AlteredNet - Final Project Report

1. Introduction

1.1 Project Template

My project is based on the Gather Your Own Dataset from the Machine Learning and Neural Networks module.

1.2 Project Concept

AlteredNet is a dataset of human photos that stores two versions of each image, original and digitally modified. AlteredNet is designed to solve binary classification tasks, making a distinction between classes:

- Class 0: digitally modified images.
- Class 1: real images.

To achieve meaningful training results, both the original and the modified version of each image are included in the dataset, allowing a direct and labeled comparison between the two.

Figure 1. Comparison between real and digitally modified images.

Each real image from Class 1 is modified based on a single criteria such as an increase in age, or facial expression. The modification criteria can be used as a sub-class, dividing the samples into 10 classes, in addition to the main 2.

1.3 Aims

In this project, I am aiming to explore whether neural networks can be trained to distinguish between real and digitally altered imagery, given only 300 input samples. I am aiming to test whether a small, but meticulously curated dataset of high quality examples, may be sufficient to accomplish the task.

Additionally, I would like to explore whether humans can classify AlteredNet better than a set of neutral networks. I am aiming to surpass the user-tested scores with the VGG-19 and AlexNet models, showing that AlteredNet can serve as an additional layer of protection from online fraud.

1.4 Motivation

- **Identification:** I would like to help identify digitally manipulated imagery, especially as Generative AI use becomes more and more prominent in daily life. While previously, producing realistically altered imagery required a level of skill and familiarity with design tools, nowadays it only requires a well-written text prompt that can produce the same result with no effort. Therefore I am motivated to help my fellow humans detect and better determine whether a given image is real.

- **Impersonation and Fraud:** I would like to help detect fraudulent online activity such as phishing accounts, and dating bots. I am motivated to help prevent individuals from sending their money or personal information to scammers.

- **Restore Trust:** I would like my project to help restore trust in the online community. Much of the social media content is distorted, altered or entirely generated from a text prompt, and it is hard to believe anything seen or heard online. If I create a tool that accurately detects image manipulation, then it might be useful in resolving the trust issues.

Figure 2. Feature addition or removal examples.

2. Literature Review

2.1 Similar Projects

2.1.1 CIFAKE: Image Classification and Explainable Identification of AI-Generated Synthetic Images[1]

CIFAKE is a dataset that combines 60,000 AI-generated images of animals and vehicles with 60,000 real images taken from the famous CIFAR-10[2] Dataset. It consists of 2 classes, real or fake, and 10 sub-classes that represent the image object: either airplane, automobile, bird, cat, deer, dog, frog, horse, ship or truck.

2.1.1.1 Methodology

The fake imagery was generated with the Stable Diffusion v1-4[3] model using text-to-image instructions. The instructions were assembled from the 10 classes of CIFAR-10, that serve as subclasses in CIFAKE, and a list of modifiers that describe similar entities. For example, to generate images of airplanes, Stable Diffusion received prompts that start with “a Photograph of {a/an}” and end with a modifier such as: “aircraft”, “airplane”,“jet’, or etc.

2.1.1.2 Evaluation Metrics

CIFAKE was evaluated with precision, recall, F1 and accuracy scores across different architecture neural networks with several combinations of fully connected layers and hidden units.

2.1.1.3 AlteredNet Comparison

There are several differences between CIFAKE and my project:

- **Domain:** the domain of my project is human subjects, while the domain of CIFAKE are animals and vehicles.
  Image Generation Process: AlteredNet will be generated with an image-to-image process, while CIFAKE was generated with text-to-image instead.
- **Curation Technique:** AlteredNet is manually generated and curated, while the synthetic image generation of CIFAKE was entirely automated. As stated in the CIFAKE paper under section A. Dataset Exploration, some images resulted in “visual imperfections”, “visual glitches", and “lack of important detail” that affected the quality of the dataset. For this reason, I intend to manually curate my images.

---

[1] Jordan J. B., Ahmad L., "CIFAKE: Image Classification and Explainable Identification of AI-Generated Synthetic Images," in IEEE Access, vol. 12, pp. 15642-15650, 2024.
[2] Alex K., Vinod N., Geoffrey H., “CIFAR-10”, Canadian Institute for Advanced Research, University of Toronto, 2019.
[3] Robin R., Andreas B., D., Patrick E., Björn O., “High-Resolution Image Synthesis with Latent Diffusion Models”, arXiv:2112.10752 [cs.CV], 2021.

- **Sample Shape:** Similar to CIFAR-10, the shape of each sample is 32 pixels wide x 32 pixels high x 3 colour channels. The shape of AlteredNet images would be 512 x 512 x 3 to allow more detail and context.

2.1.2 Real and Fake Face Detection

This dataset is composed of 1081 real images and 960 images that were created with editing tools by the Department of Computer Science, in Yonsei University. Components like eyes, nose, and mouth were extracted from one real image and placed on another to generate the fake subset.

2.1.2.1 Methodology

Very limited information is available about the dataset in English, however Kaggle states that the fake imagery was manually generated using Photoshop.

2.1.2.2 Evaluation Metrics

No evaluation was provided on Kaggle, however, several user-provided notebooks chose to evaluate with accuracy metrics.

2.1.2.2 AlteredNet Comparison

There are several differences between the Real And Fake Face Detection dataset and mine:

- **Methodology:** even though I intend to use Adobe Photoshop to alter the imagery, I will use the Generative Fill capability that was not available during the curation of the Real and Fake Face Detection Dataset. Instead of manually combining image elements, I will provide text prompts to Adobe Firefly that is integrated within the current version of Photoshop.
- **Repetition of Features:** The Real and Fake Face Detection dataset involves a high degree of repetition. One feature from a real image can repeat in multiple fake images, which I believe may present challenges during training. AlteredNet is designed to store only two images with repeating features, one real and one fake.

Figure 3. Examples of feature repetition from the Real and Fake Face Detection dataset.

2.1.3 Image Manipulation Dataset[4]

A dataset dedicated to detecting copy-move forgery that occurs when content is copied and pasted within the same image. It consists of 48 base images that were manually edited using copy-move techniques, and hundreds of samples that were auto-generated from them.

2.1.3.1 Methodology

The manipulated images were manually generated by several artists, instructed to create realistic-looking copies in high resolution. Copies of each tampered region were provided as well, allowing the authors to build a software framework around them. This framework was then used to create random variations of the tampered regions, and consequently, automating the generation of more tampered images. It is unclear, however, how the original 48 base images were obtained.

2.1.3.2 Evaluation Metrics

The Image Manipulation Dataset homepage does not present evaluation results, and the official source of the research publication, IEEE Transactions on Information Forensics and Security, vol. 7 is not publicly accessible.

2.1.3.3 AlteredNet Comparison

- **Automated Curation:** AlteredNet includes 2 manually curated variations of each image rather than hundreds of automatically-generated variations that the Image Manipulation Dataset includes.
- **Copy-move Focus:** While AlteredNet may add or remove image features, the applied technique will not necessarily be a copy-move forgery, but an array of Photoshop Generative Fill techniques or neural filters.

2.1.4 HQ-Edit: A High-Quality Dataset for Instruction-based Image Editing[5]

HQ Edit is a dataset of high resolution images of 900 × 900 pixels, that consists of 200,000 base images that were generated and modified using the GPT-4V and DALL-E 3 models in fully-automated means.

2.1.4.1 Methodology

An initial set of 293 was curated from online sources and the Emu Edit[6] dataset, and based GPT-4V was responsible for the manipulation instructions, according to which DALL-E3 altered the images. These images were then organized in triplets of input and output image descriptions

---

[4] V. Christlein, C. Riess, J. Jordan, C. Riess, E. Angelopoulou: "An Evaluation of Popular Copy-Move Forgery Detection Approaches", IEEE Transactions on Information Forensics and Security, vol. 7, no. 6, pp. 1841-1854, 2012.

