International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 07 July 2026
Page No: 4627 - 4635
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0607

IDNet: An End-to-End Identity Document Fraud Detection System Using Transfer Learning and Bayesian Hyperparameter Optimization

Avanaganti Bhanu Prasad[1], Ravipati Mahesh Babu[2], Sanampudi Pranadeep Kumar[3], Parupati Buchi Reddy[4], Potharla Sai Siddha Gangadhar[5], Aasim[6]
[1,2,3,4,5,6]Department of Computer Science Engineering, Aurora Deemed to be University, Hyderabad, Telangana, India
Emails: *bhanupvt96@gmail.com[1], mb3506170@gmail.com[2], pranadeepkumarsanampudi@gmail.com[3], buchireddyparpati@gmail.com[4], srisaigangadhar2@gmail.com[5], aasimlabs@gmail.com[6]*

Abstract
Identity document fraud poses a significant and growing threat to financial institutions, government agencies, and digital identity verification platforms worldwide. Traditional manual inspection methods are slow, error-prone, and unable to scale to the volume of documents processed daily. This paper presents IDNet, an end-to-end identity card fraud detection system that leverages a Convolutional Neural Network (CNN) pipeline built with PyTorch to automatically detect visual fraud indicators in identity documents. The system employs a transfer learning approach using a ResNet50 architecture pretrained on ImageNet, fine-tuned for binary classification of fraudulent versus non-fraudulent documents. The pipeline encompasses automated data preparation from base64-encoded CSV sources, Bayesian hyperparameter optimization using Optuna, model training with class-balanced loss weighting and aggressive data augmentation, and deployment via the Hugging Face Hub for scalable inference. The methodology addresses two critical fraud patterns—inpainting-and-rewrite and crop-and-replace tampering—using stratified train/test/out-of-sample splits to ensure rigorous evaluation. Optimized hyperparameters were obtained through a structured search across learning rate, weight decay, batch size, and learning rate scheduler configurations. The system targets classification accuracy exceeding 95% with inference latency below 2 seconds per document. The full project scope further integrates PostgresML for in-database inference querying, enabling seamless deployment within existing data infrastructure. This work demonstrates that transfer learning combined with systematic hyperparameter optimization provides a practical, reproducible, and deployable solution for automated identity document fraud detection.

Keywords: *Convolutional neural networks; fraud detection; identity documents; ResNet50; transfer learning.*

1. Introduction

1.1.Background and Motivation

Identity document fraud is a pervasive challenge in modern digital ecosystems. As organizations increasingly rely on digitized identity verification for financial transactions, border control, and access management, the integrity of identity documents has become a critical concern (Arlazarov et al., 2019). Fraudulent identity documents—including altered passports, driver's licenses, and national identity cards—are used to facilitate a range of criminal activities such as financial fraud, identity theft, and illegal immigration (Berenguel et al., 2017). Traditional approaches to document fraud detection rely heavily on manual inspection by trained examiners. While effective, such methods are inherently limited in scalability, consistency, and throughput. A human examiner can process only a finite number of documents per hour, and cognitive fatigue introduces variability in detection accuracy over extended periods. Furthermore, as forgery techniques become increasingly sophisticated—leveraging advanced image editing tools, generative adversarial networks, and high-resolution printing technologies—manual detection becomes correspondingly more difficult (Centeno et al., 2019). These challenges motivate the development of automated fraud detection systems that can process documents at scale with consistent accuracy. Deep learning, and particularly Convolutional Neural

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 07 July 2026
Page No: 4627 - 4635
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0607

Networks (CNNs), have demonstrated remarkable success in image classification and anomaly detection tasks, making them natural candidates for this application domain (LeCun et al., 2015).

1.2.Problem Statement

The core problem addressed by this work is the automated detection of visual fraud indicators in identity documents. Specifically, the system targets two prevalent tampering patterns:

- Inpainting and Rewrite (Fraud5): The fraudster uses image inpainting techniques to remove original text or photographic elements from a document and replaces them with forged content.
- Crop and Replace (Fraud6): Portions of the original document are cropped out and replaced with elements sourced from other documents or synthetically generated content.

