AI-Generated Image Detector

CHEN RONGRONG LIU WANHUA PING WANG QIAN WAN WANG, YICHEN

Abstract

This project aims to provide photography stock platforms with an image authentication tool to identify AI-generated photos, safeguarding integrity of photographers' works. It focuses on developing an end-to-end web app, using neural networks to accurately distinguish AI-generated landscape photos from human-captured ones. Model performance is evaluated based on scenery types, with explanations provided by Gram-CAM.

1. Introduction

In today's digital economy, photography stock platforms such as Adobe Stock, Shutterstock, Getty Images, and 500px have gained immense popularity as venues for monetizing photography. These platforms heavily rely on the authenticity of shared content, but the emergence of advanced AI tools like DALL-E and Mid-Journey presents a growing challenge. These technologies can produce highly realistic images that may be mistaken for genuine photographs, misleading consumers, harming careers and livelihood of professional photographers, and undermining trust in digital content.

Our project aims to address this issue by using machine learning algorithms and neural networks like ResNet50, MobileNetV2, and EfficientNetB0 to accurately distinguish between AI-generated and human-captured (referred to as "authentic" or "real") photos, focusing specifically on landscape analysis. This focus serves three primary purposes: firstly, due to the narrow focus, gathering sufficient data for effective learning of each scenery type is tangible, which ensures model accuracy. Secondly, by utilizing a specialized dataset, model's explainability is enhanced across various scenarios, aiding in fine-tuning. Additionally, this task holds significant societal value, preventing misleading information for tourists and aiding environmental monitoring efforts.

The desired outcome for the AI Image Authenticity Verification project focuses on four main objectives:

- **Accurate Detection:** Striving for superior detection rates of AI-generated images to minimize false positives and negatives, aiming to set a new industry standard that surpasses current benchmarks like C2PA.
- **End-to-End Streamlined Web Application:** Committing to traversing the entire process from data collection to deployment as a web application, ensuring ease of use.
- **High Explainability:** Guaranteeing clear explanations for why the model succeeds or fails on specific scenery types, empowering users with insights during deployment.
- **Beneficial Social Impact:** Enhancing digital trust, demonstrated by reduced misinformation spread and increased user engagement.

In summary, this project's objective is not only to provide a tool for image authentication but also to maintain individual credit and ethics, maintain professional standards, and support the integrity of photography stock platforms within the digital economy.

2. Dataset

2.1. Data Collection

Our dataset comprises two distinct sets of scenery images (Real and AI-generated). For real ones, we utilize the Kaggle dataset available at "Intel Image Classification". This dataset features finely categorized scenery images, including buildings, forests, glaciers, mountains, seas, and streets, which is ideal for detailed exploratory analysis. For AI-generated images, we source photos of the same scenery types from Freepik platform, ensuring a balanced representation across all scenery type to mimic the diversity found in the real images. This structured approach aids in our subsequent EDA and insights generation.

2.2. Exploratory Data Analysis (EDA)

In our exploratory data analysis, we compare image counts of the two classes (AI-generated and real) and perform a descriptive analysis for each scenery type using word clouds derived from image descriptions. The overall number of real images is slightly higher than AI-generated ones, and we maintain this ratio across scenery types to ensure the fair comparison of model performance between scenery types.

