# AI vs Genuine Image Detection Report

Chris Crisostomo
San Diego State University
Department of Science
Chula Vista, CA
ccrisostomo6457@sdsu.edu

Ryan Renales
San Diego State University
Department of Science
Chula Vista, CA
rrenales1971@sdsu.edu

Kalani Tran
San Diego State University
Department of Science
San Diego, CA
kalanikekaitran@gmail.com

Kevin Townsend
San Diego State University
Department of Science
San Diego, CA
ktownsend5676@sdsu.edu

***Abstract*—The advanced sophistication of artificial intelligence (AI) has allowed for increasingly realistic AI-generated images to populate the media we consume. This facet of modernization makes it difficult to ascertain whether an image was produced naturally or whether it was created by AI, bringing up concerns around deep fakes and digital misinformation. The following report details a novel machine learning model that can classify images as "real" or "AI-generated". An online, public dataset from Kaggle consisting of real images from Shutterstock and AI-generated images from DeepMedia was used to train the model, in which pre-labeled images depicting people, objects, and scenery were subjected to this supervised binary classification. The methodology was performed with a convolutional neural network (CNN) using TensorFlow, achieving accuracy >90% at the end of specified epochs and with varying batch sizes. The CNN proved effective on the dataset, and is an optimistic outlook for larger implications requiring AI image detection in the real world.**

***Index Terms*—artificial intelligence, AI, convolutional neural network, CNN, machine learning, ML, training, model, test**

## I. INTRODUCTION AND RESEARCH PROBLEM

Modern advancements in technology and the growth of artificial intelligence have introduced mass fake news, videos, and images on social media and broadcast outlets. With the rise of AI-generated photographs, comparing the differences between forged and genuine pictures may be complex. This paper details an AI-generated image detection algorithm to distinguish authentic and AI-generated images to help identify false images correlated with fake news on media platforms. The importance of this AI-generated image detection algorithm will aid in preventing false beliefs or influences from the internet.

The represented dataset consists of images from Shutterstock and DeepMedia. Shutterstock supplies genuine images from real photographers while DeepMedia yields AI-generated images using modern generative models. One-third of this dataset contains humans, while the latter depicts objects and places.

## II. RELATED WORK

With the rapid increase of artificially generated content, there have been many studies to determine

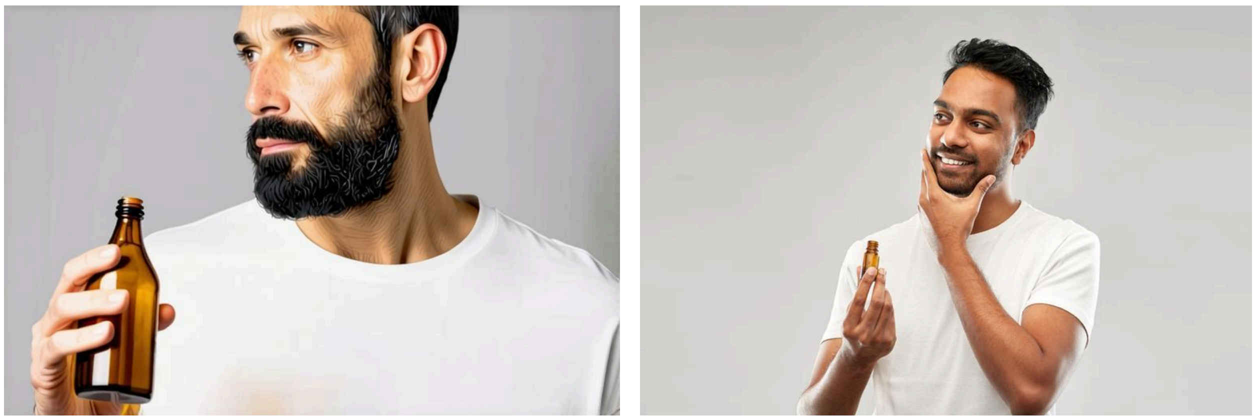

Fig. 1. Image representation of an AI-generated image (left) versus an image taken by humans (right). Upon closer inspection, the AI-generated image of a human has uncanny skin wrinkles and a missing index finger.

