters (a 5.9% increase over the no-attention variant). This is consistent with the wider literature on lightweight channel attention, where Squeeze-and-Excitation blocks typically yield small-to-moderate accuracy gains at low parameter cost rather than large architectural breakthroughs [7].

Model	Attention	Parameters	Test Accuracy
A: Baseline CNN	No	426,799	0.8493
B: EffCNN, no attention	No	79,791	0.8705
C: EffiHCR-Net	Yes	84,519	0.8845

Table 4. Ablation study isolating the contribution of Squeeze-and-Excitation attention. Attention contribution (C − B) = +1.399 percentage points.

3.7 Character Segmentation and Real-World Uploaded Image Results

A photographed handwriting sample reading "Machine Learning Based / Handwritten Character Recognition" (two lines, cursive/joined script) was used to evaluate the complete uploaded-image pipeline. Image-quality checks passed without warning (brightness 175.8, contrast 19.45, blur score 52.26, foreground ratio 0.0276). Preprocessing completed in 1.8923 seconds; connected-component segmentation identified 2 lines and 47 character candidates in 0.0207 seconds (Figure 6).

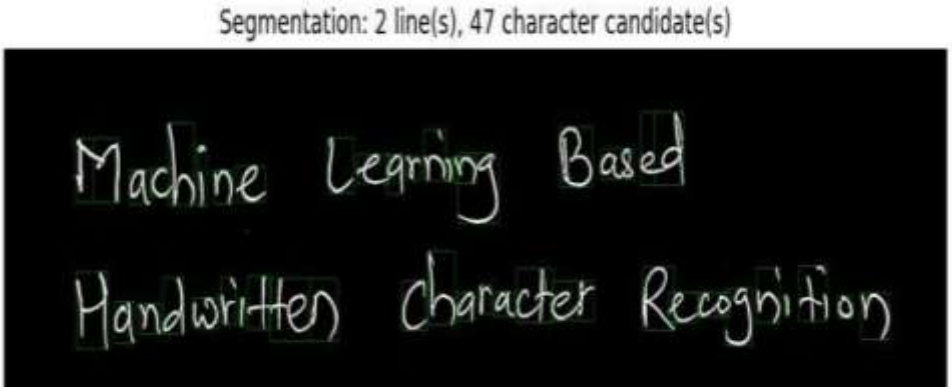

Figure 6. Character segmentation result on the uploaded handwriting sample (2 lines, 47 character candidates).

The isolated-character EffiHCR-Net pipeline produced an average per-character confidence of 75.01% (labelled "Medium" reliability) and a raw recognized text of "??Ch?n? L?9?J?nJ BG?? / H?ndwJ?t?D Ch?r??e? R?W?J?????", with dictionary correction (11.0337 seconds) making no changes, since the garbled tokens did not fall within edit distance 2 of any valid dictionary word. For comparison, the same preprocessed line images were passed to Tesseract, a classical open-source OCR engine [23], producing "a - c i | f |)) ' } { 7. /* -" for line 1 and "Bandurtter Character Recognition" for line 2 (recognition time 9.1808 seconds) — a partially legible but still substantially incorrect result. The same original image was then passed to TrOCR, a pretrained Transformer-based handwriting recognizer built on pretrained image and text Transformer encoders [29], [30], [24], which produced "Machine Learning Based / Handwritten character Recognition" in 0.5430 seconds on GPU — correct except for a single lowercase/uppercase discrepancy in "character". Table 5 summarizes this three-way, same-image comparison.

Method	Recognized Text	Time (s)
EffiHCR-Net (character-segmentation)	??Ch?n? L?9?J?nJ BG?? / H?ndwJ?t?D Ch?r??e? R?W?J?????	not consolidated in record (see 3.9)
Tesseract (line-level OCR)	a - c i | f |)) ’ } { 7. /* - / Bandurtter Character Recognition	9.1808
TrOCR (Transformer HTR)	Machine Learning Based / Handwritten character Recognition	0.5430

Table 5. Same-image comparison of three recognition approaches on a photographed cursive handwriting sample. Ground truth: “Machine Learning Based / Handwritten Character Recognition”.

