Dijlah Journal of Engineering Science (DJES) Vol. 3, No. 2, June, 2026, pp. 252-263
ISSN: Printed: **3078-9656, Online: 3078-9664**, paper ID: 164

Detecting fake facial images using deep machine learning

Mustafa M. Mohammed
Sciences & Technology University in Lebanon

Article Info

Article history:

Received Jan.,24, 2026
Revised Mar.,22, 2026
Accepted Apr.,18, 2026

Keywords:

Image forensics, Deep fake, Metadata analysis, Generative Adversarial Network, Error level analysis, Convolution Neural Network, Multilayer perception network, Principal Component Analysis.

ABSTRACT

One of the most popular phenomena is false photos that negatively affect our social lives, especially the sphere of politics and celeb. It is very easy to create fake images these days thanks to the pow-earful but easy to use apps available in a mobile device that leverage social media networks and the advancement of Generative Adversarial Networks (GAN), which might produce images that are identical to one another. Which are fake pictures and fake videos simple to fake, hard to identify, simple to viralise. Consequently, image processing and artificial intelligence have a significant role to play in the resolution of such problems. Therefore, fake image detection is an urgent issue that should be managed and to avoid all these adverse outcomes. The proposed algorithm in this research was (Convolution Neural Network) that was the most popular algorithm used in deep learning to identify the fake images.

Corresponding Author:

Mustafa M. Mohammed
Sciences & Technology University in Lebanon
Ramadi, Anbar, Iraq
Email: Mstfymndhrmhmd44@gmail.com

1. INTRODUCTION

Due to the technical enhancement, the widespread spread of deep fakes programs on digital photos with the emergence of Generative Adversarial Network (GAN). Because falsified facts and images can go viral and global on the Internet through the use of social media, photo and video editing are becoming easier to access and can be regarded as a cybercrime. So, the automatic generation and adjustment may take place. Or change the face in videos and pictures of a person on the basis of algorithms that are based on Deep may be considered deep fake, that is one of the phenomena that develop at an accelerated pace, learning technology. Now, the fee-was possible to produce the high-quality outputs through developing new multimedia materials that the human eye would hardly analyze as either original or fake. Deepfake.All multimedia contents which have undergone artificial modification or generation by generative machine learning models are known by this name [1]. It is also problematic that Deep Fake is a mix, as such, it proliferated to high levels of authenticity, rapid development, and ubiquity by November 2017, because of the United States [2]. States have gone to an extent of creating pornographic videos that have gained popularity in the internet. publicity and has generated curiosity among a broad spectrum of people. In January, Deep.One application was actually used in 2018 and it entailed the use of Fake. As a result Deep Fake proliferation. Was further enhanced. The frenzy of people was also raised and aroused by another object, which is celeb-face swapping. The leading IT companies have also taken action to avoid further rogue intrusion on the contentious debate about the potential and methods of policing Deep Fake technology [3].The computer is constantly advancing as more and more computing power is placed into it. The hardware prices are lowered, and even the false images and videos of high quality are generated according to the actual distribution of the data with the help of Generative Adversarial Networks (GAN) [7] and Conviction Encoder Automatic [6]. Some of the photographs do look real, even if the majority of them are obviously manipulated owing to pixelization and poor work by beginners. Manipulated photos have the power to create or ruin a politician's reputation, particularly in the political sphere. An expert is needed to determine the validity of a picture using current forensic techniques. Our study uses machine learning to solve the problem of identifying if a picture is authentic or not, making this data publicly available. This article will be broken

up into three sections: the first will concentrate on implementation details, the second will highlight the results of the experiment, and the last piece will highlight.