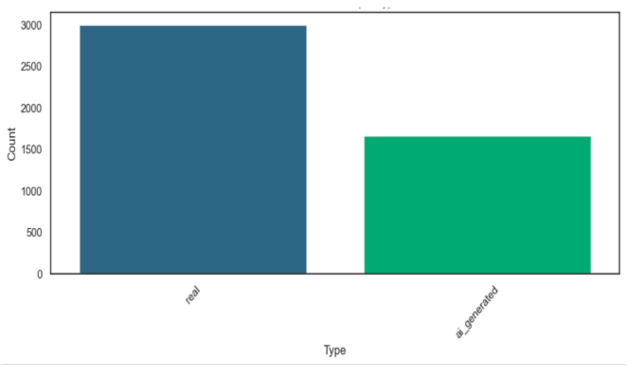


Figure 1. Number of Entries per Class

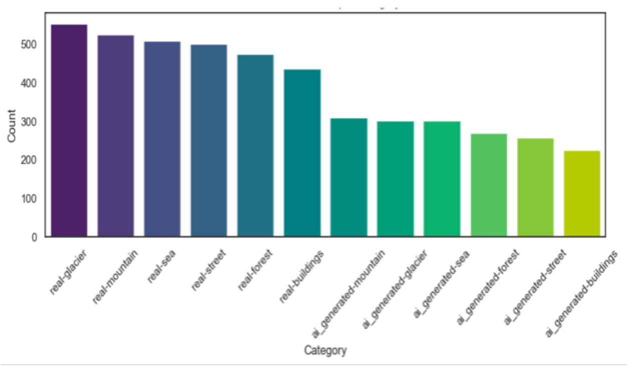


Figure 2. Number of Entries per Scenery Type

The word clouds created for each scenery type reveal distinct thematic and visual patterns. In the forest category, prevalent terms like “green”, “lush”, and “tranquil” not only suggest dense vegetation but also imply a dominant green color palette that characterizes these images. For the buildings category, words such as “downtown”, “skyscraper”, and “commercial” reflect the urban environments typically depicted, featuring architectural elements and often a more varied and sometimes grayish color tone due to the concrete and metal structures. The glacier category is marked by descriptors like “snow”, “ice”, and “mountain”, indicating imagery filled with white and blue hues that emphasize the chill and serenity of snowy landscapes. This approach allow us to gather valuable insights into the common descriptors and color schemes associated with each scenery type, aiding in the development of a more nuanced model capable of distinguishing between AI-generated and real images based on both content and stylistic elements.

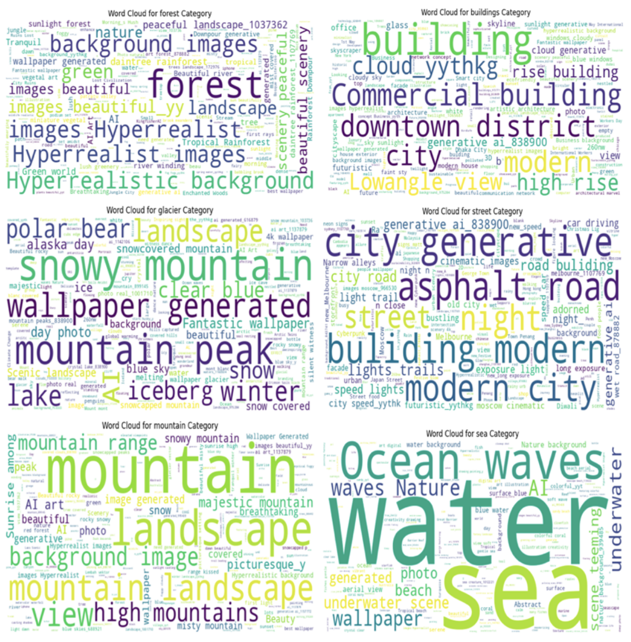


Figure 3. Word Cloud per Scenery Type

3. Modeling

3.1. Key Features of Chosen Models

3.1.1. ResNet50

ResNet50 features shortcut connections and bottleneck layers, enhancing training efficiency and performance. Key advantages include:

- **Gradient Management:** Addresses vanishing gradient problem with shortcut connections, ensuring efficient gradient flow.
- **Efficient Training:** Utilizes 1x1 convolutions in bottleneck layers to reduce parameters and computational load, enabling faster training.
- **Streamlined Complexity:** Simplifies structure compared to VGG architecture, maintaining feature capture effectiveness.

3.1.2. MobileNetV2

MobileNetV2, renowned for efficiency on mobile devices, serves as a feature extractor leveraging pretrained features from ImageNet. Key features include:

- **Efficiency:** Designed for computational efficiency without compromising performance, suitable for hardware-limited devices.

- **Transfer Learning:** Pre-trained on ImageNet, providing strong feature extraction base applicable to various tasks.
- **Adaptability:** Easily adaptable to new tasks with minimal structural adjustments, especially effective in image classification tasks.

3.1.3. EfficientNetB0

EfficientNetB0 offers a scalable architecture balancing depth, width, and resolution for optimal performance and efficiency. Key features include:

- **Efficiency and Scalability:** Designed for systematic scaling across model dimensions, achieving superior performance without excessive computational demand.
- **Transfer Learning:** Pre-trained on ImageNet, providing robust feature set for diverse image recognition tasks.
- **Adaptability:** Easily adaptable with minimal adjustments, particularly effective in image classification tasks leveraging deep learned features.