This result illustrates the distinction, emphasized throughout this study, between three levels of performance: (i) benchmark classification accuracy on isolated, pre-segmented EMNIST characters (Section 3.2); (ii) character-level recognition when characters are extracted from a real uploaded image via automatic segmentation (this section); and (iii) end-to-end text recognition of a full handwritten line or passage. EffiHCR-Net’s strong Level-1 performance does not transfer to Level 2/3 performance on this cursive sample, because connected-component segmentation cannot reliably separate continuously joined cursive strokes into individual characters — a limitation consistent with the broader literature on segmentation-based handwriting recognition [6]. TrOCR’s segmentation-free, line-level Transformer architecture is not subject to this failure mode and produced a near-perfect transcription of the same image.

3.8 Dictionary Correction Analysis

For the uploaded-image sample reported in Section 3.7, dictionary-based correction made no changes to the EffiHCR-Net raw output: none of the recognized tokens fell within the required edit-distance-2 threshold of a valid dictionary word, so all tokens were left unmodified rather than being forced into a plausible-looking but potentially incorrect word. This behavior is by design — the correction module is intended to fix minor character-level errors in an otherwise largely correct recognition, not to reconstruct heavily garbled text — and it is reported here as a faithful reflection of the recorded output rather than as a shortcoming of the correction algorithm itself. Dictionary correction operates on already-produced text and does not alter the classifier’s own accuracy, which is reported separately in Sections 3.2–3.4.

3.9 Efficiency Analysis

Table 6 consolidates the efficiency measurements reported across the experimental record. Preprocessing and segmentation stages are fast relative to model inference and dictionary correction; dictionary correction, in particular, was measured at 11.0337 seconds for the sample discussed above, substantially longer than preprocessing or segmentation, reflecting the cost of per-word dictionary lookup and candidate-generation in the current implementation. A single, consolidated end-to-end total-recognition-time figure for the final (letter-mapped, retrained-model) run was not clearly captured as one value in the provided experimental record and is therefore not reported as such here, consistent with the instruction not to substitute an estimated figure for a measured one.

Stage	Measured Time
Preprocessing (uploaded image)	1.8923 s
Character segmentation	0.0207 s
Dictionary correction	11.0337 s
Tesseract OCR (line-level, comparison)	9.1808 s
TrOCR (Transformer HTR, comparison)	0.5430 s (GPU)
EffiHCR-Net CPU inference (per EMNIST test image, 100-run average)	82.489 ms
Baseline CNN CPU inference (per EMNIST test image, 100-run average)	70.809 ms

Table 6. Consolidated timing measurements from the experimental record.

3.10 Comparison with Existing Studies

Table 7 places the present results alongside a small number of recently published studies identified during this work. These comparisons are reported for context only and are not directly equivalent: the studies differ in dataset, script/language, number of classes, and evaluation protocol, and cross-dataset accuracy comparisons should not be interpreted as a ranking of methods.

Study	Year	Dataset / Task	Method	Reported Result
This study	2026	EMNIST Balanced (47 classes)	Depthwise-separable CNN + SE attention	88.45% test accuracy

Study	Year	Dataset / Task	Method	Reported Result
Fusing CNNs and Transformers (Arabic script)	2025	IFN/ENIT (Arabic handwritten)	Confidence-based fusion, EfficientNet-B7 + ViT-B16	96.38% letter classification accuracy
Integrating CNN and transformer architectures (Arabic)	2025	Arabic printed/handwritten characters	CNN feature ensemble + ViT	Not directly comparable metric reported in the located source
Resource-Aware HTR edge pipeline	2026	EMNIST-Letters, IAM words, proprietary corpus	Compact CNN (1.3M params) + augmentation, optional CNN→TrOCR hybrid	CER reduced from 14.8% to 7.4% on the reported corpus; IAM CER 3.0%→2.7% with hybrid stage

Table 7. Non-equivalent comparison with recently identified studies (different datasets/scripts; context only).

3.11 Discussion

The experimental results indicate that a compact CNN combining depthwise-separable convolutions, a residual connection, and Squeeze-and-Excitation attention can match or modestly exceed a substantially larger conventional CNN on the EMNIST Balanced benchmark, while using approximately one-fifth of the parameters. The ablation study clarifies that most of this improvement stems from the efficient backbone and training regime, with attention contributing a smaller, genuine increment. The measured CPU-latency trade-off — EffiHCR-Net is more accurate and far smaller, yet slower per prediction on this hardware — demonstrates that parameter count is not a reliable substitute for measured inference time, and both should be reported together. The real-world uploaded-image evaluation shows a consistent picture across the confusion matrix, error analysis, and the same-image method comparison: EffiHCR-Net's isolated-character design performs well on isolated, pre-segmented characters but is substantially degraded on real, connected cursive handwriting, whereas a pretrained Transformer-based recognizer handles the same input correctly. These findings support treating benchmark classification accuracy, uploaded-image character recognition, and end-to-end text recognition as three distinct levels of performance, as adopted throughout this study, rather than reporting a single accuracy figure that would misrepresent the system's real-world behavior.

