OXFORD JOURNAL
OF STUDENT
SCHOLARSHIP

Stacking Diverse Deep Architectures for 62-Class Handwritten Character Recognition

Zubin Kalavade
zubin.kalavade@gmail.com

ABSTRACT

This work presents an approach to improving the accuracy of handwritten character recognition (HCR) across 62 classes: the uppercase alphabet, the lowercase alphabet, and the Arabic numerals. A majority of the difficulties that arose during the process were a result of structurally similar characters such as 1/l/I, O/0, and upper/lower case similarities like W/w. We evaluated our approach on a Kaggle dataset where no prior model had exceeded 85% accuracy. We utilized a stacking ensemble of three diverse models: a ResNet-18, a CBAM CNN, and a hybrid CNN-ViT, combined with a logistic regression meta-learner trained on out-of-fold predictions with temperature-scaled probabilities. Training employed focal loss, online hard example mining (OHEM), and an augmentation pipeline including CutMix and elastic distortion. The stacked ensemble achieved an accuracy of 91.2%, outperforming the previous best submission which achieved 84.6%. Future work could incorporate real-world image sources and accurate character scaling to further improve accuracy on case-ambiguous pairs.

Keywords: Convolutional Neural Network, Vision Transformer, Data Augmentation, Image Classification, Handwritten Character Recognition, Optical Character Recognition, Test-Time Augmentation, CutMix, Focal Loss, Online Hard Example Mining.

1 INTRODUCTION

Handwritten character recognition (HCR) is a core problem in optical character recognition (OCR) with applications in document processing, postal address reading, and form digitization. Beyond these, HCR enables automated bank cheque processing, digitization of historical archives and medical records, and accessibility tools that convert handwritten notes into digital text. Accurate and robust systems are, as a result, of a large value. Accurate classification remains difficult due to natural handwriting variation and visual ambiguity between character classes such as 0/O, 1/l/I, and S/5.

Most prior HCR research benchmarks on MNIST and EMNIST, large datasets with thousands of images, have largely been considered solved. However, real world use frequently involves far fewer labeled examples and requires recognition across the full alphanumeric set rather than digits alone. We address this using a 62-class Kaggle dataset with only 55 images per class, where no prior submission had exceeded 85% accuracy. This low-data, full-alphanumeric setting more closely reflects practical constraints compared to standard benchmarks, and the absence of a strong prior model indicates genuine room for improvement. We propose a stacking ensemble of three architecturally diverse models: a ResNet-18, a CBAM CNN, and a hybrid CNN-ViT, combined via a logistic regression meta-learner,

trained with focal loss, OHEM, and an aggressive data augmentation pipeline. The ensemble achieves 91.2% accuracy, more than 6 percentage points above the prior best.

Prior work on character recognition has largely focused on single-model architectures trained on large standardized datasets. To my knowledge, no prior work has applied a stacking ensemble of architecturally diverse models combined with hard-class targeted training strategies to the full 62-class alphanumeric recognition problem under a low-data regime of 55 images per class. This work differs from prior work in three ways: it operates in a low-data environment (55 images per class), it targets the full 62-class alphanumeric set rather than digits or letters alone, and it combines architectural diversity with hard-class training strategies specifically designed around the confusable character pairs that dominate classification error. The central hypothesis is that combining a ResNet-18, CBAM CNN, and hybrid CNN-ViT through a logistic regression meta-learner will outperform any single architecture, particularly on visually ambiguous pairs such as 0/O and W/w, because each model captures different feature representations and will therefore make errors on different classes.

The remainder of this paper is structured as follows. Section 2 provides background on CNNs and ViTs. Section 2.3 reviews related work. Section 3 details the methodology, including a description of the dataset, data preprocessing, model architectures, training procedure, and stacking ensemble. Section 4 presents experimental results. Section 5 discusses findings, the effect of key techniques, and limitations. Section 6 concludes the paper.

1.1 Related Work

Handwritten character recognition has been thoroughly studied on standardized datasets, the most common being EMNIST and MNIST. Cohen et al. [1] introduced EMNIST, containing the full English alphanumeric set but with far more images per class. Currently, state of the art models exceed 90% on EMNIST and MNIST data recognition has surpassed 99% and has essentially been considered solved.

CNNs have been the dominant approach for character recognition. He et al. [2] showed that residual connections allow models to be made substantially deeper without accuracy degradation, while Woo et al. [3] demonstrated that channel and spatial attention improve classification by directing the model toward stroke regions that are especially distinctive. ResNet models have exceeded 90% accuracy ranging from 90-92%. Dosovitskiy et al. [4] demonstrated that although ViTs can surpass traditional CNNs due to their ability to model global context, they require large datasets, making them suboptimal for the small dataset used in this work. ViT models have achieved similar results to other CNNs but require significantly more data to build from scratch. To achieve architectural diversity while managing data requirements, we employ a hybrid CNN-ViT instead.

Beyond architectural choices, the training procedure plays a critical role in handling the visual ambiguity and class confusion present in this dataset. Simard et al. [5] demonstrated the effectiveness of elastic distortion for handwritten recognition, while Lin et al. [6] introduced focal loss to prioritize hard examples during training — directly applicable to confusing character pairs, which were the primary source of error in this work. Shrivastava et al. [7] introduced Online Hard Example Mining (OHEM) as a complementary method for focusing on difficult examples, and Yun et al. [8] introduced CutMix, an augmentation technique that preserves local stroke structure more effectively than MixUp.