[5] Mude H., Siwei Y., Bingchen Z., Yichun S.i, Heng W., Peng W., Yuyin Z., Cihang X., "HQ-Edit: A High-Quality Dataset for Instruction-based Image Editing", arXiv:2404.09990 [cs.CV], 2024.

[6] Sheynin S., Polyak A., Singer U., Kirstain Y., Zohar A., Ashual O., Parikh D., Taigman Y.: "Emu Edit: Precise Image Editing via Recognition and Generation Tasks", arXiv:2311.10089 [cs.CV], 2023.

as well as the edit instructions to transform the input image into the output image. GPT-4 was then used to expand the original set of 293 triplets into 100,000 different instances. DALL-E then proceeded to perform the instructions, generating both input and output images to compose the dataset.

2.1.4.2 Evaluation Metrics and Results

The authors proposed a new set of evaluation metrics called Alignment and Coherence, to quantitatively evaluate the quality of editing.
Alignment measures the semantic consistency of image edits given their prompt, ensuring the modification accurately portrays the prompt. Coherence measures the overall aesthetic quality of the edited images in terms of lightning, shadow consistency, style and edge smoothness.
In addition to the training data automation, the evaluation evaluation process was also automated using GPT-4.

2.1.4.3 AlteredNet Comparison

- **Full Automation:** While the fully automated techniques are impressive and save valuable time, they may sometimes generate inaccurate output or misinterpret instructions. AlteredNet will not be automatically generated, at least not in its current iteration.
- **Evaluation Metrics:** The metrics used in the HQ-Edit dataset are inapplicable for the classification task that AlteredNet aims to perform.

2.2 Image Classification Models

I intend to use several state-of-the-art image classification models to evaluate AlteredNet:

2.2.1 AlexNet

AlexNet[7] is a Deep Convolutional Neural Network architecture that was developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. AlexNet was introduced in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) of 2012, and consequently was trained on a subset of the ImageNet[8] dataset. The subset consisted of 1,200,000 training images, 50,000 validation images, and 150,000 testing images, that described 1,000 categories of synonym sets.

Even though ImageNet stores varying-resolution images, AlexNet was designed for image inputs with a fixed size of 224 pixels × 224 pixels × 3 colour channels. As a result, ImageNet data had to be resized to 256 pixels on the short-hand side, then center-cropped to a width and height of 256 × 256 pixels, followed by an additional arbitrary region crop to 224 × 224 pixels.

---

[7] Alex K., Ilya S., Geoffrey H., "ImageNet Classification with Deep Convolutional Neural Networks", Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, Pages 1097–1105, December 2012.
[8] Jia D., Wei D., Richard S., Li-Jia L., Kai L., Li F., "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255.

The arbitrary region crop allowed random sampling and reflected that ideal classification conditions rarely occur in real-life.

2.2.2 VGG

VGG[9] is a Deep Convolutional Neural Network developed by Karen Simonyan and Andrew Zisserman from the Visual Geometry Group, University of Oxford. VGG is available with 16 and 19 layer configurations: either 13 or 16 convolutional layers, two fully connected layered and one output layer.

Similarly to AlexNet, VGG was trained on a subset of the ImageNet dataset with a distribution of 1,300,000 training images, 50,000 validation images and 100,000 test images. Each image was resized and cropped according to the same principles as AlexNet, and GPU processing was utilized as well to accelerate training.

2.4 Research Studies

The effectiveness and necessity of my project is supported by several research papers:

2.4.1 Can people identify original and manipulated photos of real-world scenes?[10]

This study explored people's ability to detect real and digitally altered imagery, spanning across 2 experiments.

The first experiment involved 707 subjects, who completed an online survey to classify the authenticity of a set of ten images. The images were randomly selected and manipulated according to several principles:

a. enhancing human subject appearance with airbrushing techniques.
b. addition or subtraction of image elements.
c. applying physically implausible geometrical inconsistencies.
d. applying shadow inconsistencies.
e. a combination of all manipulation principles.

The participants were asked to determine if an image was manipulated, allowing the option to indicate in which region of the photo the manipulation occurred. Each photo was divided into 9 regions, and subjects were given an unlimited amount of time to provide an answer.

---

[9] Karen S., Andrew Z., "Very Deep Convolutional Networks for Large-Scale Image Recognition", arXiv:1409.1556 [cs.CV], 2014.
[10] Sophie N., Kimberley W., Derrick W., "Can people identify original and manipulated photos of real-world scenes?", Cogn. Research 2, Article 30, 2017.

The results indicated that 66% of the photos were correctly classified, combining the score of photos where a manipulation was merely detected, and the photos where the manipulated region was indicated as well.

The results also show that airbrushed photos were the most difficult to detect and locate, while the combination of multiple manipulations (principle e) was the easiest to detect and locate.

The second experiment was conducted with similar principles with a total of 659 subjects. However, unlike the first experiment, selecting a manipulated region was mandatory rather than optional, and each image was divided into 12 regions rather than 9. The results showed that the subjects correctly classified only 62% of the photos.

3. Design

3.1 Usage

3.1.1 Domain

Computer vision is the domain of my project, as it includes image classification, and a digital detection of visual patterns. On a deeper level, the project targets authenticity concerns emerging from new developments in the generative AI sphere, in hopes of mitigating them.

3.1.2 Users

The AlteredNet project is designed for researchers and students, with some technical background in Python programming and machine learning. It is also intended for individuals familiar with interfaces like Jupyter Notebook or Git, as these are the preferred means of project distribution and usage.

3.3 Structure

3.3.1 Sample Structure

3.3.1.1 Shape

Each sample is an image, designed according to the following principles:

- 512 x 512 pixels in width and height, inspired by the output size of the Stable Diffusion 1-4 Model.
- RGB colour mode with a depth of 3 colour channels, to accommodate the requirements of AlexNet, and VGG.
- saved in a PNG format, inspired by the CIFAR-10 dataset specifications.

3.3.1.2 Equal criteria representation

Since different images are suitable for different kinds of manipulations, I decided to track the manipulation criteria to ensure a balanced representation in the data. For example, only images of people with hats can be manipulated based on the “remove hat” criteria.
To make sure that one criteria is not represented more than another, “remove hat” has the same number of samples as “add happiness”, or any of the rest of the criterias.

Tracking the manipulation criteria was also inspired by CIFAKE, as it tracked the CIFAR-10 classes of animals and vehicles, in addition to the CIFAKE classes of either “real” or “AI-generated”.

The 10 manipulation criterias are:

- add/remove happiness.
- add/remove mustache.
- add/remove hat.
- add/remove glasses.
- add age.
- remove skin texture.

The criteria “remove age” age was excluded given the subtle effect that the filter manifested, and “add skin texture” was excluded to avoid an odd number of 11 manipulation criterias.

3.3.2 Label Structure

3.3.2.1 Class Labels

The class labels of different subsets, training, testing and validation, are saved in separate CSV files. The CSV data stores a set of image file names, alongside the class number they represent. For example, sample “100_fake.png” is labeled with the class number of 0, which represents the class name of “fake”.

3.3.2.2 Image Labels

The name of each image file is saved in the convention of <base image id>_<class name>.png. This type of structure allows an automatic generation of label files, in the desired convention mentioned in 3.3.2.1.

3.3.3 Model Classifier Architecture

To ensure the optimal model classifier architecture is selected for VGG-19 and AlexNet, I automated a process of hyperparameter sampling. I combined different numbers of hidden layers, with different rates of dropout, across different numbers of epochs and with possible expansion to rates of weight decay. As a result, only the best of 9 model architectures was selected for evaluation, one for VGG-19 and one for AlexNet.