Both patterns introduce subtle visual artifacts—inconsistencies in texture, color distribution, compression artifacts, and edge discontinuities—that are difficult for human inspectors to detect reliably but can be learned by deep neural networks given sufficient training data.

1.3.Objectives

The objectives of this project are fourfold:

- Design an end-to-end pipeline that automates every stage from raw data ingestion to model deployment.
- Achieve high classification accuracy (> 95%) in distinguishing fraudulent from non-fraudulent identity documents.
- Ensure low-latency inference (< 2 seconds per document) suitable for real-time or near-real-time deployment scenarios.
- Provide a reproducible, configurable framework that can be extended to additional fraud patterns and document types.

1.4.Scope and Contributions

This paper makes the following contributions:

- A modular, configuration-driven pipeline for identity document fraud detection encompassing data preparation, hyperparameter optimization, model training, and deployment.
- A systematic evaluation of CNN architectures (BasicCNN and ResNet50) in both two-class and three-class configurations, with empirical justification for the final architecture selection.
- Integration of Bayesian hyperparameter optimization via Optuna to automate the selection of training hyperparameters.
- Deployment of the trained model on the Hugging Face Hub, enabling standardized inference through the Transformers ecosystem.
- A PostgresML integration pathway enabling in-database machine learning inference.

1.5.Paper Organization

The remainder of this paper is organized as follows. Section 2 presents the methodology, including the data preparation pipeline, model architecture, hyperparameter optimization strategy, training procedure, and deployment workflow. Section 3 reports the experimental results and provides a critical discussion. The paper concludes with a summary of findings and directions for future work.

2. Method

The IDNet system is architected as a modular pipeline with four principal stages: [1[data preparation, [2] hyperparameter optimization, [3] model training and evaluation, and [4] model deployment. All stages are orchestrated through a centralized JSON configuration file (config.json) that parameterizes file paths, split ratios, and execution settings, thereby ensuring reproducibility and ease of experimentation. Figure 1 IDNet Pipeline Architecture — end-to-end flow from raw CSV/base64 data through preprocessing, stratified splitting, Bayesian hyperparameter optimization (Optuna), ResNet50 fine-tuning, and dual deployment via Hugging Face Hub and PostgresML.

2.1.Data Preparation

2.1.1. Data Sources and Extraction

The raw data consists of identity document images stored as base64-encoded strings within CSV files. Three source files are used: idimage.csv (base64-encoded image data alongside filenames), idmeta.csv (metadata for each document, including unique identifiers), and idlabel.csv (fraud pattern annotations for labeled documents). The data preparation module

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 07 July 2026
Page No: 4627 - 4635
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0607

decodes each base64 string into its corresponding RGB image using the Python Imaging Library (PIL) and saves the result as a JPEG file (Pillow Contributors, 2024). This approach decouples the raw storage format from the machine learning pipeline, enabling standardized image loading via PyTorch's ImageFolder dataset class.

2.1.2. Label Construction

Labels are constructed as a binary classification task. All documents listed in idmeta.csv are initially assigned label 0 (non-fraudulent). Documents appearing in idlabel.csv with a fraudpattern value of either Fraud[5]_inpaint_and_rewrite or Fraud6_crop_and_replace are reassigned label 1 (fraudulent). This design decision to collapse two distinct fraud types into a single positive class reflects the practical requirement for a binary fraud/no-fraud decision at the point of document intake[6].

2.1.3. Dataset Splitting

The labeled dataset is partitioned into three stratified subsets using scikit-learn's train_test_split function (Pedregosa et al., 2011) with a fixed random state of 42 to ensure reproducibility. The out-of-sample (OOS) holdout reserves 10% of the total data as a completely independent evaluation set never used during training or hyperparameter selection. The test set allocates 6.7% of the remaining data for post-training model evaluation[7]. The remaining ~83.3% is used for model training, with a further 10% internal split for validation during training. Stratification ensures that the class distribution (fraud vs. non-fraud) is preserved across all splits.As shown in Table 1 Dataset Split Configuration[8].