To further improve upon individual model performance, ensemble methods provide a complementary direction by combining the strengths of multiple architectures. Wolpert [9] introduced stacking as an effective ensemble method in which a meta-learner is trained on the predictions of base models. Because

Guo et al. [10] demonstrated that neural networks are consistently overconfident, we applied temperature scaling before meta-feature construction to calibrate confidence estimates.

2 BACKGROUND

2.1 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are deep learning architectures designed to classify high-dimensional data, particularly images. CNNs work by passing images through layers, where each layer applies a convolution operation followed by a nonlinear activation function. The convolution operation slides a kernel, or filter, across the input image computing the sum of element-wise products at each position.

$$G[m,n] = \sum_{j} \sum_{k} h[j,k]\, f[m-j,\, n-k] \qquad (1)$$

where *f* is the input image, *h* is the filter, and *G*[m, n] is the resulting feature map. Each layer computes an intermediate value *Z* and applies a nonlinear activation function *g* to produce the layer's output *A*.

$$Z^{[l]} = A^{[l-1]} * W^{[l]} + b^{[l]}, \qquad A^{[l]} = g(Z^{[l]}) \qquad (2)$$

where $W^{[l]}$ contains the learned filter weights, $b^{[l]}$ is a bias term, and * denotes convolution. The activation function used in this work is the Rectified Linear Unit (ReLU), defined as *g(a) = max(0, a)*, which introduces nonlinearity while remaining computationally efficient.

Because the same filter weights are shared across all spatial positions, CNNs require fewer parameters than fully connected networks, making them well suited for image data. However, CNNs still require large amounts of training data to learn robust and generalizable features. Without sufficient data, models tend to overfit, meaning the model fits the training data too closely and loses the ability to generalize to unseen examples. In this work, overfitting is mitigated through data augmentation [11] and dropout regularization (Figure 1).

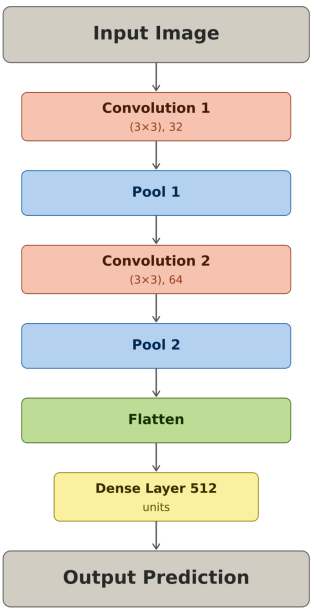

Figure 1: Convolution Neural Network Diagram

2.2 Vision Transformers

Vision Transformers (ViTs) are deep learning architectures that adapt the Transformer model [12], originally developed for natural language processing, to image classification [4]. Unlike CNNs, Vision Transformers do not apply convolutional operations. Instead, they treat an image as a sequence of fixed-size patches. An input image of size $H \times W \times C$ is divided into non-overlapping patches of size $P \times P$, yielding $N = HW/P^2$ patches. Each patch is flattened and projected to a D-dimensional embedding vector through a learned linear projection:

$$\mathbf{z}_0 = [\mathbf{x}_{\text{cls}};\ \mathbf{x}_p^1\mathbf{E};\ \mathbf{x}_p^2\mathbf{E};\ \cdots\ ;\ \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{\text{pos}} \quad (3)$$

where x^I_p is the i-th flattened patch, E is the learned patch embedding matrix, E_pos is the positional embedding, and x_cls is a learnable classification token prepended to the sequence. The core of the ViT is the Transformer encoder, which applies multi-head self-attention (MHSA) to the sequence of patch embeddings. The self-attention mechanism computes query, key, and value matrices from the input and produces a weighted combination of values:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \quad (4)$$

where Q, K, and V are the query, key, and value matrices respectively, and d_k is the dimension of the key vectors used for scaling. This permits every patch to attend to every other patch, enabling the model to capture global dependencies across the entire image — a property that CNNs cannot achieve through a single layer. However, because ViTs lack the spatial inductive biases inherent to CNNs, they require substantially more training data to learn meaningful representations. As a result, a pure ViT is not used in

this work. Instead, we utilize a hybrid CNN-ViT described in Section 3.4.2, which captures global context while remaining trainable on smaller datasets (Figure 2).

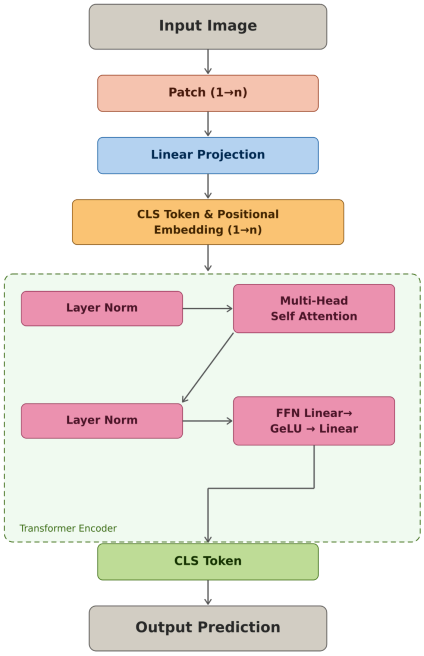

Figure 2: Vision Transformer Diagram

3 METHODOLOGY

3.1 Dataset

3.1.1 Overview

This dataset is a collection of handwritten characters and digits sourced from Kaggle, uploaded by MD Nurnabi Rana in 2025 [14]. It was created to support the development of new techniques for character recognition. Models trained on this dataset could have practical applications such as reading handwritten text in scanned documents, postal addresses, and digitized forms. This dataset was selected due to the lack of a strong existing model: no prior submission exceeded 85% accuracy, indicating genuine room for improvement.