2. Literature Review

The detection of forged sounds, pictures, and videos has received very little attention. Nonetheless, a number of studies and initiatives are underway to ascertain potential solutions to the startling proliferation of phony images on the internet. Since Adobe is aware of how Photoshop is abused, it has made an effort to offer a sort of cure [7]. A number of these investigations are summarized in the following.Celebrities are unaware of deepfakes, which include the changing of facial photographs. Additionally, it is utilized to distort politicians' faces. Initially, the picture of Abraham Lincoln's face was switched (Badale et al., 2018)[7].A model to identify deepfakes using inconsistent head positions has been developed by Yang, Li, and Lyu (2019). The faces for different people were made using that model without changing the original facial expressions[9].Using inconsistent head orientations, Yang, Li, and Lyu (2019) developed a strategy to identify deepfakes. That model was used to create the faces of many individuals without altering their original facial expressions. Some of the concealed features in phrases and photographs applied in fake news can be discovered with the help of a set of hidden features produced on the basis of this model across several layers. One recommended pattern is TI-CNN. By delivering unique and integrated characteristics in a single location, TI-CNN is trained concurrently with text and image data[10].Kuruvilla et al. have succeeded in [6] to train a neural network on the evaluation of the 4000 error level of the real and the 4000 error level of the fake images. With the high success rate of 83-percent, the trained neural network was able to accomplish the task of defining whether the picture was real or not. The results show that the implementation of this tool on the phones will significantly reduce the spreading of fake images on the social networks.Y. Wang et al. (2021) proposed two ways of detecting fakeface photos.The initial one is the Local Binary Pattern (LBP)-Net that employs global texture characteristics to detect false faces. The five models that constituted the second approach ensemble model included LBP -Net , Gram-Net , ResNet , and two models that used InceptionResnetV1 which was pre-trained on Casia-Webfaceare and vggface 2 . Findings of using the "140K Real and Fake Faces" in the identification of fakeface photos with various image augmentations, such as downsample (66.32%), brightness (81.09%), solarize (75.04%), contrast (85.42%), and color (91.06%) [9].Digital forensics techniques are required to identify alteration and phony photographs used for illicit objectives, according to Kim and Lee [11]. As a result, the researchers in this study have been developing an algorithm that uses deep learning technology to identify phony photos, which has shown impressive results in contemporary research. Initially, image processing is done using a transformed neural network. Additionally, a high pass filter is used to the image in order to access hidden characteristics rather than semantic information. Gaussian blurring, an intermediate filter, and additional white Gaussian noise are used to generate altered pictures for investigations [12].

3. RESEARCH METHODOLOGY

Pre-programmed algorithms Principal Component Analysis (PCA) are used to train the system to detect fake images. The best algorithm in this field was used as follows:

Algorithm : Analysis of Principal Components
Input: Images Output: components
Start Step 1: Create two independent variable matrices in X dimensions. Data is represented by rows, while features are represented by columns. The number of columns equals the number of dimensions. Step 2: Deduct each column's mean Step 3: Make your data consistent The centered and standardized matrix Z is obtained by dividing each value in a column by the column's standard deviation. Step 4: Obtain Z's Covariance Calculate Z's covariance matrix, CovMat = ZTZ. Step 5: Determine Eigen Values λ and Eigen Vectors E Step 6: Sort the Eigen Vectors Take the eigenvalues λ_1, λ_2, …, λp and arrange them in order of size. In the process, sort the eigenvectors in E accordingly. (For example, if λ3 If the biggest eigenvalue , then take E's third column and insert it into the place in the first column.)

This ordered matrix of eigenvectors is called E*. The columns of P* are identical to those of P, except they are arranged differently. Keep in mind that these eigenvectors are unrelated to one another.

Step 7: Calculate the new features

Calculate Z* = ZE*.

Step 8: Remove unnecessary elements from the updated collection.

To build a PCA, keep the first feature column.

END

4. THE MODEL'S OPERATIONAL PROCESS

Five stages were required to complete this project. This is a general specification of the stages. The initial process of preparing the dataset is the selection of the corresponding genuine and fake picture dataset at (kaggel.com). Picture features are selected with the help of the PCA once the dataset is divided in terms of the cross-validation (hold-out) (80:20). The second phase will involve the classification of the dataset with the help of SVM classifiers. Lastly, as Fig. 1 illustrates. Evaluate the performance of the model with the help of a set of measures, including accuracy, recall, precision and F1-score.