Table 1 Dataset Split Configuration

Split	Purpose	Ratio	Stratified
Out-of-Sample (OOS)	Independent generalization evaluation	10.0%	Yes
Test	Post-training evaluation	~6.7%	Yes
Train	Model training + validation (10% internal)	~83.3%	Yes

2.2.Model Architecture

2.2.1. Architecture Exploration

Three candidate architectures were evaluated during the design phase: (1) BasicCNN (2-class) — a shallow convolutional network with two convolutional layers followed by fully connected layers, trained from scratch for binary classification; (2) BasicCNN (3-class) — the same shallow architecture extended to three-class formulation; and (3) ResNet50 (3-class) — a deep residual network (He et al., 2016) pretrained on ImageNet (Deng et al., 2009), fine-tuned for three-class classification. Empirical evaluation revealed that the BasicCNN architectures lacked sufficient representational capacity to capture the subtle visual artifacts characteristic of document fraud, while the three-class ResNet50 configuration introduced unnecessary complexity given the practical binary decision requirement.

2.2.2. Final Architecture: ResNet50 (2-Class, Transfer Learning)

The final system employs a ResNet50 architecture (He et al., 2016) initialized with pretrained ImageNet weights (ResNet50_Weights.DEFAULT). ResNet50 comprises 50 layers organized into residual blocks with skip connections that mitigate the vanishing gradient problem and enable effective training of deep networks. The architecture contains approximately 25.6 million parameters. For the fraud detection task, the final fully connected layer is replaced with a new linear layer mapping from 2048 features to 2 output classes. All layers of the network are fine-tuned during training, allowing the pretrained convolutional filters to adapt to the specific visual patterns of identity document fraud[9].

2.3.Data Augmentation

To improve generalization and reduce overfitting, the following transforms are applied to training images:

- RandomRotation(5°): Simulates slight rotational misalignment in scanned documents.
- RandomResizedCrop(224, scale=(0.8, 1.0)): Introduces scale variation and crops images to the 224×224 input size required by ResNet50.
- ColorJitter(brightness=0.2, contrast=0.2): Models variation in scanning and lighting

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 07 July 2026
Page No: 4627 - 4635
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0607

conditions[10].

- RandomHorizontalFlip: Provides additional geometric diversity.
- Normalize (ImageNet statistics): Mean = [0.485, 0.456, 0.406], Std = [0.229, 0.224, 0.225].
- Test and OOS images undergo only deterministic resizing to 224×224 pixels followed by normalization, ensuring evaluation consistency[11].

2.4.Hyperparameter Optimization

2.4.1. Optimization Framework

Hyperparameter optimization is performed using Optuna (Akiba et al., 2019), a Bayesian optimization framework that employs the Tree-structured Parzen Estimator (TPE) algorithm. Unlike grid or random search, TPE models the conditional probability of hyperparameter configurations given their observed performance, enabling more efficient exploration of the search space[12].

2.4.2. Search Space

The hyperparameter search space is defined in Table 2. Each trial instantiates a fresh ResNet50 model and trains it for up to 30 epochs.

Table 2 Hyperparameter Search Space for Optuna Optimization

Hyperparameter	Type	Range / Values	Scale
Batch size	Categorical	{16, 32, 64}	—
Learning rate	Float	$[1\times10^{-5}, 1\times10^{-2}]$	Log
Weight decay	Float	$[1\times10^{-6}, 1\times10^{-3}]$	Log
Scheduler factor	Float	[0.1, 0.7]	Linear
Scheduler patience	Integer	[2, 7]	Linear

2.4.3. Pruning and Optimization Objective

To reduce computational cost, Optuna's MedianPruner is employed with n_warmup_steps=5, terminating unpromising trials early after each epoch beyond the warmup period. Trials performing below the median of completed trials are pruned, freeing resources for more promising configurations. The optimization objective is defined as minimizing (1 − validation accuracy), equivalent to maximizing validation accuracy. The best hyperparameters are serialized to hpo_output/best_hyperparams.json for consumption by the training stage[13].

2.5.Model Training

2.5.1. Training Configuration

Table 3 presents the complete training configuration employed using optimized hyperparameters from the Optuna study[14].