3.4 Techniques and Methods

3.4.1 Image Manipulation with Adobe Photoshop

The AlteredNet dataset samples were generated using Adobe Photoshop version 25.11, and its integrated Generative AI Model, Adobe Firefly.
Each base image was cropped and scaled to a maximum of 2500 x 2500 pixels. Then, one of 10 available alternations was implemented using a neural filter or a Generative Fill.

- **Mustache, hat, and glasses criterias:** were implemented with the Generative Fill feature, selecting the area of the image to be manipulated, and specifying the desired prompt. For example “baseball hat” or “old man mustache” suitable for the “add” modifier, along with “remove” or “shaved, smooth skin” suitable for the “remove” modifier.
- **Happiness and age criterias:** were implemented with the Smart Portrait neural filter, setting the degree of relative “happiness” between -30 and +30, or “age” to +50.
- **Remove skin texture criteria:** implemented with the Skin Smoothing neural filter, setting both the blur and the smoothness to +50.

The manipulated area was stored as a Smart Filter layer, that allows toggling between the original and altered views. The double layered Photoshop files were stored on my local file system, and from each file two samples were derived: a <id>_real.png and a <id>_fake.png.

Retaining the working files not only ensures that samples can be revised if necessary, but also provides a means of expansion into higher resolution images of up to 2500 x 2500 pixels.

3.4.2 Pytorch Classification

AlteredNet was trained and evaluated using the Pytorch deep learning framework, in the version 2.4.0+cu121. Pytorch provides the pretrained VGG-19 and AlexNet models along with the functions, classes and algorithms to facilitate fine-tuning and classification. For example, commands for transforming images, converting them into tensors and loading them into batch-based structures, expected by the neural networks. Pytorch is also compatible with parallel computing on GPU, using the CUDA platform.

3.4.3 CUDA

To accelerate the speed of training, I used a software platform named CUDA, delegating selected processing tasks from the CPU to the GPU. Since GPUs have substantially more processing cores than CPUs, more tasks can be performed in parallel.
For example, my Intel i9-13980HX CPU is capable of performing 32 tasks in parallel, given 8 processing cores that can run 2 tasks at once, and 16 efficiency cores that can only run a single task. While my NVIDIA GeForce RTX 4090 Laptop GPU is capable of performing 9,728 tasks in parallel, one for each CUDA core.

This technique allowed me to train different iterations of each model within minutes rather than hours, and perform efficient calculations on multidimensional data structures like the input tensors, expected by VGG-19 and AlexNet.

3.4.4 Pre-trained Neural Networks

The relatively small size of AltereNet, compared to similar projects, may present a training challenge. In order to find mathematical patterns within data, a large number of examples must be provided to allow proper generalization. With a small number of samples, it is hard to determine if something is a trend, or if it is just a coincidence. Therefore I chose to work with pre-trained neural networks that were already exposed to millions of samples in other domains. Instead of a custom model based on AlteredNet samples only, I decided to leverage VGG-19 and AlexNet that were trained on a subset of the ImageNet dataset, classifying 1000 different synonym sets. This technique was meant to compensate for the lack of AlteredNet samples, and allow better generalization given the prior experience of the models.

3.5 Base Image Resource

All base images were obtained from a stock image website named Freepik. I manually selected a diverse array of images of individuals with different qualities, ethnicities, age, facial expressions, and other features.

3.5.1 Resource License

Freepik holds a custom license agreemen, that allows non-commercial use of its assets. In addition to carefully reading the agreement, I reached out to Freepik via email to ensure that I am legally authorized to use their assets in my dataset. They have replied to me and confirmed my use case is appropriate.

3.6 Evaluation Plan

To evaluate the dataset quality, I performed user testing that established a baseline model, and compared it with the AlteredNet-trained versions of Vgg-19 and AlexNet. The models were compared in terms of accuracy, precision and recall, similar to other research in section 2.

3.6.1 Evaluation Metrics

3.6.1.1 Precision

The precision metric measures how many samples were correctly classified as positive out of all the samples that were classified as positive. Precision can be calculated using the following formula, where a higher precision score indicates a higher-quality classification:
Precision = TP / (TP + FP)

The highest precision score possible is 1.0, which indicates the complete absence of False Positive samples (samples that were predicted as positive, but they were actually negative).

3.6.1.2 Recall Metric

The recall metric measures how often positive samples are correctly classified, and can be calculated using the following formula:
Recall = TP / (TP + FN)

The highest recall score possible is also 1.0, which indicates the complete absence of False Negative samples (samples that were predicted as negative, but they were actually positive).

3.6.1.3 Accuracy Metric

Lastly, the accuracy metric measures how many samples were correctly classified out of all samples. This score includes both negative and positive classifications, and can be calculated using the following formula:
Accuracy = TP + TN / (TP + TN + FP + FN)

The highest accuracy score possible is also 1.0, which indicates that all samples were correctly classified with no exceptions.

3.7 Baseline Model

The baseline model that serves as a dataset quality point of reference, was established entirely through user testing. Such that if VGG-19 and AlexNet can classify AlteredNet with better accuracy, precision and recall then humans - then the project was successful.

4. Implementation

4.1 Dataset

The dataset consists of 260 training images, 40 validation images and 30 test images. All images are 512 x 512 pixels in size, 3 colour channels deep, and saved in a PNG format.

4.2.1 Train and Validation Subsets

4.2.1.1 Train and Validation Samples

The training and validation subsets were generated after the prototyping stage. A total of 150 base images were altered, resulting in 300 samples that represent 2 classes -
[0] fake and [1] real.

The samples are identified by their base image and class name. For example, the sample set with the base image id 138 consists of two image files: “138_fake.png” and “138_real.png”.

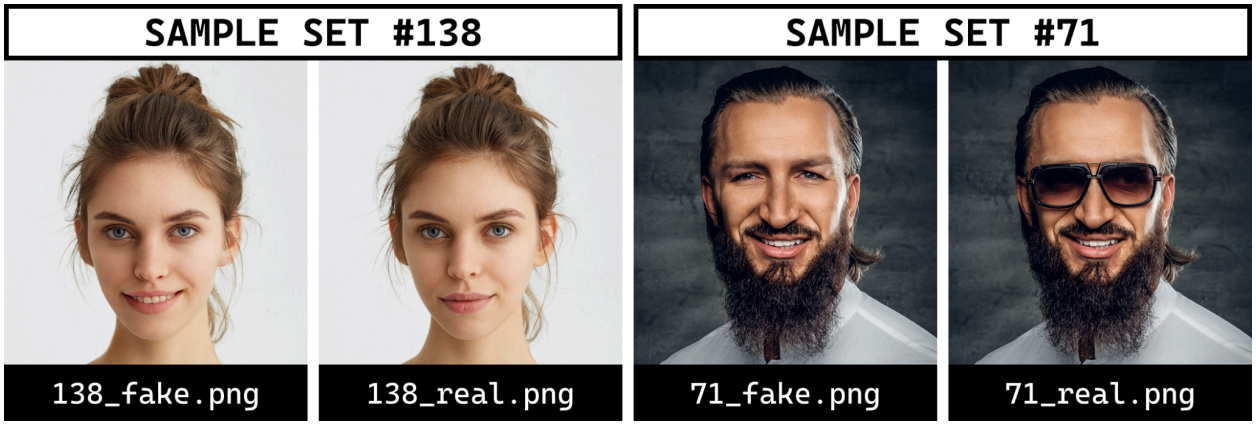

Figure 4. Training Samples and Their File Names.

4.2.1.2 Train and Validation Labels

The label files are automatically generated inside the AlteredNet_Label_Generator.ipynb notebook, located in the data directory of the project repository.
It fetches the names of all image files within the train, valid and test directories, and determines their labels from the image name. For example, the image “71_fake.png” contains the string “fake” and therefore labeled as class 0.
The automated label generator was designed for dataset scalability purposes. It accounts for a quick integration of new sample sets, as long as they are adequately named.

