

Capstone Project
Detecting AI-Generated Images

Yaqin Albirawi

Artificial Intelligence and Machine Learning, Fanshawe College

INFO-6147: Deep Learning with Pytorch

Dr. Mohammed Yousefhussien

August 3, 2025

Abstract

This project explores the use of deep learning for detecting AI-generated images using the CIFAKE dataset from Kaggle by training a binary image classifier using ResNet-18. The model was trained from random weights and used Binary Cross-Entropy, early stopping, and data augmentation techniques. Hyperparameter tuning was performed to improve model performance. The model achieved 94.87% accuracy on the test set, along with strong precision and recall scores. Grad-CAM visualizations were used to interpret model decisions, revealing distinct attention patterns between real and fake images. While the model performs well within the dataset's domain, it may struggle to generalize to other higher-resolution images or fake content generated by generative tools other than Stable Diffusion, highlighting the need for broader data diversity in future work.

Introduction

As generative AI models like Stable Diffusion, MidJourney, and DALL-E continue to improve, distinguishing between real and AI-generated images has become increasingly difficult. While humans may still rely on visual cues like blurred textures, distorted elements, or lighting inconsistencies, these flaws are becoming harder to detect with the human eye.

In contrast, machine learning models can detect subtle patterns, such as unnatural textures and edge artifacts that are often invisible to the human eye. This project explores the use of a ResNet-18 convolutional neural network for binary image classification and aims to determine whether a deep learning model can effectively learn and distinguish between real and AI-generated images.

Dataset

The dataset used to train the model is the “CIFAKE: Real and AI-Generated Synthetic Images” dataset shared by Jordan J. Bird on Kaggle. The dataset is balanced and contains 120,000 32x32 pixel RGB images, split into 60,000 real images extracted from CIFAR-10 and 60,000 fake images. The dataset folder is split into a Test folder containing 20,000 images and Train folder containing 100,000 images, each split into 50% real and 50% fake.

The fake images were generated using Stable Diffusion. Stable Diffusion is a generative model guided by text prompts that processes images in the compressed latent space instead of raw pixel space. The model produces a variety of high-resolution images. However, to maintain a controlled environment that is consistent with CIFAR-10 and allow the model to learn the domain-specific features necessary for this problem, the images generated were downsampled to 32x32 pixels and conditioned on CIFAR-10 classes.

Data Preprocessing

To ensure compatibility with the ResNet-18 architecture and improve model generalization, a series of transformations were performed:

- Images were resized to 224×224 pixels. This is required because ResNet-18 expects inputs of this shape.
- Data augmentation using random horizontal flips was applied during training to introduce variability by flipping images horizontally with a probability of 0.5.
- Images were converted from PIL format to PyTorch tensors, and pixel values were scaled from [0, 255] to [0.0, 1.0].
- The pixel values were then standardized using the mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225], which match the statistics of the ImageNet dataset that the ResNet-18 was originally trained on.

To reduce computational time, 37,500 images were used for training and evaluation purposes. An 80/20 split was used for training/testing. The training set was further divided into 80/20 split for training/validation purposes.

Model Architecture

ResNet-18 was selected as the base architecture since it is a relatively lightweight convolutional neural network that is efficient and performs well in computer vision tasks. It has proven to be able to learn a wide variety of image features, such as edges and textures, that will be useful to identify fake images.

Given CIFAKE images are only 32x32 in size and consist of synthetic images, training from random weights will ensure the model learns patterns specific to this dataset and this classification problem. Training from scratch allows the network to learn features from the ground up, which can be particularly valuable when dealing with dissimilar content like AI-generated images.

Loss Function

For this binary classification task, the Binary Cross-Entropy with Logits Loss (BCEWithLogitsLoss) function was selected. BCE is appropriate for binary classifications as it measures the distance between the predicted probabilities and the ground truth (actual) class labels, penalizing confident but incorrect predictions more.