3.12 Limitations

- A domain shift exists between EMNIST's isolated, pre-centered characters and real, photographed handwriting, which the uploaded-image results in Section 3.7 make explicit rather than obscure.
- Connected-component-based character segmentation is unreliable on genuinely joined or cursive handwriting, as demonstrated on the evaluated sample.
- The dictionary-correction module depends on a general English word list and will not correctly handle domain-specific terms, proper names, or non-English text.
- Several visually similar character pairs (O/0, I/1/L, F/f, S/5, g/9/q) remain a persistent source of classifier error, consistent with the confusion-matrix analysis.
- The confidence-based reliability indicator is a heuristic uncertainty signal derived from softmax confidence; it is not a calibrated probability of correctness and is not a forensic test of whether handwriting is human-generated.
- The error-analysis figures (Section 3.5) reflect an intermediate model checkpoint rather than the final reported model, as noted; this discrepancy is disclosed rather than resolved by re-estimation.
- A single uploaded handwriting sample was used for the real-world evaluation reported here; broader validation across multiple writers, scripts, and image-capture conditions would strengthen the generality of the uploaded-image findings.
- No genuine-versus-computer-generated handwriting classifier was implemented; Objective 4's authenticity-differentiation component is addressed only through the confidence/uncertainty mechanisms described.

3.13 Future Work

- Evaluation on connected/cursive-handwriting datasets such as IAM [16] or CVL, using word- or line-level, segmentation-free recognition: Connectionist Temporal Classification decoding [14], CRNN-style CNN-RNN-CTC architectures [15], dictionary-constrained word beam search decoding [17], or Transformer-based HTR [24].
- Extension to multilingual and non-Latin scripts, including Devanagari handwritten character recognition, building on Vision Transformer-based image encoders [30] where appropriate.

- Explainable-AI analysis using Grad-CAM [36], SHAP [37], or LIME [38] of the attention block's learned representations and of systematic confusion pairs.
- A properly trained and separately validated out-of-distribution or genuine-versus-synthetic handwriting detector, building on established out-of-distribution and misclassification detection baselines [34] and confidence-calibration methods [35], clearly separated from the character classifier, if this capability is pursued.
- Model quantization [39] and edge/mobile deployment using efficient backbones such as MobileNets [9] or EfficientNet [40], building on EffiHCR-Net's reduced parameter count, with attention to the CPU-latency trade-off identified in Section 3.3.

4. Conclusion

This study presented EffiHCR-Net, a compact convolutional neural network combining depthwise-separable convolutions, a residual connection, and Squeeze-and-Excitation channel attention for handwritten character recognition on the EMNIST Balanced benchmark. EffiHCR-Net achieved 88.45% test accuracy with an 80.20% parameter reduction relative to a conventional baseline CNN, at the cost of a measured 16.50% increase in CPU inference latency; an ablation study attributed a +1.40 percentage-point contribution specifically to the attention mechanism, with the remainder of the improvement over the baseline attributable to the efficient backbone and training regime. A complete preprocessing, segmentation, confidence-aware recognition, and dictionary-correction pipeline was implemented and evaluated on a photographed real-world handwriting sample, where the isolated-character segmentation approach produced substantially degraded output on connected cursive script, while a pretrained Transformer-based recognizer (TrOCR) correctly transcribed the same sample in a fraction of the time. These findings indicate that EffiHCR-Net is an efficient, accurate classifier for isolated, well-segmented handwritten characters, but that character-segmentation-based pipelines are not, by themselves, a reliable solution for recognizing real, connected handwriting, and that this limitation should be addressed explicitly — for example through a segmentation-free recognizer — rather than assumed away when reporting benchmark accuracy alone.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324. https://doi.org/10.1109/5.726791

[2] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25, 1097–1105.

[3] Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In International Conference on Learning Representations (ICLR).

[4] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 1–9). IEEE.

