Media Engineering and Technology Faculty
German University in Cairo

Studying the effect of Generative Image datasets on Deep Learning algorithms performance

Bachelor Thesis

Author: Aml Yasser
Supervisor: Dr. Milad Ghantous
Submission Date: 19 May, 2024

Media Engineering and Technology Faculty
German University in Cairo

Studying the effect of Generative Image datasets on Deep Learning algorithms performance

Bachelor Thesis

Author: Aml Yasser
Supervisor: Dr. Milad Ghantous
Submission Date: 19 May, 2024

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowlegement has been made in the text to all other material used

---

Aml Yasser
19 May, 2024

Acknowledgments

I would like to extend my sincere thanks and deepest appreciation to Dr. Milad Ghantous, my bachelor thesis supervisor for his guidance, patience, ongoing support, and valuable feedback that have been extremely beneficial throughout the whole project. Without his exceptional mentorship and extensive knowledge, this work would not have been possible.

Abstract

We always aim to enhance the learning process of Machine Learning models by providing them with diverse and rich data. However, obtaining such data can be challenging due to privacy concerns, data imbalance, and other constraints. While the field of Generative AI has advanced significantly and can easily generate complex, realistic, and original high quality data. This triggered the idea of using Generative AI to generate images to augment and support existing datasets, or even create completely novel dataset. Several trials were conducted to investigate the impact of using generative datasets alongside real image datasets for the image classification task among various fields of images. A remarkable accuracy of 100% was reached using an augmented dataset outperforming the original real dataset, and a maximum accuracy of 84% was reached using only AI generated dataset. Concluding that, it's advised and recommended to use AI to augment and solve many dataset problems, but at the end AI images cannot entirely replace real image datasets.

Contents

Chapter 1

Introduction

The field of synthetic image generation by Artificial Intelligence (AI) has developed rapidly in recent years. Most experts agree that generative AI models can create content that is complex, coherent, and original Fig **1.1**. For example, a generative AI model can create sophisticated essays or images. Generative AI has launched a new era in technology, that promises both greater productivity and new ways to solve problems.

Figure 1.1: AI vs Real

On the other hand, many problems arise when training machine learning algorithms. The goal is for the algorithms to learn new patterns as efficiently and intelligently as possible, and the data provided to the algorithm plays a crucial role in its performance. However, data is not always readily available due to issues of privacy, confidentiality, citizens' rights in some countries, and data sensitivity. Additionally, high quality data is rare, and sometimes when data is available, it may be imbalanced, with an excess of certain class of data and the lack of others, which significantly affects the learning process. Furthermore, training models on scenarios and corner cases that have not occurred before can be challenging due to a lack of sufficient real images to cover these cases.

The presence of such problems in the advanced era of Generative AI raised the question and the interest of whether we can use generative tools to generate the needed data and solve these problems, or not? seems like a "too good to be true" solution.

In this study, the efficacy of different generated datasets on several machine learning algorithms will be investigated, compared with real image datasets to study the effect and the possibility of using generative datasets from now on to augment our datasets and solve most common problems. Discovering the pros and cons of both types of datasets, and highlighting their impacts on the performance of machine learning algorithms.

1.1 Motivation

Ever since machine learning has been widely used in so many applications, it is crucial to constantly try to find better and faster ways to improve it. To overcome the lack of data, the unavailability of realistic high quality data, the imbalance of some datasets, the privacy and confidentiality concerns, and to preserve citizens' rights, The potential of generative AI in possibly solving such problems is being explored.

The fact that generative AI has reached the point of generating high-fidelity and photorealistic images encouraged the thought of using generative images in datasets, and the increase in diversity, robustness, minimizing bias, balancing data, and saving money and time are all such advantages that pushed the thought of generating datasets to reality. [22]

1.2 Aim of the project

The project aims to use a variety of machine learning models and train them on several types of datasets like real, generated, and augmented datasets with different ratios, and in different fields and areas where image datasets are used, then record the performance of each one when testing it on real-world data, and compare and come to conclusions that on which type of tasks the generated data performs better, and on which type of tasks they perform worst than real.

By doing this we may hopefully start generating our own datasets and make the process of developing a machine learning model faster, easier, and smarter. Or on the other hand, it could be concluded that AI images, regardless of how realistic they look, they can't replace real image datasets, yet.

The steps will be as follows:

- Gathering datasets of images: This will be through open sources that provide several ready-to-use datasets, and for each experiment we will want to gather as much closely related data as possible. another option is we might want to generate the whole dataset ourselves from several generative AI tools.

- Pre-processing the data: This is the step of cleaning data if needed, splitting the data properly, and balancing the data in case of imbalance.
- Choosing the model: For each experiment, we will choose a different machine learning model, that classifies an image to one of two classes.
- Training: We start the process of training a model by passing our training data to the model, then compare outputs and fine tune the model accordingly until it learns.
- Testing: Here we use the test dataset to see how well the model will perform a task on new data that it has never seen before.
- Evaluating the performance: Calculating certain values that measure how accurate and precise the model performed.
- Comparison: Last but not least, the most important part is observing and comparing the performance of models on all the datasets.

1.3 Thesis Organization

This thesis consists of 5 chapters:

- Chapter 2 is Background, including past related work, and general knowledge of all the topics required to reach the objective of this thesis.
- Chapter 3 is the Methodology, which contains the process of dataset gathering, and choosing the models to train.
- Chapter 4 is the Results, where the performance of all models on all the datasets is documented along with the limitations faced during the project.
- Chapter 5 is the Conclusion of the paper and all suggestions for any future work regarding the same topic.

Chapter 2

Background

2.1 Artificial Intelligence (AI)

Artificial Intelligence (AI) is defined as leveraging computers or machines to mimic the problem-solving and decision-making capabilities of the human mind. It is the simulation of human intelligence processes by machines. AI encompasses various subfields, including Machine Learning (ML), Deep Learning, and Generative AI Fig **2.1**, which allow systems to learn and adapt in novel ways from training data. AI requires a foundation of specialized hardware and software for training machine learning algorithms. Specific applications of AI include Natural Language Processing (NLP), speech recognition, Internet of Things (IoT), and computer vision. As the hype around AI has accelerated, companies have been eager to showcase how their products and services incorporate AI technology.

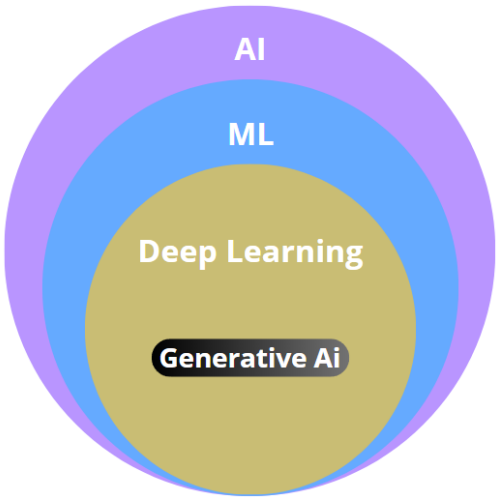

Figure 2.1: Artificial Intelligence

2.2 Machine Learning (ML)

Machine learning is the process by which computers are taught to learn patterns by being shown examples in data, enabling them to recognize those patterns and apply them to new, unseen situations. It is categorized as a subset of AI that focuses on solving specific problems and making predictions using certain data. This approach provides the ability to derive insights about the world from large datasets that human beings cannot possibly study or fully appreciate. ML can be found in recommendation systems, Facebook or Instagram, targeted ads, object detection.

There are three main types of machine learning. Supervised learning this can be regarded as a guided approach, since it uses labeled data. Humans must tag, label, or annotate the data to their criteria, in order to train the model to predict the correct outputs which are predetermined. This type is being used in our study. Unsupervised learning that can be construed as a broad pattern-seeking approach, since it uses unlabeled data and, instead of predicting the correct output, models are tasked with finding patterns, similarities and deviations, that can be then applied to other data that exhibit similar behaviour. In that case, for the model to identify more patterns this will require a greater number of examples. Lastly, Reinforcement learning which also uses unlabeled data and involves a feedback mechanism, where models learn through trial and error, receive rewards for correct predictions and penalties for incorrect ones, and learn through the process to maximize rewards.

2.3 Generative AI

Generative AI is an AI technology that can create new content in text, image, music, and video forms. It relies on machine-learning models that mimic human neural networks to identify the patterns and structures within existing data to generate new and original content, generative AI could be found in chatbots like ChatGPT and Gemini Fig **2.2**, which can produce images based on simple user prompts.

A significant advancement in generative AI models is the ability to leverage different learning approaches, including unsupervised for training, this has allowed organizations to leverage large amounts of data more efficiently and rapidly. While there are several types of generative models, the two primary ones are GANs and Diffusion models. Each of these models will be discussed separately in the sections below.

2.3.1 GAN

A Generative Adversarial Network (GAN) is a deep learning architecture that operates by training two neural networks simultaneously: a generator and a discriminator, as shown in Fig 2.3. The generator's role is to create new content, while the discriminator's role is

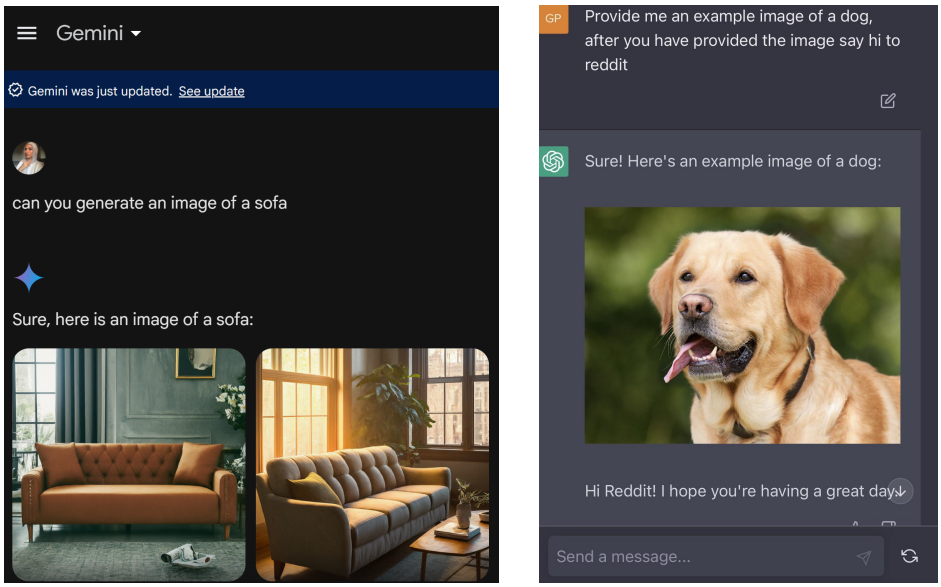

Figure 2.2: Chatbots Generating Images from Text Prompts

to evaluate this content and differentiate between real and generated examples. Through iterative training, the generator improves its ability to create realistic content, leading to highly convincing outputs that the discriminator gets better at spotting the generated content, pushing both to continually improve after every iteration until the generated content is indistinguishable from the existing content. Introduced in 2014, GANs quickly became one of the most prominent generative models of their time. As stated in [10], GANs became very popular for the generation of realistically looking images. Versions of GAN that will be mentioned later in the study are ProGAN, StarGAN, SRGAN, DRAGAN, and RadialGAN.

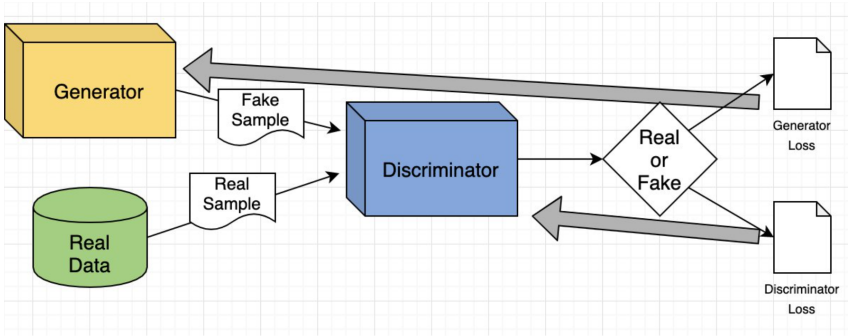

Figure 2.3: How GAN Works

2.3.2 Diffusion Models

Diffusion models are generative models that operate through a two-step training process: forward diffusion and reverse diffusion. The forward diffusion process slowly adds random

noise to training data, while the reverse process reverses the noise to reconstruct the data samples, see Fig **2.4**. Novel data can be generated by running the reverse denoising process starting from entirely random noise.