3.1.2 Class Breakdown

The dataset contains 62 classes comprising the full standard English character set: Arabic digits (0–9), lowercase letters (a–z), and uppercase letters (A–Z), as shown in Table 1. Each class contains exactly 55 images, yielding 3,410 images in total. The uniform distribution across classes means the dataset is perfectly balanced, reducing the risk of model bias toward any particular character. Several class pairs are visually confusing: the numeral 0 and uppercase O share nearly identical shapes, as do 1, lowercase l, and uppercase I. Similarly, case pairs such as W/w and Z/z are frequently confused due to their identical geometric structure.

Table 1: Class Distribution Across Character Groups

Group	Classes	Count	Images
Arabic Numerals	0–9	10	550
Lowercase Alpha	a–z	26	1,430
Uppercase Alpha	A–Z	26	1,430
Total		62	3,410

3.1.3 Size and Split

The dataset contained only 3,410 images in total and included no predefined train, test, or validation splits, requiring us to stratify it ourselves. The original images were split into train, validation, and test sets prior to any augmentation. Augmentation was applied to the training set only, expanding it from 55 images per class to 1,500 per standard class and 2,200 for each hard class, ensuring no augmented copies of validation images appeared in the training set.

This dataset differs from MNIST and EMNIST in three key ways. First, it covers all 62 alphanumeric classes rather than digits alone (MNIST) or a larger but more data-rich set (EMNIST, which contains thousands of images per class). Second, with only 55 images per class, it presents a substantially more challenging low-data regime. Third, unlike EMNIST which was derived from the NIST Special Database and contains scanned historical forms, this dataset consists of freshly collected handwritten samples, making it a distinct benchmark rather than a subset or extension of existing standards.

3.1.4 Image Properties

All images are black and white PNG files with a resolution of 1200 × 900 pixels. To reduce computational cost, images were resized to 80 × 80 pixels. Given the simple structure — black characters on a plain white background — this downscaling retained all relevant features including edges, strokes, and overall shape. A representative sample is shown in Figure 3.

Figure 3: Example image from the dataset.

3.2 Class Imbalance and Hard-Class Boosting

Prior per-class accuracy analysis identified several poorly performing classes that are consistently confused, such as o/O/0 and 1/l. To address this, the total number of augmented training images for each hard class was increased from 1,500 to 2,200. This allows the model to learn the additional nuances that distinguish these classes.

3.3 Data Augmentation

Because the dataset contained only 55 images per class, data augmentation was necessary to produce a training set of sufficient size for deep learning. Augmentation involves applying random transformations to existing images to create new, slightly different samples, simultaneously increasing the effective dataset size and reducing overfitting. Augmentation was applied in two regimes: a stronger pipeline during training, and a lighter variant at inference referred to as Test-Time Augmentation (TTA).

3.3.1 Training Augmentation

Each synthesized training sample was produced by sampling an existing image and passing it through the following sequence of transformations. First, each image was padded by 8 pixels on each side using reflection padding, and a random crop within the padded region was taken, making the model invariant to small positional shifts. Second, contrast was randomly adjusted within [0.8, 1.2] and brightness within ±0.12. Third, grid distortion was applied with 50% probability, warping the image through a nonlinear displacement field computed from a 4 × 4 control grid with a limit of ±25%, mimicking natural stroke shape variation. Fourth, elastic distortion was applied with 60% probability using Gaussian-smoothed random displacement fields ($\alpha = 10$, $\sigma = 3$), producing local deformations that mimic the irregularity of handwriting [5]. Fifth, random erasing was applied with 40% probability, masking a randomly sized and positioned rectangular region with uniform noise, forcing the model to classify from partial observations rather than relying on any single stroke region [13]. Finally, uniform pixel noise of ±0.12 was added throughout.

For hard classes specifically, a morphological step was added with 35% probability — either erosion or dilation using a 2 × 2 elliptical structuring element — to simulate pen width variation and improve robustness to stroke thickness differences, which are a primary source of confusion in visually similar pairs such as 1/l/I and 0/O/o.

3.3.2 CutMix

CutMix was applied at the batch level with 50% probability [8]. A rectangular patch from one image was replaced with the corresponding patch from a randomly selected image in the batch, and the one-hot labels were mixed proportionally to the pixel areas:

$$\tilde{x} = \mathbf{M} \odot x_A + (1 - \mathbf{M}) \odot x_B$$
$$\tilde{y} = \lambda y_A + (1 - \lambda) y_B$$
(5)

where M is a binary mask, λ is the proportion of pixels retained from image A, and x_A, x_B, y_A, y_B are the image–label pairs. CutMix was chosen over MixUp because it preserves local spatial structure within each patch, which is important for character recognition where stroke geometry in specific sub-regions carries distinctive information. The mixing coefficient was sampled from a truncated normal distribution (mean 0.4, $\sigma = 0.15$, clipped to [0.1, 0.9]).