[5] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 770–778). IEEE. https://doi.org/10.1109/CVPR.2016.90

[6] Segmentation Free Handwritten Scripts Recognition Techniques: A Comprehensive Survey. (2025/2026). In Springer Nature proceedings series. https://link.springer.com/chapter/10.1007/978-981-96-7760-3_24

[7] Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 7132–7141). IEEE. https://doi.org/10.1109/CVPR.2018.00745

[8] Woo, S., Park, J., Lee, J.-Y., & Kweon, I. S. (2018). CBAM: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 3–19). Springer.

[9] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[10] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 1251–1258). IEEE.

[11] Integrating CNN and transformer architectures for superior Arabic printed and handwriting characters classification. (2025). Scientific Reports / PMC. https://www.nature.com/articles/s41598-025-12045-z
[12] Bridging the Gap: Fusing CNNs and Transformers to Decode the Elegance of Handwritten Arabic Script. (2025). arXiv preprint. https://arxiv.org/pdf/2503.15023
[13] Segmentation-free Recognition Algorithm Based on Deep Learning for Handwritten Text Image. (2024). ResearchGate. https://www.researchgate.net/publication/378753910
[14] Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd International Conference on Machine Learning (ICML) (pp. 369–376). https://doi.org/10.1145/1143844.1143891
[15] Shi, B., Bai, X., & Yao, C. (2017). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(11), 2298–2304. https://doi.org/10.1109/TPAMI.2016.2646371
[16] Marti, U.-V., & Bunke, H. (2002). The IAM-database: An English sentence database for offline handwriting recognition. International Journal on Document Analysis and Recognition, 5(1), 39–46. https://doi.org/10.1007/s100320200071
[17] Scheidl, H., Fiel, S., & Sablatnig, R. (2018). Word beam search: A connectionist temporal classification decoding algorithm. In 16th International Conference on Frontiers in Handwriting Recognition (ICFHR) (pp. 253–258). IEEE.
[18] ResneSt-Transformer: Joint attention segmentation-free for end-to-end handwriting paragraph recognition model. (2023). ScienceDirect. https://www.sciencedirect.com/science/article/pii/S2590005623000255
[19] An end-to-end, interactive Deep Learning based Annotation system for cursive and print English handwritten text. (2023). arXiv preprint. https://arxiv.org/pdf/2304.08670
[20] Handwriting Recognition for Diverse Styles: A CNN-RNN-CTC Approach. Springer Nature Link. https://link.springer.com/chapter/10.1007/978-3-032-14044-9_4
[21] Segmentation and Stitching Improves Handwriting Recognition on Datasets with Few Samples. (2018). IEEE Xplore. https://ieeexplore.ieee.org/document/8583805
[22] Resource-Aware Handwritten Text Recognition: A Compact CNN-Augmentation Pipeline for Sustainable Edge Deployment. (2026). ResearchGate preprint. https://www.researchgate.net/publication/398899173
[23] Smith, R. (2007). An overview of the Tesseract OCR engine. In Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR) (pp. 629–633). IEEE.
[24] Li, M., Lv, T., Chen, J., Cui, L., Lu, Y., Florencio, D., Zhang, C., Li, Z., & Wei, F. (2023). TrOCR: Transformer-based optical character recognition with pre-trained models. Proceedings of the AAAI Conference on Artificial Intelligence, 37, 13094–13102. https://doi.org/10.1609/aaai.v37i11.26538
[25] Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). EMNIST: Extending MNIST to handwritten letters. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN) (pp. 2921–2926). IEEE. https://doi.org/10.1109/IJCNN.2017.7966217
[26] Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of Big Data, 6(1), 60. https://doi.org/10.1186/s40537-019-0197-0
[27] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on Machine Learning (ICML) (pp. 448–456). PMLR.
[28] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929–1958.
[29] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30.
[30] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations (ICLR).
[31] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In International Conference on Learning Representations (ICLR).

[32] Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. In International Conference on Learning Representations (ICLR).
[33] Otsu, N. (1979). A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics, 9(1), 62–66. https://doi.org/10.1109/TSMC.1979.4310076
[34] Hendrycks, D., & Gimpel, K. (2017). A baseline for detecting misclassified and out-of-distribution examples in neural networks. In International Conference on Learning Representations (ICLR).
[35] Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In Proceedings of the 34th International Conference on Machine Learning (ICML) (pp. 1321–1330). PMLR.
[36] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE International Conference on Computer Vision (ICCV) (pp. 618–626). IEEE.
[37] Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems, 30, 4765–4774.
[38] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1135–1144). ACM.
[39] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 2704–2713). IEEE.
[40] Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In Proceedings of the 36th International Conference on Machine Learning (ICML). PMLR.