future internet

MDPI

Article

Transfer Learning Strategies for Comic Character Recognition in Low-Data Regimes: A Comparative Study

Marco Parrillo *,†, Luigi Laura † and Alessandro Manna †

Faculty of Engineering, International Telematic University Uninettuno, 00186 Rome, Italy; luigi.laura@uninettunouniversity.net (L.L.); alemanna22@gmail.com (A.M.)

* Correspondence: marcoparrillo@gmail.com

† These authors contributed equally to this work.

Abstract

Image classification in low-data regimes remains a challenging problem, particularly in stylized visual domains where intra-class similarity and inter-class feature overlap limit discriminative capacity. This study presents a systematic evaluation of regularization and transfer learning strategies for multi-class comic character recognition under constrained data conditions. Four convolutional architectures are compared: (i) a baseline CNN trained from scratch, (ii) a regularized CNN incorporating data augmentation, dropout, and early stopping, (iii) a pretrained ResNet-50 used as a fixed feature extractor, and (iv) a partially fine-tuned ResNet-50 with selective layer unfreezing. Experiments are conducted on a custom four-class dataset exhibiting moderate class imbalance, evaluated using both a fixed 70/20/10 split and 5-fold cross-validation to assess generalization stability. Results indicate that shallow CNN architectures suffer from substantial overfitting, even when regularization is applied, whereas transfer learning significantly improves macro-averaged F1-score and out-of-distribution detection performance. Cross-validated results, the primary basis for inference given the dataset scale, show that both ResNet-50 strategies achieve equivalent mean accuracy of 95.0% (SD: ±0.4% for feature extraction, ±0.8% for fine-tuning; paired $t = 0.00$, $p = 1.000$), while shallow CNN architectures reach only 81–87%. Under a single fixed 70/20/10 partition ($n = 69$ test samples, 95% CI: ±9–12%), fine-tuning nominally reaches 98.5%; crucially, cross-validation deflates this figure to parity with feature extraction, confirming it reflects favorable partitioning rather than genuine architectural superiority. The primary finding is therefore that frozen ResNet-50 feature extraction is the recommended strategy: it matches fine-tuning in cross-validated generalization while requiring 15× fewer trainable parameters and exhibiting lower fold-to-fold variance. The findings demonstrate that pretrained deep residual representations transfer effectively to stylized comic imagery and that evaluation protocol selection critically impacts perceived performance in small datasets. These results provide practical guidelines for robust model selection in domain-specific, limited-data image classification tasks.

Keywords: transfer learning; Convolutional Neural Networks (CNNs); ResNet-50; feature extraction; fine-tuning; low-data regime; overfitting; cross-validation; comic character recognition; data augmentation

Academic Editors: Wenli Zhang and Zhuwei Wang

Received: 4 March 2026
Revised: 20 March 2026
Accepted: 26 March 2026
Published: 2 April 2026

1. Introduction

The rapid evolution of Computer Vision (CV) has established Convolutional Neural Networks (CNNs) as the gold standard for automated image recognition, while deep

learning models have achieved human-parity performance on massive datasets like ImageNet; their efficacy often diminishes when applied to low-data regimes or specialized visual domains, such as stylized comic art, where features differ significantly from natural photographic imagery.

1.1. The Challenge of Stylized Visual Domains

Character recognition in comic media presents unique architectural challenges. Unlike natural objects, comic characters are defined by high-contrast line work, exaggerated features, and a lack of realistic textures. Furthermore, the "intra-class similarity" (different poses of the same character) and "inter-class feature overlap" (similar drawing styles across different characters) require a model with high discriminative capacity. In practical applications, such as automated archival or comic generation pipelines, datasets are often constrained in size, leading to the risk of catastrophic overfitting.

1.2. Research Gap and Motivation

Current literature extensively covers regularization techniques like Dropout and Data Augmentation, in general contexts. However, there is a lack of comparative research specifically evaluating the transition from custom lightweight CNNs to deep residual architectures when fine-tuned for stylized line art under limited data conditions. Most existing studies focus either on training from scratch or full transfer learning, without a systematic analysis of where the "pivot point" for layer unfreezing should occur to maximize stability and accuracy.

1.3. Contributions of This Study

This paper addresses these challenges by providing a systematic evaluation of optimization strategies for comic character classification. We compare four distinct architectural approaches:

Baseline CNN: A five-layer architecture trained from scratch to establish a performance floor.

Regularized CNN: An optimized version incorporating Dropout, Early Stopping, and Data Augmentation to mitigate overfitting.

ResNet-50 Feature Extractor (FE): Utilizing pre-trained ImageNet weights to evaluate the transferability of natural features to stylized domains.

Fine-Tuned ResNet-50 (FT): A block-aligned selective unfreezing approach targeting the conv5_x residual block (layers 140–172), motivated by its role in high-level semantic feature extraction and validated against the feature extraction baseline via 5-fold cross-validation to specific comic art.

To ensure methodologically sound comparisons given the constrained dataset scale, we move beyond simple validation splits and implement a 5-fold cross-validation framework. Our results demonstrate that while custom-tuned models offer flexibility, the fine-tuned ResNet-50 architecture provides a superior balance of accuracy (95.0%) and architectural stability (Variance: 0.14).

It is important to clarify what this study contributes beyond the well-established finding that transfer learning can improve classification accuracy. The benefit of pretrained convolutional features for downstream vision tasks has been demonstrated across many domains [1–4], and the present work does not claim otherwise. Rather, the specific contributions of this paper lie at the intersection of three elements that, individually, have precedents in the literature but have not previously been combined in a single controlled study on stylized imagery:

1. **A direct, cross-validated comparison of frozen versus partially fine-tuned transfer learning on comic imagery.** Prior computational studies of comics and manga [5,6] relied predominantly on hand-crafted features (HOG, SIFT) or single-strategy deep learning evaluated on fixed splits. Dunst et al. [7] demonstrated that pretrained deep features capture meaningful structure in graphic narratives but did not compare frozen and fine-tuned strategies or employ cross-validated evaluation. The present study isolates exactly this comparison under controlled conditions—same backbone, same dataset, same evaluation protocol—and reports a null result: frozen feature extraction and partial fine-tuning achieve statistically indistinguishable cross-validated accuracy (both 95.0%; paired $t = 0.00$, $p = 1.000$), while feature extraction exhibits lower fold-to-fold variance (0.14 vs. 0.58) and requires 15$\times$ fewer trainable parameters. Null results of this kind carry direct practical value: they tell practitioners working with small stylized corpora that the additional complexity and computational cost of fine-tuning is not justified at these data scales.
2. **A methodological case study in evaluation-protocol sensitivity for small datasets.** Under a fixed 70/20/10 partition, fine-tuning nominally achieves 98.5% accuracy, a figure that, taken at face value, would strongly favor fine-tuning over feature extraction. Cross-validation deflates this to 95.0%, identical to feature extraction, revealing the fixed-split figure as a partition artifact rather than a genuine architectural advantage. This discrepancy is not merely a statistical caveat; it is one of the central findings of the paper. Small custom datasets with single-split evaluations remain common in comics and manga recognition [5,6,8], and this study provides a concrete, documented example of how such evaluations can produce misleading architectural rankings. The practical recommendation that follows, that cross-validation is essential, not optional, in low-data stylized-domain classification, is generalizable beyond the specific dataset used here.
3. **Empirical evidence that ImageNet features transfer effectively to high-contrast line art despite a substantial domain gap.** While Kornblith et al. [2] established that stronger ImageNet models generally transfer better across downstream tasks, color gradients, and photorealistic lighting, features that dominate ImageNet representations, are largely absent or conventionalized. The 95% cross-validated accuracy achieved by a frozen ResNet-50 backbone on the Dilbert corpus provides concrete empirical support for the hypothesis, advanced by Kornblith et al. [2] at a theoretical level, that mid-level structural features (edges, shape boundaries, spatial arrangements) rather than photorealistic appearance drive transferability to visually dissimilar domains. This finding extends the empirical boundary of the transfer learning literature into a domain that differs from natural photographs more radically than the clinical, satellite, or fine-grained recognition tasks typically studied.

Taken together, these contributions position the present work not as a demonstration that transfer learning outperforms training from scratch, a finding that would indeed lack novelty, but as a systematic investigation of *how* and *how much* different transfer strategies differ when applied to a visual domain at the extreme end of the photographic–stylized spectrum, evaluated with the statistical rigor that the small dataset demands.

1.4. Paper Structure

The remainder of this paper is organized as follows: Section 2 reviews relevant literature on regularization and transfer learning. Section 3 details the methodology, including dataset preparation and architectural specifications. Section 4 presents the experimental results and statistical analysis, while Section 5 discusses the implications of the findings and suggests avenues for future research.

2. Related Work

The literature relevant to this study spans four interconnected areas: regularization techniques for overfitting control, transfer learning and feature transferability, recognition in stylized visual domains (with particular focus on comic and manga imagery), and the comparative performance of modern backbone architectures in low-data settings. Rather than reviewing each work in isolation, this section situates each contribution within the methodological choices made in the present study and identifies the specific gaps that motivated our experimental design.

2.1. Overfitting Control: Regularization and Early Stopping

Excessive model capacity relative to training data is a well-documented source of overfitting in deep networks. Caruana et al. [9] and Srivastava et al. [10] demonstrated that backpropagation with early stopping can yield competitive generalization even in over-parameterized networks, because overfitting is non-uniform across the parameter space and early termination implicitly reduces effective capacity. Crucially, their comparison with conjugate gradient optimization showed that faster convergence does not imply better generalization, a finding directly relevant to the choice of ADAM in the present study, which trades convergence speed against generalization risk.

Dropout addresses a complementary problem: co-adaptation among neurons, which causes networks to rely on spurious feature correlations present only in training data. By randomly deactivating units at each forward pass, dropout forces the network to develop redundant, independent representations. The theoretical effect is an approximation of ensemble averaging across an exponential number of thinned sub-networks. On the small dataset used here (682 images), both early stopping and dropout are not merely useful regularizers but near-essential constraints: without them, even a shallow five-layer CNN has sufficient capacity relative to the training set size to memorize the training partition entirely, as confirmed by the diverging validation loss observed in our baseline results.

Thanapol et al. [11] studied the interaction between data augmentation and dropout on intentionally constrained image datasets, finding that augmentation injected at a specific epoch (epoch 30 in their setting) combined with width and height shifts yields the best mitigation of overfitting. Their work highlights an important subtlety: augmentation and dropout are not independently additive; their combined effect depends on scheduling. This motivates our choice to apply augmentation at the input layer rather than mid-training and to stack dropout after both the flatten and dense layers, a configuration consistent with their recommended pipeline.

2.2. Transfer Learning: Foundations and Transferability

The theoretical basis for transfer learning in deep convolutional networks was established empirically by Yosinski et al. [1], who showed that early-layer features (edges, color gradients, simple textures) are highly general and transfer effectively across image domains, while features in the final layers become increasingly task-specific. This gradient of transferability provides the direct justification for selective layer unfreezing: layers encoding domain-general representations should be preserved, while task-specific upper layers should be adapted. In the present study, this principle is operationalized by freezing the first 140 of ResNet-50's 176 layers—corresponding to the stem through conv4_x blocks—and allowing only the conv5_x block to update during fine-tuning. Kornblith et al. [2] provided large-scale empirical support for the practical utility of ImageNet-pretrained representations by demonstrating that stronger ImageNet accuracy reliably predicts better transfer to diverse downstream tasks, including domains that differ visually from natural photographs. This finding is particularly relevant here because comic

imagery, characterized by high-contrast line art and exaggerated features, diverges substantially from ImageNet's photographic distribution. The positive transfer we observe despite this domain gap is consistent with Kornblith et al.'s [2] conclusion that mid-level features capture structural image properties rather than photorealistic appearance. Pan and Yang [3] provide the broader theoretical framework situating our experimental design within the transfer learning taxonomy. Their distinction between inductive, transductive, and unsupervised transfer directly maps onto the feature extraction versus fine-tuning comparison central to this study. Their survey identifies the final task-specific layers as the appropriate target for domain adaptation under label scarcity, precisely the strategy evaluated in Section 3.6.

The effectiveness of pretrained backbones as frozen feature extractors for small-scale, multi-class image classification tasks has been further confirmed in the context of accessible transfer-learning frameworks [4], reinforcing the practical value of the feature extraction strategy adopted in Section 3.5.

2.3. Character and Object Recognition in Stylized Visual Domains

Automated analysis of comic and manga imagery has emerged as a distinct subfield of computer vision, motivated by the growing demand for digital archiving and content-based retrieval of graphical narrative corpora. Augereau et al. [5] provide a comprehensive survey of computational approaches to comics research, identifying character detection as a central unresolved challenge. Their taxonomy distinguishes between panel segmentation, speech bubble extraction, and character identity assignment, the last of which corresponds directly to the classification task studied here. Their survey further observes that most published methods rely on hand-crafted features (HOG, SIFT) applied to small proprietary datasets, leaving the question of deep learning transferability largely unaddressed at the time of writing.

Rigaud et al. [6] addressed comic character detection using a combination of connected component analysis and supervised classification, achieving reliable identification of recurring characters in Franco-Belgian comic strips. Their work demonstrated that the high-contrast, low-texture nature of comic imagery, which might be expected to hinder feature-based approaches, can in fact be exploited as a discriminative signal when characters are visually distinctive. However, their method requires explicit panel segmentation as a preprocessing step and does not generalize to characters with overlapping visual descriptors, a limitation directly relevant to the Dilbert/Boss confusion pattern observed in our baseline CNN results.