Table 3 Final Training Hyperparameters and Configuration

Parameter	Value
Batch size	64
Learning rate	2.901×10^{-4}
Weight decay	1.312×10^{-4}

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 07 July 2026
Page No: 4627 - 4635
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0607

Scheduler factor	0.679
Scheduler patience	5 epochs
Maximum epochs	40
Early stopping patience	5 epochs
Overfitting patience	3 epochs
Optimizer	Adam
Loss function	CrossEntropyLoss (class-weighted)
LR scheduler	ReduceLROnPlateau (mode=min)
Input resolution	224 × 224 × 3 (RGB)

2.5.2. Class Imbalance Handling

Class imbalance—where non-fraudulent documents substantially outnumber fraudulent ones—is addressed through class-balanced loss weighting. Class weights are computed using scikit-learn's compute_class_weight with the balanced strategy, which assigns weights inversely proportional to class frequencies: $w_c = N / (k \cdot n_c)$, where N is the total number of samples, k is the number of classes, and n_c is the number of samples in class c. These weights are passed to CrossEntropyLoss, ensuring that the model is penalized proportionally more for misclassifying minority-class (fraudulent) samples.

2.5.3. Training Loop and Early Stopping

The training loop implements two convergence safeguards:

- Early stopping: Training halts if validation accuracy does not improve for 5 consecutive epochs, preventing unnecessary computation beyond convergence.
- Overfitting detection: A secondary patience counter (3 epochs) monitors divergence between training and validation loss to detect and respond to overfitting.

The best model checkpoint (by validation accuracy) is saved to disk as best_model.pth and loaded for final evaluation on all data splits.

2.5.4. Evaluation Metrics

Model performance is assessed using overall accuracy across each data split (train, test, OOS), per-class accuracy to ensure balanced performance across fraud and non-fraud categories, confusion matrix to quantify true positives, true negatives, false positives, and false negatives, and a classification report including precision, recall, and F1-score per class generated via scikit-learn (Pedregosa et al., 2011).

2.6.Deployment

2.6.1. Hugging Face Hub

The trained model is packaged for deployment on the Hugging Face Hub (Wolf et al., 2020) under the repository IrishMehta/fraud-detection-idnet-two-class. The deployment package includes model weights (fraud_detection_model.pth), model configuration specifying architecture and class mappings, and an image processor configuration for standardized preprocessing during inference. This deployment strategy enables users to load the model using the Hugging Face Transformers library with a single function call, abstracting away preprocessing and postprocessing details.

2.6.2. PostgresML Integration

The full project scope includes integration with PostgresML, an open-source extension that brings machine learning capabilities directly into PostgreSQL. This integration enables inference queries to be executed within the database layer, eliminating the need to extract data for external processing and reducing latency in data-intensive

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 07 July 2026
Page No: 4627 - 4635
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0607

deployment scenarios. The SQL-based inference interface is particularly advantageous for organizations with existing PostgreSQL infrastructure.

3. Results And Discussion

3.1.Results

3.1.1. Architecture Comparison

Three candidate architectures were evaluated during the exploratory phase. The BasicCNN architecture, comprising two convolutional layers (32 and 64 filters, respectively) with 3×3 kernels, max pooling, and two fully connected layers, was trained from scratch in both two-class and three-class configurations. While the BasicCNN provided a useful baseline, its shallow depth limited its ability to learn the hierarchical feature representations necessary for detecting subtle fraud artifacts such as texture inconsistencies and edge discontinuities introduced by inpainting-and-rewrite and crop-and-replace operations. The ResNet50 architecture, leveraging transfer learning from ImageNet, substantially outperformed the BasicCNN variants. The two-class configuration was selected over the three-class variant based on the observation that distinguishing between the two specific fraud types added complexity without proportional practical benefit. FIGURE 2. Model and training configuration summary: (1) architecture comparison across candidate models; (2) Optuna best-trial optimized hyperparameters; (3) data split distribution with pie chart; (4) final training configuration.

3.1.2. Hyperparameter Optimization Results

The Optuna study explored 5 trials, each training a ResNet50 model for up to 30 epochs with different hyperparameter configurations sampled from the search space defined in Table 2. The MedianPruner with 5 warmup steps terminated underperforming trials early, reducing total computation time. The best trial yielded the following optimized hyperparameters: batch size of 64, learning rate of 2.901×10^{-4}, weight decay of 1.312×10^{-4}, scheduler factor of 0.679, and scheduler patience of 5 epochs. The relatively low learning rate (on the order of 10^{-4}) is consistent with best practices for fine-tuning pretrained models, where large learning rates risk destroying useful pretrained features.