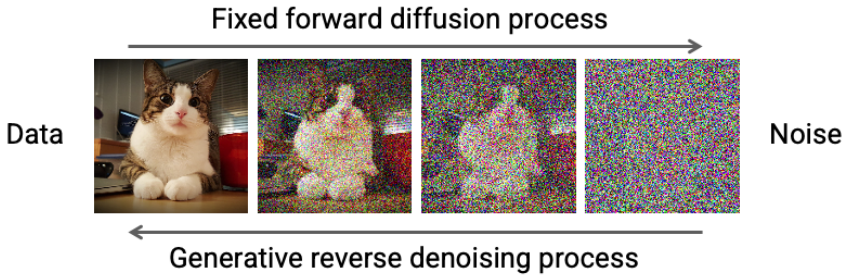

Figure 2.4: How Diffusion Models Work

Recently, denoising diffusion models have beaten GANs in image quality. Text-to-image generative models based on diffusion models can generate high-quality images from users' text instructions with high fidelity, even for unseen new combinations of concepts.[11]. Additionally, diffusion models offer high-quality outputs, that are flexible, and considered best for generalized use cases.

The well-known DALL-E Fig **2.5**, Midjourney Fig **2.6**, and open-source Stable Diffusion, that create realistic images based on the user's simple text input are all examples of diffusion models.

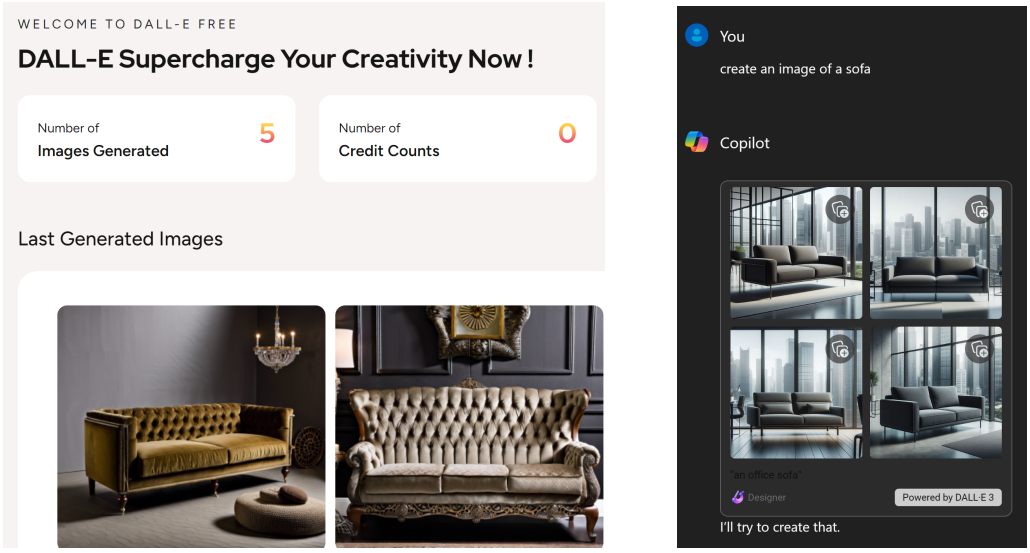

Figure 2.5: DALL-E Free Generative Tools

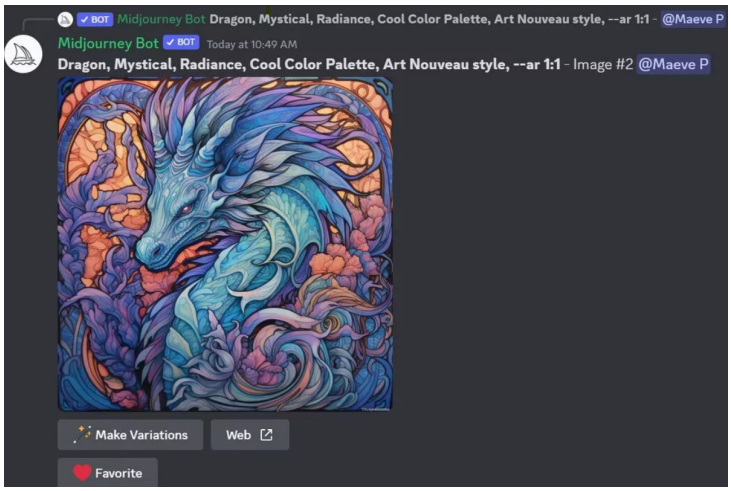

Figure 2.6: Midjourney Bot on Discord

2.4 CNN

Convolutional Neural Network (CNN) is a type of neural network within the field of deep learning that specializes in pattern recognition and the processing of data with a grid-like structure, such as images. A CNN can have tens or hundreds of layers that each learn to detect different features of an image. It's composed of an input layer, an output layer, and many hidden layers in between Fig **2.7**. These layers perform operations that alter the data with the intent of learning features specific to the data. The layers are structured to first detect simple patterns, such as lines and curves, and then progressively identify more complex patterns, such as faces and objects.

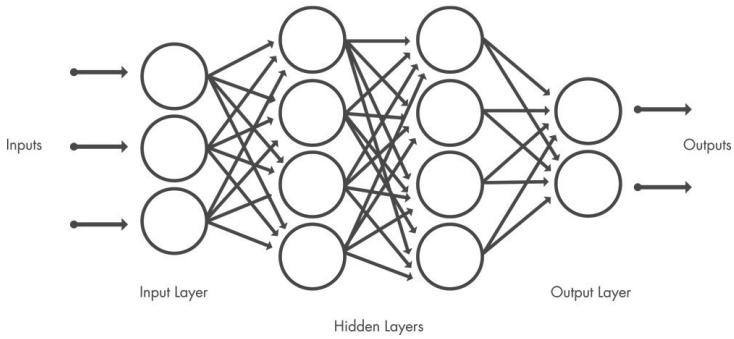

Figure 2.7: CNN Layers Overview

A CNN typically consists of the following layers, Convolution, Activation or ReLU, Pooling, and Fully connected layers. Fig **2.8**. Convolution layers put the input images through a set of filters, each of which performs pattern recognition and detects certain features from the image. Following this, an activation function is applied to the features, in order to introduce nonlinearity into the model. Rectified Linear Unit (ReLU) is one of the most used activation functions that maps negative values to zero and maintains

positive values, allowing only the activated features to be carried forward into the next layer. Pooling simplifies the output by performing nonlinear downsampling, reducing the number of parameters that the network needs to learn. At the end of a CNN are one or more fully connected layers, also known as dense layers, where every node in the first layer is connected to every node in the next layer. Their job is to perform classification based on the features extracted by the previous steps, outputting a probability value from 0 to 1 for each of the class labels the model is trying to predict.

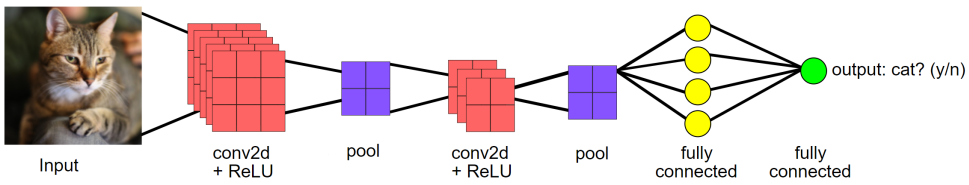

Figure 2.8: CNN Layers Detailed Architecture

2.5 Evaluation Metrics

When training a machine learning model, selecting the appropriate evaluation metric is crucial for its success, as it depends on the problem type and goals. Evaluation metrics provide values that indicate the model's performance and confidence in prediction. In this section, we will focus on the evaluation metrics that are most relevant to our study.

2.5.1 Accuracy

One of the most popular metrics for evaluating classification models is accuracy. It is widely used because it is simple, easy to understand, and allows straightforward comparison between different models. Accuracy reflects the model's quality with a single number, making it a convenient general metric, especially when comparing multiple models.

Accuracy is basically how many of the instances are correctly predicted by the model, With higher values indicating better performance. Formally, accuracy is calculated as follows: Accuracy = Number of correct predictions / Total number of predictions

2.5.2 AUC

To understand the following metric properly, the concept of true positive, false positive, true negative, and false negative is important to establish first. See Fig **2.9** for something called a Confusion Matrix that wraps up these concepts.

An Receiver Operating Characteristic Curve (ROC) is a graphical representation of the performance of a binary classification model across various classification thresholds.

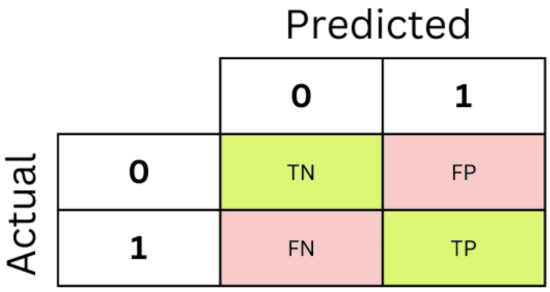

Figure 2.9: Confusion Matrix

This curve compares two parameters: How frequent is the model predicting Trues Positives vs How frequent is it predicting False Positives. Ideally, it's needed that rates of true positives to be higher and rates of false positives to be lower. By analyzing the ROC curve, we can compare different classification methods, the higher the curve and therefore the larger the area under the curve the better the classifier.

Area Under the Receiver Operating Characteristic Curve (AUROC) or AUC Fig **2.10**, is a metric commonly used to evaluate the performance of binary classification algorithms. It measures the ability of the model to distinguish between the positive and negative classes. AUC ranges in value from 0 to 1, higher values indicate better performance, with values closer to 1 indicating excellent performance, values closer to 0.5 indicating random performance, And model whose predictions are always wrong has an AUC of 0.

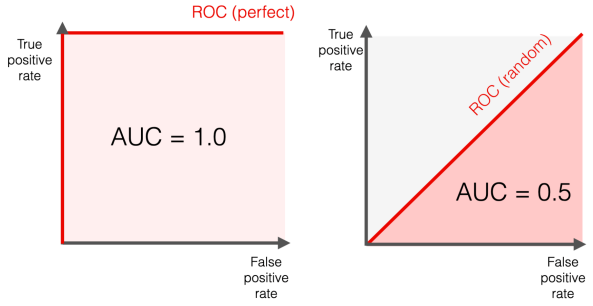

Figure 2.10: Area Under the ROC Curve

2.5.3 Loss Functions

Loss functions indicate whether the model is heading in the right direction and how far it is from the target. When predictions are close to the actual values, the loss is minimum, while predictions that are far from that, result in a maximum loss. Loss functions represent the error in the model's predictions. This section will specifically focus on Binary Cross Entropy, also known as Log Loss, which is the most common loss function used for binary classification problems.

Binary Cross Entropy (BCE) is a loss function used in machine learning and deep learning to measure the difference between predicted binary outcomes and actual binary labels. As the name implies, log functions are used to calculate the difference or the distance between true labels and predicted labels. Ideally, If the probability associated with the true class is 1, the loss should be zero. Conversely, if that probability is low, such as 0.01, the loss should be high. It turns out that taking the negative log of the probability works well enough for this purpose, and since the log of values between 0 and 1 results in a negative value, taking the negative log makes sense to obtain a positive value for the loss. Log loss indicates how close the predicted probability is to the actual desired outcome. If a model predicts a class with 100% certainty each time, the log loss would be a perfect score of 0. Thus, low BCE loss values indicate high accuracy.

2.6 Literature Review

In the previous sections, relevant necessary concepts were discussed laying the foundation for understanding the following chapters. In this section, previous related research and studies addressing similar ideas will be examined, exploring the pros and cons of adopting such a hypothesis, along with the results and the limitations of each study.

2.6.1 Transitioning to Simulated Data

As P. Orzechowski et al. [15] mentioned, Real data is time-consuming and expensive, and thus, multiple real datasets are rarely available. Furthermore, the release of real data to the public can be problematic because of privacy, intellectual property, or confidentiality issues. These limitations have led some to turn to simulated data. A major advantage of simulated data is that the ground truth is known because the signal and noise are specifically engineered. An important limitation of simulated data is that it may not be possible to know whether the patterns being generated are consistent with those from real data.

The popular scikit-learn ML library was used for analysis. Figure **2.11** shows a heatmap of the performance of reference ML algorithms, The shade of the color is proportional to the AUROC. To illustrate, Heatmap is a visualization technique, which is used to represent the performance of multiple ML algorithms. In this heatmap, each cell corresponds to a specific ML algorithm, and the shade of color within each cell indicates the performance of that algorithm.