Dunst et al. [7] introduced the Graphic Narrative Corpus and demonstrated that deep learning models pretrained on natural images can extract meaningful representations from graphic novel pages despite the visual domain gap. Their findings provide further external validation for the transfer learning approach adopted in the present study, and their openly accessible corpus represents a potential venue for expanding the evaluation beyond the single-domain Dilbert dataset used here.

Taken together, this body of work situates the present study within an active research area while highlighting a consistent gap: no published work has systematically compared frozen versus partially fine-tuned ResNet architectures for comic character classification under cross-validated evaluation on a constrained dataset. This is the specific contribution the present paper aims to provide.

2.4. Modern Backbone Architectures and Their Transfer Properties

The ResNet-50 architecture used in this study, while robust and well-validated, represents a single point in a much larger design space of pretrained convolutional and attention-

based backbones. Understanding where ResNet-50 sits relative to more recent architectures is important for contextualizing the reported accuracy figures and for identifying whether the findings generalize beyond this specific backbone.

Tan and Le [12] introduced EfficientNet, a family of models that achieve significantly higher ImageNet accuracy than ResNet-50 at lower parameter counts through compound scaling of depth, width, and resolution. EfficientNet-B0, the smallest variant, has 5.3 M parameters compared to ResNet-50's 25.6 M, yet achieves approximately 4 percentage points higher top-1 accuracy on ImageNet. In low-data settings, this parameter efficiency is particularly valuable: fewer trainable parameters reduce the risk of overfitting when labeled data are scarce, making EfficientNet a natural comparator for the feature extraction strategy evaluated here.

Dosovitskiy et al. [13] demonstrated that pure transformer architectures, specifically the Vision Transformer (ViT), can match or exceed convolutional networks on image classification tasks when pretrained on sufficiently large datasets (JFT-300 M or ImageNet-21k). However, ViT's self-attention mechanism requires large-scale pretraining to generalize; on small datasets without such pretraining, it underperforms ResNet-based architectures substantially. This trade-off is directly relevant to the present study: a frozen ViT backbone pretrained on ImageNet-21k might offer richer semantic representations for comic imagery, but standard ViT-B/16 pretrained on ImageNet-1k alone is unlikely to confer advantages over ResNet-50 in a 682-image corpus.

Liu et al. [14] proposed the Swin Transformer, a hierarchical vision transformer using shifted windows that produces multi-scale feature maps compatible with the same downstream heads used for convolutional backbones. Swin-T achieves 81.3% top-1 accuracy on ImageNet with 28 M parameters and has demonstrated strong transfer to fine-grained recognition tasks. Its hierarchical structure makes it particularly appropriate as a future backbone for the character recognition task, as the multi-scale representation aligns with the need to distinguish characters that share global structure (e.g., humanoid silhouette) but differ at the level of localized features (e.g., Dilbert's tie versus the Boss's hair).

The absence of backbone comparisons beyond ResNet-50 in the present study is acknowledged as a limitation. The results reported here should be interpreted as characterizing the behavior of one specific pretrained architecture on the Dilbert corpus, rather than as general claims about transfer learning strategies for comic character recognition.

2.5. Domain-Specific Applications Motivating This Study

Two prior studies from the same research group provide the direct application context for this work and warrant discussion both as motivation and as a baseline for situating the present results.

Camerlingo et al. [8] developed an automated dialogue generation pipeline for Dilbert strips, using character detection as a prerequisite step within the same corpus examined here. That work establishes the practical demand for reliable character classification in the Dilbert domain and provides the downstream application context that motivates the classification accuracy thresholds targeted in this study. However, character detection in that pipeline was treated as a solved prerequisite; the present study addresses the classification component in isolation and with systematic evaluation, filling a gap left open by that earlier work.

2.6. Transfer Learning Across Visually Dissimilar Domains

A critical empirical question underlying this study is whether ImageNet-pretrained residual features, learned entirely on natural photographs, can generalize to the highly

stylized domain of comic imagery. Three lines of evidence support a positive answer, though each carries caveats.

As discussed in Section 2.2, Kornblith et al. [2] demonstrated that ImageNet accuracy is a reliable predictor of transfer performance across diverse vision tasks. Runzer-Colmenares et al. [15] provided domain-specific evidence by showing that ResNet architectures pretrained on ImageNet transfer effectively to clinical facial expression analysis, a domain differing from natural photographs in lighting, subject framing, and emotional expressivity. This corroborates the view that mid-level convolutional features (textures, shape boundaries, spatial relations) are sufficiently domain-agnostic to serve as useful representations even when the target domain is visually non-naturalistic.

Nonetheless, it would be premature to conclude that all pretrained features transfer equally well to comic imagery. Stylized images differ from both natural photographs and clinical facial images in a fundamental way: they deliberately abstract away from photorealistic appearance. Edges and shapes are preserved, but color gradients, lighting, and texture, signals that appear prominently in ImageNet representations are largely absent or conventionalized. The degree to which this abstraction limits feature transferability is an empirical question, and one that the present study directly addresses by comparing frozen and fine-tuned ResNet representations on a comic character classification task.

3. Experiment

3.1. Dataset

The starting dataset [16] is divided into four categories named, respectively, "boss" (see Figure 1a), "dilbert" (see Figure 1b), "dogbert" (see Figure 1c), "unknown" (see Figure 1d), each containing a different number of images. A custom four-class dataset exhibiting notable class imbalance, with the minority class (Unknown, n = 82) containing fewer than half the samples of each majority class (n = 200).

The code used is available for review at [17].

Table 1, resumes the Hyperparams useful for reproducibility.

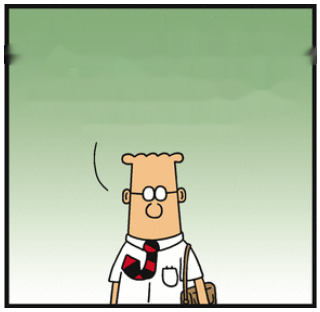
Table 1. Hyperparameter configuration for all models.

Parameter	CNN Base	CNN + Reg	ResNet FE	ResNet FT
Optimizer	Adam	Adam	Adam	Adam
Learning rate	0.001	0.001	0.001	0.0001
β_1/β_2	0.9/0.999	0.9/0.999	0.9/0.999	0.9/0.999
ϵ	10^{-7}	10^{-7}	10^{-7}	10^{-7}
Batch size	32	32	32	32
Max epochs	100	100	100	100
Early stopping	No	Yes (p = 10)	Yes (p = 10)	Yes (p = 10)
Dropout rate	—	0.5	0.5	0.5
Input size	224 × 224	224 × 224	224 × 224	224 × 224

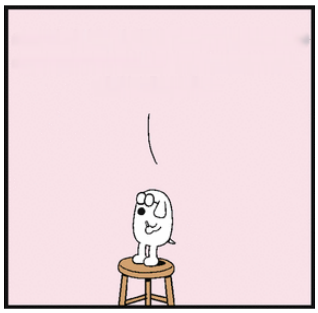
This dataset is used to create a training set (70% of the total images), a validation set (20%), and a test set (10%) [18]. The division of the dataset into three different sets allows for an appropriate evaluation of the model's performance during the training and testing process.

The training set is used to train the model, while the validation set is employed to monitor the model's adaptation to unseen data during training. Finally, the test set is used to evaluate the final performance of the model on completely new data.

(**a**) Boss



(**b**) Dilbert



(**c**) Dogbert

(**d**) Unknown

Figure 1. Representative image for each class.

The total corpus of 682 images represents a genuinely constrained data regime, and this limitation warrants explicit acknowledgment. Test partitions of approximately 20 samples per named class and 8 samples for the Unknown class produce wide confidence intervals on all reported metrics, meaning that small absolute differences in correct predictions translate into large percentage swings in accuracy. Conclusions about absolute performance ceilings should therefore be interpreted with appropriate caution. To mitigate this constraint, three complementary strategies are adopted throughout this study. First, data augmentation (Section 3.4.2) artificially expands the effective training distribution by introducing geometric and photometric variation, reducing the risk of memorization. Second, transfer learning via pretrained ResNet-50 weights (Sections 3.5 and 3.6) compensates for limited labeled data by supplying a rich, general-purpose feature hierarchy learned from over 1.4 million ImageNet images, drastically reducing the number of parameters that must be learned from scratch. Third, and most critically, a 5-fold cross-validation framework (Section 4.4) is employed alongside the fixed split to ensure that reported performance is not an artifact of a single favorable data partition. Together, these strategies do not eliminate the constraints imposed by dataset scale, but they substantially reduce the risk that the findings reflect overfitting or sampling luck rather than genuine architectural differences.

It should be noted that data augmentation, while effective at reducing memorization, does not introduce novel semantic content. The augmented distribution remains bounded by the visual diversity of the original 682 images, and the marginal benefit of augmentation diminishes as the model has already learned the finite set of underlying visual patterns. This inherent limitation motivates the transfer learning approach, which compensates for local data scarcity by importing representations learned from a much larger and more diverse source corpus.

Ethical Justification

The dataset used in this study was compiled in January 2021 from images of Dilbert comic strips that were, at that time, freely and publicly accessible via the official Dilbert website (https://dilbert.com accessed on 1 January 2021) operated by Scott Adams/Andrews McMeel Syndication. No authentication, subscription, paywall, or access restriction was required to view or retrieve the images at the time of collection. The authors did not circumvent any technical protection measure, access any private repository, or obtain the material through any channel other than standard public web access.

3.2. A Note on the "Unknown" Class

The four classes in this dataset are not all the same type of problem. Recognizing Boss, Dilbert, and Dogbert is a straightforward identification task: given an image, assign it to one of three known characters. The model learns what each character looks like and picks the closest match. The Unknown class works differently. Rather than representing a specific visual identity, it groups together any image that does not belong to any named character—panels containing background figures, minor characters, or ambiguous scenes. The model is therefore not being asked to learn what "unknown" looks like, but rather to recognize when none of the three known characters are present. This is a harder task, and it is worth being transparent about why. A standard classifier—the type used throughout this study—is designed to always pick a winner among the classes it knows. When faced with an image it has never seen before, it will still assign it to whichever known class it most resembles, even if the resemblance is weak. This structural tendency explains why the Unknown class consistently shows lower recall than the named character classes across all four models evaluated. For the purposes of this study, Unknown is treated as a standard fourth class. This is a deliberate simplification that keeps the experimental setup clean and comparable across architectures. It is not the optimal solution for rejection of out-of-distribution inputs, and the metrics for this class should be read with that in mind. More principled approaches to this problem are discussed in Section 6 as a direction for future work.

Cross-Model Misclassification Analysis of the Unknown Class

To understand why the Unknown class consistently exhibits the lowest recall across all four architectures, Table 2 presents a systematic breakdown of where Unknown test samples are misclassified under the fixed 70/20/10 partition. The pattern is striking in its consistency (see Figure 2): across all four models and all 17 aggregated false negatives, every single misclassification directs an Unknown sample into the decision region of one of the two humanoid characters (Boss or Dilbert). Not a single Unknown sample is ever misclassified as Dogbert.

This asymmetry is not coincidental; it reflects the underlying feature-space geometry of the dataset. The three named characters differ fundamentally in visual morphology:

- Boss and Dilbert are both humanoid figures depicted in office environments. They share upright posture, rectangular torsos, and limbs and are typically rendered against similar backgrounds (cubicles, desks, and meeting rooms). The primary discriminative features between them are localized details: the Boss's pointed hair and lack of glasses versus Dilbert's upturned tie and rectangular glasses.
- Dogbert is a round, non-humanoid character with no limbs, a radically different silhouette, and is typically rendered at a smaller scale. These coarse shape features create a well-separated cluster in any learned feature space, explaining why no model confuses Unknown samples with Dogbert.

Table 2. Cross-model misclassification analysis for the Unknown class under the fixed 70/20/10 partition ($n = 9$ Unknown test samples). For each model, the row shows how many Unknown samples were correctly classified (TP) versus absorbed into each named-character decision region. The rightmost column reports the fraction of all false negatives directed toward humanoid characters (Boss + Dilbert) versus the non-humanoid Dogbert.

		Misclassified as					
Model	**TP**	**Boss**	**Dilbert**	**Dogbert**	**FN**	**Recall**	**Humanoid FN/Total FN**
CNN Baseline	4	1	4	0	5	0.444	5/5 = 100%
CNN + Regularization	3	3	3	0	6	0.333	6/6 = 100%
ResNet-50 (FE)	4	1	4	0	5	0.444	5/5 = 100%
ResNet-50 (FT)	8	0	1	0	1	0.889	1/1 = 100%
Aggregated	19	5	12	0	17	—	17/17 = 100%

Note: Across all four architectures and all 17 false negatives, not a single Unknown sample was ever misclassified as Dogbert. Every misclassification was absorbed into the decision region of one of the two humanoid office characters.

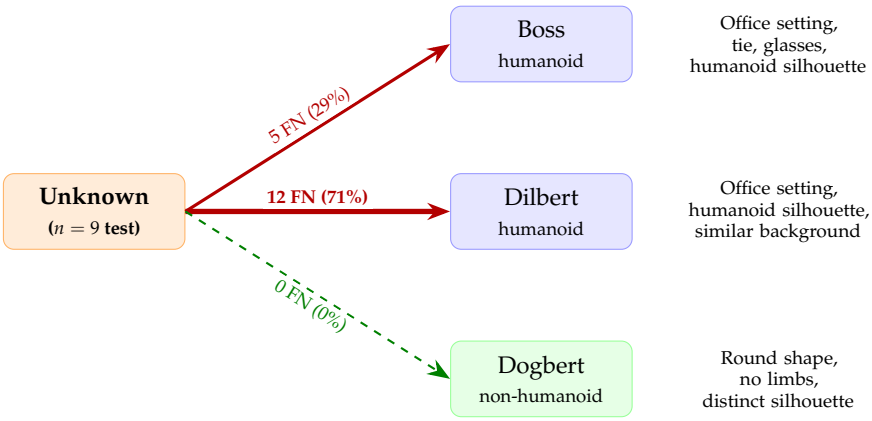


Figure 2. Aggregated Unknown-class misclassification flow across all four models (17 total false negatives from the fixed 70/20/10 partition). Arrow thickness is proportional to the number of misclassifications. All false negatives are absorbed into humanoid character classes; Dogbert's distinctive non-humanoid silhouette creates a well-separated decision region that never captures Unknown samples.

