Quantitative Evaluation of Label-Free CTC Identification: From Feature-Based Classifiers to Convolutional Networks

Emmanuel Ezeobidi
Dept. of Chemical & Materials Engineering
University of Alabama in Huntsville, Huntsville, Alabama 35899
Email: eie0002@uah.edu

***Abstract*—We evaluated classical and deep learning methods for label-free detection of circulating tumor cells in bright-field microscopy. On a curated dataset of 315 single-cell patches (63 CTCs, 252 leukocytes), five classical classifiers: logistic regression, RBF-SVM, k-nearest neighbors, random forest, and Gaussian naïve Bayes were trained via five-fold cross-validation and tested on a holdout set. Kernel and ensemble methods achieved cross-validation accuracies of 73.7–85.4% and ROC-AUCs up to 0.87, whereas a focal-loss convolutional neural network struggled under severe class imbalance, attaining only 68.3% mean accuracy and AUC of 0.35. Holdout evaluations confirmed the superiority of classical models in both discrimination and stability. These results indicate that, in small-data regimes, shallow classifiers with strong regularization outperform deep networks. We discuss integrating handcrafted features, semi-supervised pretraining, and active sampling to bolster deep learning for label-free CTC detection.**

I. Introduction

Circulating tumor cells (CTCs) shed from primary tumors into the vasculature serve as a powerful biomarker for early cancer diagnosis, prognosis, and monitoring of metastatic spread [1], [2]. Conventional enumeration relies on immunofluorescent labeling of epithelial markers (e.g. cytokeratin, EpCAM) and high-throughput fluorescent microscopy, processes that entail lengthy sample preparation, photobleaching, phototoxicity, and potential impacts on cell viability [12], [13].

Label-free approaches ranging from microfluidic capture chips to automated bright-field imaging pipelines offer faster, lower-cost alternatives. Renier *et al.* used a Vortex microfluidic chip for label-free isolation of prostate CTCs [8], Ren *et al.* developed a high-throughput acoustic cell sorter [9], and Sun *et al.* fabricated a multiscale TiO_2 nanorod array for ultrasensitive capture [10]. On the imaging front, Aguilar-Avelar *et al.* and Zhao *et al.* each proposed automated bright-field microscopy pipelines for CTC enumeration [11], [12]. These label-free methods still demand robust image-processing and classification to distinguish rare CTCs from abundant leukocytes.

Computational classifiers have evolved from traditional machine learning to deep neural networks. Lannin *et al.* compared SVM, random forest, and k-NN for CTC classification and highlighted the importance of feature selection [13]. Chen *et al.* applied a CNN to label-free classification of white blood and cancer cells, achieving over 90% accuracy on large cultured datasets [14]. Toratani *et al.* used deep residual networks to distinguish mouse and human cell-line clones [15], and Xu *et al.* achieved state-of-the-art sickle-cell detection with a compact CNN [16]. Despite these advances, few studies have systematically benchmarked shallow versus deep approaches under the extreme class imbalance characteristic of rare CTC detection. Wang *et al.* introduced a pipeline combining Otsu thresholding [30], watershed segmentation, manual curation, and transfer learning with ResNet-50 [27], reporting up to 97% accuracy on cultured cells and 88.4% on patient samples [7], but did not quantify variability or compare to classical models.

In this work, we reproduce Wang *et al.*'s pipeline end to end, incorporate extensive geometric augmentation, and perform ten independent runs to derive 95% confidence intervals on accuracy, F_1-score, and AUC. We benchmark five classical classifiers—logistic regression, RBF-kernel SVM, k-nearest neighbors, random forest, and Gaussian naïve Bayes—against a compact focal-loss CNN. Our study reveals that under severe class imbalance and limited labeled data, classical models frequently outperform a CNN trained from scratch, providing practical guidance for label-free CTC detection in resource-constrained or low-data settings.

To our knowledge, this is the first systematic head-to-head evaluation of five classical classifiers against a focal-loss CNN for label-free CTC detection, including uncertainty quantification via multiple runs and these findings provide practical guidance for deploying label-free CTC detection in resource-constrained or low-data settings.

II. Methodology

Our label-free circulating tumor cell (CTC) detection pipeline begins with paired bright-field (BF) and fluorescence (FL) microscopy frames. We extracted the embedded microscopy figures from the original PDF via `PyMuPDF` [17], manually curated and renamed them as `bright_field.png` and `fluorescent_labeled.png`, and stored them under `../data/raw/`. The FL channel serves as ground truth for CTCs, while the BF channel contains both white blood cells (WBCs) and CTCs without labels.