whether people can differentiate between what's real or not. Our goal in this research problem was to use a machine learning model that can detect AI-generated images, and it was important for us to understand previous work in this field. We discovered a research paper conducted by individuals at the atlanTTic Research Center for Telecommunication Technologies, University of Vigo in Spain. The paper titled "Detection of AI-Created Images Using Pixel-Wise Feature Extraction and Convolutional Neural Networks" has results that display the high accuracy of machine learning models. Their approach to this research problem was to use two different pixel-wise extraction features alongside the convolutional neural network to distinguish between images taken from a camera and AI-generated images. One of the extraction features they used was Photo Response Non Uniformity (PRNU), which is a noise that can identify a camera source. This noise is unique to images from a camera, meaning there should be no AI-generated images with PRNU. However, some applications can forge or change PRNU, so they trained their CNN to notice special characteristics of AI PRNU patterns. The other extraction feature they implemented was error level analysis (ELA). ELA can detect when pixels in an image have been modified. This makes ELA a good extraction feature for AI image detection because all of the pixels appear as modified. The results of their research showed that PRNU and ELA are both great extraction features for the detection of AI-generated images. With ELA having an accuracy of 98%, and PRNU right after it with 95%. What our team was able to learn from this study is that we should proceed with using CNN for our evaluation model for accurate AI image detection.

## III. METHODOLOGY AND TECHNICAL DETAILS

The AI-generated image detection algorithm uses a high-level programming language, Python, and numerous dependencies. These required libraries can be installed with Pip Installs Packages, or PIP. Using PIP allows the user to download and install the needed dependencies, such as Numpy, TensorFlow, Matplotlib, and Kagglehub.

The Numpy library, alias 'np', is used in this project for array manipulation, floating-point numbers, and expanding dimensions. The TensorFlow module was imported to scale pixel values and augment images during model training. Built using the Keras API, the binary sequential convolutional neural network (CNN) model is compiled with an optimizer, loss function, and evaluation metrics. TensorFlow handles the evaluation process using the trained model. Matplotlib is used to visualize the training performance of the CNN model. The plots created with Matplotlib show how well the model performs on the data it was trained on, and how well the model performs on unseen data. The KaggleHub Python package is an API used to load the database hosted on Kaggle.com.

Our notebook implements a binary image classifier using a CNN to differentiate label 1 (AI-generated images) and label 2 (authentic images). After importing the dependencies, the train and test CSV files are loaded. During preprocessing, each image is converted to 128 by 128 RGB and has normalized pixel values. The TensorFlow input pipeline includes mapping, batching, shuffling, and prefetching. For our CNN architecture, the discriminator model implements Conv2D layers to extract features from the image, LeakyReLU, dropout to reduce overfitting, and the flatten and dense functions to convert features to binary classification. The model is then saved and has an interface to output a label if the prediction probability is above 0.5.

During the model initialization, the model is built using 'tf.keras.Sequential'. The compilation process uses the Adam optimizer to adjust learning rates to optimize performance, the 'binary_crossentropy' loss function for binary classification, and accuracy metrics to track performance. After the data pipeline is initialized, the model starts to train. Each of the 10 epochs goes through a forward pass, outputs a prediction with a value between 0 and 1, calculates binary cross-entropy loss and compares predictions with true labels, goes through backpropagation, and updates the accuracy.

## IV. EXPERIMENTAL RESULTS AND ANALYSIS

Before experimenting with the discriminator model, the first step is to train the model on a training set. As mentioned before, the model was trained on 10 epochs. The model improved and progressed as such:

**Epoch 1/10**
858s 249ms/step - accuracy: 0.8047 - loss: 0.4293
**Epoch 2/10**
866s 248ms/step - accuracy: 0.8869 - loss: 0.2939
**Epoch 3/10**
872s 247ms/step - accuracy: 0.8919 - loss: 0.3114
**Epoch 4/10**
878s 252ms/step - accuracy: 0.9041 - loss: 0.2866
**Epoch 5/10**
872s 249ms/step - accuracy: 0.9071 - loss: 0.3101
**Epoch 6/10**
875s 250ms/step - accuracy: 0.9146 - loss: 0.3002
**Epoch 7/10**
865s 248ms/step - accuracy: 0.9085 - loss: 0.3643
**Epoch 8/10**
869s 251ms/step - accuracy: 0.9159 - loss: 0.3551
**Epoch 9/10**
863s 245ms/step - accuracy: 0.9203 - loss: 0.3542
**Epoch 10/10**
785s 214ms/step - accuracy: 0.9235 - loss: 0.3580

The model's accuracy began at 80.47%, and after 10 epochs, the model's accuracy improved to 92.35%.