This particular function applies a sigmoid activation to the model's raw output (logits) before computing the BCE loss. This bypasses the need to apply a separate sigmoid activation, which converts the logits to a probability between 0 and 1, enabling classification into "fake" if closer to 0 or "real" if closer to 1.

Optimizer

The model was optimized using the Adam optimizer, known for its efficient convergence capabilities. Different values for the learning rate were explored during hyperparameter tuning and a value of 0.0005 was found to be optimal.

Regularization

Early stopping with a patience of 3 epochs was applied to prevent overfitting. Training was terminated if the validation loss did not improve within three epochs (or a maximum of 20 epochs), and the model with the lowest validation loss was then retained as the final model. A patience of 3 was observed to allow the model to recover from temporary fluctuations, while still avoiding prolonged training time that led to overfitting or non-significant improvements, making it a reasonable and efficient choice.

Moreover, the horizontal flipping data augmentation technique explained earlier in the preprocessing section was used during training. This technique increases the diversity of the training dataset by randomly flipping each image at load time, encouraging the model to generalize better to unseen data.

Batch Size

Multiple batch size values were explored during hyperparameter tuning. A batch size of 16 was found to be optimal with respect to convergence trends as seen in the next section.

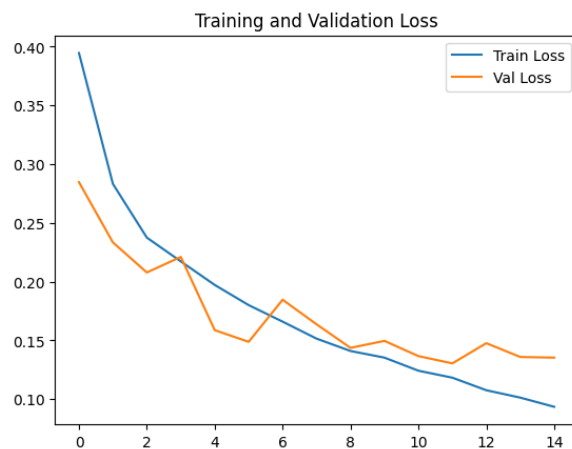
Hyperparameter Tuning

Manual hyperparameter tuning was performed on the learning rate and batch size to improve model performance. The following table summarizes the combination of hyperparameter values explored and their impact on training and validation performance.

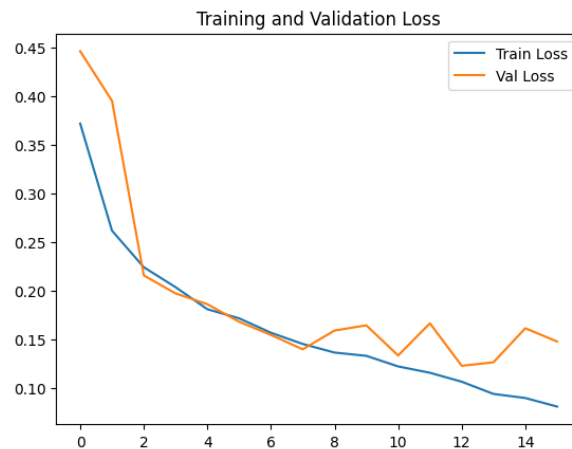
| (Lr, Batch Size) | Training | | Validation | |
|---------------------|---------------|---------------|---------------|---------------|
| | Loss | Accuracy | Loss | Accuracy |
| (0.001, 16) | 0.1798 | 93.01% | 0.1657 | 93.37% |
| (0.001, 32) | 0.1449 | 94.22% | 0.1546 | 93.90% |
| (0.0005, 16) | 0.1183 | 95.27% | 0.1305 | 95.25% |
| (0.0005, 32) | 0.1066 | 96.00% | 0.1229 | 95.03% |
| (0.0001, 16) | 0.0882 | 96.53% | 0.1601 | 94.23% |
| (0.0001, 32) | 0.1496 | 94.06% | 0.1767 | 92.70% |

The combination of 0.0001 learning rate and batch size of 16 resulted in the lowest training loss, but also showed a clear sign of overfitting. Whereas a learning rate of 0.0005 resulted in both low training and low validation loss and was chosen as the best learning rate value. Given the marginal difference in performance between the (0.0005, 16) and (0.0005, 32) settings, convergence behavior was examined to select the better-performing configuration.