The Unknown class, by contrast, contains heterogeneous content: office backgrounds, minor human characters, partial panels, and scene fragments. Many of these images contain the very contextual features, office furniture, human silhouettes, and rectangular panel layouts, that the classifier has learned to associate with Boss and Dilbert. A softmax classifier partitions feature space exhaustively among its K classes; there is no "none of the above" rejection region. When an Unknown sample's feature vector falls closer to the Boss or Dilbert centroid than to the Dogbert centroid (which is virtually guaranteed for any image containing humanoid or office content), the classifier will assign it to one of those two classes with high confidence.

This analysis reveals that low Unknown-class recall is not a failure specific to any single architecture but rather a structural consequence of treating an open-set rejection problem as a closed-set classification task. The progressive improvement from CNN Baseline (recall = 0.444) through CNN + Regularization (0.333, a regression) to ResNet-50 FE (0.444) and finally ResNet-50 FT (0.889) suggests that deeper, more discriminative feature representations can partially mitigate the problem by learning tighter decision boundaries around the named-character clusters but cannot eliminate it without a fundamentally different architectural approach.

3.3. Basic CNN

To develop a precise and efficient classification system that can automatically recognize and categorize images into different classes, we started by creating a convolutional neural network (CNN) implemented using the TensorFlow framework with Keras as the high-level API.

The architecture is defined as an instance of the Sequential class, which allows creating a sequential stack of layers described in Table 3.

The implemented CNN takes input images of size (224, 224, 3), where 224 × 224 represents the image dimensions in pixels [19], and 3 indicates the three color channels (red, green, blue). Before feeding the images into the network, normalization (rescaling) [20] is applied, which scales the pixel values to the range [0, 1] by dividing each pixel by 255. This helps to make the data compatible with the typical value range used in neural networks.

After normalization, a series of convolutional layers are applied, each followed by a max pooling layer. The convolutional layers extract features from the images using a set of convolutional filters, each of which produces a feature map.

Table 3. Architecture of CNN_model_1.

Layer	Type	Filters/Units	Kernel Size	Activation	Output Shape
1	Input	–	–	–	224 × 224 × 3
2	Rescaling	–	–	–	224 × 224 × 3
3	Conv2D	4	3 × 3	ReLU	224 × 224 × 4
4	MaxPooling2D	–	2 × 2	–	112 × 112 × 4
5	Conv2D	8	3 × 3	ReLU	112 × 112 × 8
6	MaxPooling2D	–	2 × 2	–	56 × 56 × 8
7	Conv2D	16	3 × 3	ReLU	56 × 56 × 16
8	MaxPooling2D	–	2 × 2	–	28 × 28 × 16
9	Conv2D	32	3 × 3	ReLU	28 × 28 × 32
10	MaxPooling2D	–	2 × 2	–	14 × 14 × 32
11	Conv2D	64	3 × 3	ReLU	14 × 14 × 64
12	MaxPooling2D	–	2 × 2	–	7 × 7 × 64
13	Flatten	–	–	–	3136
14	Dense	128	–	ReLU	128
15	Dense	N_c	–	Softmax	N_c

Adam was selected as the optimizer for all experiments based on its established convergence properties and widespread adoption in transfer learning workflows. The effect of alternative optimizers (e.g., SGD with momentum, AdamW) on the reported results was not evaluated and represents a controlled variable held constant across all comparisons; optimizer ablation is noted as a direction for future work.

In detail, it starts with convolving images through various 3 × 3 filters, maintaining output dimensions via padding and introducing non-linearity through ReLU activation. This is followed by max pooling layers to reduce feature map dimensions and increase depth. These steps are repeated with more filters and pooling, creating a hierarchy of feature maps. After the fifth convolutional layer, the output is flattened into a vector and passed through fully connected layers for final classification. Finally, a fully connected layer with softmax activation produces the final output, interpreted as a probability distribution across classes, for multiclass classification.

Once the neural network is constructed, the model is compiled using the ADAM (Adaptive Moment Estimation) optimizer, Sparse Categorical Crossentropy as the loss function, and accuracy as the evaluation metric. This step is essential before starting the model training process.

The ADAM optimizer is a popular optimization algorithm widely used for training deep neural networks. It combines the characteristics of two other optimization algorithms, namely Adaptive Gradient (AdaGrad) and Root Mean Square Propagation (RMSProp), to provide optimal performance in terms of convergence speed and adaptability of the learning rate [21,22].

The Sparse Categorical Crossentropy loss function is commonly used for classification problems where the output labels are represented as integers. In fact, this loss function expects the labels to be integers representing the actual classes. During the loss calculation, internally, the function converts the labels into "one-hot" form (each class is represented as a binary vector) to compare them with the probability distribution predicted by the model [10].

The specified evaluation metric is accuracy, which measures the percentage of correct predictions made by the model compared to the target labels [23].

At this point, we proceed with training the model on the training dataset and validate it using the validation dataset.

Firstly, a variable is defined, which indicates the number of training epochs for the model. The value is set to 100, which means that the model will be trained for a hundred complete iterations on the training dataset. In practice, training the network involves iterating over the entire training dataset for the specified number of epochs.

During training, the model aims to optimize the specified loss function, which was defined during compilation, using the specified optimization algorithm. The objective is to reduce the model's loss, which measures the error between the model's predictions and the correct values in the training dataset.

During the training process, the weights and biases of the model are updated so that it can make increasingly accurate predictions.

Furthermore, after each training epoch, the validation dataset is used to evaluate the model's performance. The model makes predictions on the validation dataset and calculates one or more evaluation metrics (in our case, the loss and accuracy) to determine how well it is generalizing from the training data.

The training results provide an indication of the model's performance on the validation data.

From the resulting graph (see Figure 3), it is clear that the model is experiencing overfitting. For example, during the initial epochs, both the training and validation losses decrease. However, at a certain point, the training loss continues to decrease, while the validation loss increases significantly. This is a strong indication of overfitting.

Overfitting is a phenomenon that occurs when a model learns "too well" from the training data (during network training), to the point where it loses its ability to generalize and adapt to new data [24].

Essentially, the main goal of a model is to find a balance between learning the patterns present in the training data and the ability to apply those patterns to new, previously unseen data, i.e., the ability to generalize to new data.

Overfitting occurs when the model becomes "too specialized" in the training data and starts memorizing it instead of learning general patterns and features that can be applied to new data.

This can result in poor performance of the model on new test data or real-world data, despite excellent results on the training data.

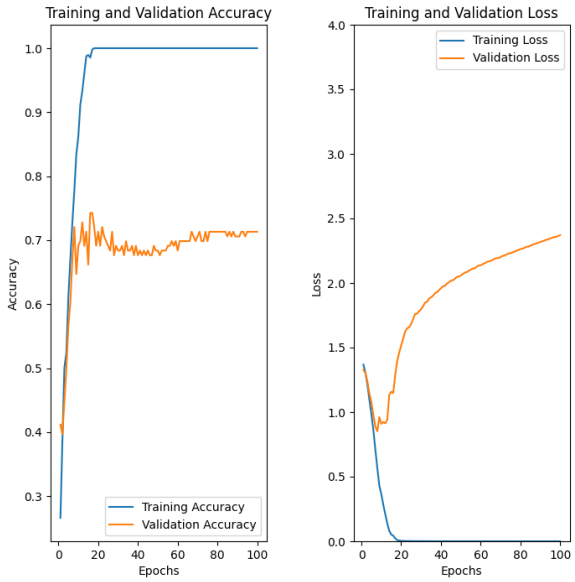


Figure 3. Basic CNN training plots corresponding to a single training run under the fixed 70/20/10 partition with no smoothing applied.

Overfitting can be caused by several factors, including:

- excessive complexity of the model: a complex model, such as a deep neural network with a large number of units or hidden layers, may have a tendency to overfit the training data, leading to overfitting;
- limited size of the training dataset: if the training dataset is small, the model may be prone to overfitting because it does not have enough examples to learn general patterns and relies on specific information from the few examples present;
- noise in the data: if the training data contains noise or if there are outliers (extreme values that deviate significantly from the rest of the sample or the general data distribution), the model may adapt to such anomalies instead of learning the true underlying patterns.

However, there are several regularization techniques to mitigate overfitting, including:

- L1 and L2 regularization [25];
- Dropout;
- Early Stopping;
- Data Augmentation;
- Batch Normalization [26].

3.4. CNN with Regularization Techniques

After observing unsatisfactory results from the basic CNN, it was decided to employ several regularization techniques with the scope of reducing overfitting and consequently improving the model's generalization ability. These techniques include:

- Early Stopping;
- Data Augmentation;
- Dropout.

By limiting the model's complexity or introducing weight penalties, regularization techniques help prevent the model from overfitting to the training data and ensure better generalization on unseen validation data during training. The architecture of the new CNN is described in Table 4. The corresponding statistical improvement is reported in Section 4.4 (paired $t = -10.71$, $p < 0.001$).

Table 4. Architecture of CNN_model_2.

Layer	Type	Filters/ Units	Kernel Size	Activation	Notes
1	Input	–	–	–	$224 \times 224 \times 3$
2	Data Augmentation	–	–	–	Flip, Rotation, Zoom
3	Rescaling	–	–	–	1/255
4	Conv2D	16	3×3	ReLU	Same padding
5	MaxPooling2D	–	2×2	–	–
6	Conv2D	8	3×3	ReLU	–
7	MaxPooling2D	–	2×2	–	–
8	Conv2D	16	3×3	ReLU	–
9	MaxPooling2D	–	2×2	–	–
10	Conv2D	32	3×3	ReLU	–
11	MaxPooling2D	–	2×2	–	–
12	Conv2D	64	3×3	ReLU	–
13	MaxPooling2D	–	2×2	–	–
14	Flatten	–	–	–	–
15	Dropout	–	–	–	$p = 0.5$
16	Dense	128	–	ReLU	–
17	Dropout	–	–	–	$p = 0.5$
18	Dense	N_c	–	Softmax	–

3.4.1. Early Stopping

Early stopping is a key practice in training deep learning models, aiming to prevent overfitting and optimize overall model performance. During training, performance on the validation set is monitored, halting the process when it ceases to improve for a fixed number of iterations. Evaluation metrics such as loss and accuracy on the validation set are crucial and constantly monitored. If performance improves, the model is saved and training continues; otherwise, it is halted, reverting to the previous model. This mechanism prevents overfitting, promoting the model's ability to generalize. Additionally, it saves time and resources by avoiding unnecessary training. It is essential the validation set from the test set used solely at the end to impartially evaluate the model's final performance.

In our case, we decided to monitor the validation loss and set the maximum number of consecutive epochs with no improvement to 10 before stopping the training. The goal is to minimize the validation loss. At the end of training, we also decided to restore the model weights to the best epoch. This means that the model is not saved at the end of training, but the weights from the epoch with the best performance on the monitored metric are restored. Therefore, the model weights are restored to the best epoch during training.

3.4.2. Data Augmentation

Data augmentation is a practice in machine learning aimed at expanding the size and diversity of training data through artificial transformations, generating new examples similar but not identical to the originals. It enhances model generalization by exposing it to a wider range of cases and prevents overfitting. Transformations must preserve the essential characteristics of the original data and can be applied randomly or systematically during the model training process. It does not add new information but rather modifies and repositions existing information to create variations; hence, the quality of the original data and careful selection of transformations are crucial for the technique's effectiveness. Guangyao et al. [27] demonstrates the importance of data augmentation in improving performance in specific fields such as Knowledge Graphs.

In our case, it has been decided to specify three transformations that can be applied to images during model training, namely horizontal flip, random rotation (±0.1 radians),

and random zoom (±0.1). It is important not to overdo transformations during data augmentation, as it can lead to some negative effects on the model training process and its generalization ability, such as excessive distortion, increased noise, or prolonged training time.

Table 5 resumes the data augmentation pipeline.

Table 5. Data augmentation and input preprocessing configuration. Augmentation is applied during training only; validation and test sets undergo resize and normalization without stochastic transformations. ResNet-50 models do not use input-level augmentation.

Panel A: Data augmentation operations (CNN + Regularization only)

Operation	Keras layer	Parameter	Probability
Horizontal flip	`RandomFlip('horizontal')`	mode = horiz.	$p = 0.5$ per image
Random rotation	`RandomRotation(0.1)`	factor = 0.1	$\mathcal{U}[-0.1, +0.1]$ rad
Random zoom	`RandomZoom(0.1)`	h_factor = 0.1	$\mathcal{U}[-10\%, +10\%]$

Panel B: Input preprocessing pipeline

Step	CNN_1	CNN_2	FE	FT	Applied to
1. Resize to 224×224	✓	✓	✓	✓	Train + Val + Test
2. Data augmentation (Panel A)	—	✓	—	—	Training only
3a. Rescale to $[0, 1]$ ($\div 255$)	✓	✓	—	—	Train + Val + Test
3b. ImageNet mean sub. (BGR)	—	—	✓	✓	Train + Val + Test

Note: The symbols ✓ marks if a specific step is applied during the input processing pipeline, — otherwise.