4.2.1.3 Train and Validation Altering Criteria

The altering criteria of each sample set is stored in criteria.txt, located within the data/criteria directory. The file contains unique base image ids along with the altering criteria they represent (as both the real and the fake versions, represent the same altering criteria).
For example, sample set 138 represents the altering criteria of “add happiness” while sample set 71 represents “remove glasses”.

The criteria file serves two purposes:

- **Equal Representation:** Maintaining an equal number of samples per criteria, ensuring a balanced representation.
- **Project Scalability:** accounting for a possible expansion of the AlteredNet dataset, application of additional use cases, or further analysis solving multi-class problems.

4.2.2 Test Subset

4.2.2.1 Test Samples

1/3 of the testing subset was generated during the prototyping stage, where 10 base images were altered, resulting in 20 samples. The rest of the test samples were generated separately

from the training and validation sets, and named according to a different numbering system, such that: <test sample id>_test_<class name>.png. For example: “10_test_real.png”.
To ensure a balanced representation, the test set consists of 15 fake images and 15 real ones.

4.2.2.1 Test Labels

Similarly to the training and validation sets, the test labels are automatically generated based on test image names.

4.2 Public Project Repository

The project, including the dataset, the classifier notebook, and the model checkpoints it produced, are publicly available on Github, in the following repository:
https://github.com/MariyaSha/AlteredNet

4.2.1 Root Directory

Contains the README file, the data and user_testing directories, as well as the classification notebook: AlteredNet_Classification.ipynb. This notebook stores an executable classification pipeline, and the results of training, validation, testing and prediction workflows, as previously executed on my local system.

4.2.1 Data Directory

The data directory includes the AlteredNet dataset, split into training, testing and validation subsets, along with a set of additional directories to allow a high degree of modularity and organization.

- **Train directory:** Contains 260 files that represent 130 base images, alongside their AI-altered counterparts. Each of the 130 real images has a matching manipulated image, stored under the same base image id but a different class name, as shown in the implementation section below.
- **Valid directory:** Contains 40 files that represent 20 base images, alongside their AI-altered counterparts.
- **Test directory:** Contains 25 files that, unlike the training and validation sets, represent either the original base image or its AI-altered counterpart. Concealing the real or fake counterpart is meant to mimic real life situations, where only one copy of the image is displayed.
- **Label directory:** Contains the automated label generator script, AlteredNet_Label_Generator.ipynb, and the 3 CSV files it generated, representing the train, test and validation set labels.
- **Checkpoint directory:** Contains 6 fine-tuned model checkpoints, saved during the epoch sampling stage of AlteredNet classification. They represent one VGG-19 and one AlexNet model, trained over 3 different numbers of epochs. For example, AlexNet_36_checkpoint.pth was trained on AlteredNet data for 36 epochs.

- **Criteria directory:** Contains the 10 subclasses to which each of the 300 samples belong, or the alternation categories discussed at section 3.3.1.2. In addition, it contains a groupby_category.ipynb notebook that counts the number of samples per category and was very helpful to me during the dataset creation process.

4.2.2 User Testing Directory

The user testing directory includes the data and the evaluation process that produced the baseline model described in section 5.1. It includes two CSV files

- **user_testing_results.csv:** stores user data from all 10 questions, and compares the total number of correct and incorrect answers for each question. This file is used to calculate the accuracy score, and find which of the samples is the most difficult or easy to detect.
- **user_testing_results_6-10.csv:** stores user data from the last 5 questions, and compares the actual labels with the predicted label, as well as the number of users who made the same prediction. This file is used to calculate the precision and recall scores.

Additionally, the user testing directory includes the jupyter notebook that was used to analyze the CSV data, **user_testing_analysis.ipynb**. This notebook is used to automate the evaluation process, and showcase the source code behind my calculations for maximum transparency.

4.3 Dataset Classification

The classification pipeline consists of several sequential processes, presented in the AlteredNet_Classification.ipynb notebook.

4.3.1 Loading Data

4.3.1.1 Custom Image Dataset

Each data subset is loaded into a CustomImageDataset object, as suggested by the Pytorch documentation. The image files along with their labels are stored as class attributes, and transformed into a tensor structure in the shape of 224 x 224 x 3.

4.3.1.2 Create Dataloader

Once each sample is transformed into the size and data type expected by VGG-19 and AlexNet, the samples are loaded into a Pytorch dataloader object. This object adds another dimension that represents a batch size, or a number of samples to be processed by the network all at once. I chose a relatively small batch size, exposing the networks to 5 samples at a time.

4.3.1.3 Dataloader Visualization

To verify that the samples were properly loaded and labeled, I created a visualization function that selects a random batch and displays it along with its class names.

4.3.2 Hyperparameter Sampling

To optimize the hyperparameters of each model, I sampled different numbers of hidden units and dropout rates. Additionally, I adapted VGG-19 and AlexNet to the classification requirements of AlteredNet.

4.3.2.1 Model Customization

Initially, each pre-trained model is loaded from Pytroch, and its classifier is customized to include only a single fully connected layer, 2 output units, and a Log Softmax activation function. Essentially, I reduced the size and dimensionality of the classifier to account for the small number of dataset samples, and prevent overfitting.

4.3.2.2 Sample Models Dictionary

To generate several model versions rather than one, I created an additional function that receives a list of dropout rates and a list of hidden unit numbers, and generates a dictionary of models. Each dictionary stores the model id, its dropout rate, its number of hidden units and the torchvision model object itself.
This dictionary facilitated the process of training multiple models, and identifying them for checkpoint saving purposes and for further exploration with epochs sampling.

```
vgg_models = sample_models("vgg", [512, 448, 372], [0.8, 0.6])
alexnet_models = sample_models("alexnet", [256, 172, 128], [0.2, 0.5])
```

Figure 5. Generate sample models dictionary, passing lists of hidden units and dropout rates.

4.3.2.3 Hyperparameter Selection

Once all the models completed their training, their accuracy and loss scores were plotted, comparing the validation and training rates. Only one of each model that displayed the most gradual reduction in validation and training loss, and the most gradual increase in validation and training accuracy, was selected to proceed to the next sampling stage.

4.3.3 Epoch Sampling

To optimize the number of epochs and ensure minimal overfitting, if any, the best performing models from the hyperparameter sampling stage were evaluated across different numbers of epochs. Additionally, to provide a wider context, instead of tracing the results of each epoch, the results were traced in a higher frequency - 5 times per epoch.
At the end of the epoch sampling stage, all model checkpoints were saved on the file system, and evaluated based on a set of metrics. Additionally, similar to the hyperparameter optimization stage, the training accuracy and loss was plotted against the respective validation metrics, which helped determine the degree of overfitting.

4.3.4 Evaluation

The evaluation was performed on 30 test images, returning accuracy, precision and recall scores, along with training performance graphs. All epoch sampling variations were evaluated, however only one was selected to proceed with the last stage of prediction.
The selected variation, in both cases, displayed the least overfitting, and surpassed the base model accuracy and recall scores. Since only one variation displayed lower precision than the base model, I chose to focus on the other metrics instead.

4.3.5 Prediction

The final prediction stage was performed on a single image from the test set, providing the means of incorporating saved model checkpoints in a production environment. The best performing VGG-19 and AlexNet checkpoints were loaded, and used to predict the class of a local image. Even though the implementation uses test samples for prediction, it is also suitable for user-provided images from any source, not just AlteredNet.
As a result, users may upload any image suspected of altering, and receive a classification result along with a confidence score. For example, “VGG predicts the image is real with 72% confidence”.

```
import numpy as np
from PIL import Image

vgg_model = load_checkpoint('data/checkpoints/VGG_24_checkpoint.pth')
alexnet_model = load_checkpoint('data/checkpoints/AlexNet_10_checkpoint.pth')

compare_predictions("data/test/10_test_real.png", 1, vgg_model, alexnet_model, topk=1)
```

```
VGG predicts the image is real with 0.97% confidence
----------------------------------------------------
AlexNet predicts the image is real with 0.54% confidence
----------------------------------------------------
```

Figure 6. Multi-model prediction functions for user provided images.