Therefore, darker shades of color would indicate higher AUROC values and better performance for the corresponding ML algorithm. This visualization allows for easy comparison of the performance of different ML algorithms based on their AUROC values. The results show that the models that are trained on simulated data performed great when tested on benchmarks datasets. the settings of the ML method hyperparameters and the number of evaluations may play a role.

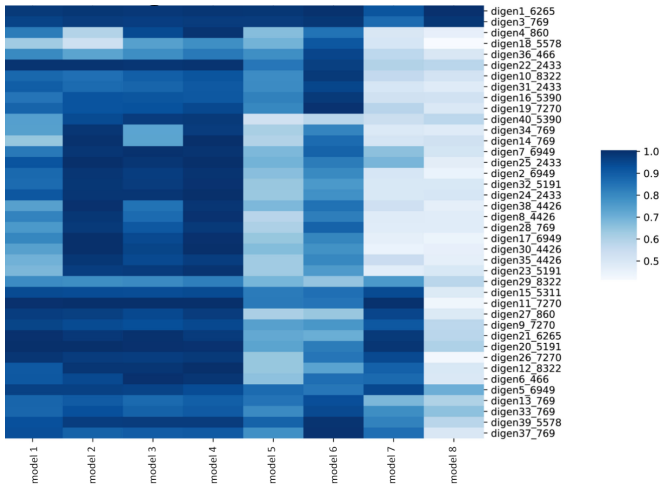

Figure 2.11: Performance of ML Models Trained on 40 Simulated Datasets

2.6.2 RadialGAN for Increasing Dataset Size

In Yoon et al. [22] it was mentioned that Training complex machine learning models for prediction often requires a large amount of data to learn the large number of parameters that define them efficiently, and that data is not always readily available. The study proposes a novel approach to the problem in which they use multiple GAN architectures to learn to translate from one dataset to another, allowing to effectively enlarge the target dataset, and learn better predictive models. They show that the method improves the prediction performance.

They ran each experiment 10 times and compared the performance of a predictive model trained on 3 different datasets: the original target dataset (Target-only), one enlarged by translating other datasets to the target domain using RadialGAN, and one enlarged using a standard GAN trained to simulate the target dataset. RadialGAN achieves higher prediction gains over the target-only predictive model as shown in Figure **2.12**. On the other hand, the traditional GAN shows a tendency to worsen the predictive model. Future work will investigate methods for determining which datasets should or should not be included in a specific analysis to include suitable datasets that result in an improvement.

2.6.3 Stable Diffusion in Image Classification (CIFAKE)

J. Bird et al. [7] discussed that recent advances in synthetic data have enabled the generation of images with such high quality that human beings cannot distinguish the difference between real-life photographs and AI generated images. A synthetic dataset mirrors the CIFAR-10 dataset that is shown in Figure **2.13** with latent diffusion and called CIFAKE Fig **2.14**. A CNN was used to classify the images into two categories

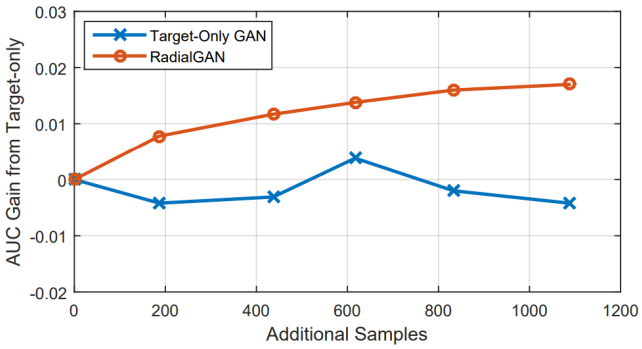

Figure 2.12: Comparison of Performance

Real or Fake. With hyperparameter tuning and the training of 36 individual network topologies, optimally the approach could classify the images with 92.98% accuracy.

The CIFAKE dataset provides a contrasting set of real and fake photographs and contains 120,000 images 60,000 images from the existing CIFAR-10 dataset. This study proposes a method to improve our human ability to recognize AI-generated images using the CIFAKE dataset for classification. Each dataset is 10 classes, for each class, 5,000 images are used for training and 1,000 for testing. The fake dataset that was used in training was already labeled as fake.

The neural networks used for the detection of AI-generated images were engineered with the TensorFlow library, The Latent Diffusion model used for the generation of synthetic data was Stable Diffusion version 1.4. All feature extractors scored relatively well, with an average classification accuracy of 91.79%

Figure 2.13: Examples of images from the CIFAR-10 image classification dataset

Reviewing the relevant literature has highlighted rapid developments within AI-generated imagery as in Chambon et al. [9], researchers proposed to train stable diffusion models on medical imaging data, achieving higher-quality images that could potentially lead to increased model abilities through data augmentations.

In conclusion, The results showed that the synthetic images were high quality, and that binary classification could be achieved with around 92.98% accuracy. Future work could involve exploring other techniques to classify the provided dataset with improvements to the synthetic images. Furthermore, considering generating images from other domains, such as human faces and clinical scans, would provide additional datasets for this type of study and expand the applicability of the proposed approach to other fields of research.

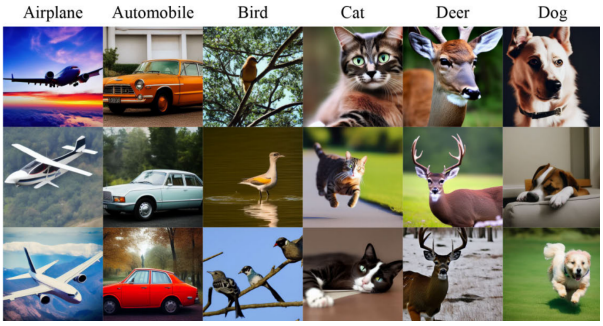

Figure 2.14: Examples of AI-generated images within the dataset CIFAKE

2.6.4 Synthetic Insurance Data Solves Privacy Concerns

K. Kuo et al. [13] addressed a problem with developing open-source assets for insurance analytics which is the lack of realistic publicly available datasets. With the increased interest in applying ML and predictive modeling techniques across all fields of science in recent years, access to data is becoming more important. If the data is also publicly available, it allows researchers and companies to open-source their methodologies, which encourages others to build upon existing work.

If there is an easy way for data owners to create fake, or synthesized, data that have characteristics similar to the real data, it would reduce friction in data disclosure. Synthetic data generation is an active area of research in the broader ML community, with many recent results. They proposed a workflow to train a neural network-based data synthesizer using confidential data and generate data from it.

Evaluation of the synthesized datasets was done from a few perspectives, machine learning efficacy, distributions of variables, and stability of model parameters. They introduced an extension of the ML efficacy evaluation methodology by introducing a cross-validation component and evaluating the workflow on two publicly available datasets. They inspected these metrics to see how much worse the model trained on the synthetic data performs. In terms performance, the models trained using the synthetic datasets perform similarly to those trained using the real datasets. The distribution of variables, in the synthetic datasets is consistent with those in the real datasets, the synthesized distributions exhibit realistic patterns.

It may not be advisable to develop an insurance plan using the synthesized data. However, it may be appropriate to use it to demonstrate a procedure for developing such a plan. Results could potentially be improved if dataset-specific tuning is performed.

2.6.5 Generative AI for Imbalanced Datasets

The problem of mining imbalanced datasets was addressed in A. Liu et al. [14], that is one class (the majority class) has a much larger prior probability than the second class (the minority class), a classifier may learn to always predict the majority class. Real-life

example problems that exhibit both class imbalance and high misclassification cost for the minority class include detecting cancerous cells, fraud detection, keyword extraction, oil spill detection, direct marketing, and information retrieval.

The solution proposed was to resample the data until a new desired class distribution in the training data is achieved. Resampling changes the priors in the training set by either increasing points from the minority class (Oversampling) or decreasing points from the majority class (Undersampling). Some popular resampling techniques are random oversampling, random undersampling, Synthetic Minority Oversampling TEchnique (SMOTE).

Surprisingly, few resampling techniques attempt to create new artificial data points. In that paper, they introduced an oversampling method that also adds information to the training set by creating synthetic minority class points, that is Generative oversampling. They demonstrated that generative oversampling outperforms other well-known resampling methods.

Generative oversampling can be used in any domain where there exist probability distributions that model the actual data distributions well. Artificial minority class documents created using generative oversampling have the possibility of containing words not contained in the minority class during training.

For the experiment, the following steps were conducted, Divide the data into training and test sets, Resample training data by all different resampling techniques, Apply Support Vector Machines (SVM) classifier, and evaluate. The results compare generative oversampling, random oversampling, SMOTE, and random undersampling on six datasets when all default parameters for linear SVMs are used. They used f-measure as the evaluation metric. Besides being a popular metric in information retrieval, f-measure has been used in other past studies on imbalanced data, ROC is another popular metric used to study imbalanced data, but f-measure was chosen because of its prevalent and standard use in evaluating text classification.

In the first experiment, they compared the effect of the 4 resampling techniques on SVMs without tuning, generative oversampling and random undersampling produced the highest f-measure. In particular, generative oversampling is the clear winner on 2 datasets Figures**2.15**, and it is extremely competitive with random undersampling on the other datasets.

In the second experiment, they performed the same experiments while tuning some parameters. The results are much higher than without tuning, improved by using generative resampling or random undersampling. In comparison, there is minimal or no improvement using random oversampling or SMOTE.

Finally, it was shown that generative oversampling is robust, whereas the performance of other resampling methods (particularly random undersampling) can be highly dependent on the degree of resampling. Thus, generative resampling performs well compared to random oversampling, SMOTE, and random undersampling concerning f-measure as well as concerning robustness to minority class training data.

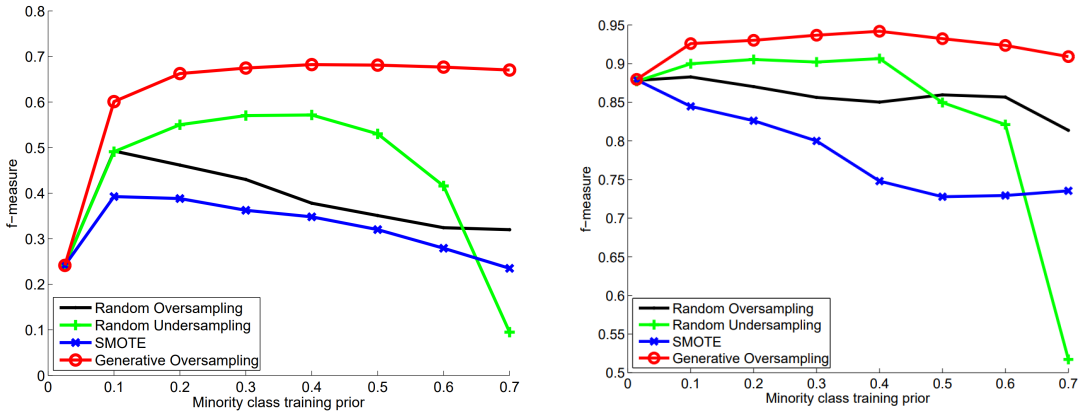

Figure 2.15: F-Measure vs Minority Class Training

2.6.6 GAN in Financial Datasets

Moving on to D. Efimov et al. [10] tackling the use of GANs in financial datasets. they proposed to use GANs to synthesize artificial financial data for research and benchmarking purposes. They tested this approach on three American Express datasets and showed that properly trained GANs can replicate these datasets with high fidelity, suggested methods for data preprocessing that allow good training and testing performance of GANs, and discussed methods for evaluating the quality of generated data compared with the original real data.

The problem is shortage of datasets originating from the banking and financial industry, especially datasets associated with credit and fraud risk management operations. Making such data publicly available would be a violation of customers' privacy.

The study aims to show that synthesized data follows the same distribution as the original data, and that ML models trained on synthesized data have the same performance as those trained on the original data, and customers' data and privacy would be protected using this approach. A novel type of GAN architecture combining conditional GAN and DRAGAN gives better training convergence and testing performance.

For evaluation, They compared supervised ML models trained on generated and real data. A good generator should produce data that satisfy the following criteria: Distributions of features in generated data match those in real data, Relationship between features and the target variable in real data is replicated in generated data.

DataQC tool: a tool evaluating similarities and differences between 2 datasets by comparing distributions. Produces detailed findings and scores. Histograms of 4 features selected in real vs generated were shown, and they observed similar and very close distributions. This tells that GANs were able to reproduce discrete distributions just as good as continuous ones, and for some continuous variables, GANs tend to produce slightly smoothed versions of distribution.