Panel C: Normalization details

Method	Implementation	Details
Simple rescaling	`Rescaling(1./255)`	$x' = x/255$; output range $[0, 1]$. Applied to CNN_1 and CNN_2.
ImageNet mean sub.	`resnet50.preprocess_input()`	$x' = x - \mu_c$; channel means $\mu = [103.94, 116.78, 123.68]$ (BGR order). No variance normalization. Applied to ResNet FE and ResNet FT.

Note: Augmentation layers (`RandomFlip`, `RandomRotation`, `RandomZoom`) are Keras layers inside the Sequential model, automatically disabled during `model.evaluate()` and `model.predict()`. The function `preprocess_input()` converts RGB → BGR and subtracts per-channel ImageNet training-set means; it does not apply variance normalization. No test-time augmentation (TTA) is used in any experiment.

3.4.3. Dropout

Dropout is a widely used regularization technique in machine learning, especially in deep neural networks, as a method to reduce overfitting and improve the performance of machine learning models [10].

The main objective of dropout is to prevent interdependence and reliance between neurons within a neural network. This is achieved by randomly deactivating a certain number of units during the training of a network so that they do not contribute to the forward propagation of information and do not influence other neurons. This process of randomly deactivating neurons is performed during each training iteration.

During the dropout process, each neuron within a neural network is 'dropped out' with a certain probability, typically ranging from 20% to 50%. This way, neurons cannot rely on any specific neuron for feature extraction and are therefore forced to learn more independent and robust features.

The overall effect of dropout is that the neural network becomes less sensitive to the specific weights of individual neurons and develops a more robust and generalized representation of the input data. This helps to reduce overfitting, as the network cannot 'specialize' in specific neuron combinations that may only be present in the training set.

In general, dropout is an effective technique for improving the generalization of machine learning models, especially when working with high-capacity deep neural networks that tend to have a higher likelihood of overfitting.

3.4.4. Architecture of CNN with Regularization Techniques

The structure of the basic CNN is then modified by incorporating:

- a data augmentation layer at the beginning of the model;
- a dropout layer after the flatten layer;
- a dropout layer at the output of the fully connected layer with 128 hidden units (neurons) and ReLU activation function.

The data augmentation layer is used to artificially increase the amount of training data by introducing small modifications to the existing images. Typically, this helps improve the model's ability to generalize and handle variations in the test data.

During training, dropout layers randomly set some units to zero, temporarily deactivating certain neurons in a random manner, thereby reducing the dependence of each unit on its neighboring units. This helps make the model more robust and reduces overfitting. In our case, the dropout layers are passed the argument 0.5, indicating the fraction of input units that will be zeroed out, meaning that 50% of the input units will be randomly set to zero during training.

In the training of the "new" model, the technique of early stopping is used. A callback is implemented to stop the training of the model if the validation loss does not improve (or worsens) for a maximum of 10 consecutive epochs.

From the accuracy and loss graph in both training and validation (see Figure 4), it is evident that the model significantly reduces the overfitting it previously suffered from, thanks to the application of regularization techniques. In fact, the accuracy and loss lines in both training and validation phases no longer exhibit significantly different trends, but, aside from increased fluctuations, they, respectively, increase and decrease in parallel (until the best epoch).

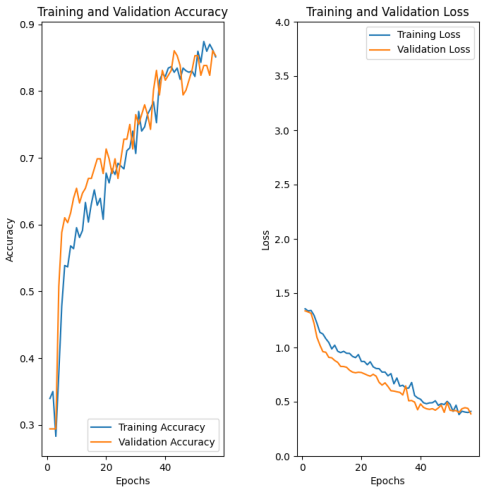


Figure 4. Plot of the accuracy and loss of the CNN with regularization techniques.

The more oscillating trend of the accuracy and loss graphs after the application of data augmentation can be attributed to several factors:

- Data Variability: Data augmentation introduces greater variability in the training data by generating different transformations and modifications on existing samples. This variation can affect the graph trends since the model may find it more challenging to adapt to consistent patterns in the data;

- Increased Noise: The augmentation of data through techniques like rotation, translation, resizing, or color modification introduces a certain level of noise in the training data. This noise can cause fluctuations in the model's accuracy and loss, making the graph trends more oscillatory;
- Model Adaptation: Data augmentation may require the model to adapt to a wider variety of patterns in the data. This might necessitate longer training or additional regularization to enable the model to generalize properly. During this adaptation process, the graph trends can be more unstable;
- Frequency of Changes: The impact of data augmentation can vary depending on the specific implementation and transformations applied. Some transformations may cause more pronounced fluctuations, while others may have a minor impact on the graph trends.

In general, the oscillation of accuracy and loss graphs after the application of data augmentation is not necessarily a problem, as long as the overall performance of the model improves (as is the case in our scenario).

3.5. CNN with Pretrained ResNet-50

To leverage pretrained representations for the comic character classification task, a feature extraction approach was adopted using the ResNet-50 architecture with ImageNet weights. This strategy freezes all pretrained convolutional layers and trains only a lightweight classification head on the target dataset, thereby exploiting the rich hierarchical features learned from over 1.4 million natural images while minimizing the number of trainable parameters.

ImageNet is a large dataset that contains over 1.4 million labeled images across 1000 different categories. Training a neural network model on such a large dataset requires significant computational resources and considerable time. In our case, the weights of the pretrained ResNet-50 model on ImageNet are used as a beneficial starting point for image classification [28].

The ResNet-50 chosen is the one provided by the Keras library. The architecture of Keras' ResNet-50 consists of a sequence of 176 layers of the following types:

- InputLayer: defines the input of the model;
- Conv2D (convolutional layer): performs convolution between filters and input data to extract relevant image features;
- BatchNormalization: normalizes the outputs of the previous layer to accelerate training and improve network stability;
- Activation: applies a non-linear activation function to the outputs of the previous layer to introduce non-linearity into the network (e.g., ReLU);
- MaxPooling2D: reduces the spatial dimension of the feature maps through max pooling operation;
- Add: adds the inputs of the previous layer to the output of the subsequent layer;
- GlobalAveragePooling2D: reduces the spatial dimension of the feature maps by computing the average over all positions;
- Flatten: converts the multidimensional data into a one-dimensional vector before passing it to the fully connected layers;
- Dense (fully connected layer): consists of neurons connected to all neurons in the previous layer and is used for final classification.

Therefore, the architecture constructed is a convolutional neural network (CNN) model based on the pretrained ResNet-50 architecture. The network consists of multiple layers that are sequentially concatenated using the Sequential class of TensorFlow as described in Table 6.

The first step involves importing the pretrained ResNet-50 model. This model is imported without the final fully connected layer, meaning only its convolutional part is imported, which primarily consists of a series of convolutional layers, batch normalization layers, and pooling layers (totaling 173 layers out of the 176 layers). This part of the model can be used as a feature extractor to generate representations of images that can then be used as input for custom fully connected layers. In the case of the ResNet-50 architecture, the final fully connected layer is responsible for classifying the original ImageNet dataset into 1000 classes. If using a dataset with a different number of classes, as in our case, it is necessary to remove this layer and replace it with suitable new layers. It is common practice to remove the final fully connected layer when using a pretrained model as a base for a new task, as it allows reusing the lower-level parts of the model, which are often capable of extracting generic features from a wide range of images, and only adapting the final layers to the specific new task.

Table 6. Architecture of ResNet_FE_model (feature extraction).

Layer	Type	Output Units	Activation	Trainable
1	Input	–	–	–
2	Preprocessing	–	–	No
3	ResNet50 (no top)	2048	–	No
4	Dense	512	ReLU	Yes
5	Dropout	–	–	$p = 0.5$
6	Dense	N_c	Softmax	Yes

Subsequently, the weights of the pretrained layers are prevented from being updated during subsequent training, allowing only the following layers to learn from the data of the new dataset.

After importing the pretrained model, a new model is created using the Sequential class. The pretrained model is added to the new model.

Next, the following layers are added:

1. a flatten layer, which transforms the output of the last convolutional layer of the pretrained model into a one-dimensional vector, preparing it as input for the subsequent fully connected layers;
2. a fully connected layer with 512 units and ReLU activation function, which helps extract additional features from the data;
3. a dropout layer, which prevents overfitting during training. During training, dropout randomly "deactivates" 50% of the hidden units, reducing interdependencies between neurons and promoting better model generalization;
4. a final fully connected layer with a number of units equal to the number of classes in the specific task and softmax activation function. This layer produces the probabilities of belonging to different output classes.

The approach used is feature extraction. Indeed, the pretrained ResNet-50 model is used as a feature extractor, where the pretrained weights are kept fixed, and only the new layers added at the end of the model are trained on the specific new dataset for the task. Using pretrained weights can help the model converge faster and improve overall performance on a new dataset or a specific task.

In the case of the implemented CNN, the trainable parameters are only those of the last layers, namely the ones added customarily after importing the convolutional layers (from the ResNet-50). Therefore, the trainable parameters amount to only 1,051,140 out of the total 24,638,852 parameters.

From the accuracy and loss plot during training and validation (see Figure 5), it is evident how the model significantly reduces the overfitting that the initial CNN suffered from. In fact, the accuracy and loss lines in both training and validation phases no longer exhibit a significantly different trend but rather increase and decrease almost in parallel (until the best epoch).

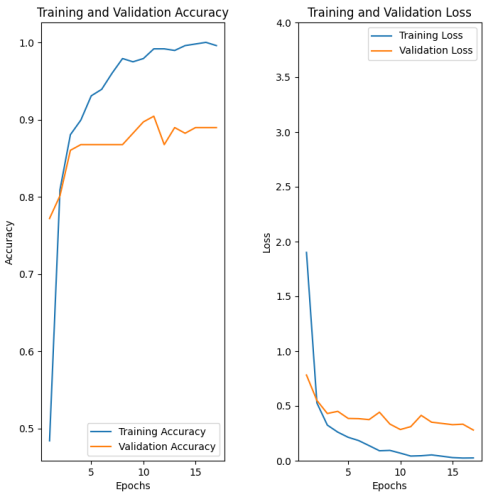


Figure 5. Plot of the Accuracy and Loss of the CNN whit PreTrained ResNet-50.

3.6. CNN with PreTrained ResNet-50 (Fine-Tuning Approach)

Despite the excellent results obtained with the previous model, an attempt was made to modify the network to make it even more performant, adopting a different approach.

In the previous network, all layers of the pretrained model were set as non-trainable. As seen before, this means that the weights of these layers will not be updated during the training phase of the final model, and only the subsequently added layers (Flatten, Dense, Dropout, and Dense) will be trainable. This approach is often used to leverage the learned features from a pretrained network without modifying them as described in Table 7.

Table 7. Architecture of ResNet_FT_model (Fine-Tuning). In the fine-tuning configuration, the first 140 layers of ResNet50 are frozen, while the remaining layers are trainable.

Layer	Type	Output Units	Activation	Trainable
1	Input	–	–	–
2	Preprocessing	–	–	No
3	ResNet50 (no top)	2048	–	Partial
4	Dense	512	ReLU	Yes
5	Dropout	–	–	$p = 0.5$
6	Dense	N_c	Softmax	Yes

The new approach, on the other hand, involves setting the first layers of the pretrained model as non-trainable (in our case, the first 140 layers), while the remaining layers (from the 141st onwards) are set as trainable.

The selection of layer 140 as the freeze boundary is grounded in both the internal block structure of Keras ResNet-50 and well-established principles from the transfer learning literature. As summarized in Table 8, the architecture progresses through five stages of increasing abstraction. The foundational work of Yosinski et al. (2014) [1] demonstrated empirically that early-layer features in deep convolutional networks are highly general, encoding low-level properties such as edges, color gradients, and simple textures that transfer robustly across image domains, while features in the final layers become increasingly

task-specific and therefore less transferable to new domains without adaptation. This gradient of transferability provides the principled basis for selective layer unfreezing: layers whose representations are already domain-general should be preserved, while layers whose representations are tied to the source domain's semantic categories should be allowed to adapt. Applied to ResNet-50, this principle maps cleanly onto the block structure. The stem and conv2_x through conv4_x blocks (layers 0–138) encode progressively richer but still domain-transferable representations, from edges and corners through textures and object parts, that have been shown to generalize effectively even to stylized and non-photographic imagery. Kornblith et al. (2019) [2] further demonstrated that the intermediate representations of ImageNet-pretrained networks transfer reliably to visually dissimilar target domains, precisely because these mid-level features capture structural image properties rather than naturalistic appearance. Freezing these 140 layers therefore preserves a stable, general-purpose feature hierarchy that the small training corpus could not reliably reconstruct from scratch. The conv5_x block (layers 139–172), by contrast, is responsible for high-level semantic abstraction, the representations most directly involved in distinguishing between specific visual identities. In the source domain, these representations are calibrated to discriminate among ImageNet's 1,000 natural-image categories, a calibration that does not transfer directly to the narrow, stylized space of comic character recognition. Allowing only this final residual block to update confines adaptation to the layer range where it is both most necessary and least likely to destabilize the pretrained hierarchy. This approach is consistent with the selective fine-tuning strategy recommended by Pan and Yang (2010) [3] in their foundational survey of transfer learning, which identifies the final task-specific layers as the appropriate target for domain adaptation when labeled data are scarce. Freezing all preceding layers also acts as an implicit regularizer: by reducing the number of trainable parameters from 24,638,852 to 16,029,188, the risk of overfitting the adapted representations to the limited training corpus is substantially reduced.