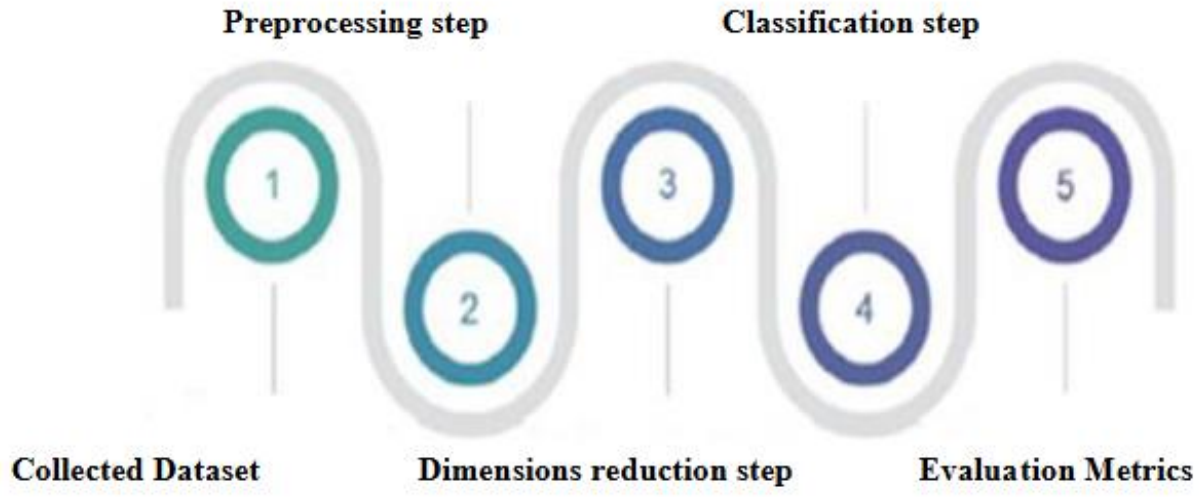

FIGURE 1. - steps in the work process of detecting fake images

5. THE PROPOSED DEEP FAKE DETECTION

A. DATASET

Deep fake detection systems are commonly based on binary classifiers that classify data into (true and fake) categories. The classification models in this method require a large volume of both authentic and fake quality data to be trained. The dataset was availed through the Kaggle site. This data consists of all the 70,000 authentic faces of the Flickr dataset that was collected by "Nvidia" itself. Moreover, 70k counterfeit faces were chosen among the 1 Million fake faces generated by Bojan (which is a product of StyleGAN). This dataset was a product of the two datasets, and each image was re-sized to 256 pixels, the data was separated into three groups namely test, validation and train. Moreover, some of the CSV files are conveniently available. In this paper, two features are used to find bogus image classifiers, i.e., pictures and labels. Label zero represents real photographs and label one represents fraudulent photographs.

B. PRE-PROCESSING STEPS

This part shows the outcomes of the pre-processing of the pictures. The data used in developing the proposed system was randomly chosen to obtain six pictures. Prior to and the outcome of the pre-processing is evidenced in the images.

The subsequent three images are counterfeit with the first three being authentic. The first column displays the images in their original form in which they were created that is RGB color space, the second column displays the transformation of the images to the YCbCr color space and the third column displays the images after the gamma correction is applied to them. The fourth column depicts the photos following the application of Canny filter. As seen in Figure 2.

FIGURE 2. - results of a preprocessing sample image from "140k Real and Fake Faces"

C. PRINCIPAL COMPONENT ANALYSIS (PCA)

Principal Component Analysis (PCA): it is one of the instances of the statistical methods to select features. It There are a number of applications where can be applied in text classification, picture compression, and face recognition . This approach for linearly The dimensional reduction of a feature dataset is standard. The primary objective of PCA is to reduce the input. several variables by determining the highest degree of correlation between the original variables . Although the final dataset is reducer, original data set characteristics are preserved and redundant data is eliminated [11].The features in the new dataset can be equal or reduce in number compared to the original dataset. The covariance matrix is they are used to estimate the key factors. The PCA could be used to improve the discriminative capabilities of the classifiers. The following steps are included in the process

- ✓ Let {X1, X2, X3,..., XN} be the training set of pictures. formula for figuring out an image's mean value

$$\bar{x} = \frac{1}{n}\sum_{i-1}^{n} xi$$

Details: n = the quantity of data, xi = the data variable

To reduce the dimension that has to be determined in the next step, the "Mean" value is calculated.

- ✓ Calculate the "Covariance Matrix" to show the dispersion degree of each feature vector connected to the average vector. The "Covariance Matrix" C is defined as follows:

$$c = \frac{1}{n}\sum_{i=1}^{n}(xi - \bar{x})(xi - \bar{x})^{T}$$

- ✓ Determine the "Eigenvalues" and "Eigenvectors" from the "Covariance Matrix"

CV=IV

The collection of "Eigenvectors" associated with its "Eigenvalue" is denoted by V.

- ✓ Based on the order of "Eigenvalues," the "Eigenvector" and "Eigenvalues" are sorted from high to low. The "Eigenvector" that corresponds to the biggest "Eigenvalue" is the primary component (k). The following equation is used to observe the vector x's primary component (k):

$$wi = vi^{T}(xi - \bar{x})$$

6. CLASSIFIER ALGORITHMS

A very important method in machine learning is the classifier algorithm, which organises or categorises data into at least one or subsets of classes . Support Vector Machines is one of the most popular and successful machine learning algorithms to classify data, used under the supervision.The primary objective of SVM is to develop as many marginal hyperplanes as possible in multi dimensional space to differentiate between different classes. The error can be reduced by constructing the hyperplane many times. Key ideas in SVM include: The data points that are closest to the hyperplane and have the biggest influence on its location are known as support vectors.