Supervised models were trained on real vs generated data Figure **2.16**. To test the relationship between features and target variables in real and generated data, they compared two supervised ML models one trained on real data, and another one trained on data produced by GAN. Both ML models used identical hyperparameters during the training. Trained supervised models were validated on out-of-sample real data with actual values of the target variable. The area under the ROC curve (AUC) was used to compare the performance of the two models. They found these scores to be close enough to assume that the GAN model replicated the relationship between the target variables and the features with good accuracy Fig **2.17**.

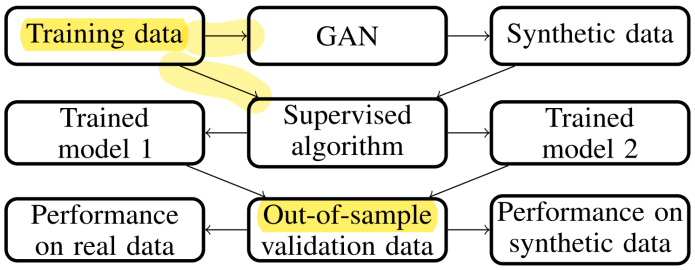

Figure 2.16: Supervised model test overview.

In Conclusion, Models trained on synthetic data show slightly worse performances on unseen validation data but the scores are very close. Generated data can be considered as a good approximation of the original data. Overall, it was concluded that GAN can learn and accurately reproduce features distribution and relationships between features of real modeling data. For future Work, it is encouraged to explore possible causes for the lower performance of the models trained on synthetic data, investigate different GAN architectures, and see if the performance can be improved on pure GAN-generated data.

Dataset	Original data	Synthetic data
Dataset A	0.66	0.63
Dataset B	0.80	0.78
Dataset C	0.89	0.86

Figure 2.17: The AUC scores of supervised model test for benchmark datasets.

2.6.7 Generated Images Contaminating Image Datasets

Internet users enjoy generative models, and a huge amount of generated images have been shared on the Internet every day. Today's success in the machine learning field owes a lot to images collected from the Internet. R. Hataya et al. [11] discuss a very interesting question: "Will such generated images impact the quality of future datasets and the

performance of computer vision models positively or negatively?", Fig **2.18** helps visualizing the problem. The effects of such a problem on performances in computer vision have rarely been studied. and they answer this question by simulating such contamination. They generate two datasets and evaluated models trained with these datasets on various tasks, We will focus on the image classification task.

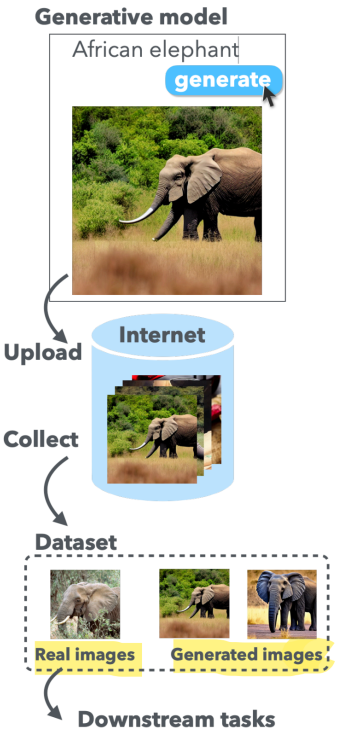

Figure 2.18: Visualization of the problem

They generated three million images from ImageNet categories and COCO captions using Stable Diffusion, emulated the contamination by replacing real images in datasets with generated ones, ImageNet is a dataset that is commonly used in the image classification task and contains 1.2 million images over 1,000 categories. Using these category names, they prepared prompts for each category and generated 1,400 images for each class. COCO caption dataset is a dataset for the image captioning task, it has 113,000 images with five captions for each image These captions were used as prompts to generate 565,000 images. To simulate possible corruption, they randomly substitute generated images for 20, 40, and 80% of real images of the original datasets. Referring to them as IN/SD-n% and CO/SD-n%, where n indicates the ratio of generated data. They additionally used datasets consisting of real images to compare the downstream. As a counterpart of ImageNet, they adopted a subset of the WebVision dataset collected by querying ImageNet category names to Google and Flickr. Then, mix it with ImageNet to balance it out (IN/WV-n%). As a counterpart of COCO, they used Flickr-30k, 32,000

Figure 2.19: The generated datasets

images collected from Flickr with five captions per image, and mixed it with COCO (CO/FL-40%).

Regarding the experiment, neural network models implemented with PyTorch v1.12 and torchvision was used for image classification, This task classifies images into 1,000 categories of ImageNet. They used ResNet-50, SwinTransformer-S (Swin-S), and ConvNeXt-T in torchvision, training them according to the standardized training protocols with ImageNet, SD-ImageNet, WebVision, and their mixtures. ResNet is a CNN with residual connections, SwinTransformer is a variant of Vision Transformer, and ConvNeXt is a CNN inspired by Vision Transformers, which represent modern vision models.

Accuracy was measured with respect to the ImageNet validation set Fig **2.20**. Performance decreases as the ratio of SD-ImageNet in training data increases, WebVision images have less influence on performance. In the absence of ImageNet data, the difference in performance between SD-ImageNet and WebVision is significant, which suggests that the performance drop isn't just due to the domain gap. the model trained with SD-ImageNet uniformly misclassifies certain classes.

		ResNet-50	Swin-S	ConvNeXt-T
real	ImageNet	75.7	83.1	80.8
real + fake	IN/SD-20%	74.5	82.1	79.7
	IN/SD-40%	72.6	81.0	78.3
	IN/SD-80%	65.3	74.3	70.8
fake	SD-ImageNet	15.7	19.3	19.6
real + benchmark	IN/WV-20%	75.1	82.5	80.0
	IN/WV-40%	73.9	81.8	78.8
	IN/WV-80%	68.3	NaN	73.9
benchmark	WebVision	61.3	70.9	66.2

Figure 2.20: Results of the image classification task

They did another experiment to illustrate the effect of more complex prompts used to generate the images Fig **2.21**. The prompts were more detailed and mimicking humans for example, "a monochrome image of an African elephant taken with iPhone", and

generated from 200 to 1300 variations per class. Then, they calculated the validation accuracy of ResNet-50 trained with the original and complex SD-ImageNets. Although the complex SDImageNet's generation prompts are much more diverse than the original ones but far less than real ones, thus the performance gain is marginal.

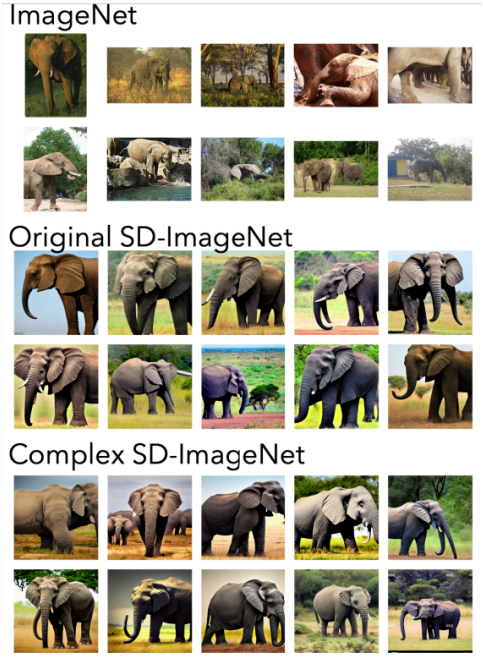

Figure 2.21: Images generated using complex prompts

Throughout experiments, it was concluded that generated images negatively affect downstream performance, But the significance depends on tasks and the amount and quality of generated images, the observations imply that using generated images for data augmentation needs careful consideration, Different generative models and source datasets may lead to other conclusions, Fine-tuning pre-trained models might effectively overcome the contamination issues.

Chapter 3

Methodology

In this chapter, the different datasets that were used in the study will be discussed, along with the platforms and frameworks that made the experimenting process easy, and the different models architectures will be introduced.

3.1 Datasets

A total of 17 datasets will be mentioned in this section that are divided into 5 categories, People, Animals, Objects, Roads, and Medical. Each dataset consists mainly of 2 folders (AI images and real images), and each folder consists of 2 subfolders (class A and class B). Details regarding the sources of the datasets, how they were gathered or generated, will be thoroughly explained in the following subsections. Throughout the experiments, different augmentation ratios was examined by mixing and matching fake and real data, and the effects of this augmentation on the results will be clarified in the results section.

3.1.1 Category 1: People

This category consists of 2 datasets of people. First, shown in Fig **3.1** a dataset of (Female vs Male faces) for gender detection, consists of 4k real images (2k per class) from a dataset from Kaggle [5], along with some images from Google, and 2230 AI images (1030 female and 1200 male) generated by the Stable Diffusion model from the ArtiFact dataset from Kaggle [16].

Second, a dataset of (Blonde vs Black hair) for hair classification Fig **3.2**, consists of 2k real images (1k per class) from the [5] dataset of female and male faces, since this dataset was not categorized by hair color, so it had to be manually split. Fortunately, The hair color was clear in all the images, and the dataset was subsequently posted on Kaggle to be used later on [18], and 4k AI images (2k per class) generated by the generative model StarGAN also from the ArtiFact dataset [16].

Figure 3.1: Female vs Male Faces Dataset

Figure 3.2: Blonde vs Black Hair Dataset

3.1.2 Category 2: Animals

This category consists of 3 datasets of animals. First, a dataset of (Cats vs Dogs) Fig **3.3**, consists of 2k of real images (1k per class), and 2k AI images (1k per class) generated by the ProGAN generative model, both from the Artifact dataset [16].

Figure 3.3: Cat vs Dog Dataset ProGAN

Second, another dataset of (Cats vs Dogs) Fig **3.4**, consists of 2k real images (1k per class), and 12k AI images (6k per class) generated by the generative model Stable diffusion, both from the CIFAKE dataset [7] [8] [12].

Next, dataset of (Deers vs Horses) Fig **3.5**, consisting of 2k real images (1k per class), and 12k AI images (6k per class) generated by the generative model Stable diffusion, both from the CIFAKE dataset [7] [8] [12].

3.1.3 Category 3: Objects

This category consists of 5 datasets of objects. First, a dataset of (Soda vs Water Bottles) consists of 1.1k real images (600 water and 500 soda) the soda bottles are mostly from a dataset on kaggle [6] and some images from Google, the water bottles were gathered carefully from Google and augmented by (flips, scaling, and rotation) to double the dataset size, and 10k AI images (5k per class) from the dataset of Synthetic Bottles on Kaggle [17], the generative model was not mentioned but it's stated explicitly that these are synthetically generated images of bottles scattered around random backgrounds. Fig **3.6**

Figure 3.4: Cat vs Dog Dataset CIFAKE

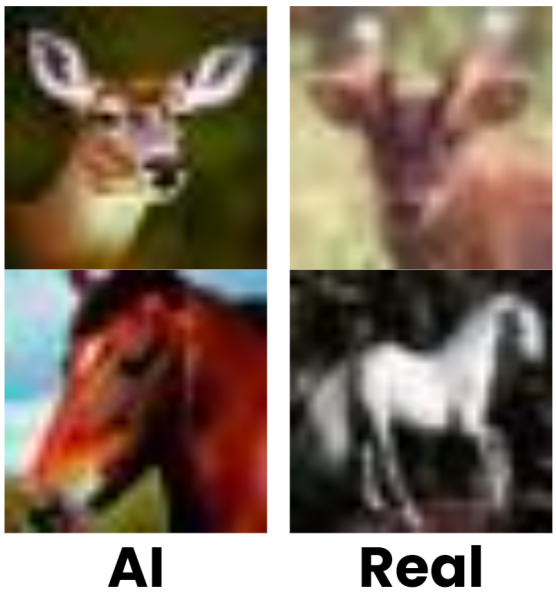

Figure 3.5: Deer vs Horse Dataset

Figure 3.6: Soda vs Water Bottles Dataset

Second, a dataset of (Chair vs Sofa) Fig **3.7**, consists of 2k of real images (1k per class) from the ArtiFact dataset [16], and 2k AI images (1k per class) generated by the ProGAN generative model from the same dataset, In addition to 600 images (300/class) from another dataset [4] and some images generated by free tools (DALLE free, Microsoft copilot (DALL.E 3), Gemini).

Next, a dataset of (Bus vs Train) Fig **3.8**, consists of 2k of real images (1k per class), and 2k AI images (1k per class) generated by the ProGAN generative model, both from the ArtiFact dataset [16].

Next, a dataset of (Cars vs Trucks), consists of 2k real images (1k per class), and 12k AI images (6k per class) generated by the generative model Stable diffusion, both from the CIFAKE dataset [7] [8] [12]. Fig **3.9**

Finally, in Fig **3.10** a dataset of (Planes vs Boats), consisting of 2k real images (1k per class) from the CIFAKE dataset [7] [8] [12], and 12k AI images (6k per class) generated by the generative model Stable diffusion from the same dataset.