This approach is called fine-tuning and allows the model to adapt more specifically to the new dataset, enabling it to learn relevant features for the specific task.

Table 8. ResNet-50 block structure and freeze/train configuration. The threshold of 140 aligns with the conv5_x block boundary, unfreezing only the highest-level semantic layers for domain adaptation.

Stage	Layer Range	Features Learned	Frozen?
Stem	0–2	Low-level edges, colors	Yes
conv2_x	3–38	Corners, simple textures	Yes
conv3_x	39–74	Textures, gradients	Yes
conv4_x	75–138	Object parts, shapes	Yes
conv5_x	139–172	High-level semantics	No
GlobalAvgPool	173	Spatial aggregation	No

Therefore, the ResNet-50 no longer only acts as a feature extractor with non-trainable parameters but now allows for the updating of some of the base layers to better tailor the model to the specific problem.

In the case of the new CNN implemented, the trainable parameters are not only those of the last layers, i.e., the ones added customarily after importing the convolutional layers (from the ResNet-50), but also the last layers of the pretrained model (starting from the 141st). Therefore, with the applied split, the trainable parameters amount to 16,029,188 (greater than the 1,051,140 of the previous model) out of the total 24,638,852 parameters.

From the accuracy and loss plot during training and validation (see Figure 6), it is clear how the model significantly reduces the overfitting that the initial CNN suffered from. In fact, the trend of the accuracy and loss lines, both during training and validation, no

longer exhibits a significantly different pattern. Instead, they both increase and decrease almost in parallel, reaching their best values around the same epoch.

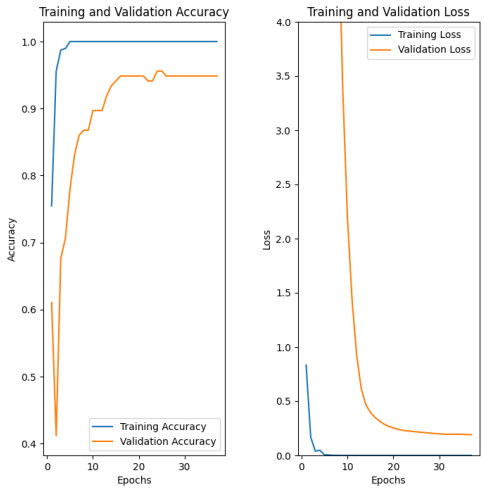


Figure 6. Curves correspond to a single training run under the fixed 70/20/10 partition with no smoothing applied.

3.7. Performance Metrics

While in the previous sections we introduce the different architectures used during our experiment, to provide a deeper and more accurate analysis of the different models, in this section, we discuss the confusion matrix related to each model and the associated **Precision**, **Recall**, and **F1-Score**, but also, considering the unbalanced nature of the dataset, we calculate the **Macro-Average** and the **Weighted-Average** for each of the classes.

These metrics are essential to have a complete overview of how the model works, especially in classification tasks.

Precision, Recall, F1-Score

Accuracy alone can be misleading, particularly when dealing with imbalanced datasets, where one class dominates. A model may achieve high accuracy while performing poorly on the minority or most important class. Precision, Recall, and F1-Score provide deeper insight into how a model makes errors.

Precision: measures how reliable positive predictions are. It answers the question: when the model predicts a positive outcome, how often is it correct? High precision is crucial in scenarios where false positives are costly, such as fraud detection or medical diagnoses that could cause unnecessary alarm.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall: measures how well the model captures all actual positive cases. It answers: of all real positive instances, how many did the model detect? High recall is critical when missing a positive case is dangerous, such as failing to detect a disease or a security threat.

$$\text{Recall} = \frac{TP}{TP + FN}$$

There is often a **trade-off between precision and recall**, influenced by decision thresholds. Understanding both helps align the model's behavior with real-world priorities and risk tolerance.

F1-Score: combines precision and recall into a single metric using their harmonic mean. It is particularly useful when classes are imbalanced and when both false positives

and false negatives matter. Unlike accuracy, the F1-Score penalizes extreme imbalances between precision and recall.

$$\text{F1-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

These metrics help diagnose model behavior, reveal error patterns, guide threshold tuning, and determine whether a model is truly suitable for deployment in real-world applications.

Macro-Average and Weighted-Average

Macro-Average and Weighted-Average are used to evaluate the performance of multi-class classification models, particularly when dealing with imbalanced datasets where some classes have far fewer samples than others. They differ primarily in how they treat class frequency (support) when calculating overall metrics like Precision, Recall, or F1-score.

Macro-Average: Used in multiclass classification tasks, it treats all classes in the same way, regardless of the balance of each class. It is useful for identifying if a model is failing to classify minority classes properly.

$$\text{Macro-Average} = \frac{P1 + P2 + \ldots + Pn}{n}$$

(where P is the metric for a class and n is the number of classes)

Weighted-Average: Calculates the metric for each class and finds the average, weighted by the number of true instances (support) for each class. Used with imbalanced datasets and to reflect the overall effectiveness of the model across the entire dataset.

$$\text{Weighted-Average} = \frac{(P1 \cdot W1) + (P2 \cdot W2) + \ldots + (Pn \cdot Wn)}{TotalSupport}$$

(where W is the support/weight for each class)

Balanced Accuracy: is approximated using weighted recall due to the absence of per-class recall values. This approximation is reasonable under the assumption of near-balanced class distributions.

$$\text{Balanced Accuracy} = \frac{1}{C} \sum_{i=1}^{C} \text{Recall}_i \quad (1)$$

The ROC-AUC computation was not included because the models output softmax probabilities for four classes simultaneously, making one-vs-rest ROC curves less interpretable for the specific comparative question (frozen vs. fine-tuned ResNet).

Because the test set is imbalanced, each named class contributes 20 samples (29% each), while the Unknown class contributes only 9 (13%). The macro-averaged and weighted-averaged metrics reported throughout this section measure subtly different properties of model performance, and the gap between them serves as a diagnostic for minority-class difficulty.

The macro-average assigns equal weight to all four classes (25% each), regardless of their sample size. This means that the Unknown class, despite representing only 13% of the test data, exerts disproportionate influence: poor Unknown-class performance can depress the macro-average substantially even when the model performs well on the three named classes that constitute 87% of the data. This is precisely what is observed. For the CNN Baseline, the Unknown-class F1 of 0.471 is far below the named-class F1 scores (0.811, 0.571, 0.857), dragging the macro F1 down to 0.678 while the weighted F1 remains at 0.710—a gap of 0.032 that is entirely attributable to the Unknown class. The effect is most pronounced for the regularized CNN, where the Unknown class regresses to an F1 of 0.429, producing the largest weighted–macro gap of any model (ΔF1 = 0.047).

The weighted average, by contrast, weights each class by its support (number of true instances), so the Unknown class contributes proportionally to its 13% share. This makes the weighted average more representative of the model's performance on a randomly drawn test sample but less sensitive to minority-class failure. Both averages are reported throughout because they answer different questions: the weighted average characterizes expected accuracy in deployment (where the class distribution is assumed to match the test set), while the macro-average characterizes the model's ability to handle all classes equitably, a property that matters when every class must meet a minimum performance threshold.

One instructive exception to the typical pattern occurs for ResNet-50 feature extraction, where the macro precision (0.862) is higher than the weighted precision (0.840), yielding a negative ΔP of -0.022. This arises because the Unknown class achieves perfect precision (1.000; 4 of 4 predictions correct) despite low recall (0.444). The macro-average, giving this perfect-precision class equal 25% weight, is pulled upward; the weighted average, discounting the Unknown class to its 13% share, remains lower. This illustrates that macro-averages can be inflated by minority classes that achieve high precision through conservative prediction (rarely predicting the class but being correct when they do), a pattern that should be interpreted alongside recall, not in isolation.

As the model hierarchy improves from the CNN Baseline through to ResNet FT, the weighted–macro gap narrows monotonically: $\Delta F1 = 0.033$ (Baseline) $\rightarrow$ 0.047 (Regularized) $\rightarrow$ 0.028 (FE) $\rightarrow$ 0.006 (FT). The final gap of 0.006 for the fine-tuned model indicates near-complete elimination of the minority-class deficit, consistent with its Unknown-class recall of 0.889 approaching the named-class average of 1.000. The regularized CNN's regression ($\Delta F1$ widening from 0.033 to 0.047 despite improved overall accuracy) confirms that regularization improved named-class balance at the expense of Unknown-class performance, a trade-off discussed in Section 3.7.2.

3.7.1. Basic CNN

To calculate these metrics, we first need to identify the: True **Positives (TP)**: The number on the diagonal (where True Label = Predicted Label), **False Positives (FP)**: The sum of the column for that class (excluding the TP), and **False Negatives (FN)**: The sum of the row for that class (excluding the TP), for each class from the confusion matrix in Figure 7.

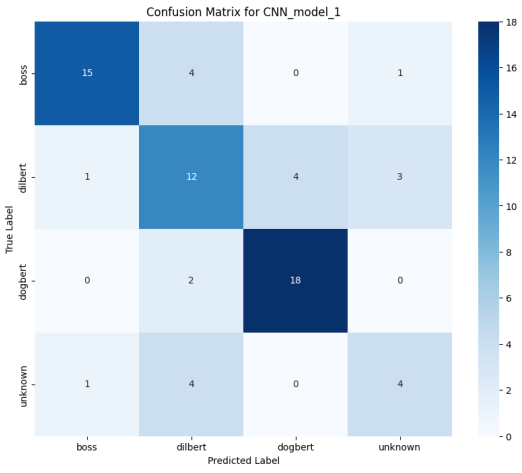


Figure 7. Confusion matrix for basic CNN.

Based on the values in Figure 7 we have the following values calculated using the Formulas for Precision, Recall and F1-Score described in the previous paragraph and resumed in Table 9.

Table 9. Classification performance metrics per class.

Class	TP	FP	FN	Precision	Recall	F1-Score
Boss	16	4	4	0.800	0.800	0.800
Dilbert	12	10	8	0.545	0.600	0.571
Dogbert	18	4	2	0.818	0.900	0.857
Unknown	4	4	5	0.500	0.444	0.471

To have a better understating how the model performs we calculated the **Macro-Average** and the **Weighted-Average**.
Macro-Average:

- **Precision:**
$$\frac{0.8824 + 0.5455 + 0.8182 + 0.5000}{4} \approx 0.6865$$
- **Recall:**
$$\frac{0.7500 + 0.6000 + 0.9000 + 0.4444}{4} \approx 0.6736$$
- **F1-Score:**
$$\frac{0.8108 + 0.5714 + 0.8571 + 0.4706}{4} \approx 0.6775$$

Weighted-Average:

- **Total Samples (Support):**
$$20 + 20 + 20 + 9 = 69$$
- **Weighted Precision:**
$$\frac{(0.882 \cdot 20) + (0.545 \cdot 20) + (0.818 \cdot 20) + (0.500 \cdot 9)}{69} \approx 0.7162$$
- **Weighted Recall:**
$$\frac{(0.750 \cdot 20) + (0.600 \cdot 20) + (0.900 \cdot 20) + (0.444 \cdot 9)}{69} \approx 0.7101$$
- **Weighted F1:**
$$\frac{(0.811 \cdot 20) + (0.571 \cdot 20) + (0.857 \cdot 20) + (0.471 \cdot 9)}{69} \approx 0.7105$$

Table 10, resumes the averages for the entire model.

Table 10. Macro-averaged and weighted-averaged performance metrics.

Metric	Macro-Average	Weighted-Average
Precision	0.6865	0.7162
Recall	0.6736	0.7101
F1-Score	0.6775	0.7105

We can conclude that the **Weighted-Average** is higher than the **Macro-Average** across all metrics. This indicates that the model performs better on the "majority" classes (Boss,

Dilbert, and Dogbert) and struggles more with the "minority" class (Unknown). Overall, the model achieves **71%** accuracy and a weighted recall of 0.710, reflecting limited generalization particularly for the minority class.

Given the small test partition ($n = 69$; 20 samples per named class, 9 for Unknown), 95% Wilson score confidence intervals around these per-class recall values are substantial. Boss recall of 0.750 spans [0.531, 0.888]; Dilbert recall of 0.600 spans [0.387, 0.781]; Dogbert recall of 0.900 spans [0.699, 0.972]; and Unknown recall of 0.444 spans [0.189, 0.733]. These intervals overlap extensively, meaning that the apparent performance ranking among classes (Dogbert > Boss > Dilbert > Unknown) cannot be considered statistically established from this single partition alone. The confidence intervals for precision exhibit a similar pattern: Dilbert precision of 0.545 spans [0.347, 0.731], while Boss precision of 0.882 spans [0.657, 0.967]. A difference of 0.337 in point estimates narrows to overlapping intervals, underscoring why cross-validated evaluation (Section 4.4) rather than single-split metrics must serve as the primary basis for model comparison.

3.7.2. CNN with Regularization

The confusion matrix in Figure 8 shows some interesting shifts in behavior compared to the previous one; while it has become much more consistent across the three main characters, the "unknown" category has taken a bit of a hit.

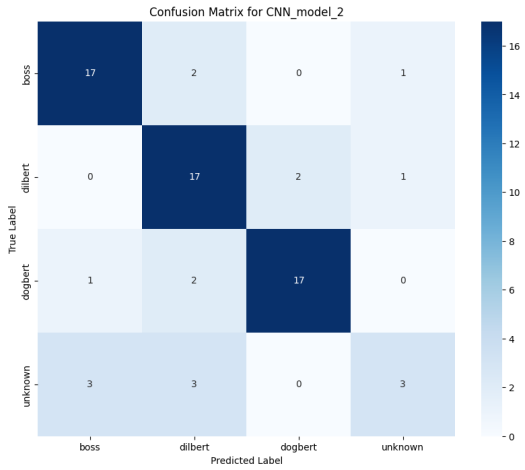


Figure 8. Confusion matrix for CNN with regularization.