3.2. Pipeline Overview

For experimentation purpose, two deep learning frameworks are used: PyTorch for ResNet50 and EfficientNetB0 while Tensorflow for MobileNetV2. We harmonize the two frameworks in the deployment stage, which will be elaborated the following section, **Web App Implementation**. The pipeline for the modeling section consists of following stages:

- **Data Loading and Preprocessing:** Images are shuffled, loaded into batches, and assigned with labels: ’0’ for human-photographed ones and ’1’ for AI-generated ones. Then, images are resized to match model input requirement (i.e. 224*224 for ResNet50 and EfficientNetB0 and 150*150 for MobileNetV2).

  Additionally, EfficientNetB0 and MobileNetV2 normalize images to align with the pixcel value distribution of corresponding pre-trained models’ expectation, while ResNet50 applies data augmentation, including ramdom flips, rotations, and color adjustments, to improve generalizability of model.
- **Data Split:** Images are stratifiedly splitted into training (60%), validation (20%), and test (20%) subsets.
- **Model Setup:** Model architectures for ResNet50, MobileNetV2, and EfficientNetB0 are configured. Output layer is modified for binary classification task. Models are compiled with the Adam optimizer due to its adaptive learning rate capabilities, alongside a binary cross-entropy loss function, which is standard for binary classification tasks.
- **Training Loop:** Each model undergoes 10-epoch training, with validation after each iteration.
- **Model Evaluation:** Model evaluation metrics, implemented on each scenery type, cover F1-score, precision, and recall, providing a comprehensive view of model performance across potential class imbalances.

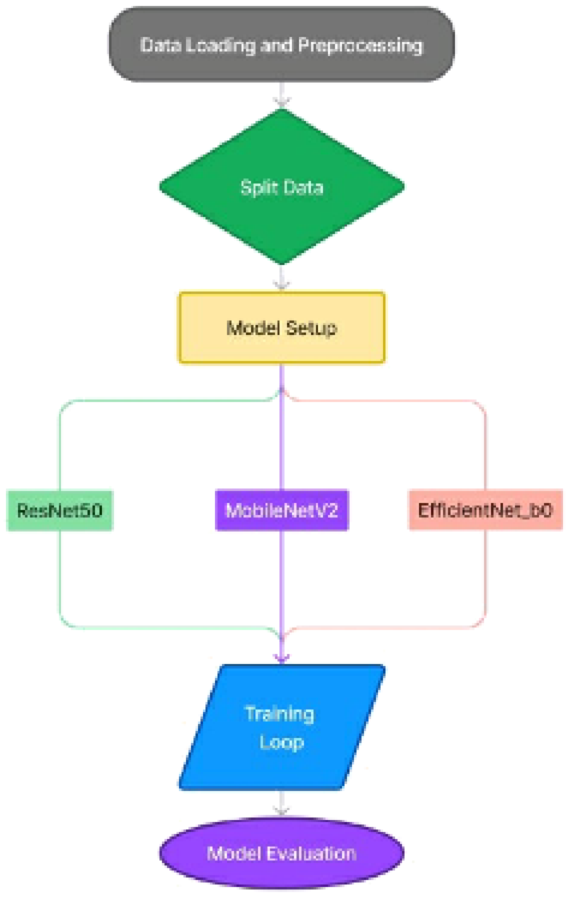


Figure 4. Modeling Pipeline

3.3. Model Architecture, Adaptation, and Finetuning

3.3.1. ResNet50

ResNet50 consists of:

- **Conv1:** 7x7 convolutional layer with 64 filters and a stride of 2, followed by max pooling.
- **Four layers of bottleneck blocks:**

- **Layer1 (9 convolutional layers):** 3×3,64 kernel convolution, another with 1×1,64 kernels, and a third with 1×1,256 kernels. These 3 layers are repeated 3 times (3 bottleneck blocks).
- **Layer2 (12 convolutional layers):** 1×1,128 kernels, 3×3,128 kernels, and 1×1,512 kernels, iterated 4 times (4 bottleneck blocks).
- **Layer3 (18 convolutional layers):** 1×1,256 kernels, and 3×3,256 kernels and 1×1,1024 kernels, iterated 6 times (6 bottleneck blocks).
- **Layer4 (9 convolutional layers):** 1×1,512 kernels, 3×3,512 kernels, and 1×1,2048 kernels iterated 3 times (3 bottleneck blocks).

- Average pooling layer followed by a fully connected layer with 1000 nodes for classification using softmax activation.

To tailor for binary classification task, we replace the last fully connected layer to an output layer with two classes.