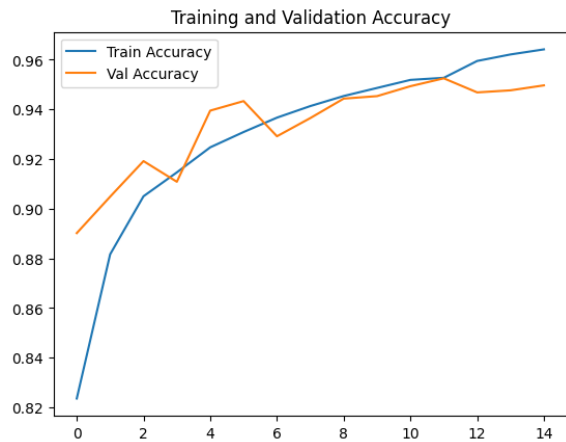
As seen in the plots below, the convergence behavior of batch size 16 model was more stable, with validation closely following training and showing fewer fluctuations. This suggests better generalization and more reliable early stopping compared to the second model, where validation loss showed more variance and potential overfitting signs or longer time to converge. Thus, despite the slightly lower validation loss that batch size 32 model portrayed, the (0.0005, 16) model proves to perform better through more stable convergence trends.



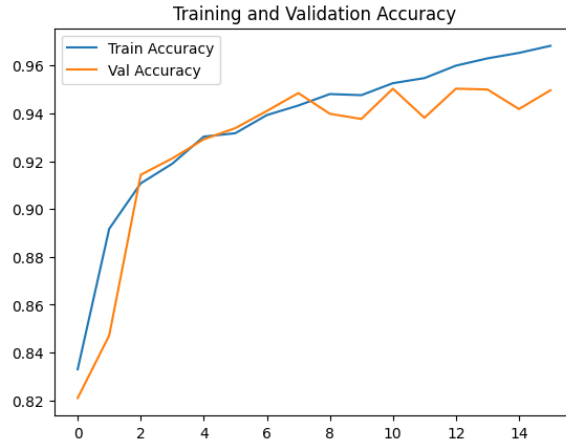
Plot 1: Loss plot for (0.0005, 16)



Plot 2: Loss plot for (0.0005, 32)



Plot 3: Accuracy plot for (0.0005, 16)



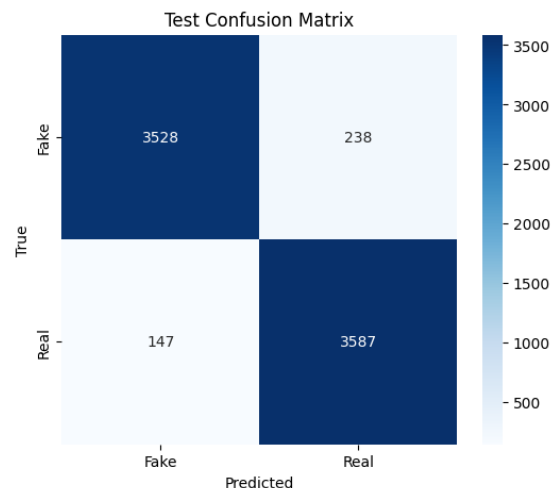
Plot 4: Accuracy plot for (0.0005, 32)

Evaluation Results

The preferred model was then evaluated on the test set using accuracy, precision, recall, and F1-score. The classifier showed strong performance in distinguishing between real and fake samples in the test dataset. Note that high precision (i.e., minimizing false positives) and high recall (i.e., minimizing false negatives) are both important for the problem of detecting AI-generated content. High precision is important in applications where trusting fake content poses a serious risk, such as failing to detect misinformation. While high recall protects real content from being wrongfully discredited. The F1-Score is a balanced overview of the two and is shown in the table below.

