Determining the Authenticity of Images

Comprehensive Technical Report

IBM-322: Analytics of Managerial and Decision-Making

Date: 28/11/2023

Submitted by:

1. Yalmarthy NV Ronnit Gupta (21117144)
2. Yathartha Rana (21117149)
3. Krish Bafna (21117065)
4. Mohit Attri (21117076)
5. Karanjit Singh (21117058)

Submitted to:
Prof. Sumit Kumar Yadav
Department of Management Studies
Indian Institute of Technology, Roorkee

Content

1. Introduction

In this modern era where Artificial intelligence and machine learning are continuously evolving, AI-generated images have caught the attention of many artists and art lovers alike. With the capability of AI to generate almost authentic images under different visual settings, its use-case in animations and art is increasing daily. Like other tools, AI Image Generators are also facing backlash because of their ability to falsify information and create doubt among people over the authenticity of images shared within the community.

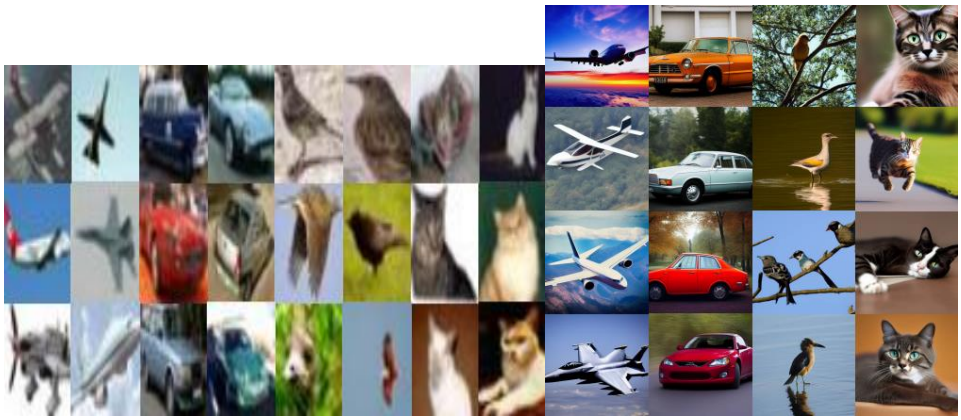
We propose using machine learning techniques to address this challenge and distinguish between AI-generated and authentic images. By harnessing machine learning, we aim to develop a solution that can differentiate between real and AI-generated images, providing a valuable tool to address concerns related to image authenticity.

2. Dataset and Preprocessing

2.1 Dataset

An extensive survey of datasets from platforms such as Kaggle, PaperswithCode, and Hugging Face was conducted to facilitate the training of our classification models. After careful consideration, the CiFAKE Dataset was identified as the most suitable for our problem.

This dataset comprises two primary classes, Real and Fake, subdivided into ten additional classes. This subdivision enhances the distinctiveness of features within the dataset, ensuring that our model is well-equipped to classify images beyond the training data. The real images in the dataset are derived from the Cifar 10 dataset, while the corresponding fake images are generated using stable diffusion 1.4.



Real Images Fake Images

2.2 Preprocessing

2.2.1 Flattening

Flattening typically refers to converting a nested data structure into a flat, one-dimensional form. Multi-dimensional arrays take more memory, while 1-D arrays take less memory, which is the most important reason why we flatten the Image Array before feeding the information to our model.

Unlike Traditional methods such as CNN utilize kernel to reduce the number of features in the images to increase training speed, we are utilizing the complete image and flattening it to avoid feature losses and improve the capability of model to classify AI generated images with real images.

2.2.2 Scaling

When referring to data or numerical values, scaling changes variable values to fall inside a given range. Standardizing or normalizing a dataset's features makes it easier to compare them or utilize them as input for machine learning methods, which is frequently the aim of scaling.

There are different scaling methods, but two common approaches are:

- **Min-Max Scaling (Normalization):** This method scales the values of a variable to a specific range, typically [0, 1]. The formula for min-max scaling is:

$$X_{\text{new}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

Where X is the original value, X_{min} is the minimum value in the dataset, and X_{max} is the maximum value in the dataset.

- **Standardization (Z-score normalization):** This method scales the values of a variable to have a mean of 0 and a standard deviation of 1. The formula for standardization is:

$$X_{\text{new}} = \frac{X - \mu}{\sigma}$$

Where X is the original value, μ is the variable's mean, and σ is the variable's standard deviation.

