A Multi-Architecture Ensemble with Test-Time Augmentation and Adaptive Batch Normalization for Image Deepfake Detection ⋆

Notebook for the ImageCLEF Lab at CLEF 2026

Ravi Solanki

Poornima University, Jaipur, India

Abstract

This paper describes our submission to the ImageCLEF 2026 Deepfake Detection task (image subtask). We treat the problem as binary classification (real vs. deepfake) using an ensemble of three diverse architectures: a Swin Transformer, an EfficientNet-B3 convolutional network, and a fine-tuned CLIP vision encoder. To improve robustness on the test distribution, we apply test-time augmentation (TTA) and adaptive batch normalization (AdaBN), followed by per-model temperature scaling and an Optuna-based search over ensemble weights and the decision threshold. Our final submission achieved a score of 0.5772. We observed strong accuracy on deepfake samples but notably weaker accuracy on real images, which we attribute to a distribution shift between our training data and the organizers' real-image set. We discuss this limitation and directions for future improvement.

Keywords

deepfake detection, image classification, ensemble learning, test-time augmentation, domain adaptation [1]

1. Introduction

Recent advances in generative models have made it possible to synthesize and manipulate images with a high degree of realism. While these technologies have many legitimate uses, they also enable the creation of deepfakes — manipulated or fully synthetic images that can be exploited for misinformation, fraud, and identity abuse. As synthetic media becomes increasingly difficult to distinguish from authentic content by eye, automatic deepfake detection has become an important research problem.

The ImageCLEF 2026 Deepfake Detection task [1,2] addresses this problem. In the image subtask, the goal is to classify each image as either real or deepfake, framed as a binary classification problem. The evaluation set consists of 24,404 images, and submissions are scored across several subsets of real and deepfake data, including deepfakes contributed by other participating teams.

In this paper we describe our submission to the image subtask. Our approach is based on an ensemble of three architecturally diverse deep learning models: a Swin Transformer, an EfficientNet-B3 convolutional neural network, and a fine-tuned CLIP vision encoder. We deliberately chose different architectures so that their errors would be less correlated, which generally benefits ensembling. Each model was trained independently and their predictions were then combined. To improve robustness on the test set, we additionally applied test-time augmentation (TTA), adaptive batch normalization (AdaBN), per-model temperature scaling, and an Optuna-based search over the ensemble weights and decision threshold.

⋆ *CLEF 2026 Working Notes, 21 — 24 September 2026, Jena, Germany*

[1] ravii222006@gmail.com (Ravi Solanki)

0009-0008-0031-9851 (Ravi Solanki)

Our final submission achieved a score of 0.5772. We found that the system was highly accurate at identifying deepfakes but noticeably weaker on real images. We attribute this imbalance to a distribution shift between our training data and the real images in the evaluation set, which we analyze further in Section 5.

The remainder of this paper is organized as follows. Section 2 briefly reviews related work. Section 3 describes our models and methodology. Section 4 presents the experimental setup and results. Section 5 discusses our findings and limitations, and Section 6 concludes.

2. Related Work

Deepfake detection has been widely studied as a binary classification problem. Early approaches relied on convolutional neural networks trained to detect visual artifacts left behind by generative models [3]. Other methods exploit frequency-domain cues, since synthetic images often contain spectral artifacts that are uncommon in natural photographs [4]. More recently, Vision Transformers and large pre-trained vision encoders have been applied to the task, as their global attention can capture subtle inconsistencies across an image [5]. A recurring challenge in this area is generalization: detectors trained on one set of generators or data sources frequently degrade when evaluated on unseen ones [6]. This limitation is directly relevant to the behavior we observe in our own system.

Our submission does not propose a new detection method; instead, it combines several established components. We build on three architecturally diverse models from the literature — the Swin Transformer [7], EfficientNet [8], and CLIP [9]. For inference-time robustness we use test-time augmentation, in which predictions over several augmented views of an input are averaged. We further apply Adaptive Batch Normalization (AdaBN) [10], which adapts batch-normalization statistics to the target data, and temperature scaling [11], a simple post-hoc calibration technique. Finally, we tune our ensemble weights and decision threshold using Optuna [12], a hyperparameter-optimization framework.

3. Methodology

3.1. Overview

Our system is an ensemble of three independently trained image classifiers. We first describe the three architectures and how they were trained (Section 3.2). We then describe the techniques applied at inference time to improve robustness and to combine the models: test-time augmentation (Section 3.4), adaptive batch normalization (Section 3.5), and probability calibration together with ensemble optimization (Section 3.6). We submitted two runs to the task: a first run based on a simple trimmed-mean ensemble, and a second run using the full pipeline described below.