The first key experiment after training the discriminator model was a brief validation test. This test was conducted by inputting images into the model, for which we had expected predictions for each. The first part included a brief list alternating AI images and real images. The validation test produced 100% accuracy for the list of 20. For the second part of the validation test, the model was tested on the training data in various batches. The batch sizes ranged from 10 images to the entire dataset of nearly 80,000 images. The first batch of 10 images produced an accuracy of 100% (Fig. 2). The second batch of 100 images produced an accuracy of 98% (Fig. 3). The third batch of 1000 images produced an accuracy of 96.3% (Fig. 4). The Fourth batch of 10,000 images produced an accuracy of 95.82% (Fig.5). The last batch of all images in the training set produced an accuracy of 95.98% (Fig.6). As you can see, the accuracy of the entire training dataset shows improved performance. This is because the model testing function performed predictions on individual images, and the training function was trained in batch sizes of 32. Nonetheless, the model performed significantly well, as validated by the tests.

As proved by the validation tests, the model performs with roughly a 90% - 95% confidence. With a larger training dataset, or an increased amount of epochs, the model may gain a much higher confidence level. With the tests concluding the model's accuracy, the last function in our notebook tests the model on random images with no expected label (see Fig. 7 for results).

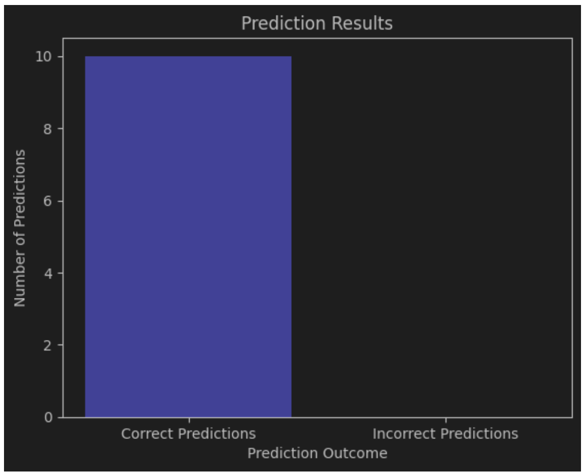

Fig 2. Prediction results of 10 images.

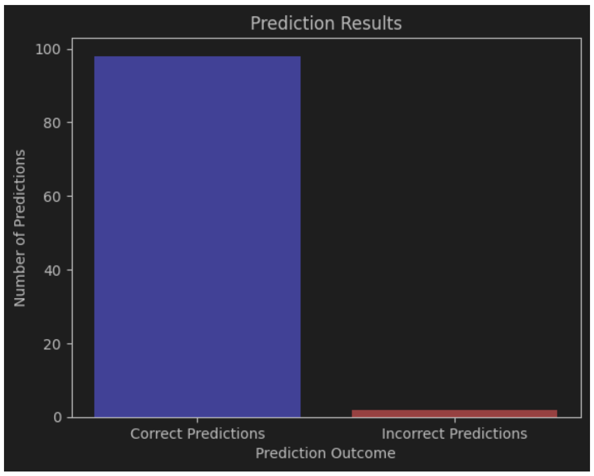

Fig 3. Prediction results of 100 images.

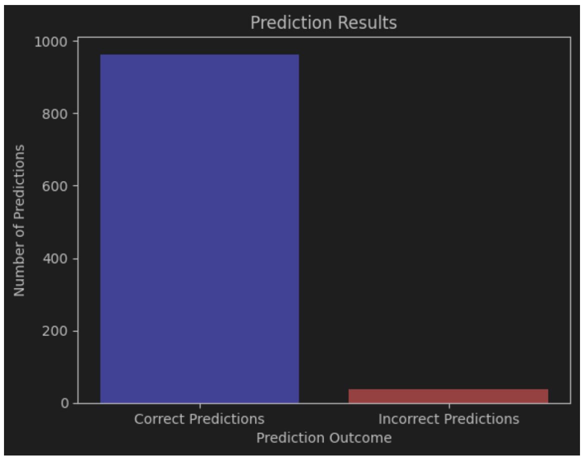

Fig 4. Prediction results of 1,000 images.

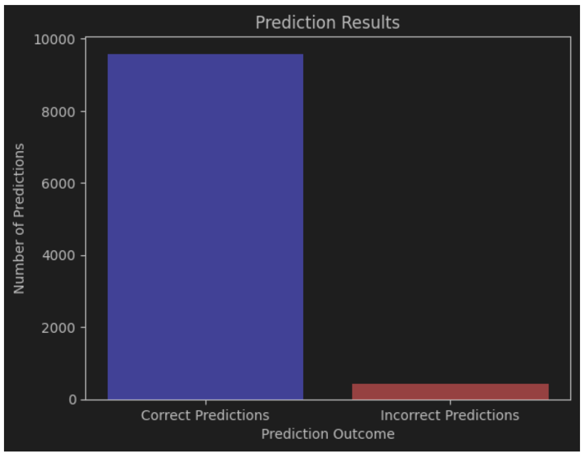

Fig 5. Prediction results of 10,000 images.

