Detecting GAN Training Data Usage in Synthetic Medical Images with Transfer Learning Approaches

Notebook for the ImageCLEF Lab at CLEF 2026

Bharathi Subrahmanian[1,*,†] , Harish M[1,†]

[1] *Shiv Nadar University, Kalavakkam, Tamil Nadu, India*

Abstract

Synthetic medical images generated using Generative Adversarial Networks (GANs) are increasingly being explored to address the shortage of publicly available medical datasets. However, an important concern is whether these generated images may still preserve hidden traces of the real data used during GAN training. In this work, we participated in the ImageCLEFmedical 2026 GANs challenge, which focuses on detecting whether a real CT image was used during GAN training. We investigated transfer learning approaches using pre-trained deep learning models including ResNet34, EfficientNet-B3, and DenseNet121. In addition, an ensemble strategy based on soft voting was explored for the final submission. Our observations show that detecting GAN training fingerprints remains a difficult task, particularly when models are evaluated on unseen data distributions. The study further highlights the challenges involved in developing robust and generalized approaches for privacy-related analysis in synthetic medical imaging.

Keywords

GAN, Medical Imaging, Transfer Learning, ResNet34, Ensemble Learning, Synthetic Images, Privacy Analysis, Fingerprint Detection

1. Introduction

Deep learning has become widely used in medical image analysis for applications such as disease diagnosis, image classification, and computer-aided detection. However, training effective deep learning models often requires large amounts of medical imaging data, which are difficult to obtain because of privacy restrictions and limited public availability. To address this challenge, Generative Adversarial Networks (GANs) are increasingly being explored for generating synthetic medical images that resemble real patient data.

Although synthetic medical images can help reduce data scarcity, they also introduce important privacy concerns. Generative models may unintentionally preserve hidden traces or "fingerprints" from the real images used during training. If such information can be identified, synthetic images may still expose sensitive patient-related information, limiting their effectiveness as a privacy-preserving solution.

The ImageCLEFmedical 2026 GANs challenge focuses on studying privacy risks associated with synthetic medical imaging. In particular, Subtask 1 aims to determine whether a specific real CT image was used during the training process of a GAN model. The task is formulated as a binary classification problem in which each image must be classified as either "used" or "not used" during GAN training.

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

* Corresponding author.

† These authors contributed equally.

bharathi25120014@snuchennai.edu.in (Bharathi Subrahmanian);

harish25120013@snuchennai.edu.in (Harish. M)

The objective of this work is to investigate whether deep learning approaches can identify subtle patterns related to GAN training fingerprints and evaluate their ability to generalize across unseen data distributions

2. Related Work

Generative Adversarial Networks, first introduced by Goodfellow et al. [11], have become one of the most widely used approaches for generating synthetic data, especially in fields where real data is scarce or difficult to access. In medical imaging, this has been particularly useful. For example, GAN-generated CT images have been used to improve model performance when access to real patient data is limited [3]. Similar approaches have also been explored for chest X-rays and lung nodule datasets, where synthetic images were used to improve training performance on limited datasets [4].

While GANs help address the problem of limited medical data, they also introduce important privacy concerns. During training, generative models may unintentionally memorize certain characteristics of the real images used in the dataset. As a result, the generated images may still preserve hidden traces or fingerprints from the original patient data. This concern is closely related to membership inference attacks, where the objective is to determine whether a particular image was part of a model's training dataset [5, 6]. The ImageCLEFmedical GANs challenge series focuses on studying such privacy-related risks in synthetic medical imaging [1, 2].

The 2026 edition of the ImageCLEFmedical GANs challenge specifically focuses on fingerprint detection in GAN-generated medical images [1]. The task evaluates whether a real CT image was used during the training process of a GAN model, making it a challenging privacy-analysis problem in medical imaging. Previous challenge results have highlighted the difficulty of reliably identifying GAN training fingerprints, especially when models are evaluated on unseen test distributions [7].