3.2. Models and Training

We trained three architecturally diverse models so that their predictions would be complementary and their errors less correlated: a Swin Transformer (swin_tiny_patch4_window7_224), an EfficientNet-B3 convolutional neural network, and a CLIP ViT-B/32 vision encoder fully fine-tuned with a custom classification head.

Each model was trained for three epochs using the AdamW optimizer (weight decay 0.01) with a linear warmup followed by cosine learning-rate decay, mixed-precision (AMP) training, and gradient clipping (maximum norm 1.0). We used a batch size of 32 with two gradient-accumulation

steps, giving an effective batch size of 64. The Swin Transformer was trained with a learning rate of 5×10^{-5} using a focal loss combined with label smoothing. For EfficientNet we selected the B3 variant after the larger B4 variant repeatedly exhausted GPU memory and failed to train stably in our setup. For CLIP, fully fine-tuning the vision encoder together with the classification head outperformed linear probing (a frozen encoder) by approximately 4.4% F1, so we adopted full fine-tuning. EfficientNet-B3 was trained with a class-weighted cross-entropy loss and label smoothing, while CLIP used the AdamW optimizer with discriminative learning rates (1×10^{-5} for the pre-trained vision encoder and 5×10^{-4} for the new classification head) and a class-weighted cross-entropy loss. The CLIP classification head maps the 512-dimensional encoder output through a LayerNorm, two hidden linear layers (512⟶256⟶64) with GELU activations and dropout, and a final linear layer to two output classes.

On the held-out validation set, the three models achieved strong performance: the Swin Transformer reached an F1 of 0.9925 (98.8% accuracy, 0.9988 AUC), EfficientNet-B3 an F1 of 0.9920 (98.7% accuracy, 0.9991 AUC), and CLIP an F1 of 0.9826 (97.3% accuracy, 0.9970 AUC)

3.3. Dataset

Our training data combined several publicly available real-face and deepfake datasets, augmented with additional modern AI-generated (diffusion-based) face images to improve coverage of recent generators. Real images were drawn from the FFHQ dataset [13] and the real portion of the 140k Real and Fake Faces dataset [14]. Deepfake and synthetic images were drawn from DFDC face crops [15], Stable-Diffusion-generated faces [16], and the synthetic portion of the 140k Real and Fake Faces dataset [14].

The combined dataset contained approximately 1.04 million images, split in a stratified manner by both class and source into roughly 726k training, 156k validation, and 156k test images using a fixed random seed. The data was class-imbalanced (approximately 30% real and 70% deepfake), which we addressed during training using a weighted random sampler. Table 1 summarises the composition of the dataset across the three splits.

Table 1: Composition of the combined dataset across the three stratified splits.

Split	Real (~30%)	Deepfake (~70%)	Total
Training	~218k	~508k	~726k
Validation	~47k	~109k	~156k
Test	~47k	~109k	~156k
Total	~312k	~726k	~1,038k

All images were resized to 224×224. We applied ImageNet normalization for the convolutional and transformer models and CLIP's native normalization for the CLIP-based model. Training augmentations included horizontal flipping, colour jitter and random brightness/contrast, small affine transformations (scaling, translation, and rotation), JPEG compression, Gaussian blur and noise, and coarse dropout (cutout).

3.4. Test-Time Augmentation

Test-time augmentation (TTA) improves robustness by averaging a model's predictions over several augmented views of the same input. At inference time we used a multi-scale TTA scheme

in which each test image was passed through each model under six different augmentations, and the resulting logits were averaged.

The six augmentations combined three scale settings, each in both its original and horizontally flipped form: (i) a direct resize to 224×224; (ii) a resize to 256×256 followed by a 224×224 center crop; and (iii) a resize to 248×248 followed by a 224×224 center crop, giving a slight zoom-in. For each model, the logits from these six views were averaged to produce a single, more stable prediction per image.

We applied this procedure independently to all three models, yielding a TTA-averaged logit vector for each model on every test image. The resulting per-model averaged logits were then used in the ensemble stage (Section 3.6).

3.5. Adaptive Batch Normalization

Adaptive Batch Normalization (AdaBN) is a simple, training-free domain-adaptation technique. Batch-normalization layers store running estimates of the mean and variance of their inputs, computed on the training data; when the test data follows a different distribution, these statistics can be mismatched. AdaBN addresses this by recomputing the batch-normalization statistics on the (unlabeled) test data, without modifying any of the learned weights.