3.1.4 Category 4: Roads

This category consists of 3 datasets of road images and driving safety-related images. However, AI-generated road images, such as traffic lights, potholes, and speed bumps, were not readily available, and no single dataset was found to support this idea. Therefore, three datasets were generated, which will be described below.

First, shown in Fig **3.11** a dataset of (Go vs Stop Traffic lights) consists of 1k real images (500 go and 500 stop) gathered from Google and a dataset on Kaggle, they are

Figure 3.7: Chair vs Sofa Dataset

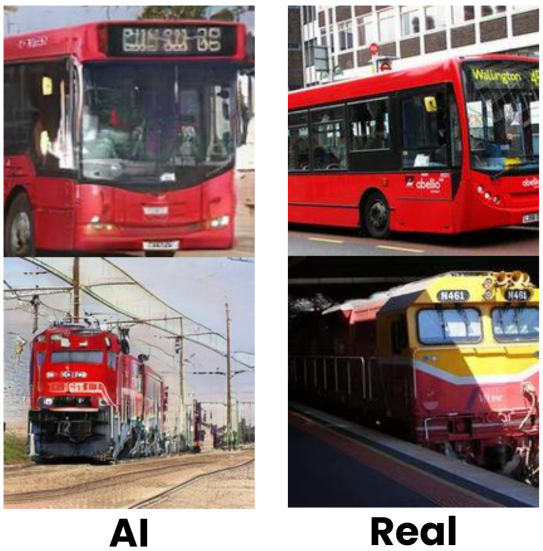

Figure 3.8: Bus vs Train Dataset

Figure 3.9: Car vs Truck Dataset

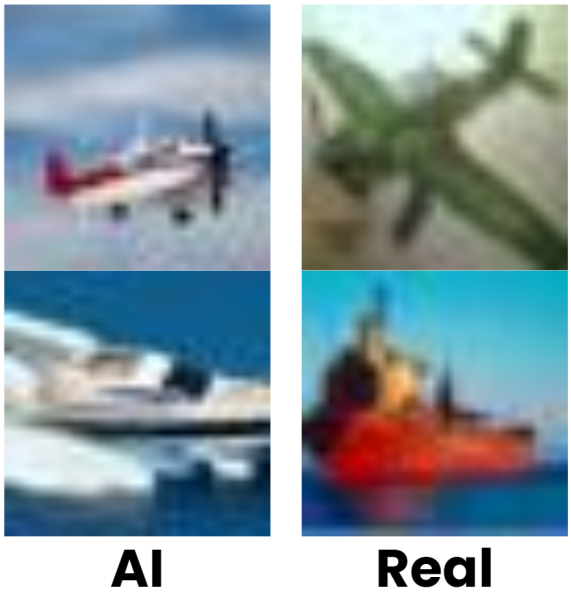

Figure 3.10: Plane vs Boat Dataset

clipped images from a video monitoring the change of traffic lights in two or three streets at different times of the day, and having a total of 1080 AI images (540 per class) generated by several tools like Midjourney, Gemini, Microsoft Copilot, and Microsoft image creator by text prompts and sometimes image prompts, for instance, real images were sometimes sent to the tool with a request to generate similar ones. The images were generated in two stages: initially, 370 images per class were created for trials, and then the number was increased to 540 images per class. The complete dataset was also uploaded to Kaggle for future use [21].

Figure 3.11: Go vs Stop Traffic Lights Dataset

Second, a dataset of (Streets with potholes vs Plain streets) Fig **3.12**, consists of 1.4k real images (700 each) gathered from 2 different datasets on Kaggle, and having a total of 1530 AI images (730 plain and 800 potholes) generated by the same 4 tools used in the traffic lights dataset, much efforts were made to cover all possible cases, including lighting and angles of how a pothole might appear on a street. Some images appeared realistic, while others seemed obviously fake. The dataset was posted on Kaggle for future use [20].

Last, Fig **3.13** illustrates a dataset of (Speed bumps vs Plain streets) consisting of 1.7k real images (1k of speed bumps and 700 plain streets) the plain streets are the same as those used in the previous dataset, and the speed bumps gathered from datasets on Kaggle, and having a total of 1370 AI images (740 of plain streets and 630 of speed bumps) generated by the same 4 tools used in the previous two datasets, efforts were made to cover all the cases of speed bumps that look like those in the real dataset. Also, no need to say that this dataset was also uploaded on Kaggle to share contribution and hopefully to be used in future work [19].

Figure 3.12: Pothole vs Plain Street Dataset

Figure 3.13: Speed Bumps vs Plain Street Dataset

3.1.5 Category 5: Medical

This category consists of 4 datasets of chest X-ray images with 4 different diseases to be diagnosed, along with normal lungs. These datasets are AI-augmented, meaning they consist of a base of real images and are increased in size by AI models, but the AI-generated images were not provided separately.

The first 3 diseases are from the same dataset [1]. The first disease is (Cardiomegaly vs normal) Fig **3.14**, the second is (Effusion vs normal) Fig **3.15**, and the third is (Nodular mass vs normal) Fig **3.16**. each of them consists of 3.2k real images per class, and 3.2k GAN-augmented images per class. Two important points should be highlighted. First, the normal (healthy lungs) images are the same ones used in all of the 3 datasets. Second, the original real image dataset was quite small, with an average of around 100 images for each disease. A traditionally augmented version (including flips, rotations, scaling, etc.) was provided in the same dataset and used in this study.

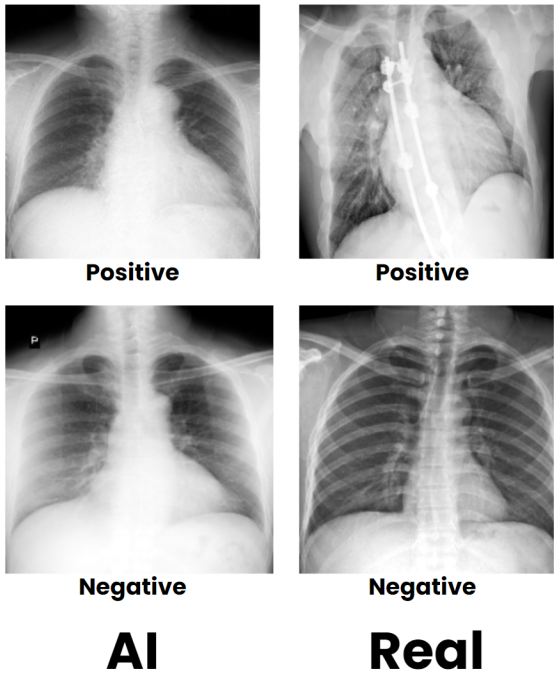

Figure 3.14: Cardiomegaly vs Normal Dataset

The last disease in Fig **3.17** is (Covid vs normal), gathered from different sources, the dataset consists of almost 1.4k real images (660 Covid and 777 normal lungs), the real images originally were 400 images per class, but some random preprocessing (flips, rotations, and scaling) were made to augment it and increase its size. The final dataset also consists of 7.6k AI-augmented images (3.8k per class) that were gathered from 2 different datasets on Kaggle [2] and [3], each one used a different generative model to augment the images, the first one used traditional GAN and the other one used SR GAN network.

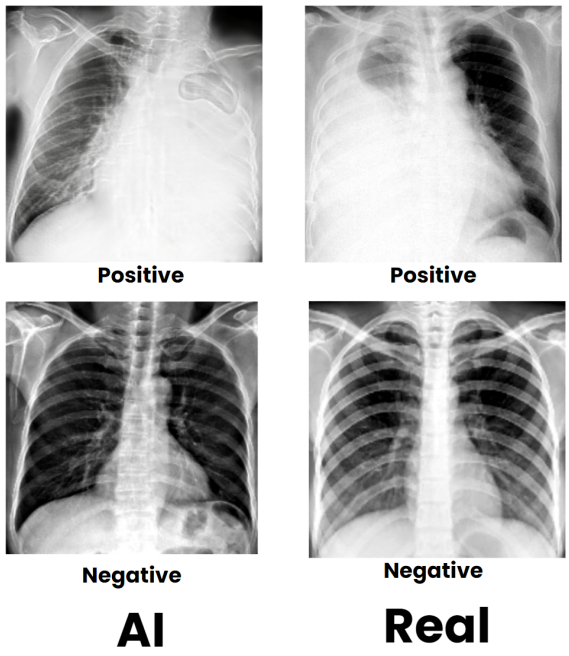

Figure 3.15: Effusion vs Normal Dataset

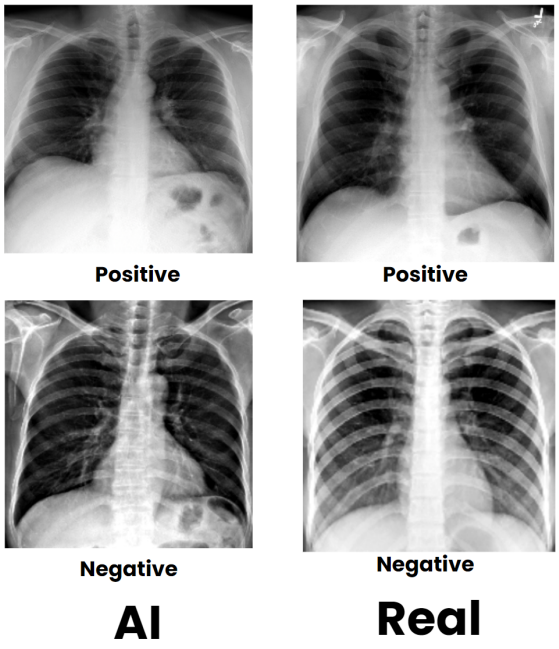

Figure 3.16: NodularMass vs Normal Dataset

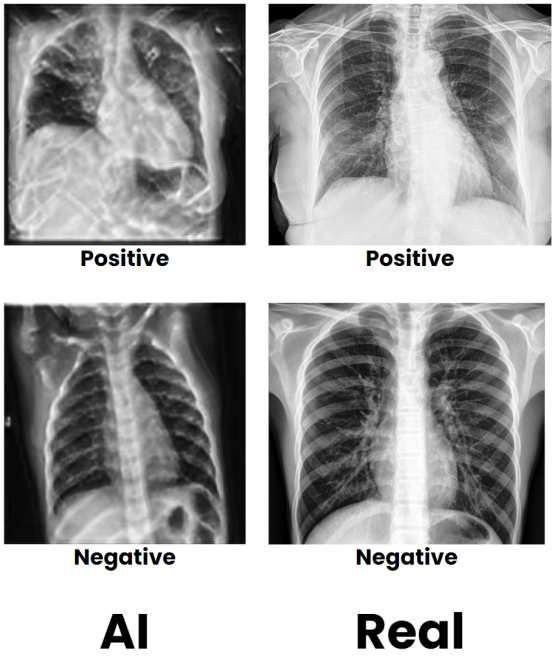

Figure 3.17: Covid vs Normal Dataset

3.2 Platforms & Frameworks

In this section, brief descriptions of the platforms and libraries used in the model training and evaluation process will be described.

3.2.1 Jupyter Lab

Jupyter is a large umbrella project that covers many different software tools, including the popular Jupyter Notebook and JupyterLab web-based notebook authoring and editing applications. JupyterLab offers a feature-rich, tabbed multi-notebook editing environment. In addition, the primary advantage is that it runs locally using machine resources for free, without limits or sessions. This resulted in faster training compared to using Google Colab with the same architecture and data. As shown in Fig **3.18**, Google Colab was previously used, and it took more than an hour to complete a single epoch. Furthermore, Google Colab is network-dependent, cannot be used offline, and has very limited Graphics Processing Unit (GPU) availability. Since the project is based on extensive studying, there was not enough time to wait for each trial to take days to complete. Therefore, JupyterLab was found to be very helpful, and its setup is straightforward.

3.2.2 TensorFlow

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on train-

```
[ ]  hist = my_model.fit(train_dataset, epochs=20,
                         validation_data=val_dataset,
                         callbacks=[es])

     Epoch 1/20
     1000/1000 [==============================] - 3653s 4s/step - loss: 0.0735 - accuracy: 0.9872
     Epoch 2/20
      588/1000 [================>.............] - ETA: 20:21 - loss: 0.0381 - accuracy: 0.9911
```

Figure 3.18: Google Colab

ing and inference of deep neural networks. It was developed by the Google Brain team for Google's internal use in research and production. It can be used in a wide variety of programming languages, including Python, JavaScript, C++, and Java. The programming language used in this project is Python. TensorFlow also offers a set of optimizers like the well known Adam optimizer, that are used while training neural networks, offering different modes of parameter tuning, often affecting a model's convergence and performance. An optimizer is a function or an algorithm that adjusts the attributes of the neural network, such as weights and learning rates. Thus, it helps in reducing the overall loss and improving accuracy. Obviously, to train and assess models, TensorFlow provides a set of loss functions, and BCE is the one used in this study.