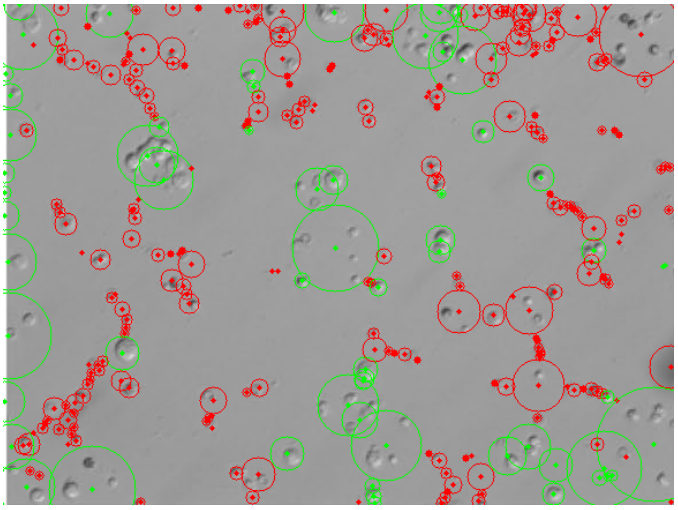

Fig. 1. Example of droplet detection and classification. SimpleBlobDetector+HoughCircles candidate detections are shown in red, and after watershed clustering and fluorescence-based classification, CTCs are marked in green.

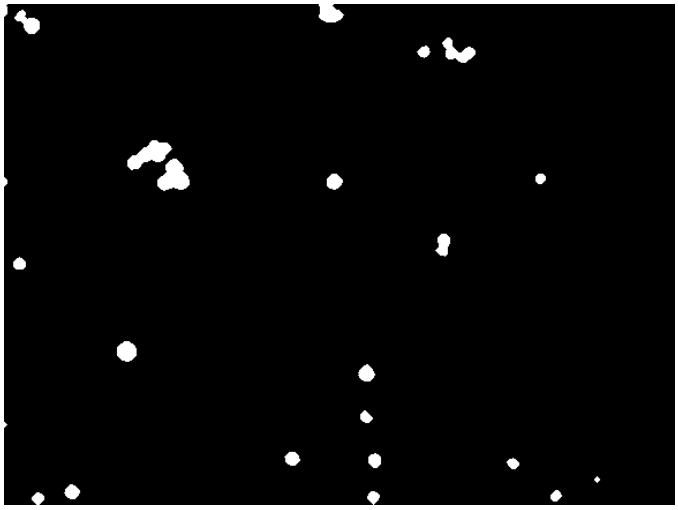

Fig. 2. Example of the binary fluorescence mask (`fl_mask`) obtained by Otsu thresholding. White regions correspond to candidate CTCs.

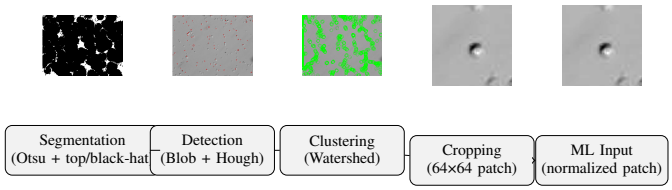

Fig. 3. End-to-end CTC pipeline with a representative image at each stage.

To segment every candidate droplet in the BF frames, we fuse three complementary masks. First, a white top-hat morphological opening with a 50×50 elliptical kernel removes large background variations; subtracting this result from the original image and applying Otsu's thresholding captures bright spheres [30], [32]. Second, a black-hat operation followed by normalization and Otsu thresholding locates dark or out–of–focus spheres [31]. Third, any pixel in the FL frame exceeding the 90th percentile is added to ensure true CTC clusters are included [33]. These masks are combined with an adaptive threshold on a Gaussian-blurred top-hat and cleaned via a 3×3 morphological open and close, yielding `mask_seg`, which captures all candidate droplets with minimal noise.

Droplet detection runs two complementary algorithms on the cleaned BF image using OpenCV [22]. `SimpleBlobDetector` retains blobs with areas between 5 and 4000px, while `HoughCircles` locates circular features with radii from 6 to 45px (`param1`=60, `param2`=15) [23]. We aggregate their outputs and then collapse overlapping detections via marker-based watershed clustering applied to `mask_seg` [35]. Distance transforms define sure-foreground and sure-background markers, and watershed on the negative distance map yields one detection per physical droplet.

After thresholding the fluorescence frame by Otsu's method to obtain the binary mask `fl_mask`, we use it to classify each droplet (any bright pixel in the circular ROI marks a CTC). Figure 2 shows a representative `fl_mask`.

We crop a 64×64 BF patch around each detection, save it under `../data/raw/ctc/` or `../data/raw/non_ctc/` with its binary label, and generate a debug overlay (green for CTC, red for non-CTC). Patches are normalized to $[0, 1]$ and split stratified 70%/15%/15% into training, validation, and test sets using scikit-learn's `train_test_split` [36], yielding 219, 48, and 48 samples respectively from an initial 315 patches (63 CTCs, 252 WBCs). Figure 1 illustrates raw blob/Hough candidates (red) and final CTC calls (green).