We have used the Standard Scaler, a specific scaling method and part of the scikit-learn library in Python. It uses the same formula as the standardization method.

3. Models

3.1 SVM

Support Vector Machine (SVM) is a supervised machine learning algorithm for classification and regression tasks. It seeks to identify the optimal hyperplane in a high-dimensional space that maximally separates data points of different classes. It maximizes the margin between the hyperplane and the nearest data points (support vectors).

SVM utilizes the hinge loss function, which promotes robust margin maximization, encouraging sparsity, resilience to outliers, and implicit regularization for effective classification.

$$L(y, f(x)) = \max(0, 1 - y \cdot f(x))$$

SVM's capabilities of classifying image data are much better than other classification models due to the kernel, which can handle linear and non-linear decision boundaries.

We are utilizing a linear kernel, and the sklearn library was utilized for training the model and testing on the test set.

```
model = SVC()
model.fit(X_T_1_scale,Y_T_1)
```

3.2 Multi-Layer Perceptron

An artificial neural network called a Multi-Layer Perceptron (MLP) comprises several layers of neurons organized in a directed acyclic graph. As a class of feedforward neural networks,

MLPs only allow information to flow in one way, without cycles or loops, from the input layer via the hidden layers to the output layer.

We employed the MLP Classifier with a random state of 1, ReLU activation function, and hidden layer sizes of (10, 5). Two models were trained, one on the standardized pixel values (`mlp_clf2`) and the other on normalized pixel values (`mlp_clf3`). The latter normalization was performed by dividing pixel values by 255.

```
mlp_clf = MLPClassifier(random_state=1, activation = 'relu',max_iter=100000,hidden_layer_sizes=(10, 5))
mlp_clf.fit(train_images,train_labels)
```

```
MLPClassifier(hidden_layer_sizes=(10, 5), max_iter=100000, random_state=1)
```

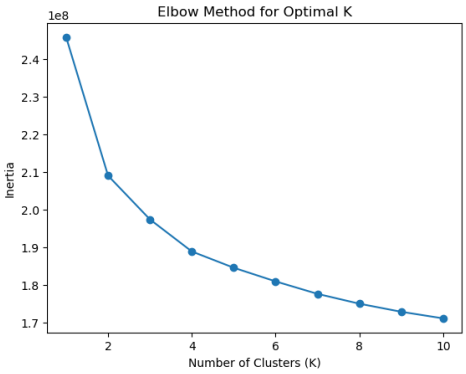
The training set consisted of real and fake images with corresponding labels (1 for real, 0 for fake). The MLP models were trained on the training sets (`mlp_clf2.` and `mlp_clf3`), and predictions were made on the test sets (`mlp_clf2.predict` and `mlp_clf3.predict`). The model's performance was evaluated using the accuracy metric, and confusion matrices were generated to assess their classification capabilities further.

3.3 K-Means Clustering

K-means clustering is a partitioning algorithm used in unsupervised machine learning to group related data points into a set number of clusters. The technique reduces variation within each cluster, making data points comparable to those in other clusters.

The Elbow Method was employed to determine the algorithm's optimal number of clusters (K). The optimal K thus chosen was determined by the point where the inertia (cost) exhibited an ‘elbow’ in the plot.

The model was then trained on the normalized training set using this optimal value of K.




```
kmeans = KMeans(n_clusters = optimal_k, random_state = 42)
kmeans.fit(train_images_normalized)
```

3.4 GMM

The Gaussian Mixture Model (GMM) is a probabilistic model used for clustering and estimating density. The data is assumed to be created by combining many Gaussian distributions, each representing a different cluster. GMM distributes probabilities to data points, allowing them to belong to numerous clusters simultaneously.

The probability density function of a GMM is a weighted sum of the probability density functions of its components. Mathematically, it is expressed as:

$$P(x) = \sum_{i=1}^{k} \pi_i \cdot \mathcal{N}(x|\mu_i, \Sigma_i)$$

Here, k is the number of components, π_i is the weight of the i-th component, and $N(x|\mu_i,\Sigma_i)$ is the Gaussian distribution for the i-th component.

Expectation-Maximization (EM) method: The Expectation-Maximization (EM) method is commonly used to estimate the parameters of a GMM. Iteratively, the EM algorithm executes two steps:

Expectation (E-step): Calculate the odds that each data point corresponds to each component (responsibility).