5. Evaluation

5.1 Baseline Model User Testing

To establish a baseline model, I conducted user testing on 10 AlteredNet samples.

5.1.1 Participants

47 viewers of a YouTube channel named Python Simplified volunteered to participate, submitting a Google form and sharing their class predictions.

These viewers belong to a community of Python learners and AI development enthusiasts - the ideal user base for the AlteredNet project.

5.1.2 Test Settings

The test was conducted during a live broadcast, by the end of which 39 form submissions were received. The remaining 8 were submitted while viewing the broadcast on-demand.
I presented the test outline in verbal and visual means, in addition to the embedded instructions and examples found within the form.

Figure 7. Criteria Examples from User Testing Form.

5.1.3 Test Content

The user input was received through a digital form with 10 multiple-choice questions regarding assorted AlteredNet samples. The participants were required to determine whether each sample was authentic or manipulated, across 2 categories of questions.

5.1.3.1 two-sample questions

The first 5 questions of the form compare between pairs of samples, real and altered, originating from the same base image. These questions are meant to mimic the validation process, where both the real and fake samples are reviewed, and classified according to a neural algorithm.

5.1.3.2 one-sample questions

The last 5 questions of the form present a single sample and inquire users about their class. These questions are meant to mimic the final evaluation process, where either the real or fake sample is reviewed. Additionally, these questions allowed collecting precision and recall scores, in addition to the accuracy score collected by all 10 questions.

5.1.4 Test Results

5.1.4.1 Evaluation Metrics

- **Accuracy:** correct / (correct + wrong) = 266 / (266 + 204) = 266 / 470 = 0.56
  based on all 10 user test samples.
- **Recall:** TP / (TP + FN) = 61 / (61 + 33) = 61 / 94 = 0.65
  based on the last 5 user test samples.
- **Precision:** TP / (TP + FP) = 61 / (61 + 70) = 61 / 131 = 0.46
- based on the last 5 user test samples.

5.1.4.2 Detection Difficulty

I also recorded the samples that were the least and most challenging to detect, based on the difference between the correct and incorrect answers for each sample.

- The **easiest** sample to detect is sample 5 by a margin of 39.
- The **hardest** sample to detect is sample 1 by a margin of 3.

Figure 8. user test samples, 1-5 are all altered, 6-10 are either altered or real.

5.1.5 Result Visualization

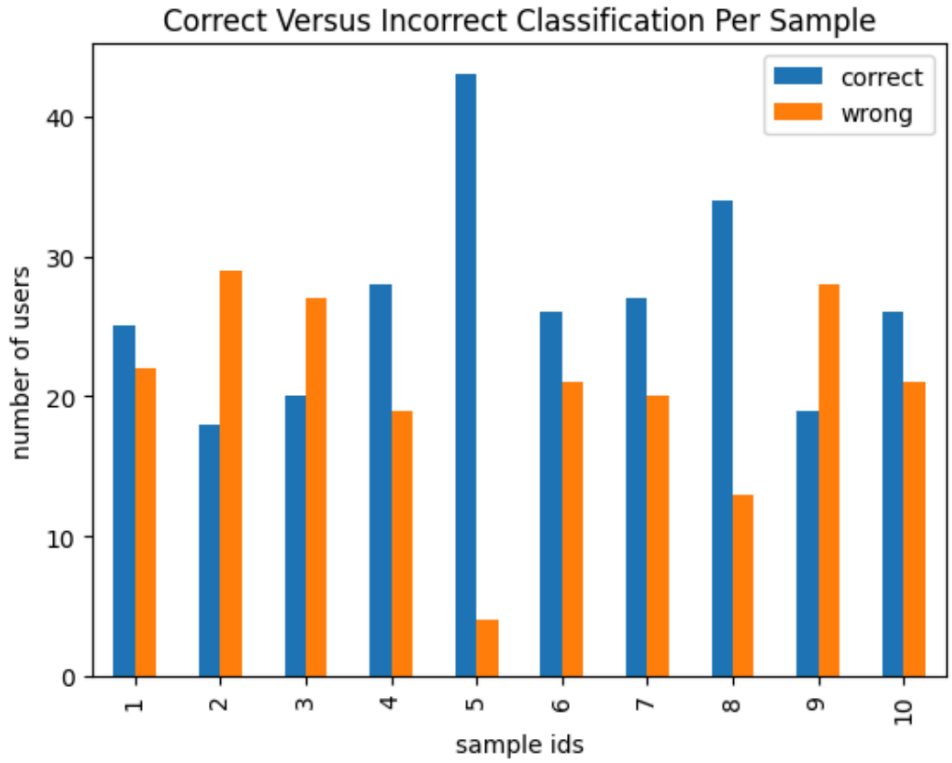

Figure 9. distribution of correct and incorrect answers per sample.

5.2 AlteredNet Evaluation

The AlteredNet evaluation for each of the pre-trained models was conducted in several stages.

5.2.1 Hyperparameter Optimization

Since VGG-19 and AlexNet were pre-trained on ImageNet, their parameters are optimized for it, and not for AlteredNet. To explore different parameter combinations, I automated a model sampling process where different versions of the same model were evaluated one against the other. The hyperparameter evaluation was performed as follows.

5.2.1.1 Comparing Training and Validation Accuracy Over Epochs

The training and accuracy scores of each model were collected at the end of each epoch, and evaluated across 6 different hyperparameter variations. Only models where both training and validation accuracy were increasing over epochs were highlighted, as shown in the VGG-19 visualization below. In model 2, the validation accuracy did not increase, and in model 4 the validation accuracy substantially surpassed the training accuracy, which is unusual.
It may indicate challenges with the regularization techniques I used during training - dropout and weight decay.

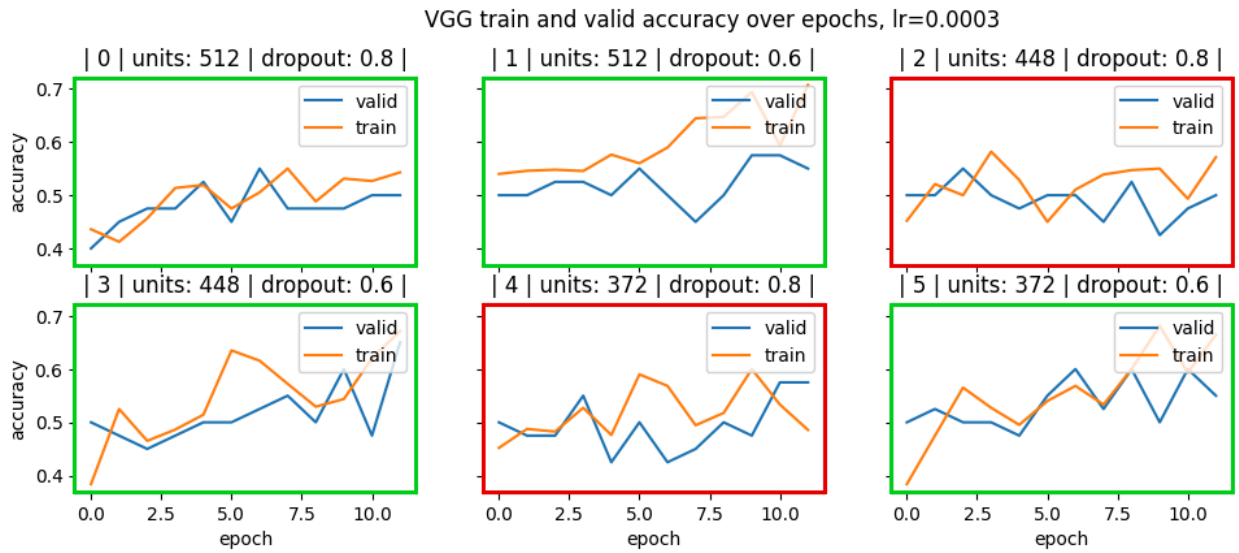

Figure 10. models highlighted in green perform as expected, models highlighted in red underperform.