Transfer learning using ImageNet-pretrained convolutional neural networks such as ResNet [8], DenseNet [9], and EfficientNet [10] has shown strong performance in medical image classification tasks, particularly when training data is limited. Motivated by these findings, our work investigates transfer learning approaches for detecting GAN training data usage in synthetic medical images. In addition, we explore whether combining multiple models through ensemble learning can improve the robustness of fingerprint detection.

3. Dataset

The dataset used in this study was provided by the organizers of the ImageCLEFmedical 2026 GANs challenge for Subtask 1: Detect Training Data Usage [1]. It consists of grayscale CT lung images released in two phases — a development phase and a final test phase — following the competition schedule.

The development dataset contains three folders:

- generated: A collection of 10,000 synthetic images produced by a Generative Adversarial Network (GAN).
- real_used: A folder containing 3,200 real CT images that were used to train the GAN. These images are assigned label 1 in our experiments.
- real_not_used: A folder containing 3,200 real CT images that were not part of the GAN training process. These images are assigned label 0.

In total, the development set provides 6,400 labeled real images with a perfectly balanced class distribution, along with 10,000 synthetic images generated by the same GAN.

The final test dataset contains one folder:

• real_unknown: A folder containing 2,140 real CT images with unknown labels. This folder includes a mix of images that were and were not used to train the GAN. The objective is to predict a binary label for each image: 1 for used and 0 for not used.

In our approach, we did not use the generated folder during training. This decision was made to keep the approach simple and avoid potential bias introduced by synthetic images during supervised training. Our models were trained only on the labeled real images from real_used and real_not_used, treating the task as a straightforward binary classification problem. All images were resized to 224×224 pixels and converted to 3-channel images to match the input requirements of ImageNet-pretrained models. Standard ImageNet normalization was applied with mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225].

4. Methodology

Our approach formulates the task as a supervised binary classification problem using transfer learning. Since we have labeled examples of real images that were and were not used to train the GAN, we treat this as a supervised binary classification problem. Rather than building complex pipelines or designing new architectures from scratch, we fine-tune well-established pretrained models on the labeled development data and observe how well they can identify GAN training fingerprints.

4.1 Preprocessing

All CT images in the dataset were grayscale and were converted to 3-channel format to match the input requirements of ImageNet-pretrained models. Each image was resized to 224×224 pixels and normalized using standard ImageNet statistics with mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225].Since the pretrained CNN models were originally trained on ImageNet, the standard ImageNet mean and standard deviation were used for input normalization to maintain compatibility with the pretrained weights. However, CT images have a substantially different intensity distribution from natural images. Therefore, exploring domain-specific normalization strategies may further improve model generalization and is considered an important direction for future work.

The models were trained only on the labeled real images from the real_used and real_not_used folders. The generated images provided in the dataset were not used during training. Images from the real_used folder were assigned the label 1, while images from the real_not_used folder were assigned label 0.

4.2 Transfer Learning with ResNet34

Our primary model is ResNet34, pretrained on ImageNet. We adopt a fine-tuning strategy where most layers are frozen and only the final convolutional blocks are updated. All layers are initially frozen, then layer3 and layer4 are unfrozen for fine-tuning. The final fully connected layer is replaced with a linear layer mapping to 2 output classes. Training used the Adam optimizer (lr=0.0003), CrossEntropyLoss with class weights [2.0, 1.0], StepLR scheduler (step size=4, gamma=0.5), batch size 64, and 10 epochs on a Tesla T4 GPU.

4.2.1. ResNet34 Fine-Tuning Pipeline

Our supervised baseline model was implemented using the PyTorch framework on Google Colab with an NVIDIA Tesla T4 GPU. The workflow is detailed below.

1. **Data Preparation and Labeling:** File paths for all images in the real_used/ and real_not_used/ directories were loaded. Each image was assigned a binary label: 1 for images in real_used/ and 0 for images in real_not_used/. The full labeled dataset of 6,400 images was used for training.
2. **Preprocessing:** All images were resized to 224x224 pixels. Grayscale CT images were converted to 3-channel RGB format by replicating the single channel to match the input

expected by ImageNet-pretrained models. ImageNet normalization (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]) was then applied.