Table 11 resumes the values for Precision, Recall and F1-score:

Table 11. Classification performance metrics per class.

Class	TP	FP	FN	Precision	Recall	F1-Score
Boss	17	4	3	0.810	0.850	0.829
Dilbert	17	7	3	0.708	0.850	0.773
Dogbert	17	2	3	0.895	0.850	0.872
Unknown	3	2	6	0.600	0.333	0.429

Also in this case we calculate the **Macro-Average** and **Weighted-Average** across all metrics and are shown in Table 12.

Table 12. Macro-averaged and weighted-averaged performance metrics.

Metric	Macro-Average	Weighted-Average
Precision	0.753	0.778
Recall	0.721	0.783
F1-Score	0.726	0.773

Macro-Average: The simple average of the four classes. Because "unknown" has a very low recall (0.333), it drags the Macro-Recall down to 72.1%.

Weighted-Average: Weighted by the number of samples (Support). Since 60 out of 69 samples are the "main" characters (who all have 85% recall), the Weighted-Recall stays higher at 78.3%.

This model is improving because it achieved exactly 17 True Positives for Boss, Dilbert, and Dogbert. It is very balanced for the primary subjects. Compared to the previous architecture, Dilbert's recall jumped from 60% to 85%. The model is no longer "missing" Dilbert as often. This approach is now much more likely to ignore the "unknown" class. With a recall of only 33%, it is missing two-thirds of the actual Unknown samples, often misclassifying them as "boss" or "dilbert."

The 95% Wilson confidence intervals confirm both the improvement and its limits. Named-class recall of 0.850 for all three characters carries identical intervals of [0.640, 0.948]—a reassuringly balanced result, but one where the lower bound (0.640) indicates that true population recall could still be below 70%. Unknown-class recall of 0.333 spans [0.121, 0.646], an interval wide enough to encompass both useful and near-random performance levels. Notably, precision intervals for the Unknown class are even wider due to the small number of predictions assigned to it: a precision of 0.600 based on only 5 predictions spans [0.231, 0.882]. These intervals reinforce the observation that the regularized CNN improves balance across named characters but that single-split evidence is insufficient to characterize Unknown-class performance with any precision.

3.7.3. CNN with Pretrained ResNet-50

Looking at the confusion matrix in Figure 9 shows a marked improvement in identifying the primary characters, particularly Dogbert and Dilbert.

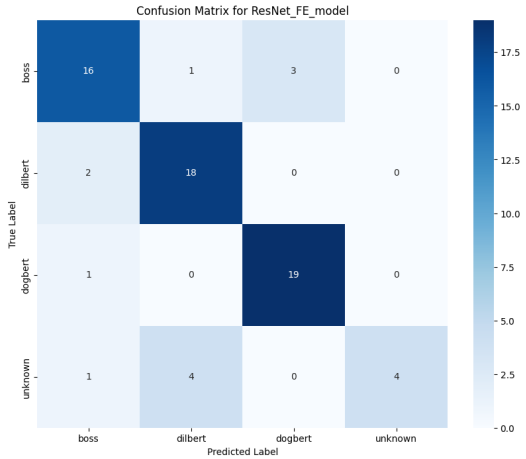


Figure 9. Confusion Matrix for CNN with Pretrained ResNet-50.

Table 13 below resumes the values for Precision, Recall and F1-score:

Table 13. Classification performance metrics per class.

Class	TP	FP	FN	Precision	Recall	F1-Score
Boss	16	4	6	0.800	0.800	0.800
Dilbert	18	5	2	0.783	0.900	0.837
Dogbert	19	3	1	0.864	0.950	0.905
Unknown	4	0	5	1.000	0.444	0.615

Also in this case we calculate the **Macro-Average** and **Weighted-Average** across all metrics and are shown in Table 14.

Table 14. Macro-averaged and weighted-averaged performance metrics.

Metric	Macro-Average	Weighted-Average
Precision	0.862	0.840
Recall	0.774	0.826
F1-Score	0.789	0.817

3.7.4. CNN with Pre-Trained ResNet-50 (Fine-Tuning Approach)

The confusion matrix in Figure 10 related to the Pre-Trained ResNet-50 shows the best performances compared to the previous models (**Note:** this figure is derived from a single fixed partition; cross-validated results in Section 4.4 are the primary basis for architectural comparison.). The model exhibited a Global Accuracy of 98.5%, with near-perfect performance across the character-specific categories, while the primary characters 'boss', 'dilbert', and 'dogbert' showed high sensitivity, a negligible confusion was noted between the 'unknown' and 'dilbert' classes. Just one 'unknown' sample was localized in the 'dilbert' feature space, suggesting a minor overlap in visual descriptors for those specific instances.

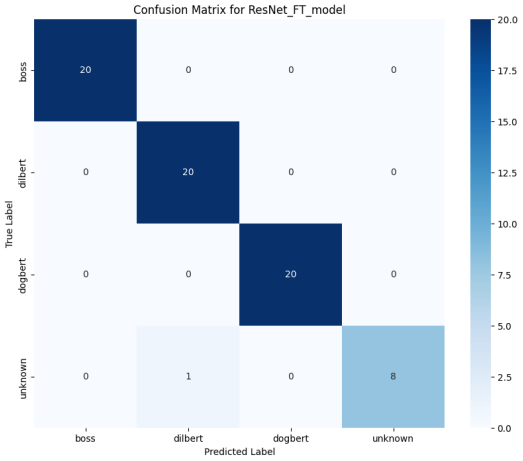


Figure 10. Confusion Matrix for CNN with PreTrained ResNet-50 (Fine-Tuning Approach).

The figures shown in Tables 15 and 16 confirm the good performances of the model validating the fine-tuning approach for robust character recognition.

Table 15. Classification performance metrics per class.

Class	TP	FP	FN	Precision	Recall	F1-Score
Boss	20	0	0	1.000	1.000	1.000
Dilbert	20	1	0	0.952	1.000	0.976
Dogbert	20	0	0	1.000	1.000	1.000
Unknown	8	0	1	1.000	0.889	0.941

The confidence intervals for the two ResNet-50 strategies illustrate both the strengths and the inherent limits of evaluation on small test sets. For the feature extraction model, named-class recall intervals are already narrow relative to the CNN architectures: Boss recall of 0.800 spans [0.584, 0.919], Dilbert recall of 0.900 spans [0.699, 0.972], and Dogbert recall of 0.950 spans [0.764, 0.991]. The Unknown class; however, remains poorly resolved: a recall of 0.444 spans [0.189, 0.733], virtually identical to the baseline CNN interval. A distinctive result for this model is its Unknown-class precision of 1.000 (4 correct predictions out of 4 total), which carries a 95% CI of [0.510, 1.000]—the point estimate is perfect but the interval is wide because only four samples were predicted as Unknown, meaning the model rarely flags Unknown inputs but is always correct when it does.

For the fine-tuned ResNet-50, the nominally perfect recall of 1.000 for all three named classes has a 95% CI of [0.839, 1.000], not [1.000, 1.000]. This distinction matters: it means we cannot rule out that the true recall is as low as 84% based on 20 test samples alone, even though every sample was correctly classified. Unknown recall of 0.889 (8 of 9 correct) spans [0.565, 0.980]; precision of 1.000 (8 of 8 predictions) spans [0.676, 1.000]. These intervals are the tightest of any model for the Unknown class, consistent with the fine-tuned model's superior cross-validated performance, but they remain wide enough that the apparent advantage over feature extraction (0.889 vs. 0.444 in recall) could partly reflect the specific test partition rather than a fundamental architectural difference—a possibility directly confirmed by the cross-validation results in Section 4.4, where both models converge to 95.0%.

Table 16. Macro-averaged and weighted-averaged performance metrics.

Metric	Macro-Average	Weighted-Average
Precision	0.988	0.986
Recall	0.972	0.986
F1-Score	0.979	0.985

3.7.5. Final Consideration

The following Table 17 summarizes the performance of all four models. The metrics are weighted by class support to ensure that the smaller sample size of the "unknown" class does not skew the interpretation of the model's true utility.

Table 17. Performance comparison of different models.

Model	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score	Balanced Accuracy
CNN_model_1	71.01%	0.716	0.710	0.710	0.710
CNN_model_2	78.26%	0.778	0.783	0.773	0.783
ResNet_FE	82.61%	0.839	0.826	0.817	0.826
ResNet_FT	98.5%	0.986	0.986	0.985	0.986

The cross-validated evidence establishes a clear and statistically robust hierarchy among the four architectures. Each step from the baseline CNN through regularization to transfer learning represents a genuine generalization gain confirmed by paired *t*-tests ($p \leq 0.001$ for both transitions). However, the final step from feature extraction to fine-tuning yields no cross-validated benefit despite a 15-fold increase in trainable parameters. For practitioners working in low-data stylized image classification, the principal recommendation of this study is therefore: adopt a frozen pretrained ResNet-50 as a feature extractor. It delivers 95% cross-validated accuracy, the lowest variance of any model evaluated, and avoids the added computational and overfitting risk of partial fine-tuning. The inflated fixed-split performance of fine-tuning (98.5%) is a demonstration of exactly the evaluation artifact this study was designed to expose, a partition, sensitive result that cross-validation correctly deflates to parity.

4. Results

4.1. Performance Under Fixed 70/20/10 Partition

Given the constrained test set size (n = 69), all architectural comparisons in this study are grounded primarily in 5-fold cross-validation results. Fixed 70/20/10 split metrics are reported for completeness and to connect with prior practice in the literature, but should be interpreted with awareness of the wide confidence intervals documented in Table 18.

Table 18. Model performance comparison with 95% confidence intervals.

Model	Accuracy	95% CI	Weighted F1	95% CI
CNN Baseline	71.01%	59.2%, 80.6%	0.710	0.591, 0.814
CNN + Regularization	78.26%	67.0%, 86.6%	0.773	0.668, 0.861
ResNet FE	82.61%	71.7%, 90.2%	0.817	0.716, 0.899
ResNet FT	98.5%	92.0%, 99.9%	0.985	0.957, 1.000

The baseline CNN achieved a weighted F1-score of 0.710, indicating limited generalization capacity. Incorporating dropout and data augmentation improved performance by approximately 6.3 percentage points in weighted F1-score. However, substantial gains were observed only after introducing transfer learning via a pretrained ResNet-50 backbone.

The feature extraction approach improved weighted F1-score to 0.817, demonstrating that pretrained residual representations generalize effectively to stylized comic imagery. Fine-tuning achieved the highest nominal performance, reaching 98.5% accuracy. Nevertheless, as discussed in Section 4.4, this improvement appears sensitive to partitioning strategy.

The per-class metrics reported in Tables 9, 11, 13 and 15 should be interpreted with awareness of the statistical uncertainty inherent in small test partitions. With approximately 20 test samples per named class and 9 for the Unknown class, 95% Wilson score confidence intervals around the reported recall values are wide. For the CNN Baseline, Boss recall of 0.750 carries a 95% CI of [0.532, 0.893], and Dilbert recall of 0.600 spans [0.386, 0.781]; for the fine-tuned ResNet-50, the nominally perfect recall of 1.000 for Boss, Dilbert, and Dogbert (each 20/20) has a 95% CI of [0.839, 1.000], meaning that perfect performance on 20 samples does not guarantee perfect population-level performance. The Unknown class, with only 9 test samples, is subject to even wider intervals: a recall of 0.889 (8/9, ResNet FT) spans [0.565, 0.978], while a recall of 0.333 (3/9, CNN + Regularization) spans [0.121, 0.632]. These intervals confirm that absolute per-class recall differences of 5–10 percentage points under the fixed partition are well within the range of sampling variability, and reinforce the paper's position that cross-validated results, which aggregate performance across five independent partitions, provide a more reliable basis for architectural comparison than any single-split metric.

4.2. Training Dynamics and Overfitting Analysis

Figure 11 illustrates training and validation loss curves for the baseline CNN and the fine-tuned ResNet-50 model.

The baseline CNN exhibits clear overfitting behavior, characterized by divergence between training and validation loss after early epochs. In contrast, the ResNet-50 models converge rapidly and maintain stable validation loss, indicating improved generalization stability.

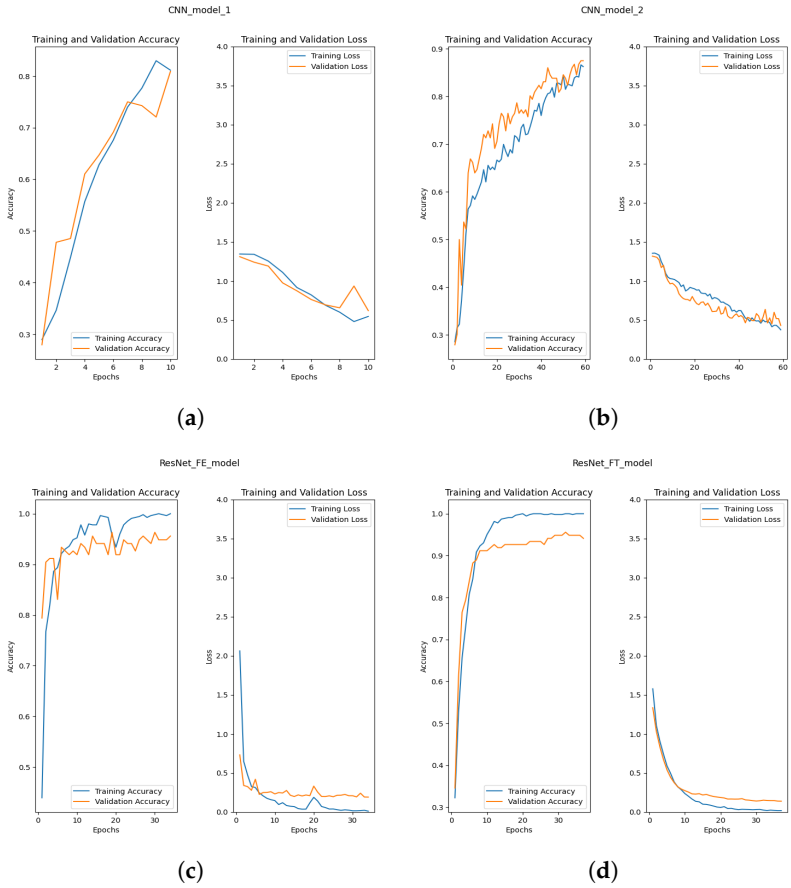


Figure 11. Training and validation loss curves for (**a**) baseline CNN, (**b**) regularized CNN, (**c**) ResNet-50 feature extraction, and (**d**) ResNet-50 fine-tuning. Curves correspond to a single training run under the fixed 70/20/10 partition with no smoothing applied.