5.2.1.2 Comparing Training and Validation Loss Over Epochs

Similarly, the loss metric was collected as well, highlighting the models where both the training and validation loss decreased over epochs. In models 0-2, the training loss decreased while the validation loss increased, which means the model was overfitting. It memorized the training data too well, and therefore it was unable to correctly classify new data.

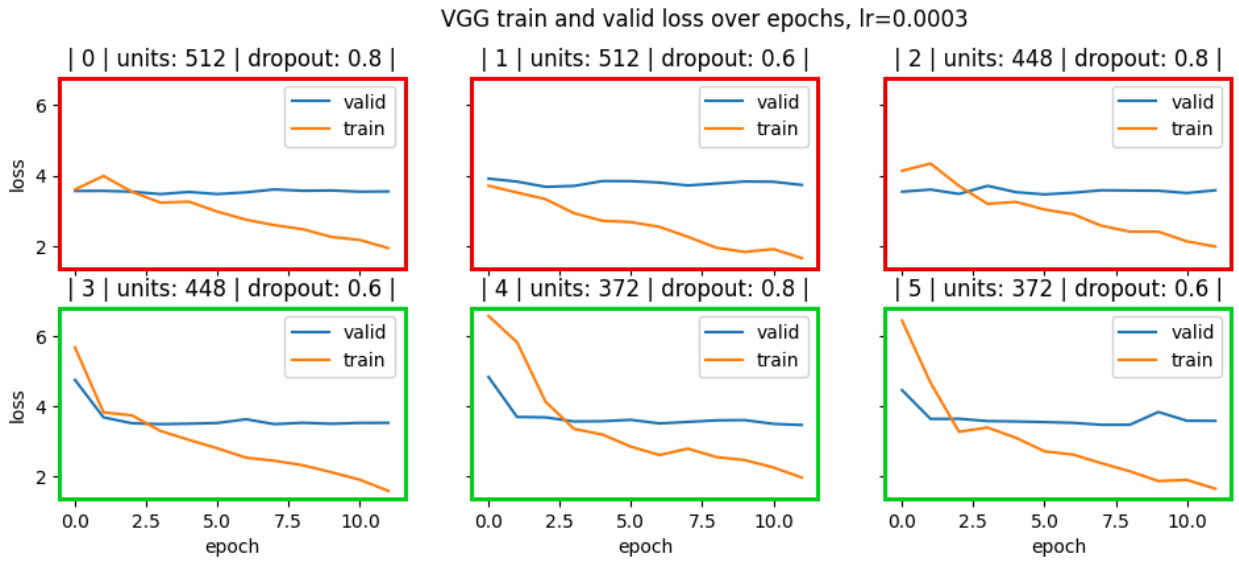

Figure 11. models highlighted in green perform as expected, models highlighted in red underperform.

5.2.1.3 Comparing Highlighted Models

Lastly, all models that performed as expected both in terms of accuracy and loss, were evaluated in the last stage of selection. Their graphs were thoroughly analyzed, and only model #3 proceeded to the epoch evaluation stage given the following factors, observed in the figure below:

- The accuracy of model #3 improved from 0.5 at epoch 0 to 0.65 at epoch 12, while the accuracy of model #5 only improved from 0.5 to 0.55.
- Even though both models show a sudden validation loss increase around epoch 6 for model #3 and epoch 9 for model #5. The loss spike from model #3 is more subtle, and occurs earlier in the training process, when the training accuracy is relatively lower, so is less significant than a higher spike that occurs at a later stage.

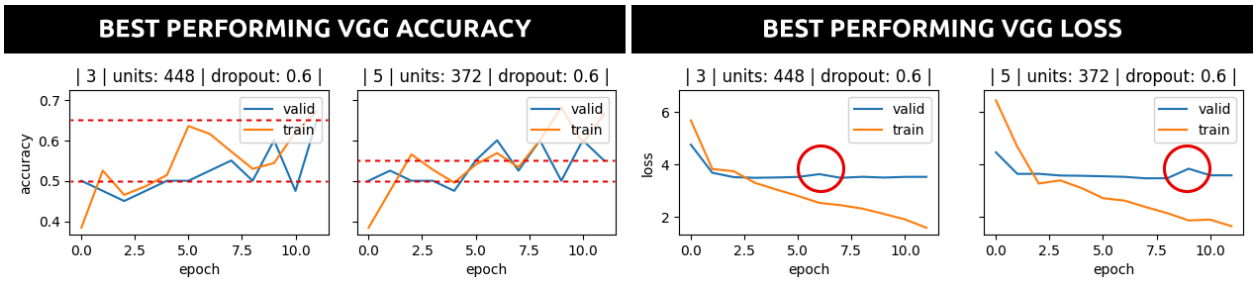

Figure 12. dashed line indicates the accuracy growth, red circle indicates the loss spike.

5.2.1.4 AlexNet Hyperparameter Evaluation

Similarly to the evaluation results of VGG-19 displayed above, an array of AlexNet models was evaluated as well, using the same principles.

5.2.1.4.1 AlexNet Hyperparameter Accuracy Evaluation Graph

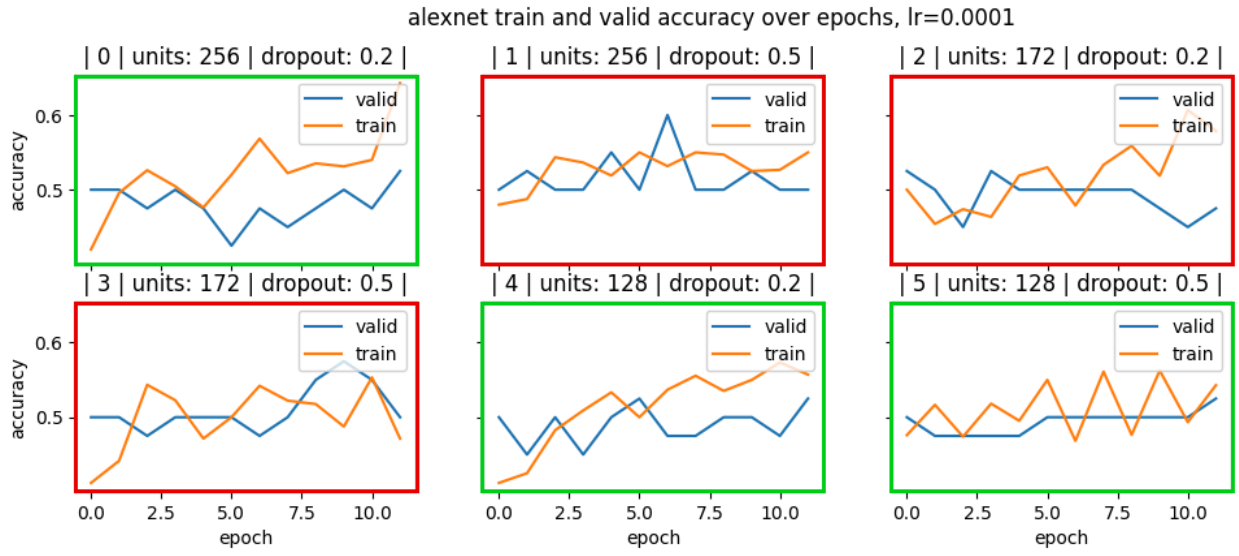

Figure 13. AlexNet Accuracy with Different Hyperparameters Over Epochs.