Concretely, we reset the running statistics of all batch-normalization layers, set those layers to update their statistics, and passed the test images through the model in a series of forward passes with all weights frozen and no gradient computation. The batch-normalization layers thereby adapted their mean and variance to the test distribution, after which we performed inference normally.

This procedure only affects models that contain batch-normalization layers. Of our three models, only EfficientNet-B3 uses batch normalization (it contains 78 such layers); the Swin Transformer and the CLIP encoder use layer normalization, whose statistics are computed per sample and are therefore unaffected by AdaBN. Accordingly, AdaBN left the Swin and CLIP predictions unchanged.

For EfficientNet-B3, however, the effect was substantial: adapting the batch-normalization statistics to the test set changed roughly 8,900 predictions and shifted the fraction of images predicted as deepfake from about 58.5% to about 27.3%. This large shift indicates a clear mismatch between the batch-normalization statistics learned on our training data and those of the evaluation set. Because the test labels are not available to us, we cannot isolate the effect of this adaptation on final accuracy; we return to its likely implications in Section 5.

3.6. Calibration and Ensemble Optimization

Neural networks are often over-confident, producing poorly calibrated probabilities. Before combining the models, we applied temperature scaling to each model independently: we searched for the scalar temperature (in the range 0.5–3.0) that minimized the log loss of the model's predictions on the validation set, and divided that model's logits by this temperature. Temperature scaling does not change the predicted class but softens the probability distribution, improving calibration and log loss without affecting F1.

For each model we then had two sets of test predictions — the multi-scale TTA logits (Section 3.4) and the AdaBN-adapted logits (Section 3.5) — which we combined by averaging the two corresponding calibrated probabilities with equal weight. For the Swin Transformer and CLIP, which are unaffected by AdaBN, this is equivalent to using the TTA predictions alone; for EfficientNet-B3 it blends the two views.

We submitted two runs. The first used a simple trimmed-mean ensemble: for each image we discarded the highest of the three models' probabilities, averaged the remaining two, and thresholded at 0.5. The second used an automatically tuned ensemble. Using the calibrated validation logits, we ran a search with Optuna (2000 trials, TPE sampler) over the combination

strategy (weighted average, trimmed mean, rank averaging, or a Swin + EfficientNet-only combination), the per-model weights, and the decision threshold (searched within 0.30–0.70). The configuration that maximized validation F1 was then applied to the combined test probabilities to produce the final binary predictions.

4. Experimental Setup and Results

4.1. Experimental Setup

All experiments were carried out on Kaggle Notebooks using NVIDIA T4 GPUs. Our models were implemented in PyTorch using the timm library, with albumentations for data augmentation and Optuna for ensemble optimization. Each model was trained for three epochs; training took several hours on the T4 hardware (approximately five hours for the Swin Transformer on a dual-T4 configuration). The inference-time stages were comparatively cheap: multi-scale test-time augmentation over all three models and the full 24,404-image test set took about 21 minutes, adaptive batch normalization about 11 minutes, and the temperature-scaling and Optuna-based ensemble optimization (run on CPU) about 15 minutes.

We evaluated our models during development on the held-out validation set using accuracy, F1, and AUC. The official task metric is computed by the organizers as a weighted combination of binary classification accuracy over several subsets of the evaluation data (organizer-generated deepfakes, organizer real images, a hidden curated real subset, and deepfakes contributed by other participating teams), producing a single final score.

4.2. Validation Results

On the held-out validation set, all three models performed strongly, with the convolutional and transformer models slightly ahead of the CLIP-based model:

Table 2: Validation performance of the three models.

Model	Accuracy	F1	AUC
Swin Transformer	98.82%	0.9925	0.9988
EfficientNet-B3	98.74%	0.9920	0.9991
CLIP (ViT-B/32)	97.29%	0.9826	0.9970

The tuned ensemble (Section 3.6) reached an F1 of approximately 0.995 on the validation set, slightly above any individual model. As we discuss in Section 5, however, this high validation performance did not transfer to the official evaluation set.

4.3. Official Test Results

We submitted two runs to the task: the first based on the trimmed-mean ensemble and the second on the temperature-scaled, Optuna-tuned ensemble combining the TTA and AdaBN predictions. The organizers reported a single official final score of 0.5772. We note that the organizers reported only a single official score; the per-run scores were not made available to us, so we cannot determine which of our two runs this score corresponds to, nor compare the two runs on the official evaluation set. The per-subset breakdown is shown in Table 3.

Table 3: Official evaluation scores by subset.