In our quest to strike a balance between performance and computational efficiency, we embark on a series of experiments involving various layer finetuning strategies. Initially, we focus on fine-tuning solely the last layer, which yields an F1-score of 0.779 with minimal parameter adjustments. However, extending the finetuning to include the fully-connected layer and the last layer of bottleneck blocks (**Layer4**) does not result in a substantial enhancement of the test F1-score. Our subsequent refinement strategy, encompassing finetuning of all layers, leads to the highest F1-score achieved. This success, however, comes at the cost of doubling the trainable parameters compared to solely finetuning the last layer and bottleneck block.

Table 1. Finetuning for ResNet50

Finetuned Layers	Test F1-Score	Trainable Parameters
Last layer	0.779	4,098
Last two layers	0.785	14,968,834
All layers	0.990	23,512,130

Accordingly, since we prioritize high performance, we chose to deploy ResNet50 model with all layers finetuned.

3.3.2. MobileNetV2

After the convolutional base, MobileNetV2 introduces several custom layers:

- **Global Average Pooling 2D:** Reduces each feature map to a single number by taking the average of all values in the feature map. It helps in reducing the model's complexity and computational cost.

- **Batch Normalization:** Normalizes the activations from the previous layer, helping to stabilize and accelerate the training process.

- **Dropout:** Randomly sets input units to 0 with a rate of 0.5 at each step during training time, which helps prevent overfitting.

- **Dense Layer with Sigmoid Activation:** Outputs a probability indicating the likelihood of the image being AI-generated.

3.3.3. EfficientNetB0

After leveraging the convolutional base of EfficientNetB0, several custom layers are added to tailor the model to the binary classification task:

- **Global Average Pooling 2D:** Condenses each feature map to a single value, effectively reducing the dimensionality and focusing on the most salient features.

- **Batch Normalization:** Normalizes the inputs of each layer to improve the stability and speed of the training process.

- **Dropout:** With a dropout rate of 0.2, randomly omits units to mitigate overfitting.

- **Dense Layer with Softmax Activation:** Outouts probabilities for each class.

3.4. Model Performance Comparison

Across all scenery types, the three models—ResNet50, MobileNetV2, and EfficientNetB0—demonstrated consistent high accuracy, with F1-scores surpassing 0.98. Notably, all models excelled in the "forest" scenery, achieving F1-scores, precision, and recall exceeding 0.99.

Each model displays diverse strengths in distinguishing different types of scenery, with no single scenery type being universally perfect or worst for all models. Among the models, EfficientNetB0 emerged as the top performer, achieving perfect F1-scores, precision, and recall in most scenery types, except "street".

Table 2. Performance Metric for ResNet50

Scenery Type	F1-Score	Precision	Recall
Buildings	1.00	1.00	1.00
Forest	0.99	0.99	0.99
Glacier	0.98	0.98	0.98
Mountain	0.98	0.98	0.98
Sea	0.98	0.98	0.98
Street	1.00	1.00	1.00

Table 3. Performance Metric for MobileNetV2

Scenery Type	F1-Score	Precision	Recall
Buildings	0.989	1.000	0.977
Forest	1.000	1.000	1.000
Glacier	0.996	1.000	0.991
Mountain	0.995	1.000	0.991
Sea	0.995	1.000	0.990
Street	0.995	1.000	0.990

Table 4. Performance Metric for EfficientNetB0

Scenery Type	F1-Score	Precision	Recall
Buildings	1.000	1.000	1.000
Forest	1.000	1.000	1.000
Glacier	1.000	1.000	1.000
Mountain	1.000	1.000	1.000
Sea	1.000	1.000	1.000
Street	0.988	0.983	0.970

4. Web App Implementation

Our web application utilizes Streamlit, a flexible framework that facilitates rapid development and deployment of data-driven applications. The backend leverages a mix of TensorFlow and PyTorch libraries, showcasing our strategic use of both ecosystems to efficiently handle different neural network architectures.

The user interface is intuitively designed to ensure ease of use, starting with a simple welcome message that explains the application's function. Users can upload images via a streamlined drag-and-drop interface or through a standard file browsing option. Following the upload, users select from three different models— **ResNet50**, **MobileNetV2**, or **EfficientNetB0**—via a sidebar menu, which dynamically loads the appropriate model based on the selection.

Uploaded images undergo a validation process for file type and are preprocessed to conform to the input requirements of the selected model. This preprocessing includes resizing and normalization steps that are essential for maintaining consistency in image analysis.