5.2.1.4.2 AlexNet Hyperparameter Loss Evaluation Graph

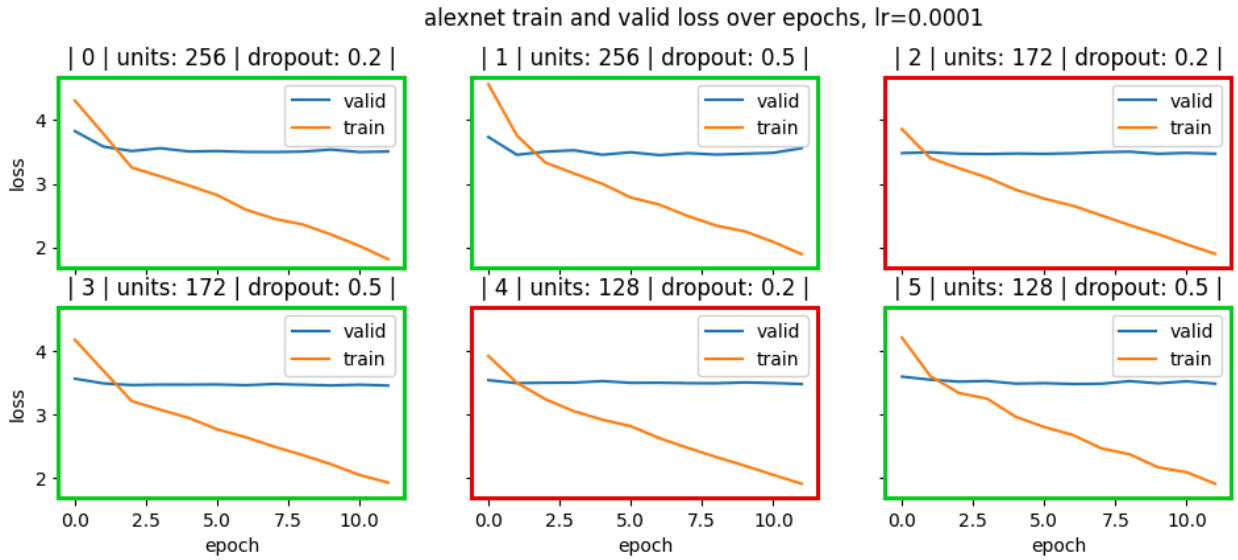

Figure 14. AlexNet Loss with Different Hyperparameters Over Epochs.

5.2.1.4.3 AlexNet Hyperparameter Highlighted Models

Since models #0 and #5 performed as expected across both metrics, they were evaluated based on the following factors:

- Even though both models display similar growth in validation accuracy, model 0 displays a substantially larger growth in training accuracy. From a nearly #0 accuracy on epoch 0, it grows to a nearly 1 accuracy at epoch 12, compared to model #5 that grows from 0.48 to 0.55.
- The validation loss of model #5 is nearly a flat line that shows an insignificant decrease, if any. On the other hand, the loss of model #0 is decreasing over epochs, even though it stabilizes around epoch 2.5.

Therefore I chose model #0 to proceed to the next evaluation stage.

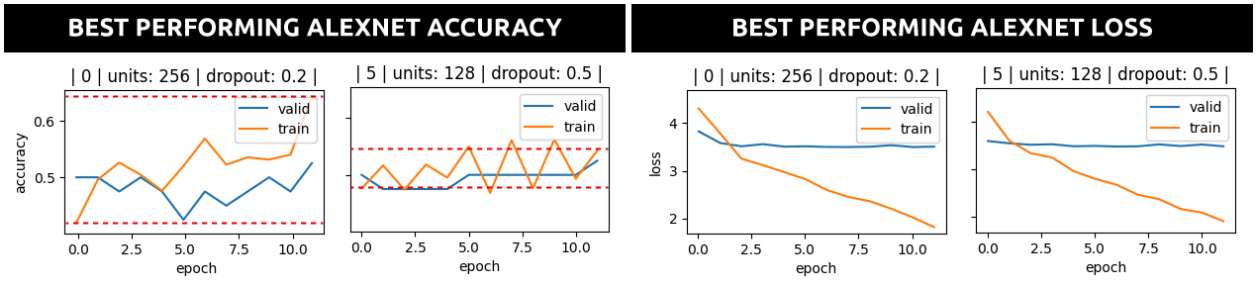

Figure 15. Highlighter AlexNet Models Selection.

5.2.2 Epoch Number Optimization

In addition to hyperparameter sampling, I explored the optimal number of epochs. Each of the best performing models was trained over 10, 24, and 36 epochs on the training and validation sets, and their results were compared to one another.

5.2.2.1 Epoch Optimization Train and Validation Accuracy

All VGG-19 and AlexNet instances displayed a gradual increase in both validation and training accuracy, as shown in the figure below.

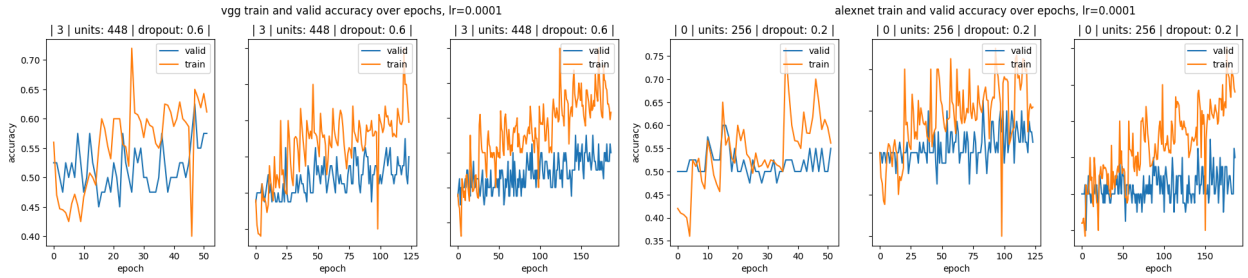

Figure 16. Epoch Optimization Accuracy Graphs.

5.2.2.2 Epoch Optimization Train and Validation Loss

Only VGG-19 trained for 36 epochs showed an increase in validation loss, which indicates overfitting. The rest of the models showed a decrease in both training and validation loss, in some cases a more significant decrease than others.

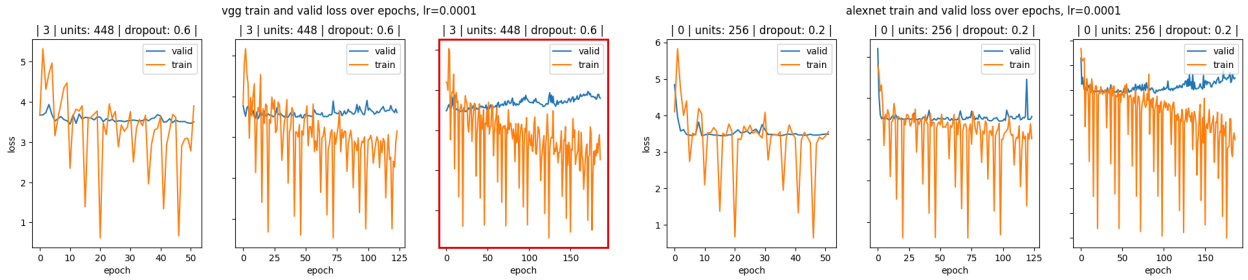

Figure 17. Epoch Optimization Loss Graphs with One Underperforming Model.

5.2.3 Evaluation on the Test Set

The evaluation on the test set was performed for all epoch-varying instances of VGG-19 and AlexNet, including the underperforming model. As the test set only consists of 30 samples, the evaluation results are obtained quickly and provide more context to the graph results. However, only one of each kind of model is selected to proceed with the final prediction stage.

5.2.3.1 VGG and AlexNet Evaluation on the Test Set

To conveniently view the evaluation results from the classifier notebook, I've arranged them in a table along with the baseline model that was obtained through user testing.