Evaluation subset	Weight	Accuracy
Baseline Deepfakes	0.1	0.5710
Organizers Real Data	0.1	0.3426
Organizers Real GT Data	0.4	0.3204
Participant Deepfake Data	0.4	0.9009
Participant Deepfake Data (Weighted)	-	0.8942
Final Score		0.5772

The results show a clear and consistent pattern: our system was highly accurate on deepfake images (around 0.90 on participant-contributed deepfakes) but performed poorly on real images (around 0.32–0.34 on the organizers' real subsets). Since the real-image subsets carry substantial weight in the final score, this weakness dominated the overall result. The large gap between our validation F1 (≈0.99) and the official score (0.5772) points to a significant distribution shift between our training/validation data and the evaluation set, which we analyze in Section 5.

5. Discussion

The most striking aspect of our results is the contrast between our system's accuracy on deepfakes (around 0.90) and on real images (around 0.32–0.34). This indicates that our system was strongly biased towards predicting "deepfake": it correctly identified most fakes, but misclassified a large fraction of real images as fake. This bias was already visible in the individual models' raw predictions on the test set, where each model labeled a high proportion of images as deepfake — approximately 65% for the Swin Transformer, 59% for EfficientNet-B3, and 78% for CLIP.

This behavior is consistent with a distribution shift between our training data and the evaluation set, particularly for real images. Our models reached very high scores on our own held-out validation set (F1 ≈ 0.99), which was drawn from the same distribution as our training data; the official score of 0.5772 is far lower, showing that this validation performance did not reflect performance on the evaluation set. Our real training images came largely from a small number of face datasets (for example, FFHQ-style portraits), which may differ substantially from the organizers' curated real images. A detector that learns features specific to our training distribution can then flag unfamiliar real images as fake.

We found further evidence of this mismatch when applying adaptive batch normalization. Recomputing EfficientNet-B3's batch-normalization statistics on the test data changed roughly 8,900 of its predictions and reduced its predicted deepfake rate from about 58.5% to about 27.3%. Such a large change implies that the feature statistics of the evaluation set differed considerably from those of our training data. Whether this adaptation improved the final score we cannot say with certainty, since the test labels are not available to us and AdaBN was applied within a larger ensemble; isolating its effect would require ground-truth labels.

More generally, the absence of test labels limited our ability to diagnose the problem during the competition. We tuned our decision threshold and ensemble weights to maximize F1 on our

validation set, but because that set was not representative of the evaluation data, the chosen threshold and ensemble configuration may have been poorly suited to the test distribution. In particular, a threshold tuned on a fake-heavy validation set is likely too aggressive in predicting "deepfake" on the evaluation set.

In hindsight, the main limitation of our approach was not the model architectures or the ensemble techniques, but the training and validation data. Future work should focus on: (i) using more diverse and representative real images, so that the detector does not over-fit to a narrow notion of "real"; (ii) constructing a validation set that better reflects the target distribution, so that threshold and ensemble choices transfer to the test data; and (iii) performing calibration and threshold selection on data closer to the evaluation distribution. Techniques aimed explicitly at generalization across unseen generators and data sources may also help close the gap we observed.

6. Conclusion

We described our submission to the ImageCLEF 2026 Deepfake Detection task (image subtask). Our approach combined three architecturally diverse models — a Swin Transformer, an EfficientNet-B3, and a fully fine-tuned CLIP encoder — with several inference-time techniques: multi-scale test-time augmentation, adaptive batch normalization, per-model temperature scaling, and an Optuna-based ensemble search. While our models achieved high accuracy on our held-out validation set (F1 $\approx$ 0.99), our official score was 0.5772, with strong performance on deepfake images but weak performance on real images.

Our analysis suggests that this gap was caused primarily by a distribution shift between our training data and the evaluation set, which led our models to over-predict the "deepfake" class. The main lesson we draw is that, for deepfake detection, the diversity and representativeness of the training and validation data matter at least as much as the choice of model architecture or ensemble strategy. In future work we plan to address this through more diverse and representative real-image data, a validation set that better reflects the target distribution, and calibration performed closer to the evaluation distribution.