3. **Model Architecture and Transfer Learning:** The architecture is built on a ResNet34 model pretrained on ImageNet. All convolutional layers were initially frozen. Only layer3 and layer4 were unfrozen for fine-tuning. The final fully connected layer was replaced with a 2-class linear output layer.
4. **Training:**The model was optimized using Adam (lr=0.0003) with CrossEntropyLoss and class weights [2.0, 1.0]. Although the dataset is balanced, class weights were applied to place greater emphasis on correctly identifying used images due to the privacy-sensitive nature of the task. A StepLR scheduler (step size=4, gamma=0.5) decayed the learning rate during training. Training ran for 10 epochs with batch size 64 on a Tesla T4 GPU.
5. **Inference and Submission:** Once trained, the model was applied to the 2,140 final test images. Predictions were saved to run.csv (format: filename, label) and compressed as Submission.zip for submission to the competition platform.

4.3 Ensemble of Three Models

For the final submission, we extended the single-model approach to an ensemble of three complementary architectures: ResNet34 [8], EfficientNet-B3 [10], and DenseNet121 [9]. Each model was independently fine-tuned using the same training protocol. Final predictions were obtained by soft voting, averaging the softmax probability outputs of all three models: avg_prob = ($prob_1$ + $prob_2$ + $prob_3$) / 3. The final label was assigned based on the highest averaged probability score. Soft voting was used because it combines prediction confidence from all three models, resulting in more robust final predictions.

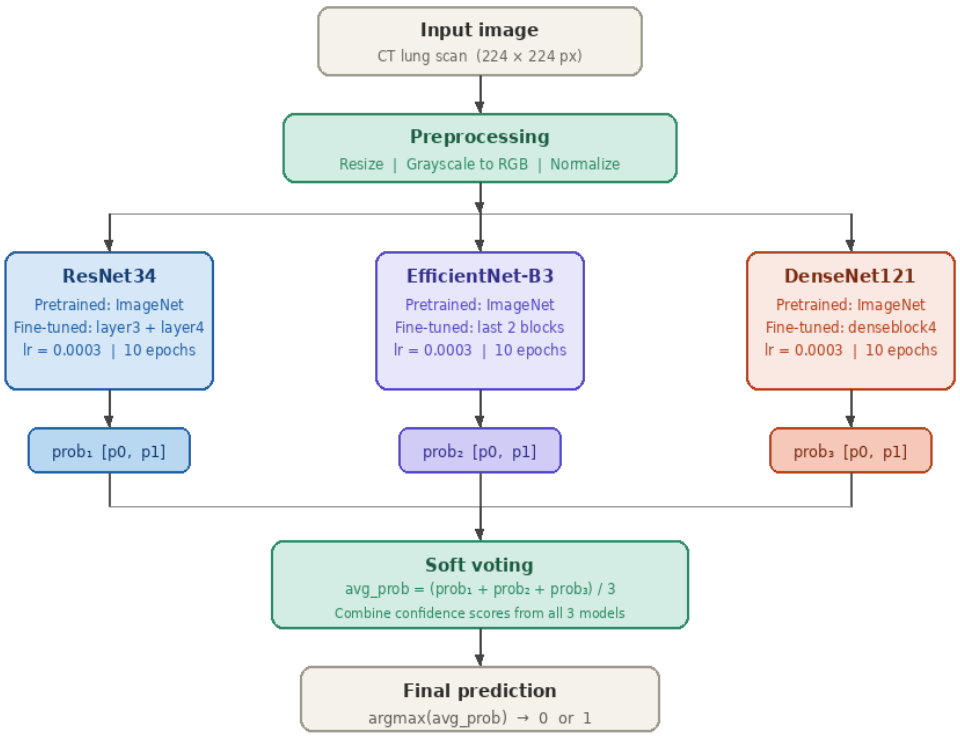

Figure1: Proposed ensemble architecture using soft voting.