Model kind	Hidden units	Dropout rate	Learning rate	Epochs	Accuracy	Recall	Precision
VGG-19	448	0.6	0.0001	10	0.70	0.67	0.71
				24	0.63	0.87	0.59
				36	0.80	0.80	0.80
AlexNet	256	0.2	0.0001	10	0.70	0.80	0.67
				24	0.36	0.67	0.42
				36	0.46	0.60	0.47
Baseline	-	-	-	-	0.56	0.64	0.46

Figure 18. Evaluation Results on the Test Set.

- **Accuracy:** most evaluated models surpassed the 0.56 baseline accuracy, with the exclusion of 24 and 36 epoch-based AlexNet.
- **Recall:** most evaluated models surpassed the 0.64 baseline recall, with the exclusion of 36 epoch-based AlexNet.
- **Precision:** most models failed to decrease the precision score below the 0.46 baseline, with the exclusion of 24 epoch-based AlexNet.

5.2.4 Prediction

The last evaluation analysis was performed on 3 assorted test images, and compared between the 24 epoch VGG-19 and the 10 epoch AlexNet. Each network was presented with a single test image and returned its class prediction and confidence score. For a convenient view of the prediction results from the classifier notebook, I arranged them in a table.

Test Sample ID	Label	Model	Predicted	Confidence
10	1	VGG-19	1	0.97
		AlexNet	1	0.54
26	0	VGG-19	0	0.55
		AlexNet	0	0.52
5	0	VGG-19	0	0.51
		AlexNet	1	0.50

Figure 19. Prediction Results.

5. Conclusions

The results of AlteredNet classification were overall satisfying, as the evaluation on the test set achieved better accuracy and recall scores than the baseline model. Additionally, all 3 test samples were correctly classified by the 24 epoch VGG-19 network during the prediction stage. However, there is a set of limitations that can be improved upon in the next iteration.

5.1 Improvement Ideas

5.1.1 Gathering More Data

330 training, testing and validation images may not be enough to achieve consistent generalization rates for never before seen data. For example, AlexNet was trained on over a million train samples, 50 thousand validation images and 150 thousand testing images, which is substantially more data than AlteredNet offers.
As a result, the training process was unpredictable, and many of the models from the hyperparameter sampling stage were underperforming and substantially overfitting.

I tried mitigating the overfitting with dropout and weight decay regularization, however, the results only showed a minimal improvement rather than a permanent fix. For that reason, I believe that generating a greater number of samples is inevitable in future iterations.

5.1.2 Performing K-Fold Validation

One solution that I couldn't implement on time, was performing K-Fold validation, that is suitable for small subsets. My validation technique required splitting the training data into validation and training dataloaders ahead of the training process. As a result, the 40 validation images were used for validation purposes only. With K-Fold validation, on the other hand, the training data is split into k equal partitions as the training performs. Where different partitions are used for training and different partitions are used for validation at any given time, such that the neural network learns both the training and the validation data.

The reason why I failed to implement K-Fold validation was reaching my monthly Git LFS quota. Since model checkpoints are files with over 100MB in size, they cannot be uploaded through git, and must be uploaded with a Git Large File System extension. I was not familiar with Git LFS prior to this project, and was not aware that a monthly quota of 1GB uploads per user exists. Unfortunately, I reached the quota by surprise, and couldn't proceed with producing more model checkpoints that were trained on the entire 300 samples.

5.1.3 Focusing on Improving Precision Score

While the 30 test samples have been classified with greater accuracy and recall than human users - a lack of precision is evident all across the board. It means that many samples that were predicted as positive, were actually negative, and therefore the neural networks failed to recognize some of the alterations. Surpassing the precision metric on the baseline model

presented a greater challenge than surpassing its recall and accuracy. Even though a lower precision usually indicates greater accuracy, this relation wasn't reflected in the results. Therefore, I believe that the main focus of the next iteration should be improving the precision metric, which may positively affect other metrics as well.

5.2 What Worked Well

5.2.1 Automatic Label Generators

Automating the label generation process was one of the most significant decisions I made. It allowed me to focus on generating samples, and effortlessly executing the classification notebook whenever new samples are introduced.

5.2.2 Hyperparameter and Epoch Sampling

The sampling process substantially improved the final evaluation results, as only the best parameters were selected for each network. Without the sampling process, I'd be engaged in guessing random parameters rather than making informed, data-driven decisions.

5.3 What Didn't Work Well

5.3.1 Custom Model for 512 x 512 Samples

Initially, I was planning to train a custom neural network on samples in their original size of 512 x 512. However, this model did not achieve impressive results and therefore I decided to discard it from my research, focusing on establishing the sampling process for the pre-trained neural networks.

5.3.2 ResNet Training

Initially, I was planning to train an additional state of the art neural network, ResNet. However, even though ResNet was inspired by AlexNet and VGG-19, it contained a classifier structure that was not compatible with theirs.

5.3.3 Binary Classification

The biggest challenge of my project was the unexpected results received by binary classification. Even though the Sigmoid activation function and Binary Cross Entropy loss were designed for 2-class problems, they did not produce meaningful training results.
Only when I switched to Log Softmax activation and Negative Log Likelihood loss, the training accuracy started to gradually increase and the loss gradually decreased.

5.4 Ideas for Further Development

- Using the 10 sub-classes from criteria.txt as labels to solve multi-class classification problems. Beyond recognizing which image is fake or real, a model can also be trained on detecting the type of alternation that has occurred. To accomplish it, a much larger

number of samples is needed which may require automating the curation process. As currently, only 15 samples exist for each alternation criteria.
- Integrating one of the saved model checkpoints in a GUI application. Using the combination of functions from the Prediction stage, predict, process_image, and compare_predictions, to classify user-provided images. This development will help expand the user base of AlteredNet beyond individuals with programming backgrounds.

6. References

The following references are based on the footnotes found throughout the report.

[1]. Jordan J. B., Ahmad L., "CIFAKE: Image Classification and Explainable Identification of AI-Generated Synthetic Images," in IEEE Access, vol. 12, pp. 15642-15650, 2024.

[2]. Alex K., Vinod N., Geoffrey H., “CIFAR-10”: Canadian Institute for Advanced Research, University of Toronto, 2019.

[3]. Robin R., Andreas B., Dominik L., Patrick E., Björn O., “High-Resolution Image Synthesis with Latent Diffusion Models”, arXiv:2112.10752 [cs.CV], 2021.

[4]. V. Christlein, C. Riess, J. Jordan, C. Riess, E. Angelopoulou: “An Evaluation of Popular Copy-Move Forgery Detection Approaches”, IEEE Transactions on Information Forensics and Security, vol. 7, no. 6, pp. 1841-1854, 2012.

[5]. Mude H., Siwei Y., Bingchen Z., Yichun S.i, Heng W., Peng W., Yuyin Z., Cihang X., “HQ-Edit: A High-Quality Dataset for Instruction-based Image Editing”, arXiv:2404.09990 [cs.CV], 2024.

[6]. Sheynin S., Polyak A., Singer U., Kirstain Y., Zohar A., Ashual O., Parikh D., Taigman Y.: “Emu Edit: Precise Image Editing via Recognition and Generation Tasks”, arXiv:2311.10089 [cs.CV], 2023.

[7]. Alex K., Ilya S., Geoffrey H., “ImageNet Classification with Deep Convolutional Neural Networks”, Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, Pages 1097–1105, December 2012.

[8]. Jia D., Wei D., Richard S., Li-Jia L., Kai L., Li F., "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255.

[9]. Karen S., Andrew Z., “Very Deep Convolutional Networks for Large-Scale Image Recognition”, arXiv:1409.1556 [cs.CV], 2014.

[10]. Sophie N., Kimberley W., Derrick W., “Can people identify original and manipulated photos of real-world scenes?”, Cogn. Research 2, Article 30, 2017.