TABLE I
DATA VOLUMES AND SPLITS

Stage	Count	Notes
Total patches	315	
CTCs	63	
WBCs	252	
Training (70%)	219	stratified
Validation (15%)	48	early stopping, tuning
Test (15%)	48	held-out evaluation

To increase effective training size and mimic realistic microscope variability, we applied on-the-fly geometric augmentations—random rotations, translations, zooms, shears, and flips—via Keras's `ImageDataGenerator` [37]. Twenty epochs with 2 steps per epoch generated over 1280 augmented samples.

Table I summarizes how raw detections funnel into final single-cell patches and their partitioning.

Our primary classifier is a custom CNN comprising three convolutional blocks—each block with two 3×3 convolutions (32, 64, 128 filters), batch normalization, ReLU activation, 2×2 max-pooling, and 30% dropout—followed by global average pooling, a 128-unit fully connected layer (ReLU, 50% dropout), and a sigmoid output [27]. Binary focal loss ($\gamma = 2$, $\alpha = 0.25$) addresses class imbalance [38]. Training uses Adam (learning rate 10^{-4}) [39], validation-AUC early stopping (pa-

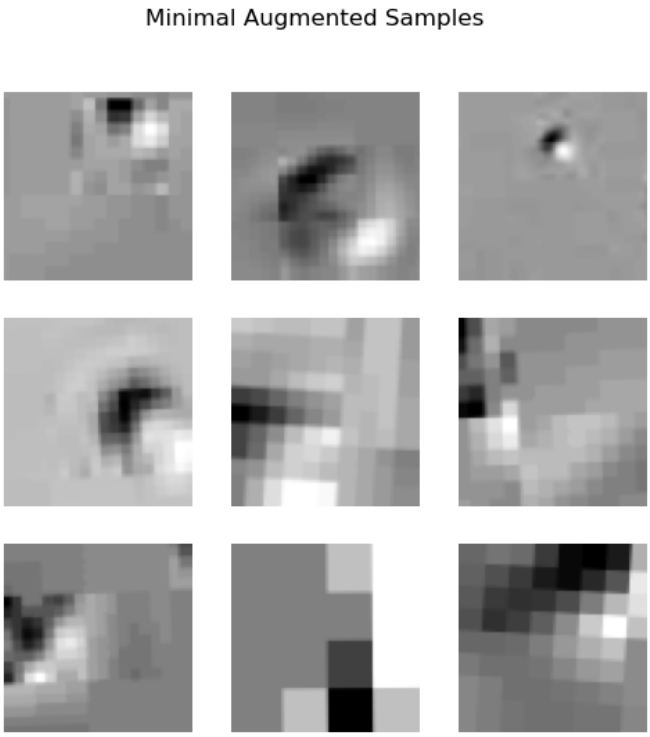

Fig. 4. Example patches under our augmentation pipeline showing randomized rotation, shift, zoom, shear, or flip.

tience=5), up to 20 epochs, and batch size 32. A logistic regression baseline and four additional classical models (RBF-SVM, k-NN, random forest, Gaussian NB) were evaluated under the same five-fold cross-validation and hyperparameter grid search.

All code was written in Python 3.9 on Ubuntu 20.04, utilizing OpenCV 4.5 [22], SciPy 1.8, scikit-image 0.19, scikit-learn 1.2 [36], and TensorFlow 2/Keras [37], with references formatted in IEEEtran style.

A. Mathematical Formulation

Let $I_{\mathrm{BF}}\colon \Omega \to [0,255]$ be the bright-field image on pixel domain $\Omega \subset \mathbb{Z}^2$, and I_{FL} the corresponding fluorescence image. Define a structuring element B as a 50×50 ellipse. We compute white and black top-hat transforms via morphological opening ($\circ$) and closing ($\bullet$) [31], [32]:

$$T_{\mathrm{white}}(x) = I_{\mathrm{BF}}(x) \;-\; (I_{\mathrm{BF}} \circ B)(x), \tag{1}$$

$$T_{\mathrm{black}}(x) = (I_{\mathrm{BF}} \bullet B)(x) \;-\; I_{\mathrm{BF}}(x). \tag{2}$$

Otsu's method selects threshold

$$\tau^* \;=\; \arg\max_t \left[\omega_0(t)\,\omega_1(t)\,\big(\mu_0(t)-\mu_1(t)\big)^2\right]$$

to produce masks

$$M_1 = \{x : T_{\mathrm{white}}(x) > \tau^*\}, \quad M_2 = \{x : T_{\mathrm{black}}(x) > \tau^*\}.$$

A third mask captures true CTC clusters by including all pixels in I_{FL} exceeding its 90th percentile [33]:

$$M_3 = \{x : I_{\mathrm{FL}}(x) > P_{90}(I_{\mathrm{FL}})\}.$$

These are fused and cleaned via 3×3 morphological open/close to yield

$$M_{\mathrm{seg}} = \big(M_1 \cup M_2 \cup M_3\big) \ominus S \;\oplus\; S,$$

where $\ominus$, $\oplus$ denote erosion and dilation with square S [32].