4.3.1. Step-by-Step Ensemble Workflow

The ensemble pipeline combines three independently trained models. The complete workflow is detailed below.

1. **Data Preparation:** The same 6,400 labeled images used for ResNet34 were reused to independently train EfficientNet-B3 and DenseNet121. Identical preprocessing was applied: resize to 224x224, grayscale to 3-channel conversion, and ImageNet normalization.
2. **Independent Model Training:** Each of the three models was independently fine-tuned using the same optimizer settings (Adam, lr=0.0003), loss function (CrossEntropyLoss, weights [2.0, 1.0]), and duration (10 epochs). Model-specific layers were selectively unfrozen: layer3+layer4 for ResNet34, last 2 feature blocks for EfficientNet-B3, and denseblock4 for DenseNet121.
3. **Saving Model Checkpoints:** After training, each model's weights were saved independently as resnet34.pth, efficientnet_b3.pth, and densenet121.pth. All three models were loaded and set to evaluation mode (model.eval()) before inference.
4. **Probability Extraction:** For each test image, the input was passed through all three models within a torch.no_grad() context. The raw logits from each model were converted to class probabilities using softmax, yielding three probability vectors: $prob_1$ (ResNet34), $prob_2$ (EfficientNet-B3), and $prob_3$ (DenseNet121).
5. **Soft Voting:** The three probability vectors were averaged element-wise: avg_prob = ($prob_1$ + $prob_2$ + $prob_3$) / 3. The final predicted label was obtained by taking argmax of avg_prob. This soft voting strategy incorporates the confidence of each model rather than just the binary decision.
6. **Output Generation:** Predictions for all 2,140 final test images were written to run.csv in the format filename,label. The file was compressed as Submission_final.zip and submitted to the competition platform.

4.4 Training Configuration Summary

For EfficientNet-B3, the last two feature blocks were unfrozen and a custom classifier with Dropout (p=0.3) was added. For DenseNet121, the final dense block (denseblock4) was unfrozen. All three models used identical optimizer settings. The Tesla T4 GPU completed training of all three models in approximately 15-20 minutes total.

4.5 Validation Strategy

In this work, the full development dataset of 6,400 labeled images was used for training without a dedicated held-out validation split. This decision was made to maximize the amount of labeled data available for learning, since the development dataset was relatively small for training deep neuralnetworks.

However, the absence of a validation split made it difficult to monitor overfitting during training and identify the most generalizable model checkpoint. In hindsight, reserving a portion of the development data — for example, using an 80/20 train/validation split — together with early stopping based on validation loss, would have provided a more principled training strategy.

Such an approach could have helped reduce overfitting, especially because two of the models reached near-perfect training accuracy during training. This limitation provided useful insight into the importance of validation-based model selection and regularization for improving generalization performance on unseen data.

5. Results and Analysis

5.1 Evaluation Metrics

The competition uses Cohen's Kappa as the primary evaluation metric, along with several supporting metrics including accuracy, precision, recall, and F1 score. These metrics together give a complete picture of how well the model is performing on the binary classification task.

Cohen's Kappa is particularly useful here because it accounts for the possibility of agreement happening by chance, making it more reliable than simple accuracy especially when class distributions may be uneven. The formal definitions of all metrics used are as follows:

$$Kappa = \frac{P_o - P_e}{1 - P_e} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \quad (4)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

5.2 Development Phase Results

During the development phase, we submitted predictions generated by our single ResNet34 model. The official results from the competition platform are shown in Table 1.

Table 1

Official Development Phase Results.

Model Description	Run ID	Submission File	Acc.	Precision	Recall	F1-Score	Cohen's Kappa
ResNet34	229	Submission.zip	0.9561	0.9212	0.9975	0.9578	0.9122

The model performed strongly during the development phase, achieving a Cohen's Kappa score of 0.9122 and an accuracy of 0.9561. The recall score of 0.9975 indicates that the model was highly effective at identifying images that were used during GAN training. These results initially suggested that the proposed approach was effective for the development dataset.