- Hyperplane: The best decision boundary for dividing or separating classes in n-dimensional space. the procedure of constructing a hyperplane with the greatest margin possible.
- SVM Kernels: To transform an entering data space into the required shape, the SVM method is implemented using a kernel.

The kernel converts non-separable problems into separable problems by adding more dimensions .

Four fundamental kernels are listed below :

- Linear : $X(yi,yj) = yi^{T} yj$
- Polynomial: $X(yi,yj) = (yi^{T} yj + r)^{d}$
- Radial basis function : $X\{yi,yj\} = \exp(-\gamma \| yi-yj \|$
- Sigmoid : $X(yi,yj) = \tanh(\gamma\, yi^{T} yj + r)^{a}$

where the kernel parameters are γ, r, and d.

The classification challenge is carried out using SVM, which was used to train the feature file. A combination of SVM hyperparameters auto-tuning between a set of parameters C, kernel, degree, and gamma values is produced by using the Grid Search tool to adjust the SVM hyperparameters. The following settings were determined to get the best results: C = 2.5, Degree = 3, Gamma = Auto, Kernel = polynomial.

The features (an array of pixel values) of the input training photos are then utilized to create a model that will eventually make classification judgments using the SVM fit technique.

Algorithm (2): Support Vector Machine (SVM)
Input: features (Pixels or PCA components)
Output: Decision of SVM (Real or Fake)
Begin Step (1): Read the dataset. Step (2): The dataset was preprocessed using Canny edge detection, gamma correction, and image color transformation. Step (3): Convert the images into features using PCA Step (4): Divide data into training and testing sets Step (5): Assign cost parameter C 2.5 Kernel K Polynomial Degree D Three Step (6): Use the current kernel K to classify the training data set. Step (7): The optimal values for C and D are found Step (8): select the testing dataset Step (9): Put the test and forecast into action Step (10): Use the Confusion Matrix to calculate testing mistakes. END

7. EVALUATION METRICS

The system's classification accuracy in detecting phony photos was evaluated using a number of assessment measures. The most popular statistic (Confusion Matrix) for identifying phony photos has been used in this part. The confusion matrix will use the categorization job classification, as shown below. where TN stands for True Negative and TP for True Positive. Furthermore, FP stands for False Positive and FN for False Negative. based on Table 1. evaluation metrics' parameters.

Table 1. - Parameters of evaluation metrics

Parameters	Description
TP (True Positive)	the total number of correctly identified phony photos.
TN (True Negative)	the total number of genuine photos that were successfully categorized
FP (False Positive)	The quantity of real photos that were incorrectly classified as fraudulent
FN (False Negative)	The quantity of phony photos that were incorrectly classified as authentic

These measures are typically employed to estimate the performance of a classifier when a variety of estimates are involved in a number of machine learning methods. Accuracy metric is used to determine the level of similarity between false images that are predicted and reality false images. Precision is the most important issue when it comes to classifying a bad photo as a fraud as this is the percentage of bad photos that have been identified as fake. This is possible by creating fewer gloomy forecasts to get enough accuracy as the true photos are not usually balanced. Recall is specifically used to estimate sensitivity or the percentage of annotated false pictures that the annotator correctly identified as fake. In particular, the larger the values, the better memory, accuracy, and precision. Finally, F1 score is also referred to as F Score or F Measure. In the F1 the P and R are well balanced

8. RESULTS AND DISCUSSION

The classification results demonstrated that the preprocessed classifier's accuracy is on par with an SVM classifier.The findings of the confusion matrix are displayed.With principal component analysis (PCA), the SVM classifier has an accuracy of 96.8%.

Table 2. - Confusion matrix of the detection of deep fake on face image using SVM only with PCA

	Positive	Negative
Positive	475	0
Negative	32	493

Accuracy =96.8
Precision =100
Recall =93.68
F1 Score=96.73

Since the goal of the research is to identify any potential manipulation in face photos, we had to conduct a test in which we preserved the image that was input into the system in order to test it and identify any modification. Some indications of alteration in the image may be buried as a result of the first processing. High classification rates, and the demonstration that any image manipulation leads to alteration or erasure of traces of manipulation, encouraged the use of established classification techniques, namely Supporting Vector Machine (SVM) without preprocessing, where the manipulation detection rate by the SVM classifier using Principal Component Analysis (PCA) reached up to 96.8%.

