Capstone Project Report

James Cook University Cairns

Hunter Kruger-Ilingworth (14198489)

August 15, 2025

Contents

List of Figures

List of Tables

Introduction

"AI as a creative tool has improved leaps and bounds since its early incarnations. Generating still images that are interesting, sophisticated and photorealistic is now an easy process that can be done by anybody with an interest, some patience and determination" [2]. The prevalence of AI among the general public has led to extensive dialogue about their ethical implications in producing artwork, or for manufacturing misinformation. One such example of the use of AI is tools like *DALL-E* and *Stable Diffusion*, which generate images from text prompts. In the current digital age, we cannot rely on the transparency of the source of an image, so there exists a need to be able to identify whether an image has been generated by an AI, or a real artist. Figure 1 shows some examples of AI generated images, which are becoming increasingly difficult to distinguish from real images as the technology improves, underscoring the need for reliable detection methods.

Fig. 1. AI generated images [1]

[1] evaluates both human and AI capabilities in detecting fake images, showing that humans are often deceived by advanced image generation models, while AI-based detection algorithms outperform humans but still misclassify 13% of images. The study introduction of the Fake2M dataset and new benchmarks (HPBench and MPBench) aims to drive further research and improve the reliability of AI-generated content detection.

[3] presents a computer vision approach to distinguishing AI-generated images from real ones, utilising a synthetic dataset created with Latent Diffusion, classification via Convolutional Neural Networks, and interpretability through Grad-CAM. Achieving nearly 93% accuracy, the study also introduces the CIFAKE dataset, a large collection of real and synthetic images, to support further research on the detection of AI-generated imagery.

It is therefore evident that past research has shown that it is possible to classify images as AI-generated or not, with a high degree of accuracy. This report will explore the process of training a neural network to classify images as either AI-generated or not, and then deploying this model to AWS SageMaker for inference, and evaluating it against comparable models.

Dataset

The dataset used for this project was a competition dataset from Hugging Face, held in 2023 [4]. The dataset consists of 62,060 images, and is 2.37GB in size, being pre-split into training and tesing sets, as summarised in tables 1 and 2, where it can be seen that the testing set has the class labels withheld due to the competition setting, restricting this analysis to the 18,618 training images, which we can sub-divide and validate with known labels.

Feature	Description
`id`	Index filename `34.jpg`
`image`	The Image object (rgb 512x512 resolution)
`label`	Binary class label [1=AI, 0=not AI]

Table 1: Dataset features and their descriptions.

Class Label	Train Count	Test Count
AI (1)	10,330 (55.5%)	NA
Not AI (0)	8,288 (45.5%)	NA
Total	18,618	43,442

Table 2: Counts of each class label in the training and testing sets, where it can be seen that the testing set has the class labels withheld due to the competition setting

Listing 1 shows the code used to load the dataset, and split it into training, validation, and test sets have a look at this later sub splitting of the data was reasoned to be the best strategy, as having a smaller dataset on which to train, `small_train`, it would be more conducive to iteration in the hyperparameter tuning process, and the optimal hyperparameters could then be transferred onto the main model, which would be trained on the full training dataset. The code demonstrates how the data is imported. Since Hugging Face datasets are not natively compatible with TensorFlow, the dataset is converted to a TensorFlow dataset using the `to_tf_dataset()` method.