3.3.3 Hard-Class Oversampling in the Training Pipeline

Beyond increasing per-class image count, hard-class samples were oversampled a second time within the training pipeline. The training set was divided into a hard and a normal class pool, and samples were

drawn with a 2.5× higher sampling weight applied to hard examples. This ensured that hard-class examples appeared more frequently within each training batch, concentrating gradient signal on the most confusable characters throughout training.

3.3.4 Test-Time Augmentation

At inference, each test image was passed through the model 8 times with different transformations: a ±4 pixel reflect-padded random crop and contrast jitter within [0.9, 1.1]. No erasing or grid distortion was applied, to avoid altering structural content. The 8 output probability vectors were averaged, reducing prediction variance without introducing bias [15].

3.4 Model Architecture

Three models were combined in an ensemble to maximize accuracy. Diversity is essential for an effective ensemble: if all models share similar architectures, they will make errors on the same classes, rendering the ensemble ineffective. Using architecturally distinct models ensures that each captures different characteristics and produces complementary errors.

3.4.1 Residual Network

Residual Networks (ResNets) are a class of CNNs in which skip connections add the input of a block directly to its output, allowing the gradient signal to flow more effectively through the network. Each block need only learn the residual difference from the identity, which is easier to optimize than learning the full mapping from scratch. This enables training of much deeper networks without accuracy degradation [2]. In this work, we utilize a ResNet-18 as one of the three ensemble models, as it captures deep hierarchical features without information loss across layers (Figure 4).

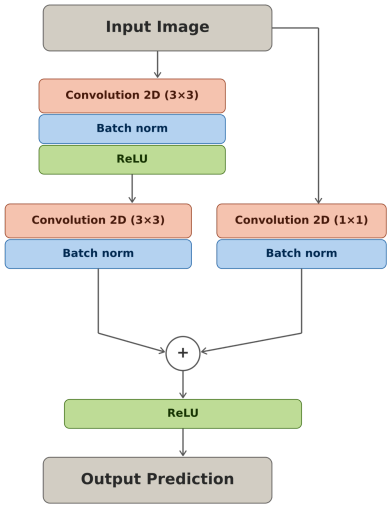

Figure 4: ResNet Architecture

3.4.2 Hybrid CNN-ViT

We utilized a custom hybrid architecture that combines a CNN backbone with a Transformer encoder. The backbone applies four stride-2 convolutional stages, reducing the 80×80 input image to a 5×5×512 feature map. This map is reshaped into 25 spatial tokens, projected to dimension 256, and prepended with a learnable CLS token. Positional embeddings are added, and 4 Transformer blocks process the sequence [4]. The self-attention mechanism attends over all spatial tokens simultaneously, giving the model access to the character's full global context, which is valuable for separating difficult classes that differ in overall structure rather than local stroke detail (Figure 5).

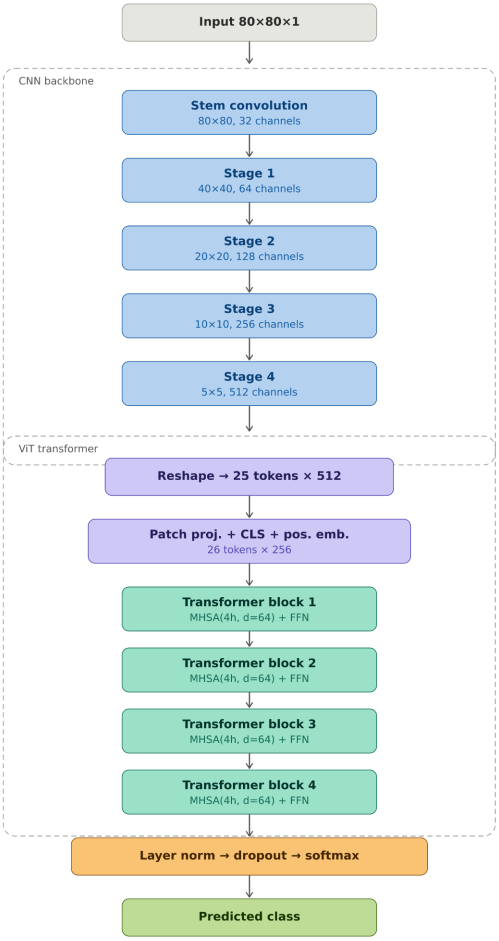

Figure 5: Custom CNN-ViT architecture.

3.4.3 CBAM CNN

The Convolutional Block Attention Module (CBAM) extends a residual block structure with an attention module consisting of two sequential components: a channel attention module and a spatial attention module [3].

The channel attention module determines which feature channels are most informative for classification. Given an intermediate feature map, both global average and global max pooling are applied, producing two channel-wise descriptor vectors. Both descriptors are passed through a shared two-layer MLP consisting of a fully connected layer that compresses the channel dimension by a factor of r, a ReLU activation, and a second fully connected layer that restores the original channel dimension $\mathbf{C}$. The outputs

of the two branches are summed and passed through a sigmoid to produce the channel attention map (Figure 6).