9. CONCLUSION

Due to the outstanding advancement of machine learning and the capacity of the GAN to produce more realistic deceptive images. To fight the deep fake phenomenon, a model of detecting fake pictures with the help of SVM and PCA was to be demonstrated, which is the final goal of our study. SVM algorithm is one of the effective machine learning algorithms that can be used to identify fake photos. The photos were first converted into the YCbCr color format prior to undergoing the gamma correction stage of preprocessing. Finally, find edges using the Canny filter on them.Next, two different detection techniques were applied: SVM with PCA and SVM alone without PCA. The greatest achievable accuracy, according to the results, is 96.8% when SVM and PCA are combined, compared to 72.2% when SVM is used alone.As a result, PCA with SVM performed better in identifying faces that were altered during classifier training procedures, and as a result, it will be used as a useful descriptor in the subsequent work.

Therefore, we infer the following inferences from the aforementioned observations.

- The preprocess on the dataset used in this article yields less acceptable results; SVM with PCA is superior to SVM only in the accuracy of classification of the fake pictures dataset. These preprocessing processes had a major influence on the decline in classification accuracy.
- The type of data utilized has a major influence on improving the classification accuracy of this study.
- The PCA features selection approach revealed reduced feature impurity; near features were eliminated to decrease feature overlap and spacing; the number of components was set at 100, where 90% of the data changes were transferred to the resulting components.
- Regular training revealed that certain spectral variables associated with the nature of the imaging system were the cause of the inaccurate classification findings.
- The method is made simpler without affecting the predictive power of the model by not using data augmentation.

10. REFERENCES

[1] A picture’s worth, Digital Image Analysis and Forensics, N Krawetz - 2007 Ph D, Hacker Factor Solutions

[2] http://imagej.net/Welcome ImageJ is an open source image processing program designed for scientific multidimensional images.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] http://forensics.idealtest.org/ CASIA v2.0 CASIA V2.0 is with larger size and with more realistic and challenged fake images by using post-processing of tampered regions. It contains 7491 authentic and 5123 tampered color images.
[4] http://neuroph.sourceforge.net/ Neuroph Framework Neuroph is lightweight Java neural network framework to develop common neural network architectures. It contains well designed, open source Java library with small number of basic classes which correspond to basic NN concepts.
[5] https://github.com/drewnoakes/metadata-extractor Metadata-extractor is a straightforward Java library for reading metadata from image files.
[6] G.Mohamed Sikandar, "100 Social Media Statistics You must know," [online] Available at: https://blog.statusbrew.com/social-media statistics-2018-for-business/ [Accessed 02 Mar 2019].
[7] Damian Radcliffe, Amanda Lam, "Social Media in the Middle East,"[online]Available:https://www.researchgate.net/publication/32318 5146_Social_Media_in_the_Middle_East_The_Story_of_2017 [Accessed 06 Feb 2019].
[8] GMI_BLOGGER,"Saudi Arabia Social Media Statistics," GMI_ blogger. [online] Available at:https://www.globalmediainsight.com/ blog/saudi-arabia-social-media-statistics/ [Accessed 04 May 2019].
[9] Kit Smith,"49 Incredible Instagram Statistics,". Brandwatch. [online] Available at: https://www.brandwatch.com/blog/instagram-stats/ [Accessed 10 May 2019].
[10] Selling Stock. (2014). Selling Stock. [online] Available at: https://www. selling-stock.com/Article/18-billion-images-uploaded-to-the-web-every d [Accessed 12 Feb 2019].
[11] Li, W., Prasad, S., Fowler, J. E., & Bruce, L. M. (2012). Locality preserving dimensionality reduction and classification for hyperspectral image analysis. IEEE Transactions on Geoscience and Remote Sensing, 50(4), 1185–1198.
[12] A. Krizhevsky, I. Sutskever, & G. E. Hinton, (2012). Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems, 1097–1105.
[13] K. Ravi, (2018). Detecting fake images with Machine Learning. Harkuch Journal [9] L. Zheng, Y. Yang, J. Zhang, Q. Cui, X. Zhang, Z. Li, et al. (2018). TI CNN: Convolutional Neural Networks for Fake News Detection. United