Maximization (M-step): Based on the responsibilities, update the settings of each component.

```
gmm = GaussianMixture(n_components=2)
gmm.fit(X_train)

GaussianMixture(n_components=2)
```

3.5 Logistic Regression

Logistic Regression is a statistical method for problems with binary categorization that predicts the likelihood that an instance belongs to a specific class. It describes data and explains the relationship between one dependent binary variable and one or more independent nominal, ordinal, interval, or ratio-level variables.

Sigmoid Function (Logistic Function): Logistic regression employs the sigmoid function to convert any real-valued integer into a value between 0 and 1. The sigmoid function reduces the output to the range (0, 1) and predicts the likelihood that a given input belongs to the positive class.

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Here, z is the linear combination of the input features and their associated weights.

Cost Function (Log Loss): The cost function in logistic regression is often expressed as the log loss (or cross-entropy loss), which measures the difference between the predicted probability and the actual class labels.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

Here, m is the number of training examples, $y^{(i)}$ is the actual class label of the i-th example, and $h_\theta(x^{(i)})$ is the predicted probability.

We have utilized the Logistic Regression model in the sklearn library Python to fit our dataset and analyze the results.

```
model.fit(scaled_images,shuffled_labels)

▾ LogisticRegression
LogisticRegression()
```

4. Results

4.1 SVM Classification

The SVM model was trained on 20000 data points and tested on a test dataset containing 5000 images divided between fake and real images.
The model could accurately classify 4050 images out of 5000 images, with an accuracy score of 81%. Furthermore, the model achieved an accuracy of 90.4% on the training data.

```
[[2033,  467],
 [ 483, 2017]],
```

Test Set

```
array([[9037,  963],
       [ 959, 9041]], dtype=int64)
```

Train Set

4.2 Multi-Layer Perceptron

MLP Model Trained on Standardized Pixel Values (`mlp_clf2`) –
The model achieved an accuracy of approximately 79.05% on the test set. The confusion matrix revealed 7964 true negatives, 2036 false positives, 2155 false negatives, and 7845 true positives.

```
0.79045
[[7964 2036]
 [2155 7845]]
```

MLP Model Trained on Normalized Pixel Values (`mlp_clf3`)-
The normalized model achieved an accuracy of approximately 74.75% on the test set. The confusion matrix showed 7837 true negatives, 2163 false positives, 2888 false negatives, and 7112 true positives.

```
0.74745
[[7837 2163]
 [2888 7112]]
```

4.3 K-Means Clustering

The normalized test set was used to predict the cluster assignments, and an evaluation was made. Confusion Matrix and Accuracy Score were two of the evaluation parameters used to test the goodness of the fit of the model.

```
Confusion Matrix:
[[3426 6574]
 [5247 4753]]
```

The best accuracy we got using the K-Means Algorithm was around 59%.

4.4 Gaussian Mixture Model

The Gaussian Mixture Model was trained on 30000 images of each class from the Training dataset and tested on 20000 images of each class from the Training dataset.
The model could accurately classify 25,549 images out of 40000 with an accuracy score of 63.87%.

```
[[ 9492 10400]
 [ 4051 16057]]
```

4.5 Logistic Regression

The Logistic Regression model was trained on 20000 images of each class and tested on a test dataset containing 5000 images of each class.
The model could accurately classify 6724 images out of 10000, with an accuracy score of 67.24%.

```
Confusion matrix for Logistic Regression model:
 [[3384 1616]
 [1660 3340]]
```

5. Inferences

The AI-generated images, although highly similar to real images, contain some features that are distinct from actual ones and can be exploited to detect them.

Linear classifiers like logistic regression generate better results than unsupervised clustering models, showcasing that clustering methods can't capture the distinct features between real and images and classify them accurately.

The Elbow Method utilized in K-means couldn’t capture the fake and real images and suggested four optimal clusters, suggesting the drawbacks of unsupervised learning models for image classification.

Although the accuracy in K-Means Clustering was low, it is to be expected since the main problem statement does not lend itself to a clustering solution but rather a classification-type approach, and so even with other clustering algorithms like GMM, we could only achieve an accuracy of up to around 63%.

References

[1]https://www.geeksforgeeks.org/impact-of-image-flattening/
[2]https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/
[3]https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/
[4] https://www.analyticsvidhya.com/blog/2019/10/gaussian-mixture-models-clustering/
[5] DATASET: CIFAKE: Real and AI-Generated Synthetic Images (kaggle.com)
[6] https://arxiv.org/pdf/2303.14126v1.pdf