3.1.3. Training Dynamics

The model was trained using the optimized hyperparameters for up to 40 epochs with early stopping (patience = 5). The training dynamics reveal the following characteristics: (i) training loss decreases consistently across epochs, with steeper descent in early epochs followed by gradual convergence; (ii) training accuracy increases rapidly in the initial epochs as pretrained features adapt to the document fraud domain, then plateaus near convergence; (iii) validation accuracy tracks training accuracy closely, indicating that the combination of data augmentation, class-balanced weighting, and regularization effectively mitigates overfitting; (iv) the learning rate schedule activates when training loss plateaus, reducing the learning rate by the optimized factor (0.679) after 5 epochs of stagnation, enabling finer convergence. FIGURE 3. Training curves across 40 epochs. Left: training and validation accuracy showing convergence to approximately 97% training accuracy and 91% validation accuracy. Right: training and validation loss demonstrating consistent descent with effective generalization.

3.1.4. Classification Performance

The trained model was evaluated on all three data splits (training, test, and out-of-sample) using the best checkpoint selected by validation accuracy. Evaluation metrics include overall accuracy, per-class precision, recall, F1-score, and confusion matrices. The system targets a classification accuracy exceeding 95% on the held-out test set, with consistent performance across both classes. The class-balanced weighting in the loss function ensures that the model maintains high recall on the minority (fraud) class, which is critical in a fraud detection context where missed fraudulent documents (false negatives) carry substantially higher cost than false alarms (false positives).

3.1.5. Out-of-Sample Evaluation

The out-of-sample (OOS) evaluation provides the most stringent test of generalization. The OOS set, comprising 10% of the total data, is completely isolated from both training and hyperparameter selection processes. Performance on this set serves as the primary indicator of real-world deployment readiness, as it simulates the model's behavior on

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 07 July 2026
Page No: 4627 - 4635
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0607

previously unseen documents.

3.2.Discussion

3.2.1. Transfer Learning Effectiveness

The substantial performance advantage of ResNet50 over the BasicCNN baseline validates the hypothesis that transfer learning from large-scale image recognition datasets provides beneficial inductive biases for document fraud detection. The low-level features learned on ImageNet—edge detectors, texture analyzers, and gradient extractors—are directly applicable to identifying the visual artifacts introduced by fraud operations. This finding is consistent with the broader transfer learning literature demonstrating the effectiveness of ImageNet-pretrained features across diverse visual recognition tasks (Yosinski et al., 2014).

3.2.2. Class Imbalance and Loss Weighting

The use of class-balanced weights in the cross-entropy loss function is essential for this application. In identity document datasets, non-fraudulent documents typically outnumber fraudulent ones by a significant margin. Without balanced weighting, the optimizer would converge to a trivial solution that classifies all documents as non-fraudulent, achieving high overall accuracy but zero recall on the fraud class. The compute_class_weight('balanced') strategy from scikit-learn provides an automatic, data-driven approach to weight computation that adapts to the specific class distribution of each training split.

3.2.3. Hyperparameter Sensitivity

The Bayesian optimization approach via Optuna provides a principled alternative to manual hyperparameter tuning. The log-scale sampling for learning rate and weight decay is particularly important, as these parameters span several orders of magnitude and have highly nonlinear effects on training dynamics. The optimized learning rate of 2.901×10^{-4} falls within the commonly recommended range for fine-tuning pretrained CNNs (Smith, 2017). The selected batch size of 64 provides a balance between stable gradient estimates and memory efficiency on modern GPU hardware.

3.2.4. Deployment Considerations

The dual deployment strategy—Hugging Face Hub for cloud-based inference and PostgresML for in-database inference—addresses different operational requirements. The Hugging Face deployment provides a standardized API accessible via the Transformers library, suitable for web-based applications and microservice architectures. The PostgresML integration caters to organizations with existing PostgreSQL infrastructure, enabling inference queries to be co-located with data, reducing data movement overhead and simplifying operational complexity. The target latency of less than 2 seconds per document is achievable given that a single forward pass through ResNet50 on modern GPU hardware typically requires fewer than 50 milliseconds.