This is the source code of 'metric learning for anti-compression facial forgery detection'. We provide the codes and the trained models.

```
#include <iostream>
#include <string>
#include <vector>
#include <cstdlib>
#include <utility>
using namespace std;
size_t screen_cnt = 10;
string DEVICE = "RF8M70MLN8N";
string RETURN = "adb -s RF8M70MLN8N shell input tap 50 90\nsleep 1\n";
string SWIPE_RET = "adb -s RF8M70MLN8N shell input swipe 100 1245 600 1247 50\nsleep 1\n";
string APPLY = "adb -s RF8M70MLN8N shell input tap 650 1400\nsleep 3\n";
string SAVE = "adb -s RF8M70MLN8N shell input tap 680 90\nsleep 2\n";

vector<string> initImgs(){
   vector<string> imgs;
   string GALLARY = "adb -s RF8M70MLN8N shell input tap 110 1370\nsleep 1\nadb -s RF8M70MLN8N
shell input tap 210 90\nsleep 1\nadb -s RF8M70MLN8N shell input tap 220 360\nsleep 1\n";
   string SWIPE_GAL = "adb -s RF8M70MLN8N shell input swipe 360 1300 360 200 850\nsleep 1\n";
   string IMG0 = "adb -s RF8M70MLN8N shell input tap 120 300\nsleep 7\n";
   string IMG1 = "adb -s RF8M70MLN8N shell input tap 360 300\nsleep 7\n";
   string IMG2 = "adb -s RF8M70MLN8N shell input tap 600 300\nsleep 7\n";
   string IMG3 = "adb -s RF8M70MLN8N shell input tap 120 541\nsleep 7\n";
   string IMG4 = "adb -s RF8M70MLN8N shell input tap 360 541\nsleep 7\n";
   string IMG5 = "adb -s RF8M70MLN8N shell input tap 600 541\nsleep 7\n";
   string IMG6 = "adb -s RF8M70MLN8N shell input tap 120 782\nsleep 7\n";
```

```
    string IMG7 = "adb -s RF8M70MLN8N shell input tap 360 782\nsleep 7\n";
    string IMG8 = "adb -s RF8M70MLN8N shell input tap 600 782\nsleep 7\n";
    string IMG9 = "adb -s RF8M70MLN8N shell input tap 120 1024\nsleep 7\n";
    string IMG10 = "adb -s RF8M70MLN8N shell input tap 360 1024\nsleep 7\n";
    string IMG11 = "adb -s RF8M70MLN8N shell input tap 600 1024\nsleep 7\n";
    string IMG12 = "adb -s RF8M70MLN8N shell input tap 120 1265\nsleep 7\n";
    string IMG13 = "adb -s RF8M70MLN8N shell input tap 360 1265\nsleep 7\n";
    string IMG14 = "adb -s RF8M70MLN8N shell input tap 600 1265\nsleep 7\n";
    vector<string> img{IMG0, IMG1, IMG2, IMG3, IMG4, IMG5, IMG6, IMG7, IMG8, IMG9, IMG10,
IMG11, IMG12, IMG13, IMG14};

    for(size_t i = 1; i < screen_cnt; i++){
        for(size_t j = 0; j < 15; j++){
            // GALLARY + SWIPE_GAL + IMG_n
            img.push_back(SWIPE_GAL + img[(i-1)*15+j]);
        }
    }
    for(size_t i = 0; i < img.size(); i++){
        imgs.push_back(GALLARY + img[i]);
    }
    return imgs;
}

vector<vector<string>> initOpns(){

    string OP1_ = "adb -s RF8M70MLN8N shell input tap 70 1250\nsleep 1\n";
    string OP2_ = "adb -s RF8M70MLN8N shell input tap 210 1250\nsleep 1\n";
    string OP3_ = "adb -s RF8M70MLN8N shell input tap 350 1250\nsleep 1\n";
    string OP4_ = "adb -s RF8M70MLN8N shell input tap 490 1250\nsleep 1\n";
    string OP5_ = "adb -s RF8M70MLN8N shell input tap 630 1250\nsleep 1\n";
    string OP6_ = "adb -s RF8M70MLN8N shell input swipe 500 1250 335 1250 100\nsleep 1\nadb -s
RF8M70MLN8N shell input tap 350 1250\nsleep 1\n";
    string OP7_ = "adb -s RF8M70MLN8N shell input swipe 500 1250 335 1250 100\nsleep 1\nadb -s