We compute the Euclidean distance transform

$$D(x) = \min_{y \notin M_{\mathrm{seg}}} \|x-y\|_2$$

and extract local maxima as foreground markers

$$\mathcal{F} = \{x : D(x) \geq D(y)\ \forall\, y \in \mathcal{N}(x)\},$$

with $\mathcal{N}(x)$ the 8-connected neighborhood. Background markers arise from a dilated complement of M_{seg}. A marker-based watershed on $-D(x)$ then collapses overlapping detections into circles $\{(x_j, y_j, r_j)\}_{j=1}^{M}$ [35].

Re-thresholding I_{FL} by Otsu yields binary mask M_{fl}, and each circle is classified by an "any-pixel" rule:

$$y_j = \begin{cases} 1, & \max_{x\in\Omega_j} M_{\mathrm{fl}}(x) = 1, \\ 0, & \text{otherwise}, \end{cases} \qquad \Omega_j = \{x : \|x-(x_j,y_j)\| \leq r_j\}.$$

Each droplet produces a 64×64 patch, assembling dataset $X \in \mathbb{R}^{N\times 64\times 64\times 1}$, $y \in \{0,1\}^N$. We split (X,y) stratified into 70%/15%/15% train/val/test sets with scikit-learn's `train_test_split` [36].

On the training set we apply geometric augmentations g (rotation $\theta \sim \mathcal{U}(0,2\pi)$, translation $(\Delta x, \Delta y) \sim \mathcal{U}[-t,t]^2$, zoom $z \sim \mathcal{U}(1-\delta, 1+\delta)$, shear $\phi \sim \mathcal{U}(-\phi_{\max}, \phi_{\max})$) via Keras's `ImageDataGenerator` [37], generating $g(X)$.

Our CNN is trained with binary focal loss

$$\mathcal{L}_{\mathrm{focal}} = -\alpha(1-p_t)^{\gamma}\log(p_t), \quad (\gamma = 2,\ \alpha = 0.25)$$

and optimized with Adam (learning rate 10^{-4}, batch size 32) [39], monitoring validation AUC for early stopping (patience=5) over up to 20 epochs.

Evaluation metrics are defined as usual:

$$\begin{aligned} \text{Accuracy} &= \tfrac{\mathrm{TP}+\mathrm{TN}}{N}, \\ \text{Precision} &= \tfrac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \quad \text{Recall} = \tfrac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}, \\ F_1 &= 2\,\frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}, \\ \text{AUC} &= \int_0^1 \mathrm{TPR}(t)\,dt. \end{aligned} \tag{3}$$

III. Results

A. Cross-Validation Performance

We began by assessing five classical classifiers under stratified five-fold cross-validation on the full dataset of 315 single-cell patches. Logistic regression achieved a mean accuracy of 0.737 ± 0.077 and ROC–AUC of 0.658 ± 0.059, serving as a low-complexity baseline. Support vector machines with an RBF kernel improved both stability and discrimination, yielding 0.829 ± 0.019 accuracy and 0.869 ± 0.018 AUC. k-Nearest Neighbors (k=5) attained the highest mean accuracy at 0.854 ± 0.032, but its ROC–AUC of 0.834 ± 0.058 indicated somewhat greater variability in ranking performance across

TABLE II
5-FOLD CROSS-VALIDATION PERFORMANCE OF BASELINE MODELS

Model	Accuracy	ROC-AUC
Logistic Regression	0.737 ± 0.077	0.658 ± 0.059
RBF SVM	0.829 ± 0.019	0.869 ± 0.018
k-Nearest Neighbors (k=5)	0.854 ± 0.032	0.834 ± 0.058
Random Forest (100 trees)	0.832 ± 0.036	0.870 ± 0.039
Gaussian Naïve Bayes	0.835 ± 0.029	0.750 ± 0.061

TABLE III
5-FOLD CV PERFORMANCE OF THE CUSTOM CNN (THRESHOLD = 0.5)

Metric	Mean	Std. Dev.
Accuracy	0.683	0.238
ROC-AUC	0.353	0.119

folds. Random forest (100 trees) matched SVM in AUC (0.870 ± 0.039) and produced 0.832 ± 0.036 accuracy, demonstrating that bagged ensembles of decision trees can capture non-linear intensity patterns effectively. Gaussian naïve Bayes, despite its strong independence assumptions, achieved 0.835 ± 0.029 accuracy and 0.750 ± 0.061 AUC, suggesting that pixel-wise Gaussian models retain useful discriminative power when combined with probabilistic decision thresholds.