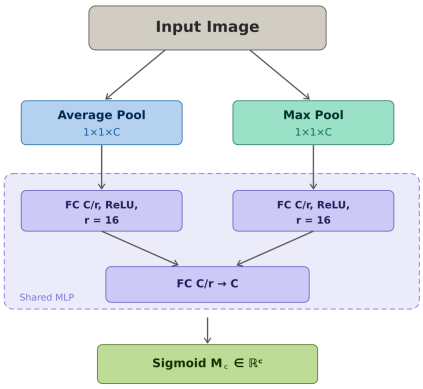

Figure 6: Channel attention module architecture.

The spatial attention module then identifies which spatial locations within the refined feature map are most relevant. Average and max pooling are applied along the channel axis, producing two single-channel maps. These are concatenated to form a 2-channel descriptor, which is passed through a 7×7 convolution to produce the spatial attention map (Figure 7).

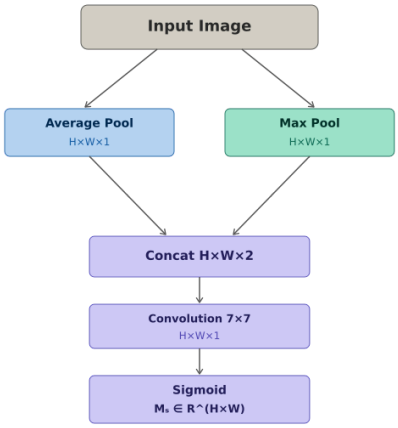

Figure 7: Spatial attention module architecture.

The final output is obtained by element-wise multiplication of the spatial attention map with the channel-refined feature map, directing the network toward the most discriminative spatial regions.

3.5 Training Procedure

Several techniques were applied during model training to maximize validation accuracy, including focal loss, online hard example mining, and warmup cosine decay.

3.5.1 Focal Loss

Unlike standard cross-entropy (CE) loss, which treats all samples equally, focal loss introduces a modulating factor to the CE formula [6]:

$$FL(p_t) = -(1 - p_t)^{\gamma} \log(p_t) \quad (6)$$

where *p_t* is the model's predicted probability for the correct class and γ is the focusing parameter. Focal loss downweights easy examples such as 2, A, or i, and directs training toward frequently misclassified characters such as o, 1, or W. This allows the model to improve accuracy on hard classes without substantially reducing accuracy on easier ones.

3.5.2 Online Hard Example Mining

Online Hard Example Mining (OHEM) [7], similarly to focal loss, prioritizes training on examples causing high loss. OHEM operates through the following algorithm:

1. Run a forward pass through all samples in the batch.
2. Compute per-sample loss for each example.
3. Select the top κ examples with the highest loss.
4. Backpropagate only through those κ samples.

Unlike focal loss, which reweights all samples continuously, OHEM applies a hard threshold, either including or excluding examples entirely. Because they operate differently, the two methods are complementary and can be combined for further improvement on difficult classes.

3.5.3 Warmup Cosine Decay

Warmup cosine decay was used as the learning rate scheduler. Starting from a base learning rate of 3×10^{-4}, the rate was linearly increased from epoch 1 to epoch 10, then decreased following a cosine curve to a minimum of 1×10^{-6}. For the CNN-ViT hybrid, the base learning rate was reduced to 1×10^{-4}, as Transformer architectures are sensitive to large learning rates before positional embeddings and attention weights have stabilized.

This scheduler was preferred over reactive approaches such as *ReduceLROnPlateau*, which depend on validation signal and apply abrupt reductions that can destabilize training. The cosine decay profile provides a smooth descent that decreases quickly when far from the optimum and slows near convergence, matching the geometry of the loss landscape more efficiently. Early stopping with a patience of 8 epochs was applied based on validation accuracy. The resulting learning rate schedule is shown in Figure 8.

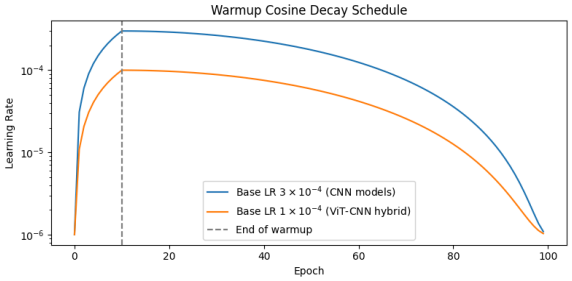

Figure 8: Warmup cosine decay learning rate schedule.

3.6 Stacking Ensemble

The models were combined through a stacking procedure [9] using a logistic regression meta-learner. Three techniques were applied: out-of-fold (OOF) prediction, temperature scaling, and meta-feature construction.

3.6.1 Out-Of-Fold Prediction

Out-of-fold prediction was used to generate meta-learner training features without data leakage. The training set was partitioned into 5 stratified folds, each containing a proportional representation of all 62 classes. For each fold, all 3 models were run in inference mode on the held-out images. Each image was passed through the model 8 times with different augmentations — a process called test-time augmentation (TTA) [15] — and the 8 resulting probability vectors were averaged to produce a more reliable prediction. After cycling through all 5 folds, every training image had been predicted exactly once, as shown in Figure 9. These predictions became the training data for the meta-learner.