5.3 Final Phase Results

For the final phase, we submitted two runs. Run 843 used the single ResNet34 model, while Run 857 used the ensemble of three models. The official results are shown in Table 2.

Table 2

Official Final Phase Results for All Submitted Runs.

Model Description	Run ID	Submission File	Acc.	Precision	Recall	F1-Score	Cohen's Kappa
ResNet34 (Single Model)	843	Submission.zip	0.4808	0.4786	0.4290	0.4524	-0.0383
Ensemble (ResNet34+	857	Submission_final.zip	0.4612	0.4573	0.4150	0.4351	-0.0776

Both submissions performed poorly on the final test set. Run 843 achieved a Cohen's Kappa score of -0.0383, while Run 857 achieved -0.0776. These results represent a significant drop compared to the development phase performance and indicate that the models struggled to generalize effectively to the unseen evaluation data.

5.4 Analysis

A large performance gap was observed between the development phase (Kappa = 0.9122) and the final phase (Kappa = -0.0383 and -0.0776). Several factors may have contributed to this behavior. One possible reason is domain shift. The final test images appear to follow a different distribution compared to the development dataset. As a result, the models learned patterns that worked well on the development data but did not generalize effectively to the unseen final test images. This is a common challenge in machine learning, where models that achieve high performance on one dataset may fail to generalize when evaluated on data from a different distribution. Another contributing factor may be overfitting. During training, both ResNet34 and DenseNet121 reached a training accuracy of 1.0000, suggesting that the models may have learned characteristics specific to the development dataset rather than generalized fingerprint features. In hindsight, introducing a validation split and early stopping earlier in the training process could have helped detect overfitting and preserve a more generalizable checkpoint instead of relying on the final epoch model. This may have limited the ability of the models to adapt to the final test distribution.

We also observed that the ensemble model (Run 857) performed slightly worse than the single ResNet34 model (Run 843). This suggests that the three architectures may have learned similar feature representations and correlated prediction errors. Since all three models were trained on the same development dataset using similar optimization settings, they likely learned correlated feature representations rather than complementary ones. As a result, combining their predictions through soft voting did not provide the expected improvement in generalization performance.

The negative Cohen's Kappa scores obtained during the final evaluation further indicate that the models struggled to generalize reliably to the unseen test distribution. Similar observations were reported in the 2025 edition of this challenge [7], where supervised approaches also showed difficulty in generalizing GAN fingerprint detection across different data distributions.

Overall, these findings highlight the difficulty of detecting GAN training fingerprints in synthetic medical images. While the models achieved strong performance on the development dataset, the final results demonstrate that learning robust and generalizable fingerprint representations remains a challenging problem.

6. Conclusion

In this work, we explored whether deep learning models trained on labeled development data could detect whether a real CT image was used during GAN training. We approached this as a binary classification problem and investigated two strategies — a single fine-tuned ResNet34 model and an ensemble of three models combining ResNet34, EfficientNet-B3, and DenseNet121 using soft voting.

During the development phase, our ResNet34 model performed strongly, achieving a Cohen's Kappa of 0.9122 and a recall of 0.9975. These results gave us confidence that transfer learning was a promising direction for this task.

However, the final phase told a different story. Both submissions achieved negative Kappa scores, meaning the models performed worse than random guessing on the unseen test data. After reflecting on these results, we believe the main reason was domain shift — the final test images came from a different distribution than the development data, and our models were not able to adapt to that difference. The fact that our models reached near-perfect training accuracy also suggests that overfitting played a role, with the models memorizing development data patterns rather than learning truly generalizable features.

These results are humbling but important. They show that strong development phase performance does not always translate to real-world generalization, especially when the test distribution is unknown. They also confirm what teams in the 2025 edition found [7] — that detecting GAN training fingerprints is a genuinely difficult problem that current supervised approaches struggle to solve reliably.