RF8M70MLN8N shell input tap 490 1250\nsleep 1\n";
    string OP8_ = "adb -s RF8M70MLN8N shell input swipe 500 1250 335 1250 100\nsleep 1\nadb -s
RF8M70MLN8N shell input tap 630 1250\nsleep 1\n";

    string OP_0 = "adb -s RF8M70MLN8N shell input tap 85 1245\nsleep 3\n";
    string OP_1 = "adb -s RF8M70MLN8N shell input tap 253 1245\nsleep 3\n";
    string OP_2 = "adb -s RF8M70MLN8N shell input tap 421 1245\nsleep 3\n";
    string OP_3 = "adb -s RF8M70MLN8N shell input tap 589 1245\nsleep 3\n";
    string OP_4 = "adb -s RF8M70MLN8N shell input tap 700 1245\nsleep 3\n";
    string OP_5 = "adb -s RF8M70MLN8N shell input swipe 500 1245 370 1245 100\nsleep 1\nadb -s
RF8M70MLN8N shell input tap 635 1245\nsleep 3\n";
    string OP_6 = "adb -s RF8M70MLN8N shell input swipe 500 1245 250 1245 100\nsleep 1\nadb -s
RF8M70MLN8N shell input tap 295 1245\nsleep 3\n";
    string OP_7 = "adb -s RF8M70MLN8N shell input swipe 500 1245 250 1245 100\nsleep 1\nadb -s
RF8M70MLN8N shell input tap 465 1245\nsleep 3\n";
    string OP_8 = "adb -s RF8M70MLN8N shell input swipe 500 1245 250 1245 100\nsleep 1\nadb -s
RF8M70MLN8N shell input tap 635 1245\nsleep 3\n";

    string op0_0 = SAVE + RETURN;
    vector<string> op0s{op0_0};

    string op1_0 = OP1_ + SWIPE_RET + OP_0;
    string op1_1 = OP1_ + SWIPE_RET + OP_1;
```

```
string op1_2 = OP1_ + SWIPE_RET + OP_2;
string op1_3 = OP1_ + SWIPE_RET + OP_3;
string op1_4 = OP1_ + SWIPE_RET + OP_4;
vector<string> op1s{op1_0, op1_1, op1_2, op1_3, op1_4};

string op2_0 = OP2_ + SWIPE_RET + OP_0;
string op2_1 = OP2_ + SWIPE_RET + OP_1;
string op2_2 = OP2_ + SWIPE_RET + OP_2;
string op2_3 = OP2_ + SWIPE_RET + OP_3;
string op2_4 = OP2_ + SWIPE_RET + OP_4;
vector<string> op2s{op2_0, op2_1, op2_2, op2_3, op2_4};

string op3_0 = OP3_ + SWIPE_RET + OP_0;
string op3_1 = OP3_ + SWIPE_RET + OP_1;
string op3_2 = OP3_ + SWIPE_RET + OP_2;
string op3_3 = OP3_ + SWIPE_RET + OP_3;
string op3_4 = OP3_ + SWIPE_RET + OP_4;
vector<string> op3s{op3_0, op3_1, op3_2, op3_3, op3_4};

string op4_0 = OP4_ + SWIPE_RET + OP_0;
string op4_1 = OP4_ + SWIPE_RET + OP_1;
string op4_2 = OP4_ + SWIPE_RET + OP_2;
string op4_3 = OP4_ + SWIPE_RET + OP_3;
string op4_4 = OP4_ + SWIPE_RET + OP_4;
string op4_5 = OP4_ + SWIPE_RET + OP_5;
vector<string> op4s{op4_0, op4_1, op4_2, op4_3, op4_4, op4_5};

string op5_0 = OP5_ + SWIPE_RET + OP_0;
string op5_1 = OP5_ + SWIPE_RET + OP_1;
string op5_2 = OP5_ + SWIPE_RET + OP_2;
string op5_3 = OP5_ + SWIPE_RET + OP_3;
string op5_4 = OP5_ + SWIPE_RET + OP_4;
string op5_5 = OP5_ + SWIPE_RET + OP_5;
string op5_6 = OP5_ + SWIPE_RET + OP_6;
string op5_7 = OP5_ + SWIPE_RET + OP_7;
string op5_8 = OP5_ + SWIPE_RET + OP_8;
vector<string> op5s{op5_0, op5_1, op5_2, op5_3, op5_4, op5_5, op5_6, op5_7, op5_8};

string op6_0 = OP6_ + SWIPE_RET + OP_0;
string op6_1 = OP6_ + SWIPE_RET + OP_1;
string op6_2 = OP6_ + SWIPE_RET + OP_2;
vector<string> op6s{op6_0, op6_1, op6_2};

string op7_0 = OP7_ + SWIPE_RET + OP_0;
string op7_1 = OP7_ + SWIPE_RET + OP_1;
string op7_2 = OP7_ + SWIPE_RET + OP_2;
string op7_3 = OP7_ + SWIPE_RET + OP_3;