Depending on the model chosen, the detection process varies. For MobileNetV2, images are processed using TensorFlow's image processing capabilities, whereas for ResNet50 and EfficientNet, PyTorch's advanced model evaluation techniques are employed. This allows us to utilize specific features from each framework, such as TensorFlow's efficient image handling and PyTorch's robust model manipulation.

Once the analysis is complete, the application provides immediate feedback by displaying whether an image is likely AI-generated or real. If the image is suspected to be AI-generated, a prominent alert warns the user, reinforcing the integrity of digital content.

To optimize performance, the app uses caching techniques to avoid reloading models during each session, significantly speeding up response times. Further optimizations include streamlined image preprocessing and the use of efficient prediction methods tailored to each model's specifications.

In summary, by integrating multiple Machine Learning frameworks and optimizing for performance and usability, our web application could effectively identify suspicious images and prevent user deception.

5. Model Explanation Using Grad-CAM

Despite their high classification accuracy, the CNN models we used function as "black boxes", which means that the decision-making processes of the models are non-transparent. To address this, we employed Grad-CAM (Gradient-weighted Class Activation Mapping), a visual explanation algorithm, to pinpoint which parts of an image most significantly influence model's final decision. Heatmap is used to demystify the model's operations by highlighting the image regions that are key to its predictions. To be consistent with the model implementing environment, PyTorch Grad-CAM is applied on ResNet50 and EfficientNetB0, while Tensorflow Grad-CAM is applied on MobileNetV2.

5.1. ResNet50

5.1.1. Model Decision Explanation

In theory, the heatmap for the last layer (**Layer4** for ResNet50) should display the most accurate visual explanation of the class predicted by the model. To understand how the ResNet50 model processes decisions, we initially apply the Grad-CAM heatmap method in PyTorch to the last bottleneck block of **Layer4**. However, from the results obtained, it is evident that there is no discernible pattern indicating how the model differentiates between real and AI-generated images.

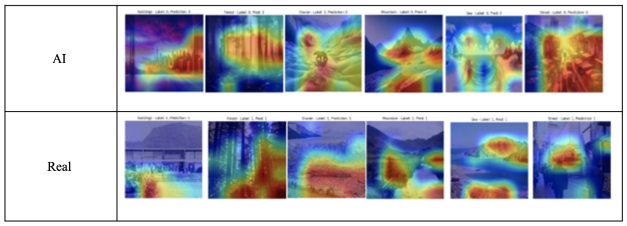


Figure 5. ResNet50: AI vs Real

While the last layer of ResNet50 model captures abstract features critical for classification, it might not offer the clearest visual explanations. Middle layers, like **Layer2** and **Layer3**, reveal finer spatial details and provide more interpretable visual patterns of specific image features. As we progress from **Layer2** to **Layer4**, the heatmaps become more focused, moving from basic structural outlines to specific areas and details. In the context of building images, **Layer2** emphasizes edges and outlines, **Layer3** focuses on specific parts like corners and windows, and **Layer4** intensely highlights unique architectural features.

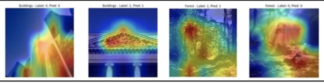
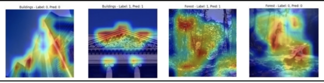
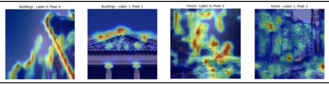
Last layer	
Layer3	
Layer2	

Figure 6. ResNet50: Layer2 vs Layer3 vs Layer4

5.1.2. Model Error Explanation

Real images wrongly predicted as AI-generated (False Negative): When compared to correctly classified images, those that are wrongly classified exhibit focus on relatively smaller areas and tend to overlook the main parts of the image. This observation suggests that inaccuracies in classification might stem from the model concentrating on less relevant features rather than the key elements of the image.