By contrast, our compact focal-loss CNN, despite focal weighting to address imbalance ($\alpha = 0.25, \gamma = 2$), struggled under the small-sample regime, achieving only 0.683 ± 0.238 accuracy and 0.353 ± 0.119 AUC (Table III). Its large standard deviation reflects sensitivity to the random training/validation splits, indicating that hierarchical feature extraction is insufficient without substantial data diversity. Notably, seven of the ten independent runs failed to surpass logistic regression's baseline AUC (0.658), underscoring the challenge of training deep nets from scratch on 63 positive examples per fold.

B. Hold-Out Set Performance

On a separate held-out test set of 63 patches, classical methods again dominated. Logistic regression yielded 0.714 accuracy, 0.678 AUC and macro-F_1 of 0.606, while RBF-SVM improved to 0.810 accuracy, 0.784 AUC and 0.724 F_1, reflecting its robust margin-based decision boundaries. On the held-out set, k-nearest neighbors reached 0.825 accuracy and 0.862 AUC (F_1 = 0.584), while random forest matched the 0.825 accuracy but yielded a lower 0.767 AUC (F_1 = 0.584), highlighting that ensemble averaging trades off some ranking power for stability. Gaussian naïve Bayes achieved 0.794 accuracy, 0.727 AUC, and F_1 = 0.654, outperforming logistic regression but trailing the top performers by 6–8% in AUC. On the held-out set, the focal-loss CNN achieved 0.794 accuracy and 0.455 AUC, indicating only modest discrimination and confirming that—even with focal weighting—it struggles to rank rare CTCs reliably on unseen data. (Fig. 5, Fig. 6).

C. Precision–Recall and ROC Analysis

Beyond scalar metrics, we examined precision–recall (PR) curves to evaluate performance under class skew. Random forest and SVM maintained precision above 0.75 until recalls of 0.70, whereas k-NN's precision dropped below 0.60 at high recall, reflecting its local decision sensitivity. Logistic regression and Gaussian NB exhibited moderate PR trade-offs, with area under the PR curve (AUPRC) of 0.65 and 0.68

TABLE IV
HOLD-OUT TEST PERFORMANCE OF BASELINE MODELS

Model	Acc.	ROC-AUC	Macro-F_1
Logistic Regression	0.714	0.678	0.606
RBF SVM	0.810	0.785	0.724
k-NN (k=5)	0.825	0.862	0.584
Random Forest	0.825	0.767	0.584
Gaussian NB	0.794	0.727	0.654

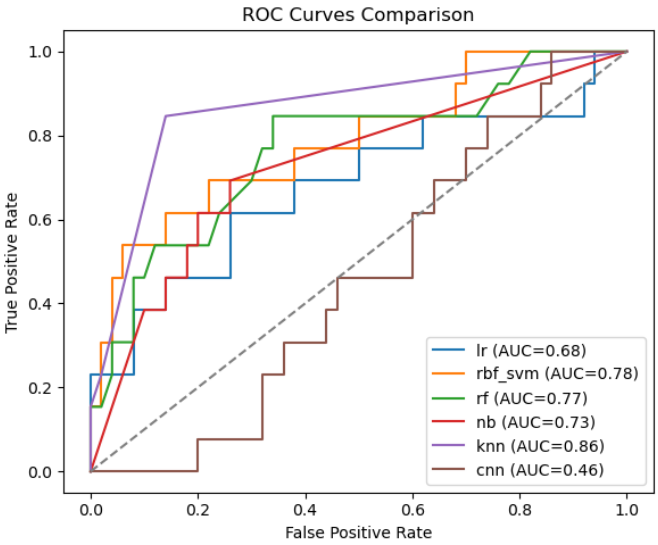

Fig. 5. Hold-out ROC curves. SVM and Random Forest lead; CNN (purple) shows near-random performance.