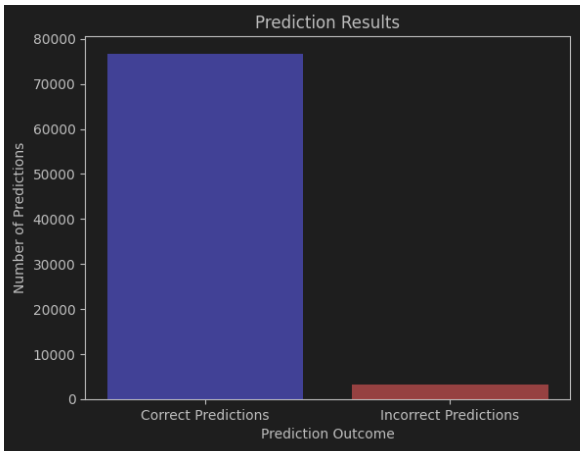

Fig 6. Prediction results of all 80,000 images.

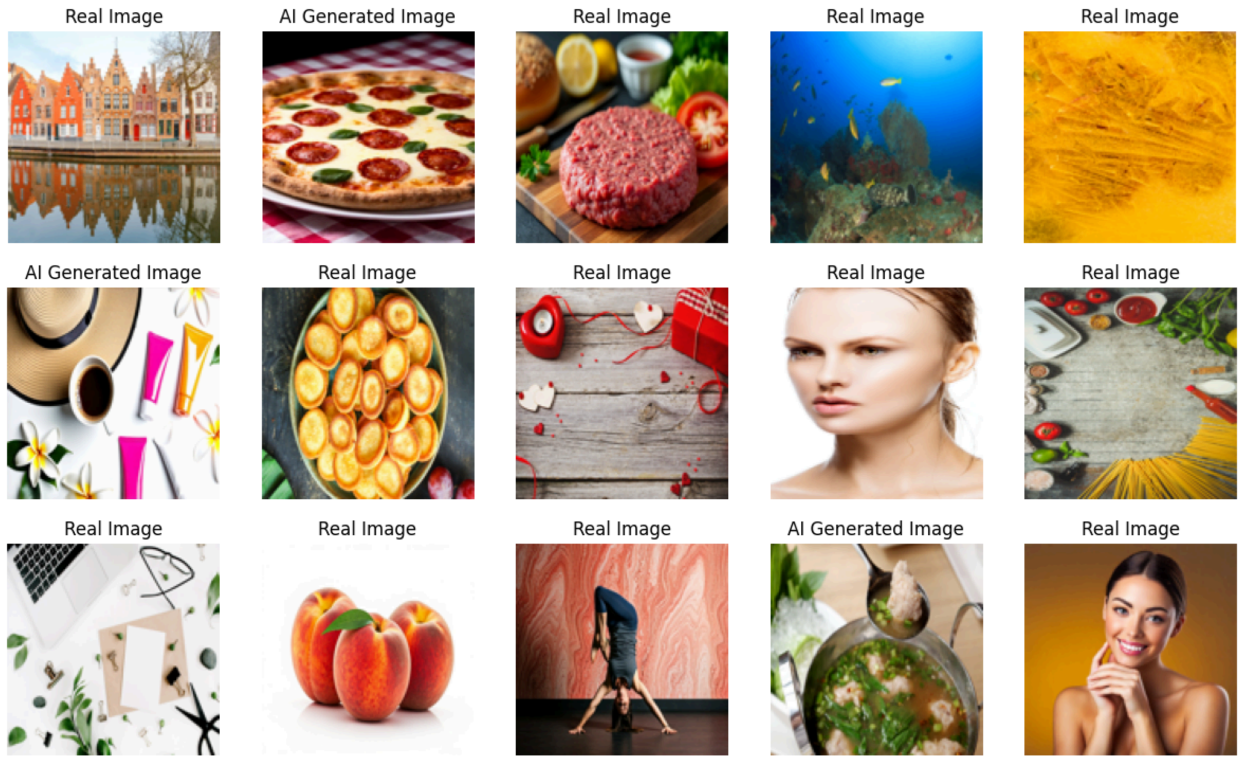

Fig 7. Sample output of the trained model on a random input image.