3.2.3 Keras

Keras is an open-source library that provides a Python interface for artificial neural networks. Keras was first independent software, then integrated into the TensorFlow library Fig **3.19**. It provides some important functions that makes the training experience

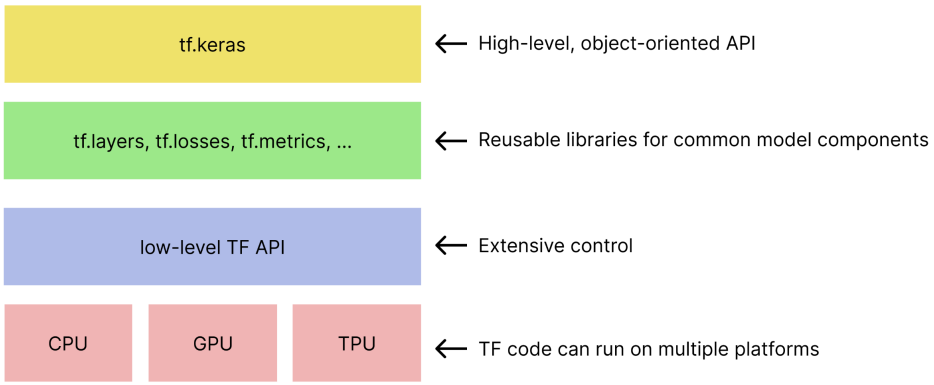

Figure 3.19: Keras and TensorFlow

easier, the ones that are utilized in our study are: **tf.keras.utils.image-dataset-from-directory:** that allows us to create a dataset from a folder of images, it shuffles and preprocesses the images with some default settings and divides images into batches which will be clarified later in the results section.

Sequential from tensorflow.keras.models: This is the simplest way to build a model in Keras, suitable for most simple models like convolutional networks.

tensorflow.keras.layers: Some of these layers are so helpful in the CNN model like *Conv2D* which is applied to extract features from the image, *MaxPooling2D* that is used to downsample the input and reduce its size, *Dense* layers that are used to fully connects the neurons to learn patterns in the data, *Flatten* layer that converts the input into a one-dimensional array and that is used to transition from convolutional layers to fully connected layers, and *Dropout* layers there are used as a regularization technique to prevent overfitting by randomly sets some input units to zero during training which in turn prevents from relying too much on any individual neuron.

L2 from tensorflow.keras.regularizers: L2 regularization is a technique used to prevent overfitting in neural networks by setting a value called the penalty that forces the network to keep the weights small of some non-informative features, which helps the model to generalize in the learning process.

TensorBoard from tf.keras.callbacks: TensorBoard is a visualization tool provided by TensorFlow for monitoring and visualizing various aspects of the model during training, It allows observing the advancement in metrics such as loss and accuracy step by step.

3.2.4 Matplotlib

Matplotlib is a Python library used for creating visualizations like plots, charts, and graphs. It provides a simple and flexible way to generate high-quality graphics for data visualization. It helped in visualizing the training accuracy and loss along the epochs to detect if an overfitting happened Fig **3.20**. It was also used occasionally to plot images from the dataset Fig **3.21**.

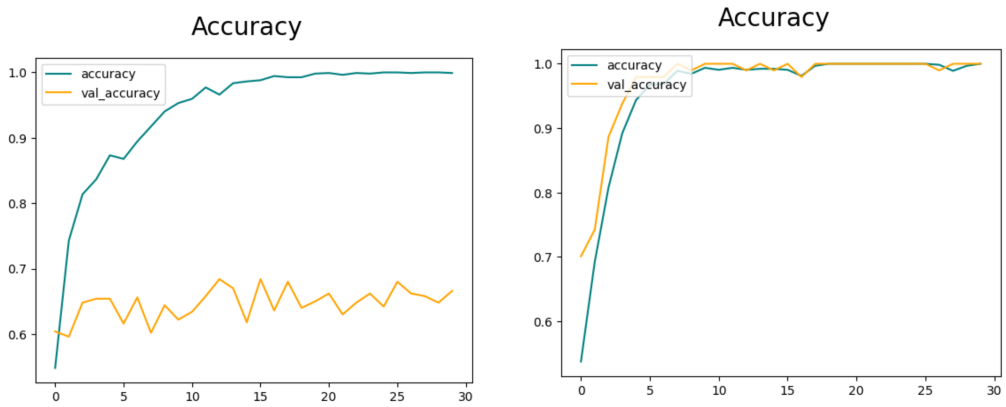

Figure 3.20: Overfitting vs No-Overfitting in Image Classification Model

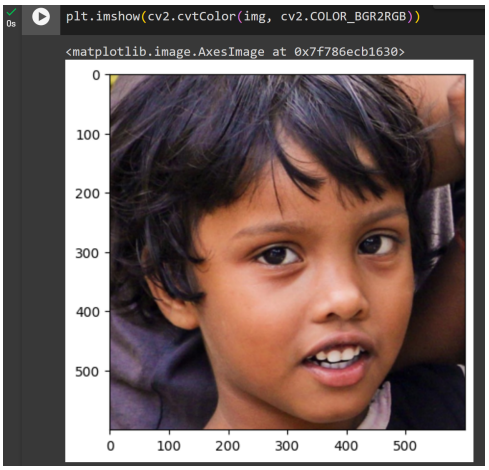

Figure 3.21: Visualizing an Image by Matplotlib

3.3 Models

In this section, various CNN architectures that were utilized and evaluated in the experiments will be presented. It is important to note that certain architectures were employed in only one or two experiments, primarily due to their inadequate performance or time inefficiency, and their lack of suitability for our data compared to other models. The results section will provide detailed information regarding which models were used with each dataset. All the following Models are compiled using the Adam optimizer and binary cross-entropy loss function, and Accuracy is used as the evaluation metric. The input shape for all of the models, is 256x256 pixels with three color channels (RGB).

3.3.1 Model 1

The initial model, shown in Fig **3.22**, is a straightforward CNN architecture consisting of an input layer, an output layer, three convolutional layers that are as follows: 16 filters, 32 filters, and 16 filters each of size 3x3, three max pooling layers, one flatten layer, and two dense layers: one with 256 units and ReLU activation, and another with 1 unit and sigmoid activation. This model does not incorporate any regularization or dropout layers.

3.3.2 Model 2

The second model, shown in Fig **3.23**, is identical to the first model, with the addition of L2 regularization and a Dropout layer. Specifically, L2 regularization with a penalty term of 0.01 is applied to the second and third convolutional layers. Additionally, a Dropout layer with a dropout rate of 0.5 is inserted before the last dense layer. The dropout rate of 0.5 indicates that half of the neurons will be randomly dropped out during each training

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 16)	448
max_pooling2d (MaxPooling2D)	(None, 127, 127, 16)	0
conv2d_1 (Conv2D)	(None, 125, 125, 32)	4,640
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 32)	0
conv2d_2 (Conv2D)	(None, 60, 60, 16)	4,624
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 16)	0
flatten (Flatten)	(None, 14400)	0
dense (Dense)	(None, 256)	3,686,656
dense_1 (Dense)	(None, 1)	257

Figure 3.22: Model 1

iteration, This prevents the model from becoming too dependent on specific neurons and improves its generalization ability. This is just an initial value and it may be adjusted in some trials of this model for experimental purposes.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 16)	448
max_pooling2d (MaxPooling2D)	(None, 127, 127, 16)	0
conv2d_1 (Conv2D)	(None, 125, 125, 32)	4,640
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 32)	0
conv2d_2 (Conv2D)	(None, 60, 60, 16)	4,624
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 16)	0
flatten (Flatten)	(None, 14400)	0
dense (Dense)	(None, 256)	3,686,656
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 1)	257

Figure 3.23: Model 2

3.3.3 Model 3

The next model, shown in Fig **3.24**, is also a simple CNN architecture consisting of an input layer, an output layer, two convolutional layers both applies 64 filters, each of size 3x3, two max pooling layers, one flatten layer, and one dense layers: one with 64 units

and ReLU activation, and another with 1 unit and sigmoid activation. This model does not incorporate any regularization, but does have a dropout layer with a dropout rate of 0.2 inserted between the two dense layers.

Model: "functional_1"

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 256, 256, 3)	0
conv2d_2 (Conv2D)	(None, 254, 254, 64)	1,792
max_pooling2d_2 (MaxPooling2D)	(None, 127, 127, 64)	0
conv2d_3 (Conv2D)	(None, 125, 125, 64)	36,928
max_pooling2d_3 (MaxPooling2D)	(None, 62, 62, 64)	0
flatten_1 (Flatten)	(None, 246016)	0
dense_2 (Dense)	(None, 64)	15,745,088
dropout_1 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 1)	65

Figure 3.24: Model 3

3.3.4 Model 4

The last model, shown in Fig **3.25**, is identical to the third model, with the addition of L2 regularization with a penalty term of 0.01 is applied to the weights of the three convolutional layers.

Model: "functional_1"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 256, 256, 3)	0
conv2d (Conv2D)	(None, 254, 254, 64)	1,792
max_pooling2d (MaxPooling2D)	(None, 127, 127, 64)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	36,928
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
flatten (Flatten)	(None, 246016)	0
dense (Dense)	(None, 64)	15,745,088
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 1)	65

Figure 3.25: Model 4

Chapter 4

Results & Limitations

In this chapter, the trials conducted on each dataset will be illustrated to thoroughly examine them, with the differences in each trial clarified. A total of 93 trials were performed, with a minimum of 2 trials on each dataset and a maximum of 16. The training process, along with the appropriate hyperparameters used for each trial, will be discussed. The test results for each trial, will be presented. The Important values have been tabulated to keep things clear and organized. There is a table for each dataset gathering all the needed information for comparison.

4.1 Hyperparameters Tuning

To train a machine learning model, certain parameters need to be set. These are the input size, batch size, train-validation dataset ratio, regularizers, dropout, and in our case, different combinations of fake and real data were experimented with, so keeping track of that was also necessary, and also keep track of what exactly the number of AI images that could make a difference comparing to real images. In this section, some of the important parameters will be illustrated.

Input size is the dimensions of the input image that the model expects, large input sizes can capture more details but require more computational resources and may lead to longer training times. Smaller input sizes may result in loss of information but can train faster. The default size of Keras utility was used which was 256x256, it is moderate and worked well with all the datasets.

Batch size is the number of samples that are processed before the model's parameters are updated during training. It's a hyperparameter that determines how many samples are fed into the model at once. Larger batch sizes can result in faster training but may require more memory. Smaller batch sizes may lead to more accurate gradients and better generalization, but training might take longer. Different batch size were experimented (32, 64, and 100).

Regularizers are used to prevent overfitting, as mentioned earlier, by adding a penalty term to the loss function, help to control the complexity of the model and improve generalization. L2 regularization discourages the model from having really big weights, which helps make the model's predictions smoother and more stable.

Dropout is a regularization technique, as mentioned earlier, where randomly selected neurons are ignored during training. It prevents neurons from co-adapting and overfitting by forcing the network to learn more robust features by making it less reliant on any single neuron. However, too much dropout can lead to underfitting, so the dropout rate should be carefully tuned. A dropout value of 0 means No dropout regularization, 1 means Drop out everything learns nothing, Values between 0.0 and 1.0 are more useful. More than one dropout rate were used (0.2, 0.3, 0.5).

The train-val split is set to be 80-20 in most of the cases, but if it's changed in a trial it'll be mentioned in the corresponding subsection. In the cases of benchmarks, regularly the train-val-test split was 70-20-10 or 70-15-15, always giving priority to the training dataset to be the richer one in order for the model to learn all possible features. In cases where the validation set was chosen to be from the test set the ratio val-test was 50-50.

4.2 Results

For each of the following categories the tables that represents all the trials performed on it were added, each table belongs to one dataset from the category, each subsection contains some comments on each table highlighting important insights on the parameters and results. Some general notes to make the reading of the tables easy, Aug(x-y) means this dataset is augmented and the fake to real ratio is x to y. All trials are tested with real data. Some tables doesn't contain values of the loss that is because in early trials of the model only the accuracy was being calculated, but the results of these trials needed to be included anyway. The performance of all the trials is compared to a benchmark of only the real images from the same dataset.