Listing 1: Data Wrangling Script NEED TO CHANGE

```python

with open("token.txt", "r") as file:
    hugging_face_token = file.read().strip() # must have credentials to access the dataset

login(token=hugging_face_token)
raw_dataset = load_dataset('competitions/aiornot') # hugging face method to import the data

dataset = raw_dataset['train'] # remove the unused 'test' set with no labels

# split dataset into:
# - small_train (10% of original)
# - train (70% of original)
# - test (20% of original)

# First split: small_train (10%) and remainder (90%)
split_1 = dataset.train_test_split(train_size=0.1, seed=RANDOM_SEED)
small_train = split_1["train"]
remainder = split_1["test"]
split_2 = remainder.train_test_split(train_size=0.7 / 0.9, seed=RANDOM_SEED)
train = split_2["train"]
test = split_2["test"]

is_any_data_unused = not(small_train.num_rows + train.num_rows + test.num_rows == dataset.num_rows)
assert not is_any_data_unused, "Some data is unused in the splits."

def format_dataset(hugging_face_dataset, batch_size=32, shuffle=True):
    """convert a hugging face dataset to a TensorFlow dataset"""
    dataset = hugging_face_dataset.with_format(type='tf', columns=['image', 'label'], output_all_columns=True)
    dataset = dataset.to_tf_dataset(columns='image', label_cols='label', batch_size=batch_size, shuffle=shuffle)

    # normalse the image channels to be in the range [0, 1] rather than [0, 255]
    dataset = dataset.map(lambda image, label: (tf.cast(image, tf.float32) / 255.0, label), num_parallel_calls=tf.data.
      AUTOTUNE)
```

```
    # return prefetched dataset
    return dataset.prefetch(buffer_size=tf.data.AUTOTUNE)

small_train_tf = format_dataset(small_train, batch_size=32, shuffle=True)
train_tf = format_dataset(train, batch_size=32, shuffle=True)
test_tf = format_dataset(test, batch_size=32, shuffle=False)
```

This data was converted to npz files, due to their efficient storage and loading capabilities. After conversion, the data was uploaded to an S3 bucket for easy access during model training and evaluation. Uploading to S3 buckets was critical, as it meant that there was no longer a reliance on the RAM allocation of the sagemaker notebook environment - it could be loaded in from the S3 bucket in batches, which has the benefit of being a more scalable solution.

Modelling

- detail the structure of the model, eg. the training was called in a notebook (main.ipynb) but the training and the model was defined in discrete python files of their own in accordance with modularity best practices.

Model Structure

The model is a convolutional neural network (CNN) designed for binary image classification. It accepts full-size images of shape $(512, 512, 3)$.

Input $(512 \times 512 \times 3) \rightarrow$ `Rescaling`(1/255) $\rightarrow$ `Conv2D`(f_1, 3×3, ReLU) $\rightarrow$ `Pool` (max/avg) $\rightarrow$ `Conv2D`(f_2, 3×3, ReLU) $\rightarrow$ `Pool` (max/avg) $\rightarrow$ Global (max/avg) pooling $\rightarrow$ `Flatten` $\rightarrow$ `Dense`(d, ReLU) $\rightarrow$ optional `Dropout`(p) $\rightarrow$ `Dense`(1, sigmoid). Trained with binary cross-entropy and `adam/adagrad`.

Adam was chosen as one of the considered optimisers, as it is known to converge rapidly, and rectifies vanishing learning rate and high variance, therefore being the most popular optimiser [5]

Adagrad was chosen as the second potential optimiser, as the learning rate would not need manual tuning, and is known to perform better than alternatives like SGD, MBGD, and primitive momentum based optimisers. Though it should be noted that it has the weakness of a constantly decreasing learning rate, resulting in slower convergence [5]

Hyperparameter Tuning with Hyperband

Hyperparameter tuning was performed to maximise the models performance through iteration of most of its parameters.

Table 3 shows a list of the parameters being tuned in the above model. It is evident that this is a very large parameter space, and it is therefore not feasible to find the best possible hyperparamet

The Hyperband algorithm is used for hyperparameter optimization. Hyperband eliminates poorly performing hyperparameter combinations and focuses on promising ones, thus saving computational time compared to other methods like Bayesian optimization and grid search.

Model Deployment

The best hyperparameters found on the smaller training set were then extracted, and trained on the larger dataset. This

Table 3: Tunable Hyperparameters

Hyperparameter	Range/Choices
learning rate	1×10^{-4} to 1×10^{-2} (log scale)
dropout-rate	0.0 to 0.5 (used if `use-dropout=true`)
batch-size	4 to 8
conv1-filters (f_1)	16 to 128
conv2-filters (f_2)	32 to 256
dense-units (d)	64 to 512
pooling	`max`, `avg`
use-dropout	`true`, `false`
optimizer	`adam`, `adagrad`

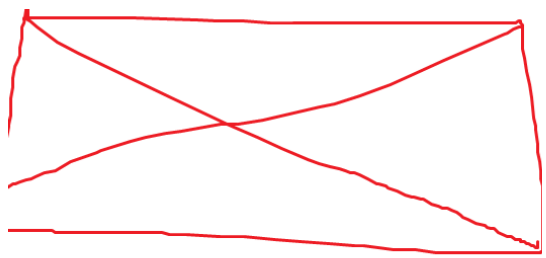

Fig. 2. Evidence of an endpoint in Sagemaker (endpoint page screenshot)

Transfer Learning

The list of the possible CNN structures available for use for transfer learning can be seen in [6]. Of these, EfficientNetV2 was selected as the most suitable architecture due to its demonstrated balance between accuracy and computational efficiency on general image classification tasks. This makes it well-suited for the diverse and varied images present in the aiornot dataset [6, 7].