Looking ahead, we think there are several directions worth exploring. Training with augmentation strategies that simulate distribution shift could help models become more robust. Using the generated synthetic images during training — which we did not do in this work — might also provide useful signals. Additionally, approaches that directly compare real and synthetic images in feature space, rather than treating this purely as a classification problem, could offer a more principled solution.

We hope our experience — both the successes and the failures — contributes to a better understanding of what makes this task so challenging, and helps future participants approach it with more informed strategies.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT for language refinement, grammar correction, and formatting assistance. After using this tool, the authors carefully reviewed and edited the content and take full responsibility for the final publication.

References

[1] A.-G. Andrei, M.-G. Constantin, M. Dogariu, D. Karpenka, Y. Prokopchuk, A. Radzhabov, D.-C. Stanciu, L.-D. Stefan, V. Kovalev, H. Muller, B. Ionescu, Overview of the 2026 ImageCLEFmedical GANs task: Fingerprint detection, latent space analysis, and privacy-preserving medical image synthesis, in: CLEF 2026 Working Notes, CEUR Workshop Proceedings, CEUR-WS.org, Jena, Germany, September 21-24, 2026.

[2] B. Ionescu, H. Muller, D.-C. Stanciu, A. Radu, R.-G. Bolborici, M. Negru, A.-F. Ene, V.-M. Vasilescu, A.-A. Nicolae, L.-D. Stefan, M.-G. Constantin, M. Dogariu, A.-G. Andrei, H. Damm, T. M. G. Pakull, A. Ben Abacha, A. Garcia Seco de Herrera, C. M. Friedrich, R. Brungel, L. Reinartz, H. Schafer, C. S. Schmidt, B. Bracke, P. Nath, B. Eryilmaz, M. Hjuler, D. Fabre, C. Lemaire, B. Lecouteux, D. Schwab, D. Dimitrov, M. S. Hee, M. Ahsan, S. Ahmad, D. Zlatkova, G. Pachov, Z. Xie, P. Nakov, I. Koychev, J. E. Heras Rivera, D. K. Low, W. Yim, J. Ruzevick, D. Child, M. Kurt, Z. Sun, F. Xia, M. Yetisgen, A. Radzhabov, Y. Prokopchuk, V. Kovalev, D. Karpenka, S. A. Hicks, S. Gautam, M. A. Riegler, V. Thambawita, P. Halvorsen, M. El Sakka, J. Mothe, A. Baicoianu, C.-N. Florea, M. Ivanovici, Overview of ImageCLEF 2026: Multimodal Challenges in Medicine, Science, Agritech, and Security, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction, Proceedings of the Seventeenth International Conference of the CLEF Association (CLEF 2026), Springer Lecture Notes in Computer Science LNCS, Jena, Germany, September 21-24, 2026.

[3] M. Frid-Adar, I. Diamant, E. Klang, et al., GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification, Neurocomputing 321 (2018) 321-331.

[4] H.-C. Shin, N. A. Tenenholtz, J. K. Rogers, et al., Medical image synthesis for data augmentation and anonymization using generative adversarial networks, arXiv:1807.10225 (2018).

[5] J. Hayes, L. Melis, G. Danezis, E. De Cristofaro, LOGAN: Membership inference attacks against generative models, in: ACM SIGSAC, 2019, pp. 579-595.

[6] B. Hilprecht, M. Harterich, D. Bernau, Monte Carlo and anomaly-based membership inference attacks against generative models, in: IEEE TPS-ISA, 2019, pp. 174-183.

[7] K. S. Saravanan, M. Subramaniam, R. B. A. Narayanan, B. P., K. Ayyamperumal, Detecting training data usage in synthetic images using machine learning techniques, in: CLEF 2025 Working Notes, CEUR-WS.org, 2025.

[8] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: CVPR, 2016, pp. 770-778.

[9] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: CVPR, 2017, pp. 4700-4708.

[10] M. Tan, Q. V. Le, EfficientNet: Rethinking model scaling for convolutional neural networks, in: ICML, 2019, pp. 6105-6114.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, et al., Generative adversarial nets, in: NeurIPS, 2014.

[12] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980 (2014).