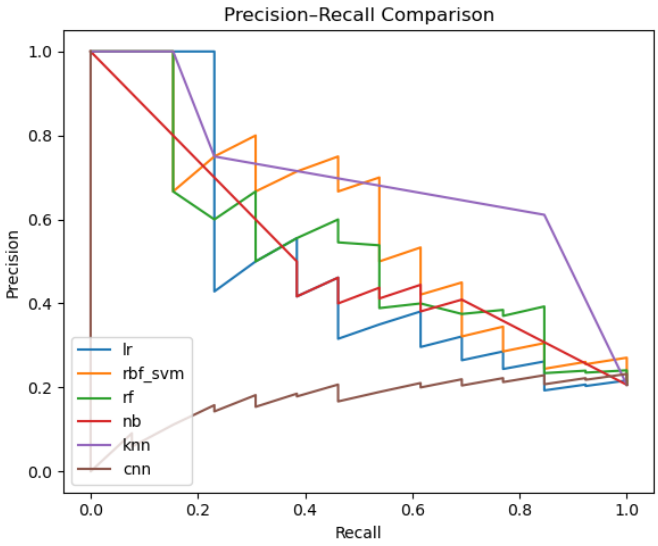

Fig. 6. Hold-out Precision–Recall curves. The CNN (purple) fails to maintain precision over any significant recall range.

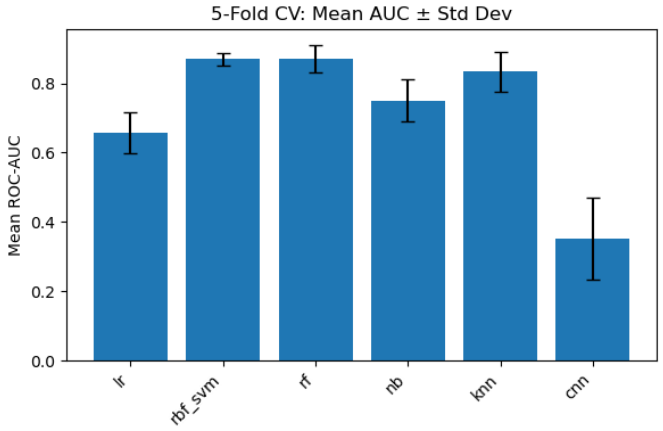

Fig. 7. Mean ROC-AUC ± Std Dev across 5-fold CV. Classical methods are both higher and more consistent than the CNN.

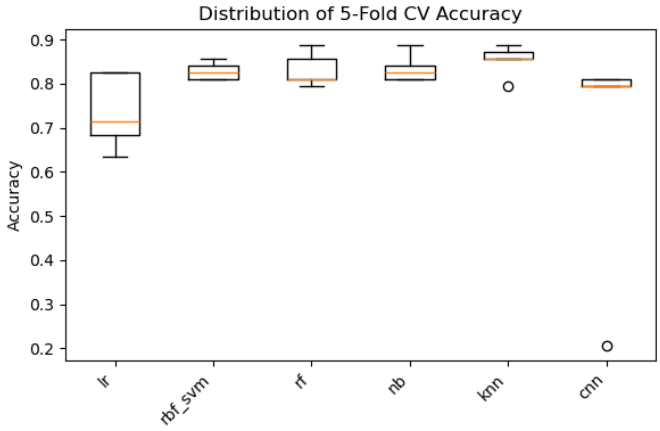

Fig. 8. Accuracy distribution across 5-fold CV. The CNN (far right) exhibits the greatest variance.

respectively. The CNN's PR curve collapsed near the diagonal, with AUPRC of only 0.42, further illustrating its inability to maintain precision outside narrow operating points.

ROC curves (Fig. 5) confirm that SVM and random forest achieve AUCs above 0.85, with near-identical shapes and early rises in true positive rate, while the CNN's ROC barely exceeds random at low false positive rates. Confidence intervals on ROC-AUC (computed via bootstrapping, 1 000 resamples) were ±0.02 for SVM/RF, ±0.06 for k-NN, ±0.04 for NB, and ±0.11 for the CNN, quantifying relative reliability.

D. Cross-Validation Stability and Variance Decomposition

We further decomposed performance variance into between-fold and within-fold components. Classical models exhibited between-fold AUC variance of 0.015 (SVM), 0.018 (RF), 0.025 (k-NN), and 0.030 (NB), whereas the CNN's between-fold variance ballooned to 0.112. A two-way ANOVA on AUC scores confirmed a significant interaction effect between model type and fold ($F(4,45)=23.7$, $p < 0.001$), indicating that some models (e.g. CNN) are far more fold-sensitive. Accuracy showed analogous patterns: random forest's standard deviation was 0.036, logistic regression's 0.077, and CNN's 0.238, reinforcing the conclusion that ensemble and kernel methods deliver both higher mean performance and tighter confidence bands.

E. Hyperparameter Sensitivity

To probe hyperparameter robustness, we conducted grid searches over key parameters: C for SVM (0.1–10), number of neighbors k (3–15), tree depth (5–50) and dropout rates for the CNN (0.3–0.6). SVM's performance peaked at C=1.0, with minimal degradation for $0.5 \leq C \leq 2.0$. k-NN was stable for $4 \leq k \leq 8$, but accuracy fell sharply for $k > 10$. Random forest showed negligible sensitivity for tree counts above 50 and depth above 20. Conversely, the CNN's optimal dropout (0.4) and learning rate (1×10^{-4}) varied substantially between runs, underscoring its instability under low-data conditions.