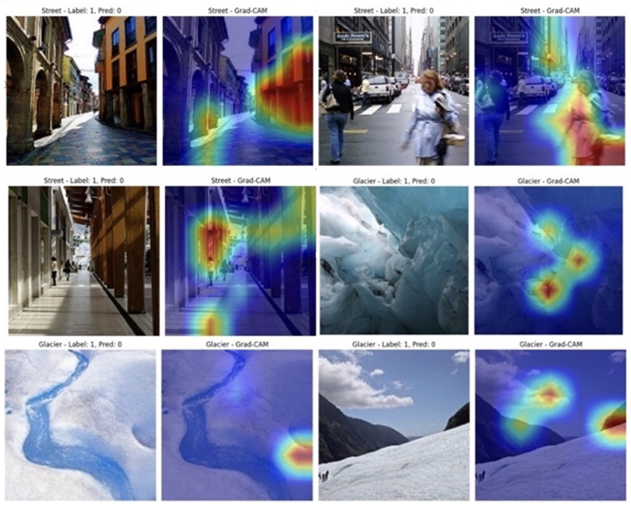

Street - Label: 1, Pred: 0
Street - Grad-CAM
Street - Label: 1, Pred: 0
Street - Grad-CAM
Street - Label: 1, Pred: 0
Street - Grad-CAM
Glacier - Label: 1, Pred: 0
Glacier - Grad-CAM
Glacier - Label: 1, Pred: 0
Glacier - Grad-CAM
Glacier - Label: 1, Pred: 0
Glacier - Grad-CAM

Figure 7. ResNet50: False Negative

AI-generated images wrongly predicted as real (False Positive): We can tell from the classification report that our model recall of class 0 is 1, which stands for outstanding model capability of identifying AI-generated images. Browsing through the whole dataset, only 4 AI-generated images are wrongly predicted as real. Using the Grad-CAM to root the cause, we can see that the model seems to focus meaninglessly on the side and corner of the image.

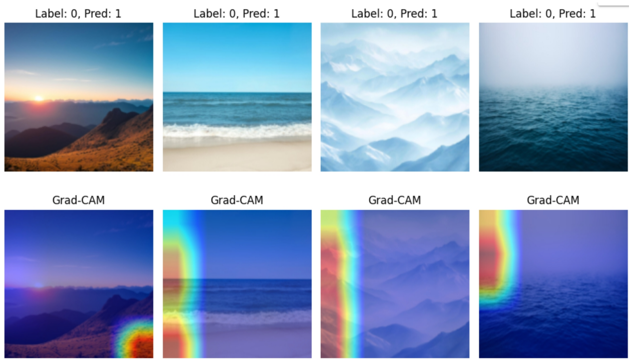


Figure 8. ResNet50: False Positive

5.2. EfficientNetB0

5.2.1. Model Decision Explanation

Comparison with ResNet50: ResNet50's heatmaps tend to be more spread out and scattered, highlighting multiple areas of the image. In contrast, EfficientNetB0 often focuses on more intense regions. Additionally, the areas of focus of two CNN models are different for the same image, indicating differences in how they identify and process key image features.

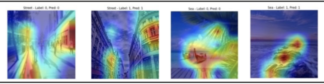
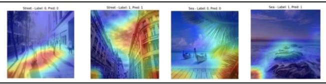
ResNet50	
EfficientNet_b0	

Figure 9. Real images that ResNet50 predicted wrongly and EfficientNetB0 classified correctly

5.2.2. Model Error Explanation

The EfficientNetB0 model outperforms the ResNet50 in accurately identifying real images. To understand why ResNet50 often misclassifies, we find out some of those real images that ResNet50 wrongly predicted but correctly classified by EfficientNetB0, and compare their heatmaps differences. Although no global rule can be extracted, this comparison highlights specific areas within individual images that help differentiate AI from real images and areas that are less effective in making this distinction.

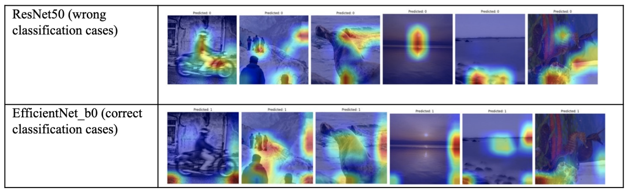

ResNet50 (wrong classification cases)
EfficientNet_b0 (correct classification cases)

Figure 10. ResNet50 vs EfficientNetB0

5.3. MobileNetV2

We also leverage the Grad-CAM technique through Tensorflow to analyze how MobileNetV2 model processes image classification task. Specifically, we compute heatmaps by tracking the gradients of the highest-impact predictions at the **"out_relu"** layer of the MobileNetV2 model, highlighting the most influential areas in the model's decision-making. Then, we overlay the heatmaps on the original images to showcase where the model focuses its attention, thus aiding in the interpretation of whether an image is AI-generated or real. This approach provides a clear visualization of the model's reasoning by accentuating the most relevant features in the image classification process.