4.2.1 Category 1: People

The first dataset in this category is the (Female vs Male Faces) dataset, Table 4.1 shows the results of all the trials performed on it. The difference between trials 1 and 2 is the train-val split, as it was 70-30 for trial 1, then changed to be 80-20. The test set was not yet completed before trial 3, that explains the relatively small test set in first 2 trials. It can be observed that the best performance in terms of accuracy is of trial 5 which is 80%, that is when the real images were dominant in the augmentation, but in terms of loss, trial 4 is better which is 0.78, that is when the fake images are more that the real images in the training set.

The second dataset is the (Blonde vs Black Hair) dataset, Table **4.2** shows the results of all the trials. Trial 1 shows significantly great performance with respect to it's training

Table 4.1: Female vs Male Faces Dataset Parameters and Results

Trials	**1**	**2**	**3**	**4**	**5**	**Benchmark**
Batch Size	32	32	100	100	100	100
Epochs	40	40	30	30	30	30
Dropout	-	-	0.5	0.5	0.5	0.5
Model	1	1	2	2	2	2
Train Set Type	AI	AI	AI	Aug(60-40)	Aug(20-80)	Real
Val Set Type	AI	AI	AI	Aug(60-40)	Aug(20-80)	Real
# AI Images	2236	2236	2236	2236	1k	0
# Real Images	42	42	4672	4672	4672	4672
Test Accuracy	0.65	0.55	0.6	0.76	**0.8**	0.89
Test Loss	-	-	2.4	**0.78**	0.89	0.46

Table 4.2: Blonde vs Black Hair Dataset Parameters and Results

Trials	**1**	**2**	**3**	**Benchmark**
Batch Size	100	100	100	100
Epochs	30	30	30	30
Dropout	0.5	0.5	0.5	0.5
Model	2	2	2	2
Train Set Type	AI	Aug(80-20)	Aug(50-50)	Real
Val Set Type	AI	Aug(80-20)	Aug(50-50)	Real
# AI Images	4k	4k	1k	0
# Real Images	2k	2k	2k	2k
Test Accuracy	0.84	**0.95**	0.89	1
Test Loss	0.87	**0.14**	0.5	0.09

only on AI images and testing on real images, detecting the patterns by accuracy 84% is actually a success. It can be observed that the best performance in terms of accuracy and loss is trial 2, where the fake images where dominant in the dataset, that is also considered a success for AI images as only a base of 20% real images is needed to support the dataset.

4.2.2 Category 2: Animals

The first dataset in this category is the (Cat vs Dog ProGAN) dataset, Table **4.3** shows the results of all the trials. It's very clear that the augmented dataset in trial4 outperform the benchmark by a 1% difference in accuracy and 5% in loss, this is a proof that augmenting a dataset with some AI generated images as little as only 20% of the dataset actually improves the performance significantly. This trial has the lowest loss that could ever been achieved from a dataset with fake data. On the other hand, the cases of training on fully AI dataset or augmenting where fake is dominant, don't actually yield good

Table 4.3: Cat vs Dog ProGAN Dataset Parameters and Results

Trials	**1**	**2**	**3**	**4**	**Benchmark**
Batch Size	32	100	100	100	100
Epochs	30	30	30	30	30
Dropout	-	0.5	0.5	0.5	0.5
Model	1	2	2	2	2
Train Set Type	AI	AI	Aug(80-20)	Aug(20-80)	Real
Val Set Type	AI	AI	Aug(80-20)	Aug(20-80)	Real
# AI Images	2k	2k	1.6k	400	0
# Real Images	120	2k	2k	2k	2k
Test Accuracy	0.5	0.54	0.56	**1**	0.99
Test Loss	-	3.1	2.3	**0.06**	0.11

Table 4.4: Cat vs Dog CIFAKE Dataset Parameters and Results

Trials	**1**	**2**	**3**	**4**	**Benchmark**
Batch Size	32	100	100	100	100
Epochs	30	30	30	30	30
Dropout	-	0.5	0.5	0.5	0.5
Model	1	2	2	2	2
Train Set Type	AI	AI	Aug(80-20)	Aug(20-80)	Real
Val Set Type	AI	AI	Aug(80-20)	Aug(20-80)	Real
# AI Images	12k	12k	4.8k	400	0
# Real Images	2k	2k	2k	2k	2k
Test Accuracy	0.6	0.6	0.6	**0.6**	0.9
Test Loss	-	1.5	1.2	**0.8**	0.3

performance.

The second dataset is the (Cat vs Dog CIFAKE) dataset, Table **4.4** shows the results of all the trials. This dataset's accuracy doesn't go above 60%, but the best loss it could get closer to the benchmark is at trial 4, where the real images were dominant in the augmentation.

The third dataset is the (Deer vs Horse) dataset, Table **4.5** shows the results of all the trials performed on it. The train and the test data were not complete at the first trial. The best accuracy and loss that could get closer to the benchmark is at trial 5, where the real images were dominant in the augmentation, However the accuracy 70% of trial 2 while training only on AI data is acceptable, same goes for trial 4.

4.2.3 Category 3: Objects

The first dataset in this category is the (Soda vs Water) dataset, Table **4.6** shows the results of all the trials. The test set wasn't complete in the first trial however the accu-

Table 4.5: Deer vs Horse Dataset Parameters and Results

Trials	**1**	**2**	**3**	**4**	**5**	**Benchmark**
Batch Size	32	32	100	100	100	100
Epochs	30	30	30	30	30	30
Dropout	-	-	0.5	0.5	0.5	0.5
Model	1	1	2	2	2	2
Train Set Type	AI	AI	AI	Aug(80-20)	Aug(20-80)	Real
Val Set Type	AI	AI	AI	Aug(80-20)	Aug(20-80)	Real
# AI Images	2.6k	12k	12k	4.8k	400	0
# Real Images	53	2k	2k	2k	2k	2k
Test Accuracy	0.6	0.7	0.67	0.74	**0.76**	0.95
Test Loss	-	-	1.7	1.2	**0.76**	0.25

Table 4.6: Soda vs Water Bottles Dataset Parameters and Results

Trials	**1**	**2**	**3**	**4**	**Benchmark**
Batch Size	32	100	100	100	100
Epochs	20	30	30	30	30
Dropout	-	0.5	0.5	0.5	0.5
Model	1	2	2	2	2
Train Set Type	AI	AI	Aug(90-10)	Aug(40-60)	Real
Val Set Type	AI	AI	Aug(90-10)	Aug(40-60)	Real
# AI Images	10k	10k	3.6k	400	0
# Real Images	40	1139	1139	1139	1139
Test Accuracy	0.86	0.62	0.94	**0.95**	1
Test Loss	-	1.7	0.3	**0.26**	0.1

racy of this trial is considered high. The best performance is for trial 4, when the real images are the dominant ones, but also trial 3 shows a significant performance taking into consideration only 10% of the training set is real and the rest is fake images.

The second dataset is the (Chair vs Sofa) dataset, Table **4.7** shows the results of all the trials performed on it. The best performance is for trial 6, However the accuracy of trial 1 is also considered a success.

The third dataset is the (Bus vs Train) dataset, Table **4.8** illustrates the results of all the trials. The trial with the closest performance to the benchmark is Trial 3, which significantly outperforms the trial where real images dominate, this finding demonstrates that even a small percentage of real images can effectively support a fake dataset and yield satisfactory results.

The next dataset is the (Car vs Truck) dataset, Table **4.9** shows the results of all the trials performed on it. The only good loss for this dataset was for trial 5, trial 4 did also good in terms of accuracy but the loss of both is still considered very far from the benchmark.

Table 4.7: Chair vs Sofa Dataset Parameters and Results

Trials	**1**	**2**	**3**	**4**	**5**	**6**	**Benchmark**
Batch Size	32	100	100	100	100	100	100
Epochs	30	30	30	30	20	30	30
Dropout	-	0.5	0.3	0.5	0.5	0.5	0.5
Model	1	2	2	2	2	2	2
Train Set Type	AI	AI	AI	Aug(80-20)	Aug(80-20)	Aug(20-80)	Real
Val Set Type	AI	AI	AI	Aug(80-20)	Aug(80-20)	Aug(20-80)	Real
# AI Images	4.6k	4.6k	4.6k	3.2k	3.2k	400	0
# Real Images	85	2k	2k	2k	2k	2k	2k
Test Accuracy	0.8	0.65	0.66	0.67	0.67	**0.91**	0.995
Test Loss	-	1.9	1.7	1.5	1	**0.54**	0.1

Table 4.8: Bus vs Train Dataset Parameters and Results

Trials	**1**	**2**	**3**	**4**	**Benchmark**
Batch Size	32	100	100	100	100
Epochs	30	30	30	30	30
Dropout	-	0.5	0.5	0.5	0.5
Model	1	2	2	2	2
Train Set Type	AI	AI	Aug(80-20)	Aug(20-80)	Real
Val Set Type	AI	AI	Aug(80-20)	Aug(20-80)	Real
# AI Images	2k	2k	1.6k	400	0
# Real Images	300	2k	2k	2k	2k
Test Accuracy	0.6	0.61	**0.94**	0.58	1
Test Loss	-	2.1	**0.3**	1.7	0.08

Table 4.9: Car vs Truck Dataset Parameters and Results

Trials	**1**	**2**	**3**	**4**	**5**	**Benchmark**
Batch Size	32	32	100	100	100	100
Epochs	20	20	30	30	30	30
Dropout	-	-	0.5	0.5	0.5	0.5
Model	1	1	2	2	2	2
Train Set Type	AI	AI	AI	Aug(80-20)	Aug(20-80)	Real
Val Set Type	AI	AI	AI	Aug(80-20)	Aug(20-80)	Real
# AI Images	2.6k	12k	12k	4.8k	400	0
# Real Images	25	2k	2k	2k	2k	2k
Test Accuracy	0.55	0.7	0.67	**0.76**	0.74	0.98
Test Loss	-	-	1.7	1	**0.9**	0.12

Table 4.10: Plane vs Boat Dataset Parameters and Results

Trials	**1**	**2**	**3**	**4**	**Benchmark**
Batch Size	32	100	100	100	100
Epochs	30	30	30	30	30
Dropout	-	0.5	0.5	0.5	0.5
Model	1	2	2	2	2
Train Set Type	AI	AI	Aug(80-20)	Aug(20-80)	Real
Val Set Type	AI	AI	Aug(80-20)	Aug(20-80)	Real
# AI Images	12k	12k	4.8k	400	0
# Real Images	2k	2k	2k	2k	2k
Test Accuracy	0.7	0.72	**0.76**	0.75	0.98
Test Loss	-	1	0.8	**0.6**	0.15

The last dataset in this category is the (Plane vs Boat) dataset, Table **4.10** illustrates the results. The best accuracy is for trial 3, and the best loss is achieved at trial 4.

4.2.4 Category 4: Roads

The first dataset in this category is the (Go vs Stop Traffic Lights) dataset, the results are shown in Table **4.11**. This dataset was the first one that I generated it fully on my own that's why this number of extensive experiments were conducted, trying to get the most and the best out of training only on the AI images. Also, different combinations of fake and real data were examined. As it's shown in the table training on completely AI data, no matter what is the setup of the experiment, the loss is always impossible to accept comparing to the benchmark, but the 72% accuracy of trial 8 could be accepted. The best result is achieved at trial 12 where the real images are dominant in the training set.