## V. CONCLUSION AND FUTURE WORK

### *A. Future Work*

The results of the algorithm lead to approximately 95% accuracy of the image classification. The next steps in the project would be to improve this percentage to be closer to 100%, which will require slight changes to the methodology. This can be accomplished by using more epochs when training the algorithm, so the model can learn more efficiently and make accurate predictions. However, with the increase in epochs, the dataset has to be large and varied enough to reduce overfitting so the algorithm can be useful across various datasets. Going further beyond academic purposes, this model can be expanded to be used as a real-world interface in various fields such as:

1. Social Media - Determine whether a post uses AI-generated images or not
2. News Outlets - Reduce the likelihood of fake news and stories by detecting real images from AI
3. Legal System - Ensure that all evidence presented is real and was not generated by AI

### *B. Conclusion*

The exploration of this research problem led to the development of a machine learning model capable of performing binary classification between AI-generated and real images. The “AI vs. Human-Generated Images” dataset from Kaggle was ideal for the model to train and test due to the amount and quality of the images included. Through the utilization of Python and its various dependencies, the convolutional neural network implemented was able to handle overfitting and properly calculate loss to make the predictions accurate on both seen and unseen data. As more AI-generated content appears across the internet, models such as this one will be pivotal for discerning what is real or not.

## VI. CONTRIBUTIONS OF EACH GROUP MEMBER

### *A. Software Engineer / Codebase Maintainer*

For our project, Chris Crisostomo was the main software engineer as well as the codebase maintainer. Chris possesses a strong background in software development through his Software Engineering and other upper-division CS classes, where he has completed and helmed similar coding projects. Chris was responsible for creating and maintaining the GitHub repository, as well as coaching all other team members on proper GitHub etiquette. This role also encompassed ensuring there was proper version control of our model, as well as coordinating reliable integration across all group members' environments. The software engineer also corroborated with the team to follow a consistent structure and documentation style, which was commonly subject to disparities due to the different online resources procured for this project.

### *B. Model Trainer / ML Engineer*

Amidst preparation for this assignment, Ryan Renales was the most capable for this role as he had previously finished a Computer Vision course that heavily involved many of the ML topics we implemented for this project. Ryan used ML tools like TensorFlow and Numpy to implement an effective convolutional neural network (CNN). Ryan was adept at optimizing the model's performance by tuning the appropriate hyperparameters and cost functions within the architecture. As our model trainer, Ryan ran multiple trainings of the model as well as validation experiments using 10 epochs. This role also required helping formulate the discriminator model and evaluating all other relevant metrics for tracking the performance of the model.

### *C. Data Curator / Analyst*

Kevin Townsend was a choice fit for this role as he possessed an extensive background through his Database Systems and Data Science courses, wherein his skills with Python were extremely invaluable towards the project. Kevin was able to productively analyze data with high attention to detail, aiding the ML engineer in using Pandas and NumPy to clean and visualize the dataset. Kevin was responsible for researching and proposing the selected dataset after our group decided on a topic, in which Kevin ensured the dataset was suitable for binary classification tasks. The data curator also checked the images themself to establish compatibility with the model pipeline. This role also required skills such as exploratory data analysis (EDA) to grasp topics related to ML, such as class distribution and other dataset characteristics.

### *D. Evaluation Lead / Report Analyst*

Kalani was selected as the evaluation lead and the report analyst for this project, as his previous Data Science and Algorithms courses granted him critical skills related to technical writing with a CS perspective. Kalani was able to summarize the results of the model and effectively communicate with the group to derive notable findings and identify errors that required fixing. Kalani has experience in other research-based group projects in which he applied this knowledge to help critique and present model behavior for this assignment. The report analyst designed the performance evaluation pipeline according to the tenets of accuracy, precision, and recall. This role also required knowledge of Matplotlib to create and detail helpful visualizations. Kalani also helped the analyst to interpret the statistical analysis of false positives/negatives that the model outputted.

## REFERENCES

[1] F. Martin-Rodriguez, and M. Fernandez-Barciela. "Detection of AI-Created Images Using Pixel-Wise Feature Extraction and Convolutional Neural Networks." *National Institutes of Health (NIH)*, 8 Nov. 2023, https://pmc.ncbi.nlm.nih.gov/articles/PMC10674908/. Accessed 2025.

[2] A. Sala, "AI vs. Human-Generated Images," *Kaggle.com*, 2025. https://www.kaggle.com/datasets/alessandrasala79/ai-vs-human-generated-dataset/data (accessed May 10, 2025).