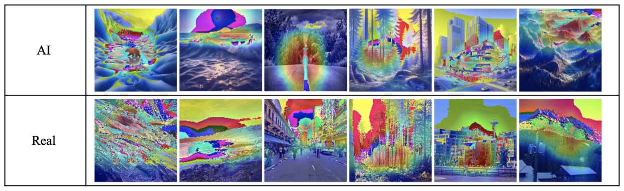


Figure 11. MobileNetV2: AI vs Real

The Grad-CAM heatmaps vividly show the contrast between AI-generated and real images: AI-generated images exhibit chaotic and complex color distributions, reflecting the synthetic complexity, whereas real images feature more organized and focused hotspots, highlighting the model's ability to directly recognize natural features. This difference is pivotal in effectively distinguishing between the two classes, offering a clear visualization of how the neural network processes and interprets varying image characteristics.

According to the evaluation of MobileNetV2, "building" exhibits the lowest F1-score and recall among six scenery types. To better understand the cause, we employ the Grad-CAM method on three layers to determine which offers the most accurate detection. Furthermore, we use the 99th percentile for heatmap normalization instead of the maximum value, providing a more robust and balanced visualization to better inform how our model makes decision. By selecting the optimal layer based on this analysis, we aim to get a better understanding why MobileNetV2 has relatively poorer performance on classifying "building".

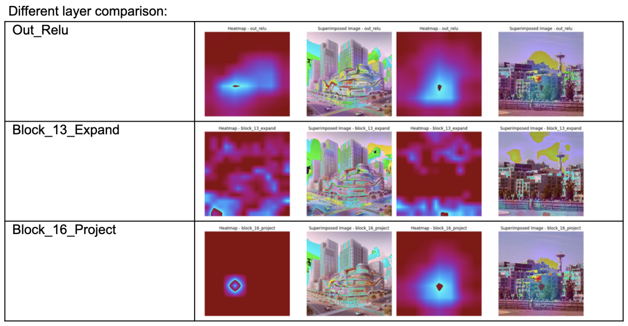


Figure 12. Out_Relu vs Block_13_Expand vs Block_16_Project

Based on the comparison of heatmaps corresponding to different layers as shown in Figure 12, we have decided to select **"Block_13_Expand"** layer because of its distinct clarity and more precise localization of relevant features compared to the other layers tested. This layer is likely more effective in providing more accurate insights about how MobileNetV2 makes decision, which is particularly useful for improving model performance in classifying "building".

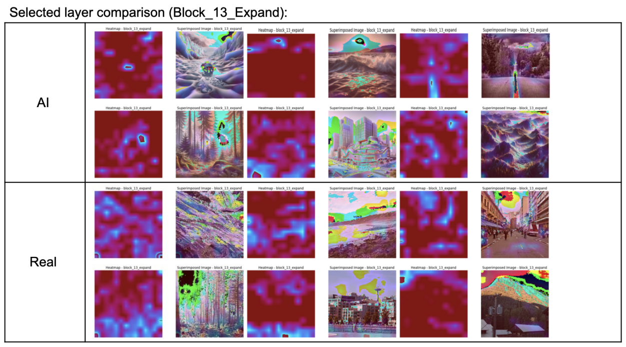


Figure 13. Block_13_Expand: AI vs Real

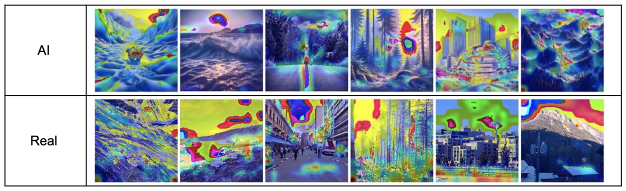


Figure 14. Grad-CAM via Block_13_Expand: AI vs Real

Based on the visual comparison shown in Figure 14, applying heatmap on the **“Block_13_Expand”** layer has enhanced the distinction between AI-generated and real scenery images. This improvement in Explainable AI is marked by more refined and complex activation patterns in both AI-generated and real images. AI-generated images now feature richer textures and gradients, closely mimicking the complexity of real scenes, while real images display clearer and more accurate natural elements. These enhancements suggest MobileNetV2’s ability to discern finer textural details, crucial for accurately differentiating between AI-generated and authentic scenery. However, **“Block_13_Expand”** layer also reveals that our model struggles with the accurate identification of buildings due to its inability to clearly delineate architectural contours and details, which may lead to lower classification accuracy for “building” scenes.

6. Conclusion

6.1. Achievement of Goals

Our project has developed an end-to-end web application proficient in discerning AI-generated images from authentic photographs with exceptional accuracy. Integrated with advanced machine learning models—ResNet50, MobileNetV2, and EfficientNetB0—the tool surpasses industry benchmarks like C2PA.