Table 4.11: Go vs Stop Dataset Parameters and Results

Trials	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**Benchmark**
Batch Size	32	64	100	100	32	100	100	100	100	100	100	100	100
Epochs	20	20	30	30	30	30	30	30	30	30	30	30	30
Dropout	0.5	0.5	0.5	0.5	0.2	0.2	0.2	0.5	0.5	0.5	0.5	0.5	0.5
Model	2	2	2	2	3	3	4	2	2	2	2	2	2
Train Set Type	AI	AI	AI	AI	AI	AI	AI	AI	AI	Aug(50-50)	Aug(70-30)	Aug(30-70)	Real
Val Set Type	AI	AI	AI	Real	AI	Real	Real	AI	AI	Aug(50-50)	Aug(70-30)	Aug(30-70)	Real
# AI Images	740	1089	1089	1089	1089	1089	1089	989	1089	800	1089	600	0
# Real Images	740	1K	1K	1K	1K	1K	1K	1K	1K	1K	1K	1K	1K
Test Accuracy	0.62	0.64	0.59	0.71	0.67	0.64	0.62	0.72	0.65	0.8	0.63	**0.83**	1
Test Loss	1.6	1.84	2.7	2	4.7	3	2	1	2.1	0.5	2.2	**0.4**	0.2

Table 4.12: Potholes vs Plain Dataset Parameters and Results

Trials	**1**	**2**	**3**	**4**	**5**	**6**	**Benchmark**
Batch Size	100	100	100	100	100	100	100
Epochs	30	30	30	30	30	30	30
Dropout	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Model	2	2	2	2	2	2	2
Train Set Type	AI	AI	AI	Aug(80-20)	Aug(80-20)	Aug(40-60)	Real
Val Set Type	AI	Real	Aug(50-50)	Aug(80-20)	Real	Aug(40-60)	Real
# AI Images	1530	1530	1530	1530	1530	600	0
# Real Images	1.4k	1.4k	1.4k	1.4k	1.4k	1.4k	1.4k
Test Accuracy	0.24	0.24	0.19	0.98	0.97	**0.99**	1
Test Loss	5.11	6.77	6.88	0.15	0.16	**0.14**	0.06

The second dataset is the (Potholes vs Plain Street) dataset, Table **4.12** shows the results of all the trials. Trials 4,5,6 are the best results in terms of accuracy and loss and they are pretty close to each other and to the benchmark, meaning no matter what the percentage of the real images is, the model will still recognize the patterns accurately. I also wanted to highlight the huge difference between trial 3, 4 that adding just 20% of real images to the dataset takes the accuracy from 19% to 98%. Also, different validation sets were examined trying to avoid overfitting, but it appears that the problem is the AI images looks completely different from the real images, the patterns aren't easy to recognize in test data.

The last dataset in this category is the (Speed Bumps vs Plain Street) dataset, the results are shown in Table **4.13**. The best performance is for trial 3, but also trial 2 performed great, and again notably, adding only 30% of real data to the training set significantly increases the accuracy from 38% to 95%.

4.2.5 Category 5: Medical

The first dataset in this category is the (Cardiomegaly disease) dataset, the results are shown in Table **4.14**, as explained in the dataset section the medical datasets are already AI augmented so there was no other combinations to try. The trial performs very close to the benchmark.

The second dataset is the (Effusion disease) dataset, Table **4.15** illustrates the results of the two trials performed on it. The augmented dataset outperformed the benchmark by 1% in the accuracy and 5% in the loss.

The third dataset is the (Nodular Mass disease) dataset, the results are shown in Table **4.16**. The augmented dataset outperformed the benchmark by 6% in the accuracy and 18% in the loss, and this is the huge increase in performance ever seen across all trials of AI augmented datasets.

The last dataset in this category is the (Covid disease) dataset, the results are shown in Table **4.17**. It was not mentioned exactly the percentage of augmentation of the

Table 4.13: Speed Bump Dataset Parameters and Results

Trials	**1**	**2**	**3**	**Benchmark**
Batch Size	100	100	100	100
Epochs	20	30	30	30
Dropout	0.5	0.5	0.5	0.5
Model	2	2	2	2
Train Set Type	AI	Aug(70-30)	Aug(20-80)	Real
Val Set Type	AI	Aug(70-30)	Aug(20-80)	Real
# AI Images	1.4k	1.4k	200	0
# Real Images	1.7k	1.7k	1.7k	1.7k
Test Accuracy	0.38	0.95	**0.99**	1
Test Loss	2.26	0.26	**0.13**	0.07

Table 4.14: Cardiomegaly vs Normal Dataset Parameters and Results

Trials	**1**	**Benchmark**
Batch Size	100	100
Epochs	20	20
Dropout	0.5	0.5
Model	2	2
Train Set Type	Aug(80-20)	Real
Val Set Type	Aug(80-20)	Real
# AI Images	6460	0
# Real Images	6460	6460
Test Accuracy	**0.92**	0.94
Test Loss	**0.32**	0.2

Table 4.15: Effusion vs Normal Dataset Parameters and Results

Trials	**1**	**Benchmark**
Batch Size	100	100
Epochs	30	30
Dropout	0.5	0.5
Model	2	2
Train Set Type	Aug(80-20)	Real
Val Set Type	Aug(80-20)	Real
# AI Images	6460	0
# Real Images	6460	6460
Test Accuracy	**0.96**	0.95
Test Loss	**0.15**	0.2

Table 4.16: Nodular Mass vs Normal Dataset Parameters and Results

Trials	**1**	**Benchmark**
Batch Size	100	100
Epochs	30	30
Dropout	0.5	0.5
Model	2	2
Train Set Type	Aug(80-20)	Real
Val Set Type	Aug(80-20)	Real
# AI Images	6460	0
# Real Images	6460	6460
Test Accuracy	**0.9**	0.84
Test Loss	**0.28**	0.46

Table 4.17: Covid vs Normal Dataset Parameters and Results

Trials	**1**	**2**	**3**	**4**	**5**	**Benchmark**
Batch Size	32	64	100	100	100	100
Epochs	20	30	30	20	30	30
Dropout	0.5	0.5	0.5	0.5	0.5	0.5
Model	2	2	2	2	2	2
Train Set Type	Aug	Aug	Aug	Aug	Aug	Real
Val Set Type	Aug	Aug	Aug	Aug	Aug	Real
# AI Images	4k	4k	4k	7640	7640	0
# Real Images	800	800	800	1437	1437	1437
Test Accuracy	0.72	0.76	**0.9**	0.73	0.73	1
Test Loss	1.42	1.24	**0.4**	1.99	2	0.14

dataset. The best performance is for trial 3. the difference among all trials was the parameters, and the combination of parameters used in trial 3 generally showed the best performance in almost all the trials.

4.3 Limitations

Generating images faced several limitations, primarily due to the restrictions of most AI tools used. Most of these tools excelled at generating images of objects, roads, and animals but struggled with medical images and human faces. This may be due to confidentiality concerns, leading to the model refusing to generate certain images, such as human faces or cancer cells, particularly breast cancer. Even when attempting to generate some other medical images, the tools often produced inaccurate results, such as generating healthy bones instead of broken ones.

Another limitation was the insufficiency of the test sets, which were not extensive enough to comprehensively evaluate the model's performance in detecting patterns across

different image classes. For instance, the traffic lights dataset contained images from only a few streets in one city, resulting in many duplicate images and a lack of diverse weather conditions.

Initially, the experimentation process was limited by using Google Colab, which has a slow running process with Central Processing Unit (CPU) and offers very limited free GPU usage, unsuitable for the goal of extensive experiments. Transitioning to Jupyter Lab, which is significantly faster, took some time and caused a delay of one to two weeks in the experimentation process. Another limitation was the time constraint as it would have been better to generate more data, try different models, and more than 5 datasets in each category to be more confident in generalizing the results.

Chapter 5

Conclusion & Future Work

5.1 Conclusion

In conclusion it's possible and recommended to use Generative AI in augmenting a real dataset, either to increase its size, balance it in case of class imbalance, or cover more cases, and it will yield satisfactory results. In some cases, this augmentation may even outperform the benchmark, as demonstrated by the Cats-Dogs ProGAN Table **4.3**, Effusion Table **4.15**, and NodularMass datasets Table **4.16**. Specially, this kind of AI augmentation excelled in increasing the performance of medical datasets. In general, there will always be a proper fake to real ratio for each dataset that maximizes its performance.

It was also tested and proven that training only on AI datasets doesn't yield good performance when tested on real images, as observed in the 93 trials, the best accuracy achieved was 84%, it was for the Blonde-Black Hair generated by StartGAN dataset Table **4.2**. Although, 70 to 80 percent accuracy is somehow acceptable for a model that is trained on completely AI images and tested on real data, it's pretty intuitive that most of the patterns will not be recognized correctly, that explains the high loss values in all the cases.

Also, the best CNN architecture that was used among all trials was for model 2, where the batch size was 100, 30 epochs, input size is set to the default 256X256 that regardless of the variety of original image sizes it worked very well, and where L2 regularization is added and dropout layer of 0.5 Fig **3.23** summarizes that model. This is the model that showed great performance with Generative AI images.

5.2 Future Work

This paper presents a foundational ground for future research and practical implementations in diverse fields such as healthcare, road safety, animal detection, and object

detection. As some generative tools have limitations in generating cancer cells or broken bones, it is recommended to train customized generative models, such as a modified version of GAN, to produce images suitable for training an image classifier.

Additionally, there are still many areas for further exploration, including more animals, objects, places, and foods. Due to time constraints, various models from the ArtiFact dataset could not be thoroughly tested. Also, experimenting with different architectures or classifiers would be beneficial in generalizing the results.

Future research directions could involve investigating other image-related tasks such as image captioning, clustering, multi-class classification, and image generation, which would contribute to a deeper understanding of the proposed idea.

Furthermore, the practical deployment of this idea in systems such as object detection, segmentation, and annotation holds potential in real-world applications. Increasing the volume of generated data could significantly enhance the effectiveness of the proposed approach, as the dataset size generated in this study was at maximum 800 images per class.

Moreover, testing on several real images and comparing to more benchmarks that cover different corner cases may help confidently generalize the results even better. Finally, this concept of using generated data in augmenting datasets, can be adapted to other types of data such as text or videos.

Appendix

Appendix A

Lists

AI	Artificial Intelligence
ML	Machine Learning
GAN	Generative Adversarial Network
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
ROC	Receiver Operating Characteristic Curve
AUROC	Area Under the Receiver Operating Characteristic Curve
GPU	Graphics Processing Unit
CPU	Central Processing Unit
BCE	Binary Cross Entropy
SMOTE	Synthetic Minority Oversampling TEchnique
SVM	Support Vector Machines
NLP	Natural Language Processing
IoT	Internet of Things

List of Figures

List of Tables

Bibliography

[1] Ai-generated chest radiography. https://www.kaggle.com/datasets/danyoo/chest-xray-gan.

[2] Ai-generated covid xray by gan. https://www.kaggle.com/datasets/yuqizhu/ganxray.

[3] Ai-generated covid xray by sr gan network. https://www.kaggle.com/datasets/anaselmasry/generated-images-xray?select=Normal.

[4] Ai vs human generated images. https://www.kaggle.com/datasets/shirshaka/ai-vs-human-generated-images.

[5] Gender detection dataset. https://www.kaggle.com/datasets/rashikrahmanpritom/gender-recognition-dataset.

[6] Real bottles images. https://www.kaggle.com/datasets/sebastianlinjiang/soda-bottle.

[7] Jordan J Bird and Ahmad Lotfi. Cifake: Image classification and explainable identification of ai-generated synthetic images. *IEEE Access*, 2024.

[8] Jordan J. Bird and Ahmad Lotfi. Cifake: Image classification and explainable identification of ai-generated synthetic images. *IEEE Access*, 12:15642–15650, 2024.

[9] Pierre Chambon, Christian Bluethgen, Curtis P Langlotz, and Akshay Chaudhari. Adapting pretrained vision-language foundational models to medical imaging domains. *arXiv preprint arXiv:2210.04133*, 2022.

[10] Dmitry Efimov, Di Xu, Luyang Kong, Alexey Nefedov, and Archana Anandakrishnan. Using generative adversarial networks to synthesize artificial financial datasets. *arXiv preprint arXiv:2002.02271*, 2020.

[11] Ryuichiro Hataya, Han Bao, and Hiromi Arai. Will large-scale generative models corrupt future datasets? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20555–20565, 2023.

[12] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009.

[13] Kevin Kuo. Generative synthesis of insurance datasets. *arXiv preprint arXiv:1912.02423*, 2019.

[14] Alexander Liu, Joydeep Ghosh, and Cheryl Martin. Generative oversampling for mining imbalanced datasets. *DMIN*, 7:66–72, 2007.

[15] Patryk Orzechowski and Jason H Moore. Generative and reproducible benchmarks for comprehensive evaluation of machine learning classifiers. *Science Advances*, 8(47):eabl4747, 2022.

[16] Md Awsafur Rahman, Bishmoy Paul, Najibul Haque Sarker, Zaber Ibn Abdul Hakim, and Shaikh Anowarul Fattah. Artifact: A large-scale dataset with artificial and factual images for generalizable and robust synthetic image detection. In *2023 IEEE International Conference on Image Processing (ICIP)*, pages 2200–2204, 2023.

[17] Lanz Vincent Vencer. Bottles synthetic images, 2022.

[18] Aml Yasser. Blonde vs black hair images, 2024.

[19] Aml Yasser. Generated speed bumps, 2024.

[20] Aml Yasser. Generated street potholes, 2024.

[21] Aml Yasser and Mariam. Generated traffic lights, 2024.

[22] Jinsung Yoon, James Jordon, and Mihaela Schaar. Radialgan: Leveraging multiple datasets to improve target-specific predictive models using generative adversarial networks. In *International Conference on Machine Learning*, pages 5699–5707. PMLR, 2018.