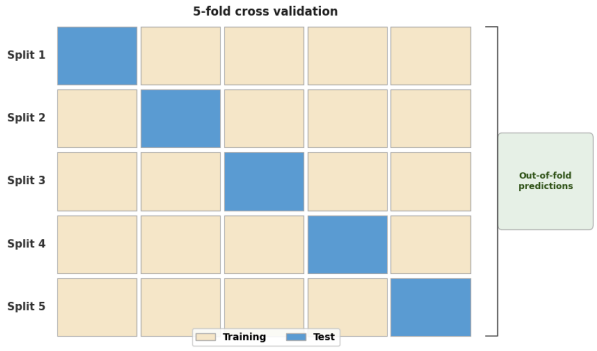

Figure 9: Out-of-fold prediction visualization

After cycling through all 5 folds, every training image had been predicted exactly once. The resulting OOF accuracies across folds are shown in Table 2. These predictions became the training data for the meta-learner.

Table 2: OOF Prediction across folds

Fold	Accuracy
1	92.09%
2	92.01%
3	92.12%
4	91.95%
5	92.46%

3.6.2 Temperature Scaling

Since neural networks tend to be overconfident, the meta-learner would otherwise receive distorted confidence estimates. Temperature scaling [10] corrects this by introducing a scalar **T** that adjusts the sharpness of each model's output probability distribution. The optimal value of **T** was selected by minimizing calibration error on the OOF predictions.

3.6.3 Meta-Feature Construction

After calibrating the probability vectors via temperature scaling, those vectors were converted to log-probability space, which better matches the linear assumptions of the meta-learner. The log-probability vectors for each model were then standardized to prevent a more confident model from dominating the meta-learner by virtue of larger output magnitudes alone. The standardized blocks were concatenated, yielding one row per training image with 186 features in total: 62 log-probabilities from each of the 3 models.

3.6.4 Meta-Learner Training and Selection

A logistic regression was chosen as the meta-learner. This linear model takes the 186 features described in Section 3.6.3 and learns a weighted combination to predict the final class. It was selected for its computational efficiency and low susceptibility to overfitting on the meta-feature space. The model has one tunable hyperparameter **C**, controlling regularization strength, with smaller values constraining the model more tightly. The optimal value of **C** was selected by evaluating 6 candidates {0.05, 0.1, 0.3, 0.5, 1, 2} via a second round of 5-fold cross-validation on the OOF meta-features. The final meta-learner was trained using the value of **C** that produced the highest average accuracy.

4 RESULTS

"The ensemble model [16] that resulted achieved 91.20% accuracy, successfully outperforming the previous baseline [17], which had achieved a consistent 84.60%. The baseline approach employed a pretrained VGG-19 fine-tuned on this dataset using standard cross-entropy loss, without any ensemble strategy, hard-class augmentation, or targeted training techniques for visually ambiguous character pairs. The proposed method builds directly on this foundation, replacing the single fine-tuned model with a stacking ensemble of three architecturally diverse models trained with focal loss, OHEM, and an aggressive augmentation pipeline specifically designed for the confusable character pairs that limited the baseline's performance."

4.1 Model Performance

The individual models performed as follows: the ResNet-18 achieved the lowest accuracy at 84.16%; the CBAM CNN achieved 88.29%; and the hybrid CNN-ViT performed best among the base models at 89.31%. The stacked ensemble using the meta-learner reached 91.2%, while a simple average of the three models reached only 89.2% (Figure 10).

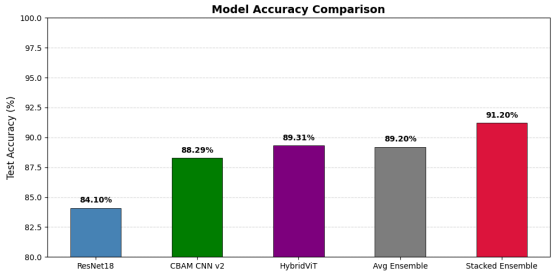

Figure 10: Model Accuracy Comparison.

4.2 Individual Class Performance

While overall accuracy reached 91.2%, performance varied considerably across the 62 classes. The most difficult characters were those involved in visually ambiguous pairs, where structural similarity made consistent discrimination difficult. Table 3 lists the five worst-performing individual classes; 1 and W were especially difficult, each falling below 50% accuracy.

Table 3: Top 5 Worst Performing Classes

Character	Accuracy
1	44.0%
W	46.0%
0	62.0%
w	63.6%
z	63.6%

The errors were not randomly distributed across all class pairs; rather, they clustered strongly around specific confusable pairs. Table 4 shows the five most frequently confused character pairs. The W/w pair alone accounted for 131 errors, more than any other pair by a wide margin, reflecting the fact that uppercase and lowercase forms are geometrically identical at the same scale.

Table 4: Top 5 Most Confused Pairs

Pair	Errors
W ↔ w	131
V ↔ v	81
I ↔ l	73
O ↔ 0	68
S ↔ s	66

All five of the most confused pairs involve either case ambiguity (W/w, V/v, S/s) or characters that share nearly identical stroke geometry (I/l, O/0). Because all images are resized to the same fixed resolution, relative scale — the primary cue humans use to distinguish uppercase from lowercase — is entirely removed. This structural limitation of the dataset is the dominant source of residual error.

4.3 Training Dynamics

Training and validation accuracy and loss curves across all three models are shown in Figures 11 and 12.