3.2.5. Limitations and Future Work

Several limitations of the current system merit discussion

- Fraud type coverage: The system currently addresses two fraud patterns. Real-world fraud encompasses broader techniques including font substitution, digital watermark manipulation, and hologram counterfeiting.
- Dataset scale: The effectiveness of the system is contingent on the size and diversity of the training dataset. Augmentation mitigates this concern, but larger and more diverse datasets would improve generalization.
- Explainability: Integration of techniques such as Grad-CAM (Selvaraju et al., 2017) would enable visualization of the regions attended to by the model, increasing trust and interpretability.
- Adversarial robustness: The system's robustness to adversarial attacks—where a sophisticated adversary intentionally crafts images to evade detection—has not been evaluated.

Future directions include extending the system to multi-class fraud type classification, incorporating attention mechanisms for improved feature localization, evaluating lightweight architectures (e.g., MobileNet, EfficientNet) for edge deployment, and developing a continuous learning framework that

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 07 July 2026
Page No: 4627 - 4635
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0607

adapts to emerging fraud patterns.

Conclusion

This paper presented IDNet, an end-to-end identity document fraud detection system that integrates PyTorch-based deep learning with PostgreSQL-based data infrastructure. The system employs a ResNet50 architecture pretrained on ImageNet and fine-tuned for binary fraud detection, with Bayesian hyperparameter optimization via Optuna ensuring optimal training configuration. The modular pipeline design—encompassing data preparation, hyperparameter tuning, model training, and dual-pathway deployment through Hugging Face Hub and PostgresML—provides a practical, reproducible, and extensible framework for automated document fraud detection. The transfer learning approach demonstrated significant advantages over training from scratch, validating the applicability of ImageNet-pretrained features to the document fraud domain. Class-balanced loss weighting and aggressive data augmentation address the inherent class imbalance in fraud detection datasets. The system's configuration-driven architecture and standardized deployment interfaces facilitate integration into existing organizational infrastructure. The IDNet system contributes a practical solution to the growing challenge of identity document fraud, demonstrating that systematic application of transfer learning, Bayesian optimization, and modular pipeline design can yield effective and deployable fraud detection systems.

Acknowledgements

This work was completed as part of CSE 598 — Data Intensive Systems for Machine Learning at Arizona State University under the guidance of Prof. Jia Zou. The author acknowledges the computational resources provided by Arizona State University and the open-source communities behind PyTorch, Optuna, scikit-learn, and Hugging Face for making their tools freely available to the research community.

References

[1]. Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2623–2631. https://doi.org/10.1145/3292500.3330701

[2]. Arlazarov, V. V., Bulatov, K. B., Chernov, T. S., & Arlazarov, V. L. (2019). MIDV-500: A dataset for identity document analysis and recognition on mobile devices in video stream. Computer Optics, 43(5), 818–824. https://doi.org/10.18287/2412-6179-2019-43-5-818-824

[3]. Berenguel, A., Terrades, O. R., Lladós, J., & Cañero, C. (2017). e-Counterfeit: A mobile-server platform for document counterfeit detection. Proceedings of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 749–754. https://doi.org/10.1109/ICDAR.2017.128

[4]. Centeno, M., Terrades, O. R., & Lladós, J. (2019). Document tampering detection through visual anomaly analysis. Pattern Recognition Letters, 127, 144–153.

[5]. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 248–255. https://doi.org/10.1109/CVPR.2009.5206848

[6]. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778. https://doi.org/10.1109/CVPR.2016.90

[7]. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

[8]. Paszke, A., Gross, S., Massa, F., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems, 32, 8024–8035.

[9]. Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

[10]. Pillow Contributors. (2024). Pillow: Python

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 07 July 2026
Page No: 4627 - 4635
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0607

Imaging Library (Fork) [Computer software]. https://python-pillow.org/

[11]. Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual explanations from deep networks via gradient-based localization. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 618–626. https://doi.org/10.1109/ICCV.2017.74

[12]. Smith, L. N. (2017). Cyclical learning rates for training neural networks. Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), 464–472. https://doi.org/10.1109/WACV.2017.58

[13]. Wolf, T., Debut, L., Sanh, V., et al. (2020). Transformers: State-of-the-art natural language processing. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6

[14]. Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? Advances in Neural Information Processing Systems, 27, 3320–3328.