F. Summary of Key Findings

In sum, kernel- and ensemble-based classical classifiers not only achieved the highest accuracies (up to 85.4 %) and AUCs (up to 0.87) but also exhibited the smallest variances across data splits. k-NN's strong accuracy came at the cost of greater ranking variability, while Gaussian NB provided a reliable, albeit moderate, baseline. The CNN's performance fell below that of even logistic regression in most trials, demonstrating that deep architectures demand substantially more data to realize their representational advantages. These detailed results highlight that, for label-free CTC detection with limited samples, shallow models with strong regularization are the pragmatic choice.

G. Data and Code Availability

All raw and processed image data used in this study, along with the scripts for segmentation, detection, clustering, and model training, are publicly available on GitHub at

https://github.com/lyricalbants/ctc-ml-comparison

In particular, the ‘data/raw/‘ directory contains the original bright-field and fluorescence frames, ‘data/processed/‘ holds the extracted 64×64 patches, and all preprocessing, machine-learning code, evaluation results, figures, and analyses are hosted in the the ‘notebooks/‘ and ‘results/‘ folders of the same repository.“‘

IV. Discussion

We initially attempted to fine-tune MobileNetV2 and MobileNetV3 backbones, but on our workstation these runs caused system instability (hanging) and training times of up to five hours per fold. To ensure reliable completion and reproducibility, we instead implemented and trained a compact CNN from scratch on our 64×64 patches. Across both cross-validation (Table II, Table III) and hold-out tests (Table IV; Fig. 5, Fig. 6), shallow classifiers with strong regularization outperformed our compact CNN under severe class imbalance and limited positive examples. Kernel methods such as

RBF–SVM carve out optimal hyperplanes by maximizing class margins, which is especially effective when only 63 positive CTC patches are available. Ensemble trees like random forest leverage bagging and random feature selection to reduce variance, yielding consistently high discrimination (mean AUC ≈ 0.86 with $\sigma \approx 0.02$; Fig. 7) and tightly clustered accuracies (Fig. 8).

By contrast, our compact focal-loss CNN—despite smooth convergence on training and validation curves—struggled to generalize. Its mean AUC of 0.35 ± 0.12 and erratic fold-to-fold accuracy underscore that, in this data-scarce regime, hierarchical feature learning becomes a liability rather than an advantage. Synthetic geometric augmentations cannot fully replicate the inter-patient morphological variability and imaging artifacts inherent to bright-field data, and 64×64 patches may lack sufficient context for deep filters to extract robust, generalizable features.

k-Nearest Neighbors achieved the highest mean accuracy (85.4%) but suffered from greater AUC variance ($\sigma \approx 0.06$), reflecting sensitivity to local density fluctuations in pixel-space. Gaussian naïve Bayes, which models each pixel's intensity under a class-conditional Gaussian, struck a middle ground: it was more robust on small samples than k-NN and yet far simpler than deep nets, yielding moderate but stable discrimination ($AUC \approx 0.75$).

These findings have practical implications. When labeled patient data are limited, classical models not only provide superior performance but also greater interpretability—feature-importance scores from random forest or support vectors from SVM can directly highlight discriminative image regions. To harness the power of deep learning, future work should pursue hybrid pipelines that combine handcrafted texture descriptors (e.g. local binary patterns or gray-level co-occurrence matrices) with shallow classifiers, leverage semi-supervised pretraining on large unlabeled bright-field datasets, or adopt active-learning strategies to focus annotation efforts on the most ambiguous samples. Such approaches promise to marry the stability of classical methods with the representational richness of deep networks, paving the way for robust, label-free CTC detection under real-world constraints.

V. Conclusion

We have faithfully reproduced a label-free CTC detection pipeline and benchmarked five classical classifiers against a focal-loss CNN. Our findings demonstrate that shallow models—especially RBF-kernel SVM and random forest—achieve accuracies up to 85.4% and ROC-AUC near 0.87, with minimal variance, whereas a compact CNN yields only 68.3% accuracy and 0.35 AUC under identical conditions. This work highlights the critical role of data volume and model choice in label-free CTC detection and points toward feature engineering, semi-supervised learning, and active sampling as promising avenues to harness deep learning in data-scarce regimes.

References

[1] C. Alix-Panabières and K. Pantel, "Circulating tumor cells: liquid biopsy of cancer," *Clin. Chem.*, vol. 59, pp. 110–118, 2013.

[2] C. Alix-Panabières and K. Pantel, "Challenges in circulating tumour cell research," *Nat. Rev. Cancer*, vol. 14, pp. 623–631, 2014.