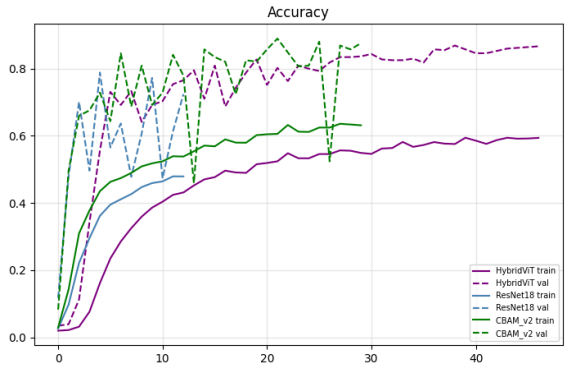

Figure 11: Training and Validation Accuracy across Models.

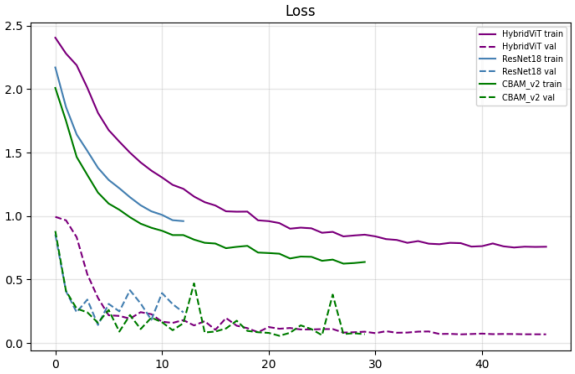

Figure 12: Training and Validation Loss across Models.

5 DISCUSSION

5.1 Ensemble Behavior

The gap between the simple average (89.2%) and the stacked ensemble (91.2%) highlights the value of the meta-learner. A simple average treats all three models equally; because the ResNet-18 was notably weaker (84.16%) than the other two, its predictions dragged down the average. The logistic regression meta-learner, trained on out-of-fold predictions, learned to down-weight ResNet outputs and place greater emphasis on CBAM and CNN-ViT predictions. This demonstrates that stacking is most beneficial when base models differ substantially in quality, as it allows the meta-learner to compensate for weaker components rather than averaging them in equally.

5.2 Persistent Confusion on Hard Classes

Despite the additional training images, oversampling, focal loss, and OHEM applied to hard classes, the worst-performing characters (Table 2) still achieved substantially lower accuracy than the overall average. The dominant source of error, as seen in Table 3, is case ambiguity: character pairs such as *W/w*, *V/v*, and *S/s* are geometrically identical at the same scale. Because all dataset images are resized to the same fixed resolution, relative size — the natural cue humans use to distinguish uppercase from lowercase — is entirely eliminated. This is a fundamental limitation of the dataset rather than a modelling failure, and it cannot be fully overcome through augmentation or loss weighting alone.

Similarly, 1/l/I confusion is extremely difficult even for humans when characters are not drawn with sufficient distinction. The residual error on these pairs likely represents a near-irreducible floor given the available data.

5.3 Effect of Key Techniques

5.3.1 Data Augmentation

Data augmentation was applied primarily to increase the effective training set size and reduce overfitting. This was successful: training accuracy never substantially exceeded validation accuracy throughout training, indicating that the models generalized well rather than memorizing training examples (Figures 11 and 12). The augmentation pipeline was particularly important given the dataset's small size of only 55 images per class; without it, all three models would have been at severe risk of overfitting.

5.3.2 Focal Loss

Focal loss produced a substantial improvement on the worst-performing classes. In the baseline model trained without focal loss, the five worst classes all fell below 46% accuracy (Table 5), with the lowest at 36.4%.

Table 5: 5 Worst Classes Without Focal Loss

Class	Accuracy
l (lowercase L)	36.4%
W	36.4%
5	45.5%
I (uppercase I)	45.5%
X	45.5%

With focal loss, the worst performers improved substantially, and several characters that previously appeared in the bottom five (notably 5 and X) no longer do. This confirms that focal loss successfully redirected training signals toward the most ambiguous examples without harming performance on easy classes.

5.3.3 Online Hard Example Mining

Introducing OHEM on top of focal loss produced a further measurable improvement. Table 6 shows the five worst classes with focal loss but without OHEM.

Table 6: Worst Performing Classes: Focal Loss Only (No OHEM)

Class	Accuracy
9	36.4%
K	45.5%
S	45.5%

O	45.5%
x	54.5%

The average accuracy of the five worst classes with both OHEM and focal loss is more than 10 percentage points higher than with focal loss alone, confirming the complementary benefit of combining both methods. Focal loss provides continuous reweighting across all samples, while OHEM applies a hard selection threshold that forces the model to focus exclusively on the most difficult examples in each batch.

5.3.4 Progression of Model Iterations

Three earlier models were evaluated before arriving at the final ensemble. The baseline Keras Sequential model reached 79.0%: without skip connections, gradients degraded as depth increased, and the absence of any attention mechanism meant all spatial regions were treated equally, leaving hard classes with poor performance. An EfficientNet-inspired model improved on this, reaching 81.0% by introducing depthwise separable convolutions and squeeze-and-excitation channel attention, but lacked spatial attention and residual connections, limiting its expressive capacity on the small dataset. The standalone ResNet-18 reached 84.0% by adding skip connections and enabling deeper feature learning. Each step in this progression demonstrated that architectural improvements alone provided diminishing returns, motivating the move to a diverse ensemble with targeted training strategies.