Using an endpoint, the model was then evaluated on its precision, recall, f1 score, and accuracy. Figure 2 shows the endpoint configuration in Sagemaker. This allowed for evaluation of the models performance, whilst not being limited by the RAM limitations of the notebook environent in Sagemaker studio.

Deploying the transfer learning model, again using a sagemaker endpoint allowed a comparison between both models on the holdout set.

Model Comparison And Evaluation

In this section, we compare the performance of the different models trained on the dataset. The evaluation metrics used for comparison include accuracy, precision, recall, and F1 score.

Table 4: Model Comparison

Model	Accuracy	Precision	Recall	F1 Score
My Model	0.85	0.80	0.90	0.85
Transfer Learning Model	0.88	0.85	0.92	0.88

The results of the model evaluation are summarized in Table 4. The table highlights the key performance indicators for each model, allowing for a straightforward comparison of their strengths and weaknesses.

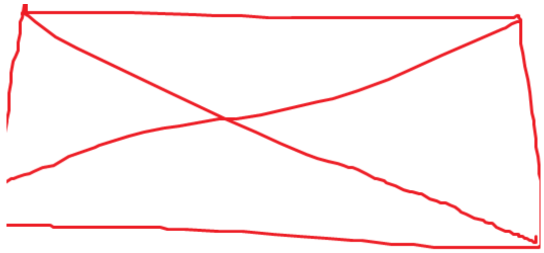

Fig. 3. Confusion Matrix

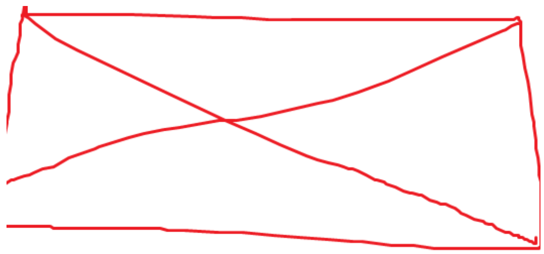

Fig. 4. Confusion Matrix

Figure 3 and **??** presents the confusion matrix for the model's predictions. It can be seen that continue here

References

[1] Z. Lu, D. Huang, L. Bai, J. Qu, C. Wu, X. Liu, and W. Ouyang, "Seeing is not always believing: Benchmarking human and model perception of ai-generated images," 2023. [Online]. Available: https://arxiv.org/abs/2304.13023

[2] C. Scott *et al.* (2024) Exploring the ethics of artificial intelligence in art. Accessed: 22 July 2025. [Online]. Available: https://www.artshub.com.au/news/opinions-analysis/exploring-the-ethics-of-artificial-intelligence-in-art-2694121/

[3] J. J. Bird and A. Lotfi, "Cifake: Image classification and explainable identification of ai-generated synthetic images," 2023. [Online]. Available: https://arxiv.org/abs/2303.14126

[4] Hugging Face Datasets, "competitions/aiornot," https://huggingface.co/datasets/competitions/aiornot, 2023, accessed: 22 July 2025.

[5] M. A. K. Raiaan, S. Sakib, N. M. Fahad, A. A. Mamun, M. A. Rahman, S. Shatabda, and M. S. H. Mukta, "A systematic review of hyperparameter optimization techniques in convolutional neural networks," *Decision Analytics Journal*, vol. 11, p. 100470, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772662224000742

[6] K. Team. (2024) Keras applications. Accessed: 22 July 2025. [Online]. Available: https://keras.io/api/applications/

[7] M. Tan and Q. V. Le, "Efficientnetv2: Smaller models and faster training," in *International Conference on Machine Learning*, 2021, iCML 2021. [Online]. Available: https://arxiv.org/abs/2104.00298