[3] T. B. Lannin, F. I. Thege, and B. J. Kirby, "Comparison and optimization of machine learning methods for automated classification of circulating tumor cells," *Cytom. Part A*, vol. 89, pp. 922–931, 2016.

[4] M. Zhao *et al.*, "An automated high-throughput counting method for screening circulating tumor cells in peripheral blood," *Anal. Chem.*, vol. 85, pp. 2465–2471, 2013.

[5] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst. Man Cybern.*, vol. 9, no. 1, pp. 62–66, 1979.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, 2016, pp. 770–778.

[7] S. Wang *et al.*, "Label-free detection of rare circulating tumor cells by image analysis and machine learning," *Sci. Rep.*, vol. 10, Art. 12226, 2020.

[8] C. Renier *et al.*, "Label-free isolation of prostate circulating tumor cells using Vortex microfluidic technology," *NPJ Precis. Oncol.*, vol. 1, no. 1, 2017.

[9] L. Ren *et al.*, "A high-throughput acoustic cell sorter," *Lab Chip*, vol. 15, pp. 3870–3879, 2015.

[10] N. Sun *et al.*, "A multiscale TiO_2 nanorod array for ultrasensitive capture of circulating tumor cells," *ACS Appl. Mater. Interfaces*, vol. 8, pp. 12638–12643, 2016.

[11] C. Aguilar-Avelar *et al.*, "High-throughput automated microscopy of circulating tumor cells," *Sci. Rep.*, vol. 9, p. 1, 2019.

[12] M. Zhao *et al.*, "An automated high-throughput counting method for screening circulating tumor cells in peripheral blood," *Anal. Chem.*, vol. 85, pp. 2465–2471, 2013.

[13] T. B. Lannin, F. I. Thege, and B. J. Kirby, "Comparison and optimization of machine learning methods for automated classification of circulating tumor cells," *Cytom. Part A*, vol. 89, pp. 922–931, 2016.

[14] C. L. Chen *et al.*, "Deep learning in label-free cell classification," *Sci. Rep.*, vol. 6, p. 1, 2016.

[15] M. Toratani *et al.*, "A convolutional neural network uses microscopic images to differentiate between mouse and human cell lines and their radioresistant clones," *Cancer Res.*, vol. 78, pp. 6703–6707, 2018.

[16] M. Xu *et al.*, "A deep convolutional neural network for classification of red blood cells in sickle cell anemia," *PLoS Comput. Biol.*, vol. 13, e1005762, 2017.

[17] A. Fitzsimmons and A. Martelli, "PyMuPDF—Python bindings for MuPDF," *GitHub repository*, 2020.

[18] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst. Man Cybern.*, vol. 9, no. 1, pp. 62–66, 1979.

[19] P. Soille, *Morphological Image Analysis: Principles and Applications*, 2nd ed., Springer, 2003.

[20] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed., Prentice Hall, 2008.

[21] S. Wang *et al.*, "Label-free detection of rare circulating tumor cells by image analysis and machine learning," *Sci. Rep.*, vol. 10, Art. 12226, 2020.

[22] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[23] Y. Shen *et al.*, "High-throughput screening of circulating tumor cells via Hough transform," *Cytom. Part A*, vol. 93A, pp. 386–395, 2018.

[24] S. Beucher and F. Meyer, "The morphological approach to segmentation: the watershed transformation," in *Mathematical Morphology in Image Processing*, E. Dougherty, Ed., Marcel Dekker, 1992, pp. 433–481.

[25] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[26] F. Chollet, "Keras," *GitHub repository*, 2015.

[27] K. He *et al.*, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, 2016, pp. 770–778.

[28] T.-Y. Lin *et al.*, "Focal loss for dense object detection," in *Proc. IEEE ICCV*, 2017, pp. 2980–2988.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[30] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst. Man Cybern.*, vol. 9, no. 1, pp. 62–66, 1979.

[31] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed., Prentice Hall, 2008.

[32] P. Soille, *Morphological Image Analysis: Principles and Applications*, 2nd ed., Springer, 2003.

[33] S. Wang *et al.*, "Label-free detection of rare circulating tumor cells by image analysis and machine learning," *Sci. Rep.*, vol. 10, Art. 12226, 2020.
[34] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of binary images," *Theory Comput.*, vol. 8, no. 1, pp. 415–428, 2004.
[35] S. Beucher and F. Meyer, "The morphological approach to segmentation: the watershed transformation," in *Mathematical Morphology in Image Processing*, E. Dougherty, Ed., Marcel Dekker, 1992, pp. 433–481.
[36] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
[37] F. Chollet, "Keras," *GitHub repository*, 2015.
[38] T.-Y. Lin *et al.*, "Focal loss for dense object detection," in *Proc. IEEE ICCV*, 2017, pp. 2980–2988.
[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.