5.4 Limitations

Two main limitations affect the results reported here. First, the majority of training data consisted of augmented images derived from a small set of 55 originals per class. While the augmentation pipeline introduced substantial variation, all synthesized samples share underlying features with their source images; the effective diversity of the training set is lower than its size suggests. Second, uniform image resizing removes character scale as a discriminating feature, making it structurally impossible for the model to distinguish case pairs that differ only in size. These two limitations are connected: even an ideal model trained on this data cannot fully resolve scale-ambiguous pairs.

A third consideration concerns the choice of training strategy. Although pretraining on MNIST or EMNIST followed by fine-tuning is a common approach for low-data settings, it was not adopted here for two reasons. First, MNIST covers only 10 digit classes and EMNIST, while broader, uses a different image distribution (scanned historical forms normalized differently from this dataset's freshly collected samples), meaning pretrained weights may introduce domain mismatch rather than useful priors. Second, the aggressive augmentation pipeline — including elastic distortion, CutMix, and morphological transforms — was designed to simulate the same diversity that pretraining would otherwise provide, making it a more controlled and dataset-specific alternative.

Additionally a third limitation affects the generalization of these results. The dataset does not document the numbers of individual writers that contributed to the dataset. However, visual inspection of the images indicate different sizes, styles and shapes as shown in Figure 13.

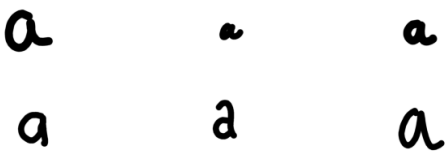

Figure 13: Diversity of handwriting on the lowercase letter a.

6 CONCLUSION

This work proposed a stacking ensemble of three architecturally diverse models for 62-class handwritten character recognition, achieving 91.2% accuracy and surpassing the previous best of 84.6% by more than 6 percentage points. The results demonstrate that architectural diversity, combined with focal loss, OHEM, and a logistic regression meta-learner, can significantly improve accuracy on visually similar character classes. Analysis of the remaining errors reveals that the primary bottleneck is case ambiguity arising from the uniform image rescaling in the dataset, which eliminates size as a discriminating feature.

6.1 Future Improvements

Future work could address the identified limitations in two principal ways. First, constructing a more diverse dataset under realistic conditions — incorporating background noise, ink bleed, or characters sourced from real-world scanned documents and street signs — would reduce the dependence on augmented copies of a small original set and expose the model to the natural variation encountered in deployment. Second, constraining characters to a scaled reference, such as ruled lines of fixed width, would restore relative size as a discriminating feature and is expected to substantially improve accuracy on case-ambiguous pairs such as *W/w*, *V/v*, and *O/0*, which currently account for the largest share of classification errors. Taken together, these improvements would address both the data diversity limitation and the structural ambiguity limitation identified in this work.

References

1] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: Extending MNIST to Handwritten Letters," IJCNN, pp. 2921–2926, 2017. https://doi.org/10.1109/IJCNN.2017.7966217

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," CVPR, pp. 770–778, 2016. https://arxiv.org/abs/1512.03385

[3] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional Block Attention Module," ECCV, pp. 3–19, 2018. https://arxiv.org/abs/1807.06521

[4] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," ICLR, 2021. https://arxiv.org/abs/2010.11929

[5] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis," ICDAR, pp. 958–963, 2003. https://doi.org/10.1109/icdar.2003.1227801

[6] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection," ICCV, pp. 2980–2988, 2017. https://arxiv.org/abs/1708.02002

[7] A. Shrivastava, A. Gupta, and R. Girshick, "Training Region-based Object Detectors with Online Hard Example Mining," CVPR, pp. 761–769, 2016. https://arxiv.org/abs/1604.03540

[8] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features," ICCV, pp. 6022–6031, 2019. https://arxiv.org/abs/1905.04899

[9] D. H. Wolpert, "Stacked Generalization," Neural Networks, vol. 5, no. 2, pp. 241–259, 1992. http://dx.doi.org/10.1016/S0893-6080(05)80023-1

[10] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," ICML, pp. 1321–1330, 2017. https://arxiv.org/abs/1706.04599

[11] C. Shorten and T. M. Khoshgoftaar, "A Survey on Image Data Augmentation for Deep Learning," J. Big Data, vol. 6, no. 1, p. 60, 2019. https://doi.org/10.1186/s40537-019-0197-0

[12] A. Vaswani et al., "Attention Is All You Need," NeurIPS, vol. 30, 2017. https://arxiv.org/abs/1706.03762

[13] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random Erasing Data Augmentation," AAAI, vol. 34, no. 07, pp. 13001–13008, 2020. https://doi.org/10.1609/aaai.v34i07.7000

[14] MD Nurnabi Rana, "Handwritten Character Recognition Dataset," Kaggle, 2025. https://www.kaggle.com/code/mdnurnabirana/alphabet-digit-recognition-using-cnn/notebook

[15] D. Shanmugam, D. Blalock, G. Balakrishnan, and J. Guttag, "Better Aggregation in Test-Time Augmentation," arXiv:2011.11156, 2020. https://arxiv.org/abs/2011.11156

[16] Z. Kalavade, "Stacking Diverse Deep Architectures for Handwritten Character Recognition," Kaggle Notebook, 2025. https://www.kaggle.com/code/zubinkalavade/ensemble-model-hcr

[17] MD Nurnabi Rana, "Handwritten Character Recognition Notebook," Kaggle, 2025. https://www.kaggle.com/code/mdnurnabirana/alphabet-digit-recognition-using-cnn/notebook