string op7_4 = OP7_ + SWIPE_RET + OP_4;
vector<string> op7s{op7_0, op7_1, op7_2, op7_3, op7_4};

string op8_0 = OP8_ + SWIPE_RET + OP_0;
string op8_1 = OP8_ + SWIPE_RET + OP_1;
string op8_2 = OP8_ + SWIPE_RET + OP_2;
string op8_3 = OP8_ + SWIPE_RET + OP_3;
string op8_4 = OP8_ + SWIPE_RET + OP_4;
string op8_5 = OP8_ + SWIPE_RET + OP_5;
```

```
    string op8_6 = OP8_ + SWIPE_RET + OP_6;
    string op8_7 = OP8_ + SWIPE_RET + OP_7;
    string op8_8 = OP8_ + SWIPE_RET + OP_8;
    vector<string> op8s{op8_0, op8_1, op8_2, op8_3, op8_4, op8_5, op8_6, op8_7, op8_8};

    vector<vector<string>> opns{op0s, op1s, op2s, op3s, op4s, op5s, op6s, op7s, op8s};
    return opns;
}

vector<pair<int, int>> rng(vector<int> sizes, int seed){
    srand(seed);
    int i0 = rand() % (sizes.size()-1) + 1;
    int i1 = rand() % (sizes.size()-1) + 1;
    while(i1 == i0) i1 = rand() % (sizes.size()-1) + 1;

    int i2 = rand() % (sizes.size()-1) + 1;
    int i3 = rand() % (sizes.size()-1) + 1;
    while(i3 == i2) i3 = rand() % (sizes.size()-1) + 1;
    int i4 = rand() % (sizes.size()-1) + 1;
    while(i4 == i3 || i4 == i2) i4 = rand() % (sizes.size()-1) + 1;
    int i5 = rand() % (sizes.size()-1) + 1;
    while(i5 == i4 || i5 == i3 || i5 == i2) i5 = rand() % (sizes.size()-1) + 1;
    pair<int, int> p0 (i0, rand() % (sizes[i0]-1) + 1);
    pair<int, int> p1 (i1, rand() % (sizes[i1]-1) + 1);
    pair<int, int> p2 (i2, rand() % (sizes[i2]-1) + 1);
    pair<int, int> p3 (i3, rand() % (sizes[i3]-1) + 1);
    pair<int, int> p4 (i4, rand() % (sizes[i4]-1) + 1);
    pair<int, int> p5 (i5, rand() % (sizes[i5]-1) + 1);
    vector<pair<int, int>> result {p0, p1, p2, p3, p4, p5};
    return result;
}

vector<string> initOps(vector<string> imgs, vector<vector<string>> opns){
    vector<string> ops;
    vector<int> sizes;
    for(size_t i = 0; i < opns.size(); i++){
        sizes.push_back(opns[i].size());
    }
    for(size_t count = 0; count < imgs.size(); count++){
        vector<pair<int, int>> random = rng(sizes, count+15486);
        string op;
        op += opns[0][0];
        op += SWIPE_RET + opns[random[0].first][random[0].second] + APPLY + SAVE + RETURN;
        op += SWIPE_RET + opns[random[0].first][0] + APPLY;
        op += SWIPE_RET + opns[random[1].first][random[1].second] + APPLY + SAVE + RETURN;
        op += SWIPE_RET + opns[random[1].first][0] + APPLY;

        op += SWIPE_RET + opns[random[2].first][random[2].second] + APPLY;
        op += SWIPE_RET + opns[random[3].first][random[3].second] + APPLY;
        op += SWIPE_RET + opns[random[4].first][random[4].second] + APPLY;
        op += SWIPE_RET + opns[random[5].first][random[5].second] + APPLY + SAVE + RETURN;

        ops.push_back(op);
    }
    return ops;
}
```

```
int main(){
    vector<string> imgs = initImgs();
    vector<vector<string>> opns = initOpns();
    vector<string> ops = initOps(imgs, opns);

    for(size_t i = 0; i < imgs.size(); i++){
        cout << imgs[i];
        cout << ops[i];
        cout << RETURN;
        cout << "echo " << DEVICE << ": Finished img " << i << endl;
    }
    cout << endl;
}"/home/user/ff/images")
```