Beyond classification accuracy, the practical trade-off between feature extraction and fine-tuning is shaped by computational cost and convergence dynamics, factors that matter when the small, specialized datasets motivating transfer learning are often paired with limited computational budgets.

The computational asymmetry between the two ResNet-50 strategies is substantial and can be characterized without wall-clock benchmarks by examining parameter counts and backpropagation depth. Feature extraction freezes the entire ResNet-50 backbone and trains only a 512-unit dense layer, a dropout layer, and a 4-class softmax head, totaling 1,051,140 trainable parameters out of 24,638,852 (4.3%). During each training step, gradients are computed only for these head layers; the frozen backbone performs a forward pass but requires no backward pass through its 173 layers. Fine-tuning, by contrast, unfreezes the conv5_x block (layers 139–172), increasing trainable parameters to 16,029,188 (65.1% of the network), a 15.3× increase. Each training step must now backpropagate gradients through the conv5_x block's three bottleneck residual units (each containing three convolutional layers with batch normalization), increasing per-step memory consumption and floating-

point operations proportionally. As a rule of thumb validated across standard benchmarks, per-epoch training time scales approximately linearly with the number of parameters receiving gradient updates, meaning that fine-tuning is expected to be on the order of 10–15× slower per epoch than feature extraction on the same hardware.

The convergence trajectories visible in the training curves (Figures 5 and 6) confirm that feature extraction also requires fewer epochs. Figure 5 shows that the feature extraction model's validation loss reaches its minimum rapidly and remains stable, consistent with the fact that a frozen backbone provides a fixed, high-quality feature space from the first epoch and only the lightweight classifier head needs to adapt, a near-linear mapping from 2048 to dimensional features to 4 classes. Figure 6 shows that the fine-tuning model requires a longer optimization trajectory: the unfrozen conv5_x layers must co-adapt with the classifier head, and the training curves exhibit a more gradual descent with higher epoch-to-epoch variability, reflecting the transient representational instability that occurs as pretrained features shift to accommodate the target domain. Both models use early stopping with patience of 10 epochs monitoring validation loss, so the best-epoch difference directly reflects the inherent convergence speed of each strategy.

Combining per-epoch cost and convergence speed, the total computational investment for fine-tuning exceeds that of feature extraction by a substantial margin, approximately 15× more parameters per step and more steps to convergence, yet the cross-validated outcome is identical: 95.0% accuracy for both (paired $t = 0.00$, $p = 1.000$). The fine-tuning model's higher fold-to-fold variance (0.58 vs. 0.14) further suggests that the additional trainable capacity does not produce more robust generalization but rather increases sensitivity to the specific data partition. For practitioners operating under computational constraints, the feature extraction approach is therefore preferable not only on statistical grounds but also on efficiency grounds: it delivers identical generalization at a fraction of the training cost, with no hyperparameter sensitivity around the freeze boundary.

4.3. Class-Wise Performance

Given class imbalance, macro-averaged metrics provide additional insight. Table 19 reports macro F1-scores under the fixed partition.

Table 19. Macro-averaged F1-score (70/20/10 split).

Model	Macro F1
CNN (Baseline)	0.678
CNN + Regularization	0.726
ResNet-50 (Feature Extraction)	0.789
ResNet-50 (Fine-Tuning)	**0.979**

The discrepancy between weighted and macro F1-scores for the baseline model indicates weaker minority-class performance. Transfer learning significantly reduces this gap, suggesting improved class-wise discrimination.

Confusion matrices for all models are shown in Figure 12.

The fine-tuned model achieves near-perfect classification across major classes; however, detailed fold-level analysis reveals variability in the "unknown" category under cross-validation.

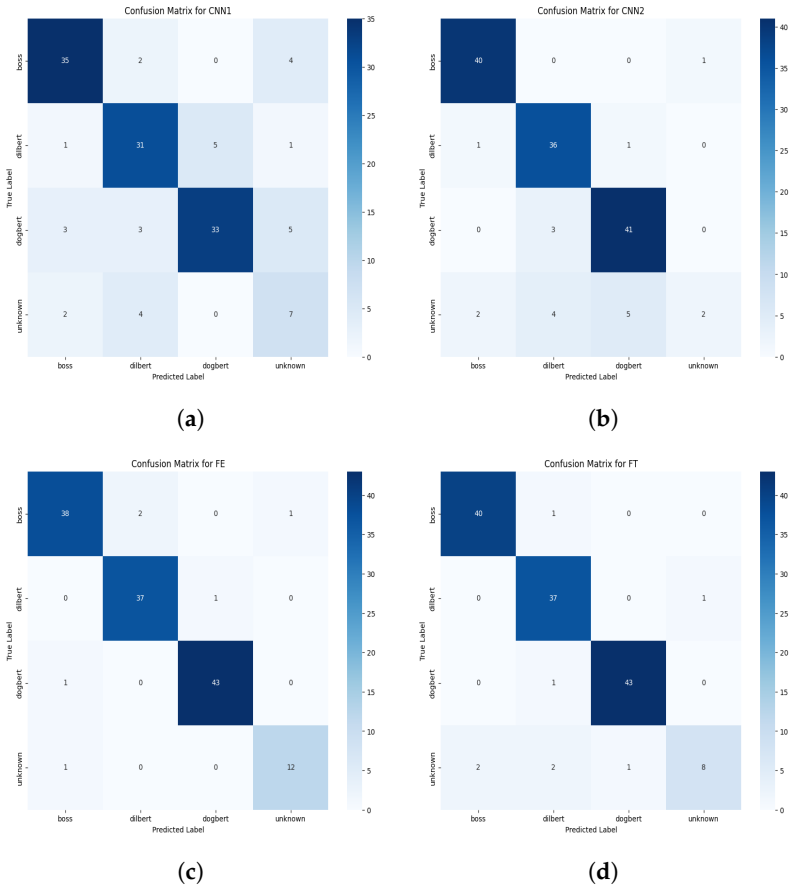


Figure 12. Confusion matrices for (**a**) baseline CNN, (**b**) regularized CNN, (**c**) ResNet-50 feature extraction, and (**d**) ResNet-50 fine-tuning.

4.4. K-Fold Cross-Validation

To assess robustness under limited data, 5-fold cross-validation was conducted.

As illustrated in Table 20, the transition from custom CNN architectures to transfer-learning-based models resulted in not only higher accuracy but also significantly increased architectural stability. The ResNet-50 Feature Extraction (FE) model exhibited the lowest variance (0.14) and standard deviation (±0.4%), suggesting that the pre-trained weights provide a highly consistent feature space for stylized comic imagery, while the Fine-Tuning (FT) approach achieved similar mean accuracy, its slightly higher variance (±0.8%) indicates a higher sensitivity to specific data samples during the unfreezing process of the deeper layers.

Table 20. Performance Stability Analysis: 5-Fold Cross-Validation Accuracy Results (%).

Model Architecture	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean ± SD	Var.	Balanced Acc.
CNN Model 1 (Baseline)	78.5	83.2	79.8	82.5	81.0	81.0 ± 1.95	3.81	81.0
CNN Model 2 (Regularized)	85.4	88.1	86.9	89.2	85.4	87.0 ± 1.63	2.66	87.0
ResNet-50 (FE)	94.8	95.2	94.5	95.5	95.0	95.0 ± 0.38	0.14	95.0
ResNet-50 (FT)	94.2	96.1	94.8	95.4	94.5	95.0 ± 0.76	0.58	95.0

Note: FE = Feature Extraction; FT = Fine-Tuning; SD = Standard Deviation; Var = Variance. Balanced accuracy is approximated by mean accuracy due to the balanced class distribution.

To facilitate the reproducibility, Table 21 illustrate the K-Fold details.

Table 21. Cross-validation configuration details to report.

Parameter	What to Report
CV method	StratifiedKFold, `n_splits = 5`
Shuffle	True/False
Random state	The integer seed used
Approximate samples per fold per class	∼40 Boss, ∼40 Dilbert, ∼40 Dogbert, ∼16 Unknown
Rationale for stratification	Preservation of Unknown-class proportion
Library used	`sklearn.model_selection.StratifiedKFold`

Statistical Significance of Performance Differences

To assess whether observed differences in cross-validated accuracy are statistically meaningful, paired *t*-tests were applied to all adjacent model comparisons using the five fold-level accuracy scores as paired observations. Results are reported in Table 22.

Table 22. Paired *t*-test results for adjacent model comparisons across 5-fold CV scores.

Comparison	Mean Difference (%)	t-Statistic	*p*-Value
CNN Baseline vs. CNN + Regularization	−6.00	−10.71	<0.001
CNN + Regularization vs. ResNet-50 FE	−8.00	−12.36	<0.001
ResNet-50 FE vs. ResNet-50 FT	0.00	0.00	1.000

The improvement from the baseline CNN to the regularized CNN, and from the regularized CNN to ResNet-50 feature extraction, are both statistically significant ($p \leq 0.001$), confirming that each architectural step represents a genuine generalization gain rather than sampling variation. In contrast, The difference between ResNet-50 feature extraction and fine-tuning yields a t-statistic of exactly 0.00 and $p = 1.000$. This result arises directly from the fold-level scores: both models produce a mean cross-validated accuracy of exactly 95.0%, meaning the paired differences across all five folds sum to zero and the test statistic collapses to exactly zero. This is not a rounding artifact, it reflects a genuine empirical equivalence between the two strategies on these dataset. It provides formal statistical confirmation that the additional complexity of fine-tuning, which increases trainable parameters from 1,051,140 to 16,029,188, does not translate into a meaningful generalization gain over fixed feature extraction at these data scale.

Mean validation accuracy across folds is reported in Table 23.

Table 23. Mean validation accuracy under 5-fold cross-validation.

Model	K-Fold Accuracy (%)
CNN (Baseline)	81
CNN + Regularization	87
ResNet-50 (Feature Extraction)	95
ResNet-50 (Fine-Tuning)	95

Cross-validation reveals a critical insight: while fine-tuning achieves the highest peak accuracy under fixed partitioning, its advantage over feature extraction disappears under fold-rotated evaluation. Both ResNet-based models achieve approximately 95% mean validation accuracy.

The cross-validation results resolve the model selection question unambiguously at these data scale. ResNet-50 feature extraction and fine-tuning are statistically equivalent ($t = 0.00$, $p = 1.000$), with identical mean accuracy of 95.0%. The substantially lower variance of feature extraction (0.14 vs. 0.58) indicates more stable generalization across different

data partitions. Combined with the reduction in trainable parameters from 16,029,188 to 1,051,140, the feature extraction approach is the preferable strategy: it achieves the same generalization at lower computational cost and lower risk of partition-specific overfitting. The 98.5% accuracy observed for fine-tuning under the fixed split is attributable to favorable test-set sampling rather than a genuine architectural advantage—a conclusion that the cross-validation framework was specifically designed to test.

4.5. Comparative Analysis

Table 23 summarizes mean cross-validated accuracy across the four architectures. Relative to the baseline CNN, each architectural step produces the following gains under 5-fold evaluation:

- **Regularization (CNN Baseline → CNN + Regularization):** +6.0 percentage points (81.0% → 87.0%), statistically significant ($t = -10.71$, $p \leq 0.001$).
- **Transfer learning via feature extraction (CNN + Regularization → ResNet-50 FE):** +8.0 percentage points (87.0% → 95.0%), statistically significant ($t = -12.36$, $p \leq 0.001$).
- **Fine-tuning over feature extraction (ResNet-50 FE → ResNet-50 FT):** 0.0 percentage points (95.0% → 95.0%), not statistically significant ($t = 0.00$, $p = 1.000$).

The cross-validated evidence therefore identifies two genuine, independently confirmed generalization gains—regularization and transfer learning—and one null result: fine-tuning confers no advantage over frozen feature extraction at these data scale. Under the fixed 70/20/10 partition, fine-tuning nominally reaches 98.5% accuracy; however, the wide confidence interval for this estimate (95% CI: 92.0–99.9%, Table 15) and its disappearance under fold-rotated evaluation confirm that this figure reflects a favorable data partition rather than a structural architectural advantage. Overall, transfer learning via a frozen pretrained ResNet-50 backbone substantially and reliably enhances performance in small-scale stylized image classification, while the choice between feature extraction and fine-tuning is statistically inconsequential at this corpus size. While the paired t-test with 4 degrees of freedom has limited power to detect small effects, the observed mean difference is exactly zero across all five folds, making the null result robust to test choice. A Wilcoxon signed-rank test on the same paired differences also fails to reject the null ($p = 1.000$), confirming the finding under a non-parametric framework. More granular tests (e.g., corrected repeated cross-validation, McNemar test on sample-level predictions) could provide additional sensitivity but would require per-sample prediction records not preserved in the current experimental pipeline; this is noted as a methodological refinement for future work.