| Metric | Accuracy | Precision | Recall | F1-Score |
|--------|----------|-----------|--------|----------|
| Result | 94.87% | 93.78% | 96.06% | 94.91% |

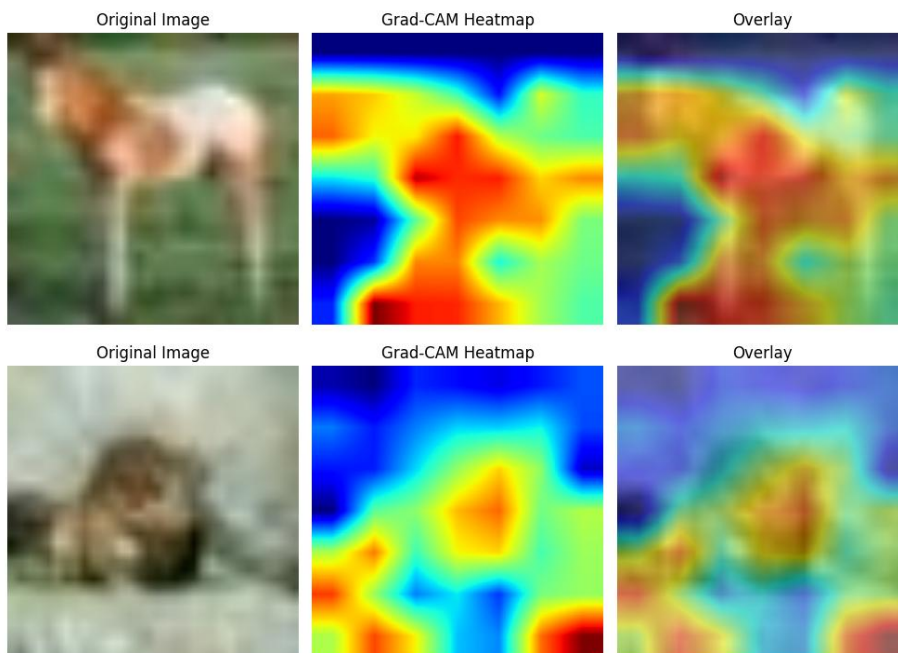
The confusion matrix below further summarizes the performance on the test set, showing how many predictions were correct or incorrect, and what kinds of errors were made.