Declaration on Generative AI

During the preparation of this work, the author used multiple generative AI assistants — ChatGPT (OpenAI), Gemini (Google), and Claude (Anthropic). During the research and development phase, these tools were used to discuss and develop the methodology and to assist with writing and debugging code. During the preparation of the manuscript, they were used to draft content (including the abstract, introduction, related work, methodology, results, and discussion sections), generate a literature review, paraphrase and reword text, improve writing style, check grammar and spelling, format citations, and assist with formatting. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] B. Ionescu, H. Müller, D.-C. Stanciu, A. Radzhabov, A. García Seco de Herrera, A.-G. Andrei, A. Băicoianu, A. Neacşu, A. Storås, A. Ben Abacha, B. Bracke, L. Reinartz, B. Lecouteux, C. M. Friedrich, C. S. Schmidt, C.-N. Florea, D. Fabre, D. Schwab, D. Dimitrov, E. Esperança-Rodier,

G. Constantin, H. Damm, H. Schäfer, I. Koychev, J. Mothe, L.-D. Ștefan, M. J. Hjuler, M. Kurt, M. Yetisgen, M. A. Riegler, M. Dogariu, M. Ivanovici, M. S. Hee, M. El Sakka, M. Ahsan, O. Pelka, P. Halvorsen, P. Nakov, R. Brüngel, S. Ahmad, S. A. Hicks, S. Gautam, T. M. G. Pakull, B. Eryılmaz, V. Thambawita, V. Kovalev, W.-W. Yim, Y. Prokopchuk, Z. Xie, Overview of ImageCLEF 2026: Multimodal Challenges in Medicine, Science, Agritech, and Security, in: Advances in Information Retrieval, Proc. of the 48th European Conference on Information Retrieval (ECIR 2026), Lecture Notes in Computer Science, vol. 16486, Springer, Cham, 2026, pp. 336–344. doi:10.1007/978-3-032-21321-1_44.

[2] D.-C. Stanciu, A. Radu, R.-G. Bolborici, M. Negru, A.-F. Ene, V.-M. Vasilescu, A.-A. Nicolae, B. Ionescu, L.-D. Ștefan, M.-G. Constantin, M. Dogariu, A.-G. Andrei, Overview of ImageCLEF 2026 Deepfake Task: Multimodal Detection and Generation of Deepfakes, in: CLEF 2026 Working Notes, CEUR Workshop Proceedings, CEUR-WS.org, Jena, Germany, 2026.

[3] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, M. Nießner, FaceForensics++: Learning to detect manipulated facial images, in: Proc. IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 1–11. doi:10.1109/ICCV.2019.00009.

[4] J. Frank, T. Eisenhofer, L. Schönherr, A. Fischer, D. Kolossa, T. Holz, Leveraging frequency analysis for deep fake image recognition, in: Proceedings of the 37th International Conference on Machine Learning (ICML), PMLR, vol. 119, 2020, pp. 3247–3258.

[5] D. Wodajo, S. Atnafu, Deepfake video detection using convolutional vision transformer, arXiv preprint arXiv:2102.11126 (2021). doi:10.48550/arXiv.2102.11126.

[6] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, A. A. Efros, CNN-generated images are surprisingly easy to spot... for now, in: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 8692–8701. doi:10.1109/CVPR42600.2020.00872.

[7] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, Swin transformer: Hierarchical vision transformer using shifted windows, in: Proc. IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 10012–10022.

[8] M. Tan, Q. V. Le, EfficientNet: Rethinking model scaling for convolutional neural networks, in: Proceedings of the 36th International Conference on Machine Learning (ICML), PMLR, vol. 97, 2019, pp. 6105–6114.

[9] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, I. Sutskever, Learning transferable visual models from natural language supervision, in: Proceedings of the 38th International Conference on Machine Learning (ICML), PMLR, vol. 139, 2021, pp. 8748–8763.

[10] Y. Li, N. Wang, J. Shi, X. Hou, J. Liu, Adaptive batch normalization for practical domain adaptation, Pattern Recognition 80 (2018) 109–117.

[11] C. Guo, G. Pleiss, Y. Sun, K. Q. Weinberger, On calibration of modern neural networks, in: Proceedings of the 34th International Conference on Machine Learning (ICML), PMLR, vol. 70, 2017, pp. 1321–1330.

[12] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), ACM, 2019, pp. 2623–2631.

[13] T. Karras, S. Laine, T. Aila, A style-based generator architecture for generative adversarial networks, in: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4401–4410. doi:10.1109/CVPR.2019.00453.

[14] xhlulu, 140k Real and Fake Faces, Kaggle, 2019. URL: https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces.

[15] B. Dolhansky, J. Bitton, B. Pflaum, J. Lu, R. Howes, M. Wang, C. C. Ferrer, The DeepFake Detection Challenge (DFDC) dataset, arXiv preprint arXiv:2006.07397 (2020). doi:10.48550/arXiv.2006.07397.

[16] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer, High-resolution image synthesis with latent diffusion models, in: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 10684–10695. doi:10.1109/CVPR52688.2022.01042.