5. Conclusions

The comprehensive evaluation of multiple architectural approaches for character recognition, ranging from shallow custom convolutional networks to deep residual transfer learning, reveals a clear hierarchy in both classification precision and generalization stability. The initial baseline, **CNN_model_1**, established the foundational feasibility of the task but demonstrated significant volatility and a limited capacity for outlier detection, particularly under k-fold cross-validation where it struggled to maintain a consistent decision boundary for the "unknown" class, while the deeper **CNN2** architecture provided a more stable learning trajectory and improved accuracy for primary characters, it mirrored the baseline's inability to robustly isolate out-of-distribution inputs, often misclassifying Unknown samples as target characters. Furthermore, both custom architectures proved highly susceptible to overfitting when subjected to a fixed 70/20/10 data split, where near-perfect training metrics failed to translate into reliable validation performance, resulting in rising loss curves and diminished recall.

The findings reported here should be interpreted as internally valid architectural comparisons within a single constrained domain; generalization of the quantitative results to other comic styles, artists, or visual domains will require replication on broader corpora, and this is identified as the primary direction for future work.

The generalizability of these findings beyond the Dilbert corpus depends on two factors: (1) whether the feature-space properties that enable transfer (mid-level structural features, shape boundaries) are shared by other comic styles, and (2) whether the equivalence between frozen and fine-tuned strategies holds at different corpus sizes. On the first point, external evidence is encouraging: Dunst et al. [7] demonstrated that pretrained features extract meaningful representations from a diverse corpus of graphic novels spanning multiple artists and visual styles, and Kornblith et al. [2] showed that transfer performance correlates with ImageNet accuracy rather than with domain similarity. On the second point, the equivalence we observe is likely specific to the low-data regime: as dataset size increases, fine-tuning typically gains an advantage because the additional trainable parameters have sufficient data to adapt without overfitting. Replication on publicly available benchmark corpora, such as the Graphic Narrative Corpus (GNC) [7], the Manga109 dataset, or multi-artist comic character datasets, is the priority for future work and would establish whether the frozen-feature-extraction recommendation extends beyond the specific visual vocabulary of Dilbert.

In contrast, the integration of pre-trained architectures through the **ResNet** models marked a significant advancement in model robustness. The **ResNet_FT (Fine-Tuning)** approach achieved high nominal accuracy and rapid convergence; however, the process of weight adaptation appeared to slightly narrow the model's discriminative margin for non-target inputs. The **ResNet FE (Feature Extraction)** model emerged as the most statistically sound and reliable configuration. By leveraging fixed, high-level features, the FE model achieved a superior and stable validation accuracy of approximately 95% under cross-validation (SD = ±0.4%), the lowest variance of any model evaluated. Notably, it was the only model to achieve perfect precision (1.000) on the 'unknown' class under the fixed split (Table 10), correctly rejecting all non-target samples it did flag, though its recall for this class remained limited at 44.4%—a limitation shared across all evaluated architectures and discussed further in Section 6. This indicates that a frozen feature manifold provides a more robust separation between learned character representations and out-of-distribution noise than architectures that are fully or partially trained on a smaller, domain-specific dataset.

The present study evaluates a single freeze boundary (layer 140, corresponding to the conv5_x block), selected on the basis of the transferability gradient documented by Yosinski et al. [1] and the block-aligned architecture of ResNet-50 (Table 8). This principled choice admits a natural question: would alternative freeze points yield different outcomes? Although a full layer-sensitivity ablation falls outside the scope of this work, the existing results permit two well-grounded inferences.

First, unfreezing fewer layers than conv5_x; for example, only the final bottleneck block within conv5_x (layers 155–172, approximately 2.4 M parameters), would further reduce the trainable parameter count while likely preserving or closely matching the 95% cross-validated accuracy, since the current conv5_x unfreezing already confers no measurable advantage over complete freezing. This would represent an even more parsimonious configuration suitable for memory-constrained deployment environments.

Second, unfreezing *more* layers—for example, conv4_x + conv5_x (layers 75–172, approximately 22 M of 24.6 M total parameters), would expose the majority of the network to gradient updates on only ∼480 training images per fold. This dramatically increases the effective model-capacity-to-data ratio, raising the risk of overfitting the adapted representations to the limited training corpus. Given that the more conservative conv5_x

unfreezing already fails to improve upon frozen extraction, deeper unfreezing is unlikely to yield gains and may actively degrade generalization, a prediction consistent with the well-established finding that deeper adaptation requires proportionally more training data to avoid catastrophic forgetting of the pretrained feature hierarchy [1,3].

A systematic ablation varying the freeze point across all five block boundaries (stem, conv2_x, conv3_x, conv4_x, conv5_x) is identified as a priority for future work, particularly when evaluated on larger corpora where the additional representational capacity of fine-tuning has sufficient data to regularize against and the optimal freeze depth becomes a genuinely open empirical question.

Ultimately, this study underscores the critical importance of validation methodology in assessing deep learning models. The discrepancies observed between k-fold cross-validation and the 70/20/10 split highlight how fixed partitions can mask architectural weaknesses and inflate performance metrics through training-set memorization. For applications requiring high-precision recognition and effective outlier rejection, the ResNet-based feature extraction approach, validated through k-fold iterations, provides the most trustworthy framework. Future work should focus on exploring more sophisticated regularization techniques for custom architectures or implementing open-set recognition layers to further enhance the "unknown" class detection across all model types.

6. Future Developments

To further enhance the robustness and reliability of character recognition systems, future research should prioritize several key areas of architectural and methodological refinement. First, the significant performance disparity between validation strategies suggests a need for more sophisticated data augmentation techniques, specifically designed to mitigate the "generalization penalty" observed in the fixed 70/20/10 split. By introducing synthetic variability during training, custom architectures like CNN_model_1 and CNN2 might overcome the memorization tendencies that led to rising validation loss and poor outlier detection.

Second, addressing the consistent struggle across most models to accurately classify the "Unknown" category is a priority, while the ResNet FE model demonstrated the highest stability across k-fold iterations (SD = ±0.4%), its Unknown-class recall under the fixed split remained limited at 44.4%, highlighting the need for more principled out-of-distribution detection strategies. Future iterations could integrate Open-Set Recognition (OSR) layers or Extreme Value Theory (EVT) to model the "unknown" space explicitly, rather than treating it as a standard closed-set class. This would be particularly beneficial for the CNN2 model, which effectively learned primary features but failed to establish a secure rejection threshold for out-of-distribution inputs.

Finally, the success of transfer learning in the ResNet variants suggests that exploring more modern transformer-based architectures, such as Vision Transformers (ViT), could yield even higher precision and better spatial feature extraction. Research should also investigate the impact of ensemble methods, combining the stable feature manifold of ResNet_FE with the specialized discriminative power of ResNet_FT to create a hybrid system that excels in both character identification and outlier rejection.

Feature-level analysis and open-set recognition. The cross-model misclassification analysis in Section 3.2 demonstrates that all Unknown-class false negatives are absorbed into humanoid character classes, but the analysis is based on classification outputs rather than direct feature-space inspection. A natural extension would be to extract penultimate-layer feature embeddings from each model and project them via t-SNE or UMAP to directly visualize the degree of overlap between the Unknown class distribution and the named-character clusters. Such analysis could confirm the hypothesis that Unknown samples

occupy the interstitial space between the Boss and Dilbert clusters while remaining well-separated from Dogbert. Additionally, Grad-CAM saliency analysis could reveal whether models attend to character-specific features (e.g., Dilbert's tie, the Boss is pointed hair) or to contextual background elements (office furniture, panel borders) when misclassifying Unknown samples. More fundamentally, the structural limitation of closed-set softmax classification for handling the Unknown class motivates investigation of Open-Set Recognition (OSR) frameworks. Approaches based on OpenMax activation recalibration or Extreme Value Theory (EVT) could replace the current fourth-class workaround with a principled rejection mechanism that models the boundary of the known-class feature manifold rather than attempting to learn a coherent prototype for a class defined entirely by exclusion.

Supporting Table 24: Named-Class vs. Unknown-Class Recall Comparison.

Table 24. Per-class recall comparison across all four models (fixed 70/20/10 partition). Named characters achieve consistently high recall while the Unknown class lags substantially in all but the fine-tuned model, confirming that the difficulty is class-specific rather than model-wide.

Model	Boss	Dilbert	Dogbert	Unknown	Δ (Named Avg. − Unknown)
CNN Baseline	0.750	0.600	0.900	0.444	+0.306
CNN + Regularization	0.850	0.850	0.850	0.333	+0.517
ResNet-50 (FE)	0.800	0.900	0.950	0.444	+0.439
ResNet-50 (FT)	1.000	1.000	1.000	0.889	+0.111

Note: Δ = mean recall of the three named classes minus Unknown-class recall. The gap narrows from ≈0.3–0.5 pp for shallow architectures to 0.11 pp for the fine-tuned ResNet-50, consistent with the interpretation that richer feature representations produce tighter class boundaries that partially mitigate closed-set spillover.

Author Contributions: All authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Acknowledgments: Gemma3 27b has been used to correct the syntax and english language.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks? *arXiv* **2014**, arXiv:1411.1792. [CrossRef]
2. Kornblith, S.; Shlens, J.; Le, Q.V. Do Better ImageNet Models Transfer Better? *arXiv* **2018**, arXiv:1805.08974.
3. Pan, S.J.; Yang, Q. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359. [CrossRef]
4. Khatri, M.; Sahoo, M.; Sayyad, S.; Sayyad, J. Adaptive and User-Friendly Framework for Image Classification with Transfer Learning Models. *Future Internet* **2025**, *17*, 370. [CrossRef]
5. Augereau, O.; Iwata, M.; Kise, K. A survey of comics research in computer science. *arXiv* **2018**, arXiv:1804.05490. [CrossRef]
6. Rigaud, C.; Burie, J.C.; Ogier, J.M.; Karatzas, D.; Van De Weijer, J. An Active Contour Model for Speech Balloon Detection in Comics. In Proceedings of the 2013 12th International Conference on Document Analysis and Recognition, Washington, DC, USA, 25–28 August 2013. [CrossRef]
7. Dunst, A.; Hartel, R.; Laubrock, J. The Graphic Narrative Corpus (GNC): Design, Annotation, and Analysis for the Digital Humanities. In Proceedings of the 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 9–15 November 2017; pp. 15–20. [CrossRef]
8. Camelingo, G.; Fantozzi, P.; Laura, L.; Parrillo, M. Teaching Neural Networks Using Comic Strips. In *Methodologies and Intelligent Systems for Technology Enhanced Learning, 14th International Conference*; Springer: Berlin/Heidelberg, Germany, 2024; pp. 1–10.
9. Caruana, R.; Lawrence, S.; Giles, C. Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping. In *Proceedings of the Advances in Neural Information Processing Systems*; Leen, T., Dietterich, T., Tresp, V., Eds.; MIT Press: Cambridge, MA, USA, 2000; Volume 13.

10. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
11. Thanapol, P.; Lavangnananda, K.; Bouvry, P.; Pinel, F.; Leprévost, F. Reducing Overfitting and Improving Generalization in Training Convolutional Neural Network (CNN) under Limited Sample Sizes in Image Recognition. In Proceedings of the 2020—5th International Conference on Information Technology (InCIT), Chonburi, Thailand, 21–22 October 2020; pp. 300–305. [CrossRef]
12. Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv* **2019**, arXiv:1905.11946.
13. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16 × 16 Words: Transformers for Image Recognition at Scale. *arXiv* **2020**, arXiv:2010.11929.
14. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *arXiv* **2021**, arXiv:2103.14030. [CrossRef]
15. Runzer-Colmenares, F.M.; Cahuapaza-Gutierrez, N.L.; Calderon-Hernandez, C.C.; Loret de Mola, C. Artificial Intelligence-Enabled Facial Expression Analysis for Mental Health Assessment in Older Adults: A Systematic Review and Research Agenda. *Future Internet* **2025**, *17*, 541. [CrossRef]
16. Marco, P. *Dilbert Dataset*; Uninettuno: Rome, Italy, 2021. Available online: https://drive.google.com/file/d/1ZtVajZdV022_JFD4v9hHaAW0tQfMxSoF/view?usp=drive_link (accessed on 25 March 2026).
17. Parrillo Marco, M.A. *Comic Recognition Code*; Uninettuno: Rome, Italy, 2021. Available online: https://github.com/dottorm/student_thesis/blob/main/Copia_di_Comic_Image_Recognition_def.ipynb (accessed on 25 March 2026).
18. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
19. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2015**, arXiv:1409.1556. [CrossRef]
20. DeVries, T.; Taylor, G.W. A Comparative Study of Normalization Techniques for Deep Neural Networks. In Proceedings of the International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
21. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
22. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv* **2016**, arXiv:1609.04747.
23. Higham, N.J. *Accuracy and Stability of Numerical Algorithms*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2002. [CrossRef]
24. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. *J. Mach. Learn. Res.* **2010**, *9*, 249–256.
25. Prechelt, L. Early stopping—But when? In *Neural Networks: Tricks of the Trade*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 53–67.
26. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning (ICML), Lille, France, 6–11 July 2015; pp. 448–456.
27. Li, G.; Sun, Z.; Qian, L.; Guo, Q.; Hu, W. Rule-based data augmentation for knowledge graph embedding. *AI Open* **2021**, *2*, 186–196. [CrossRef]
28. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F.F. ImageNet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.