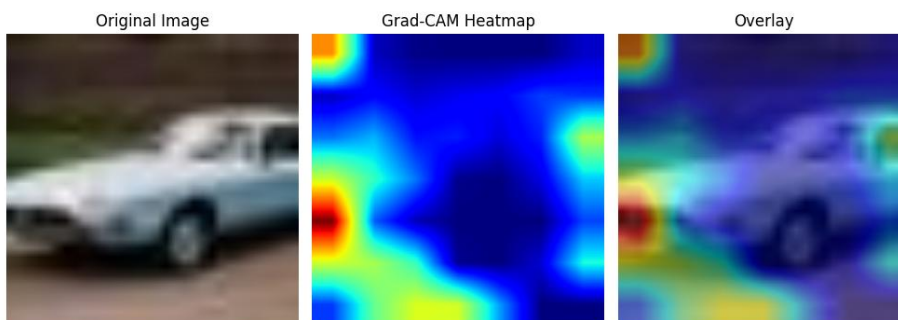
Interpretability using Grad-CAM

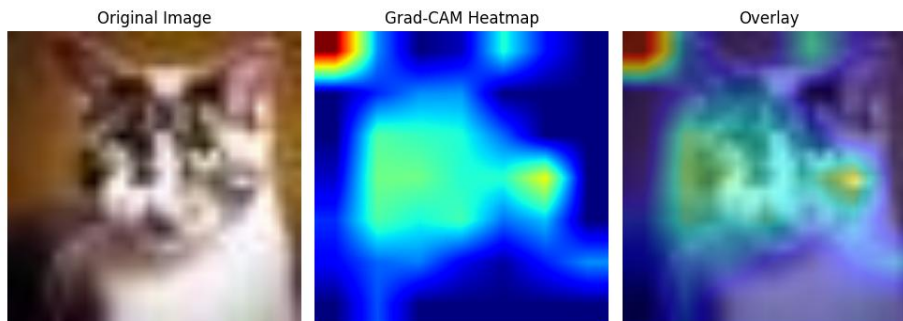
To interpret the model's decisions, Grad-CAM visualizations were generated. Grad-CAM is used to visually interpret deep learning models by highlighting which parts of an image the model focused on when making a prediction. Regions highlighted in red represent the most salient features that guided the model's decision, whereas blue areas had minimal impact on the output.

The Grad-CAM heatmaps revealed notable differences in how the model interprets real versus fake images. The model typically focused on broader regions of real images, including the main object itself. This is reflected in the widespread red areas shown in the real images below.




In contrast, for fake images, the model often concentrated on small regions, typically in the background and outer edges rather than on the main object. These areas likely contain texture inconsistencies or blurring, which the model has learned to associate with fake images.





Comparison Against ChatGPT

To compare the model performance against ChatGPT, multiple independent chat sessions were conducted using a sample set of 20 images, 10 real and 10 fake, manually presented in random order. In some sessions, ChatGPT labeled all images as AI-generated, while in others, it labeled all as real. Although a few sessions included a mix of predictions, ChatGPT frequently misclassified images, as seen with the session sample results below. This highlights the limitations of ChatGPT on detecting authenticity of low-resolution images.

| Image |  |  |  |  |  |  |  | ... |  |  |  |  |  | ... |  |
|---------|---|---|---|---|---|---|---|-----|--|---|---|---|---|-----|---|
| Actual | Fake | Fake | Real | Fake | Real | Fake | Real | ... | Fake | Fake | Real | Real | Real | ... | Real |
| ChatGPT | Real | Real | Real | Real | Real | Real | Real | ... | Real | Real | Fake | Real | Fake | ... | Fake |

The screenshot of the chat above can be viewed in the folder here:

<https://github.com/yalbirawi/Detecting-AI-Generated-Images/tree/main/assets>

Limitations and Improvements

One limitation of the current model is that it was trained on low-resolution images, which may limit its ability to generalize to higher-resolution content seen in real-world cases. Moreover, the fake images in the dataset were generated using Stable Diffusion so the model may not perform well on fake images created by other generative tools such as DALL-E or MidJourney.

To improve generalization, future work should incorporate fake images from a diverse range of generative models and include training on higher-resolution datasets.

Conclusion

This project demonstrated the use of deep learning for detecting AI-generated images. With proper preprocessing and hyperparameter tuning, the model achieved strong performance (94.87% accuracy) on images generated by Stable Diffusion. Grad-CAM visualizations showed that the model focused on meaningful features, distinguishing real from fake images using subtle visual cues. However, its reliance on low-resolution data and a single generative model source limits generalization to real-world scenarios. Future work should explore more diverse datasets and resolutions. Overall, the results highlight deep learning's potential in combating misinformation and detecting fake content.

References

Github Page: <https://github.com/yalbirawi/Detecting-AI-Generated-Images/tree/main>

CIFAKE Dataset on Kaggle: <https://www.kaggle.com/datasets/birdy654/cifake-real-and-ai-generated-synthetic-images>

CIFAKE: Image Classification and Explainable Identification of AI-Generated Synthetic Images: https://www.researchgate.net/publication/377538637_CIFAKE_Image_Classification_and_Explainable_Identification_of_AI-Generated_Synthetic_Images

Grad-CAM Adaptation: <https://github.com/jacobgil/pytorch-grad-cam>