Utilizing Grad-CAM for explainable AI has enhanced transparency in classification decisions. While ResNet50 and EfficientNetB0 showed no discernible patterns, MobileNetV2 revealed differences: AI-generated images exhibited chaotic color patterns, while photos captured by humans displayed more organized hotspots, underscoring likely model’s recognition of natural or nonsynetic features. Grad-CAM also aids in explaining individual image decisions and identifying potential model errors by highlighting prediction-focused areas, especially useful in complex scenarios like building scenes.

Successful deployment of this project holds promise for beneficial social impact, bolstering digital trust and reducing misinformation by accurately identifying and labeling AI-generated content.

6.2. Encountered Difficulties

Throughout the project, we encountered several challenges primarily related to technical aspects of model implementation. Processing power and memory usage posed significant constraints, especially with the demanding computational requirements of sophisticated deep learning models. This challenge became apparent when handling large datasets and conducting extensive training sessions, necessitating high-performance GPUs. To address these issues, we optimized our models and selected a manageable dataset size, striking a balance between performance and available resources.

Applying Explainable AI presented a key challenge in discerning globally applicable rules for model decision-making. Since our objective is to train models to distinguish between AI-generated and human-photographed images, interpreting model decision-making processes from heatmaps—more commonly used for object detection tasks—proved challenging. To overcome this obstacle, we compared heatmaps across layers and models to gain insights into their respective contributions.

6.3. Limitation

One critical deficiency in the modeling session was the lack of coordination in data transformation methods. Specifically, MobileNetV2 and EfficientNetB0 underwent data normalization, while ResNet50 did not. Conversely, ResNet50 underwent data augmentation, whereas the other two models did not. This inconsistency renders the detection results incomparable. However, the varied approaches also provided valuable experimentation opportunities, as all three models achieved high accuracy.

Additionally, the use of two different deep learning frameworks, PyTorch for ResNet50 and EfficientNetB0 and TensorFlow for MobileNetV2, posed challenges in explainable AI and model integration. Despite these challenges, navigating these diverse frameworks provided our team with unique experiences into experimentation with different platforms.

The key limitation of applying Explainable AI is the subjectivity of qualitative interpretations, due to the less intuitive nature of our task compared to object detection tasks. This leads to a lack of robust and generalizable rules for explaining model’s decision, which can be applied to photographs other than landscapes.

Furthermore, the lower resolution of images captured by humans compared to AI-generated ones, resulting in blurred representations in Explainable AI analysis. Improving data quality is essential to enhance xAI effectiveness.

6.4. Future Improvements

Looking ahead, there are several areas where our project can be improved:

- **Enhancing Model Generalization:** Expanding our dataset to include a wider variety of image types and incorporating more diverse AI-generated images will improve the models' ability to generalize across different contexts and reduce bias.
- **Upgrading Computational Resources:** Overcoming hardware constraints by seeking additional funding or partnerships for access to more advanced computational resources will enable more extensive training and handling of larger datasets.
- **Incorporating More Advanced AI Tools:** Harmonizing frameworks like PyTorch and TensorFlow and deploying a hybrid approach that combines different models can enhance performance and xAI capabilities. Tailoring strategies to specific scene types will boost accuracy and provide a more nuanced understanding through xAI, improving our ability to explain the model's decision-making processes.
- **Enhancing User Experience:** Further enhancing the user interface and experience to make it more intuitive and responsive is a priority. Feedback loops from users will be essential for refining the app's functionality and usability.

By continuing to refine and enhance our approach, we aim to maintain the relevance and effectiveness of our tool in the evolving landscape of digital media authenticity. Our commitment remains steadfast in supporting the integrity of content creators' work and ensuring the trustworthiness of digital imagery in the marketplace.

7. Reference

Rastogi, A. (2022, March 19). ResNet50 - Dev Genius. *Medium.* https://blog.devgenius.io/resnet50-6b42934db431

Reiff, D. (2022, May 12). *Understand your Algorithm with Grad-CAM - Towards Data Science.* Medium. https://towardsdatascience.com/understand-your-algorithm-with-grad-cam-d3b62fce353

ResNet-50: the basics and a quick tutorial. (2023, May 22). Datagen. https://datagen.tech/guides/computer-vision/resnet-50/

8. Codebase and Data

Github Link: *https://github.com/onlymeTerri/ai-artwork-detector*